Improving Thermodynamic Property Estimation through Volume Translation

by
Kurt Frey

M.S. Chemical Engineering Practice
Massachusetts Institute of Technology, 2005

B.S. Chemical Engineering
Ohio State University, 2004

Submitted to the Department of Chemical Engineering in partial fulfillment of the requirements for the degree of Doctor of Science in Chemical Engineering at the Massachusetts Institute of Technology.

February 2010

Signature of Author .....
Department of Chemical Engineering
January 4, 2010

Certified by .....
Jefferson W Tester
Professor of Chemical Engineering
Thesis Supervisor

Certified by .....
Michael Modell
Research Affiliate
Thesis Supervisor

Accepted by .....
William M. Deen
Professor of Chemical Engineering
Chairman, Committee for Graduate Students

Improving Thermodynamic Property Estimation through Volume Translation

by
Kurt Frey

Submitted to the Department of Chemical Engineering in partial fulfillment of the requirements for the degree of Doctor of Science in Chemical Engineering at the Massachusetts Institute of Technology.

Abstract

Steady state process simulation is used throughout the chemical industry to guide development and help reduce uncertainty and risk. Volumetric equations of state (EOS)s are the most robust of the numerous methods available for estimating various thermodynamic properties of interest because they are valid for the entire fluid phase. In some operating regimes, such as those near or above a component's critical point, EOS methods are the only option available.

Improving the property estimation accuracy of EOSs through volume translation is an attractive approach because mathematical translations can be layered onto currently implemented models without altering the underlying (untranslated) equation. However, unconstrained volume translation functions can lead to nonphysical results, such as negative heat capacities, in the translated model. This project has created a framework for modifying EOSs through volume translation so that the translated model still retains the global validity characteristic of EOS models. A novel volume translation method dependent on both temperature and density was then developed and applied to the Soave-Redlich-Kwong EOS.

This modified translation provides good molar volume accuracy (within five percent of accepted values) for a wide variety of pure compounds. The improvement in accuracy in the region around the critical point is particularly noteworthy, as the modified translation is more accurate than other, extended virial-type EOSs that require more than twice as many adjustable parameters. Phase equilibrium predictions remain largely unaffected by the translation and extension of the translated EOS to multicomponent systems proceeds in the same manner as the original model. Unknown parameter values can be reliably estimated from critical properties and ambient fluid properties without extensive regression. Significantly, mixture densities can be calculated rapidly and with good accuracy even at supercritical conditions.

Thesis Supervisor: Jefferson W Tester
Title: Professor of Chemical Engineering

Thesis Supervisor: Michael Modell
Title: Research Affiliate

Acknowledgements

I would like to express my gratitude to my advisors Jeff Tester and Mike Modell for giving me the opportunity to work on this project. One of my main reasons for going to graduate school was to get involved with energy research. The other members of my thesis committee, Ken Smith, Greg Rutledge, Paul Barton, and Alan Hatton have all been very supportive as well.

My project sponsors and associates at BP have been very generous with their time and support. In particular, I would like to thank Sue Little, Malcolm Woodman, Richard Bailey, Fatosh Gozalpour, Chuck Greco, Roger Humphreville, George Huff, and Ozie Owen.

Jason Ploeger, Chad Augustine, Scott Paap, Andy Petersen, Rocco Ciccolini, Hidda Thorsteinsson, Russell Cooper, and Michael Johnson all helped to make the basement a fun place to be, despite my constant complaining about the lack of windows. Chad, Scott, and Rocco deserve particular kudos for getting this project started while I was away at Practice School.

Gwen Wilcox had done more to keep the lab running and all of us sane than could possibly be mentioned here. I deeply appreciate all that she does, especially since I'm likely aware of less than half of it.

I'm glad to have worked with Kevin Brower, Nate Aumock, David Adrian, David McClain, John Angelos, Sohan Patel, Melanie Chin, Matt Abel, and many others from my class in the chemical engineering department. They've all had the good humor to put up with talking shop over lunch at one time or another.

My roommates and good friends, Dan Livengood, Sid Rupani, Nandan Sudarsanam, and Ben Scandella helped make Boston a lot of fun on those days I managed to escape the lab. Thanks guys.

Table of Contents

Chapter 1 – Introduction

Chapter 2 – Equilibrium Calculations

Chapter 3 – Translation Methodology

Table of Contents

Table of Contents

Chapter 7 – Conclusions and Recommendations

Appendices

Chapter 1 - Introduction

Experimentally determining unknown chemical properties is an expensive process. There are an infinite number of unknown quantities and limited resources available for investigation. Identifying which properties are valuable enough to warrant investigation is the first and perhaps most challenging step in any study. This chapter introduces the formalism of chemical thermodynamics and describes the role of equations of state in estimating thermophysical properties. Several models used for property estimation are addressed, along with the procedures and data needed to effectively use these models.

1.1 – Thermodynamic Properties

Property models and estimation methods for chemical thermodynamics predate a unified description of the field. A series of papers published in the 1870s by Josiah Willard Gibbs was the first time that many of the physical relationships between what are now considered thermodynamic properties were presented in a unified manner. Neither conservation of energy, the first law of thermodynamics, nor the tendency of entropy toward a maximum, the second law of thermodynamics, was proposed by Gibbs; however, his presentation of these laws in a simple, elegant manner served to formally define the field. The Fundamental Equation of thermodynamics (alternatively: Gibbs Fundamental Equation) is given as equation 1.1. This relationship describes a thermodynamic potential as a function of a specific set of independent variables.

$$\underline{U} = f_{\underline{U}}(\underline{S}, \underline{V}, N_1 \ldots N_n) \tag{1.1}$$

The Fundamental Equation relates the total internal energy, $\underline{U}$, with the total entropy, $\underline{S}$, the total volume, $\underline{V}$, and the molar extents of the n components present, N_j. Specifying these $n + 2$ independent variables is sufficient to completely describe the potential; a thermodynamic state is any unique set of the $n + 2$ independent variables along with the value of the thermodynamic potential at that point. The differential form of equation 1.1, which is presented as equation 1.2, is more useful.

$$d\underline{U} = \left(\frac{\partial f_{\underline{U}}}{\partial \underline{S}}\right)_{\underline{V},N} d\underline{S} + \left(\frac{\partial f_{\underline{U}}}{\partial \underline{V}}\right)_{\underline{S},N} d\underline{V} + \sum_{j=1}^{n}\left(\frac{\partial f_{\underline{U}}}{\partial N_j}\right)_{\underline{V},\underline{S},N_{\neq j}} dN_j \tag{1.2}$$

Underscored variables represent total, extensive properties; these quantities depend on the size of the system. The intensive counterparts to these variables, such as U the internal energy per mole of substance, do not have underscores.
Each of the partial derivatives of the Fundamental Equation has its own unique attributes, as indicated in equation 1.3.

$$T = \left(\frac{\partial f_{\underline{U}}}{\partial \underline{S}}\right)_{\underline{V},N} = f_{\underline{U}1}(\underline{S},\underline{V},N_1 \ldots N_n)$$

$$-P = \left(\frac{\partial f_{\underline{U}}}{\partial \underline{V}}\right)_{\underline{S},N} = f_{\underline{U}2}(\underline{S},\underline{V},N_1 \ldots N_n) \qquad (1.3)$$

$$\mu_j = \left(\frac{\partial f_{\underline{U}}}{\partial N_j}\right)_{\underline{V},\underline{S},N_{\neq j}} = f_{\underline{U}2+j}(\underline{S},\underline{V},N_1 \ldots N_n)$$

The temperature, T, the pressure, P, and the chemical potentials of the n components, μ_j, all correspond to the various partial derivatives of the Fundamental Equation. All of these variables are intensive quantities. Equation 1.4 combines equation 1.2 and 1.3 into a single expression.

$$d\underline{U} = Td\underline{S} - Pd\underline{V} + \sum_{j=1}^{n} \mu_j dN_j \qquad (1.4)$$

The Fundamental Equation completely describes all of the stable equilibrium states of a simple system, which is any system that has no internal constraints (e.g., impermeable barriers) and is not being acted on by external forces or fields. Unfortunately, external fields that could make a system non-simple are always present. Explicitly including these fields can be accomplished by additively introducing further terms into the Fundamental Equation. However, including additional terms in this manner also requires additional variables beyond the $n + 2$ identified in equation 1.1 (e.g., a gravitational field would require an independent variable for spatial position). These fields have been neglected. Only changes in the thermodynamic potential, as described by equation 1.2 or 1.4, are of interest, not the absolute value of that potential described by equation 1.1. If the external fields are not contributing to a change in the internal energy, then explicitly including those variables adds needless complexity.

1.2 – Variable Selection

The Fundamental Equation presented in equation 1.1 uses the total entropy, total volume, and molar extents as the $n + 2$ variables to describe the internal energy. Applying Legendre transformations to equation 1.1 is useful because, as indicated in equation 1.3, each of the partial derivatives of the Fundamental Equation has its own physical interpretation.

Equation 1.5 introduces three alternative thermodynamic potentials besides the total internal energy, and relates them to the internal energy through Legendre transformations.

$$\underline{H} = \underline{U} - \underline{V}\left(\frac{\partial \underline{U}}{\partial \underline{V}}\right)_{\underline{S},N} = \underline{U} + P\underline{V}$$

$$\underline{A} = \underline{U} - \underline{S}\left(\frac{\partial \underline{U}}{\partial \underline{S}}\right)_{\underline{V},N} = \underline{U} - T\underline{S} \qquad (1.5)$$

$$\underline{G} = \underline{U} - \underline{V}\left(\frac{\partial \underline{U}}{\partial \underline{V}}\right)_{\underline{S},N} - \underline{S}\left(\frac{\partial \underline{U}}{\partial \underline{S}}\right)_{\underline{V},N} = \underline{U} + P\underline{V} - T\underline{S}$$

These additional potentials are the total enthalpy, $\underline{H}$, the total Helmholtz free energy, $\underline{A}$ (alternatively: $\underline{F}$), and the total Gibbs free energy, $\underline{G}$. The differential forms of these expressions are shown in equation 1.6.

$$d\underline{H} = Td\underline{S} + \underline{V}dP + \sum_{j=1}^{n} \mu_j dN_j$$

$$d\underline{A} = -\underline{S}dT - Pd\underline{V} + \sum_{j=1}^{n} \mu_j dN_j \qquad (1.6)$$

$$d\underline{G} = -\underline{S}dT + \underline{V}dP + \sum_{j=1}^{n} \mu_j dN_j$$

The independent variables for the total enthalpy are the total entropy, pressure, and n molar extents; the total volume, temperature, and n molar extents are independent in the expression for the total Helmholtz free energy; temperature, pressure, and the n molar extents are independent in the expression for the total Gibbs free energy.

Selection of independent variables is a subjective choice and is usually dictated by the system under investigation. The four thermodynamic potentials, $\underline{U}$, $\underline{H}$, $\underline{A}$, and $\underline{G}$, all provide equivalent information. Other potentials can be defined by transforming the n molar extent coordinates, but situations where it is convenient to independently specify chemical potential are not common.

1.3 – Volumetric Equations of State

Each of the partial derivative expressions in equation 1.3 are considered equations of state because they serve to relate the $n + 2$ independent variables necessary to define a unique thermodynamic state. Equivalent relationships can be derived from the total enthalpy, total Helmholtz free energy, and total Gibbs free energy. Two of these relationships are of particular interest:

$$P = \left(\frac{\partial f_{\underline{A}}}{\partial \underline{V}}\right)_{T,N} = f_{\underline{A}2}(\underline{V}, T, N_1 \ldots N_n)$$
$$\underline{V} = \left(\frac{\partial f_{\underline{G}}}{\partial P}\right)_{T,N} = f_{\underline{G}2}(P, T, N_1 \ldots N_n) \tag{1.7}$$

Both of the relationships presented in equation 1.7 are volumetric equations of state (EOS)s. These relationships involve only experimentally accessible quantities: the temperature, the pressure, the total volume, and the molar extents. Other equations of state involve less intuitive variables, such as the total entropy or chemical potential. The most important model for this relationship is the ideal gas equation of state, which is presented as equation 1.8 .

$$P = \frac{RT}{V} \quad \text{or} \quad V = \frac{RT}{P} \tag{1.8}$$

The gas constant, R, is a constant of proportionality equal to the product of Avogadro's number and Boltzmann's constant: approximately 8.314 J/mol/K. The molar volume, V is equal to the total volume, $\underline{V}$, divided by the summation of all n molar extents, N. Similarly, mole fractions, x_j, are equal to molar extents N_j divided by the total number of moles, N. The absence of any mole fractions in equation 1.8 is notable because it implies that the ideal gas EOS is independent of composition.

All substances behaves as an ideal gas in the limit that their molar volume becomes arbitrarily large (i.e., $V \to \infty$ or $P \to 0$). The relationship presented in equation 1.8 was verified empirically and synthesized from several other gas laws; it can also be derived explicitly based on the kinetic theory of gases. The independence with respect to composition is a unique characteristic of the ideal gas EOS. Similarly, this EOS can be made explicit in either pressure or volume, which is not generally true of all EOS models.

Identifying equation 1.8 as specific to ideal gases is the first time that any of these thermodynamic relationships have been constrained to a particular phase of matter or aggregation. A sufficiently robust EOS could apply to all possible states of a system. Volumetric equations of state are not as descriptive as the complete Fundamental Equation, but they still provide a great deal of information about a system.

Many thermodynamic properties beyond pressure, temperature, total volume, and molar extents are accessible from a volumetric equation of state. These properties can be calculated through integration and differentiation of the volumetric equation of state.

1.4 – Calculating Properties from an Equation of State

Volumetric equations of state explicitly relate the pressure, molar volume, temperature, and composition of a substance. Other properties are accessible through this relationship because of its derivation from the Fundamental Equation of thermodynamics. For the purposes of calculating these properties, it is convenient to start with the extensive presentation of the Helmholtz free energy and separate it into two parts as in equation 1.9. The Helmholtz free energy is defined using the same variables as a pressure explicit volumetric equation of state.

$$\underline{A} = f_{\underline{A}}(T, \underline{V}, N_1 \ldots N_n) = f_{\underline{A}}^{0}(T, \underline{V}, N_1 \ldots N_n) + f_{\underline{A}}^{res}(T, \underline{V}, N_1 \ldots N_n) \tag{1.9}$$

The two parts of the Helmholtz free energy function in equation 1.9 represent the ideal gas contribution, $f_{\underline{A}}^{0}$, and the residual contribution $f_{\underline{A}}^{res}$. The ideal gas contribution is defined as the Helmholtz free energy of a substance in a specified thermodynamic state if behaved as an ideal gas; the residual contribution is the difference between the ideal gas value for that state and the real value. A superscript of *0* is used to denote an ideal gas state property.

As presented in equation 1.6, the partial derivative of the Helmholtz free energy with respect to total volume is the pressure

$$\left(\frac{\partial \underline{A}}{\partial \underline{V}}\right)_{T,N} = -P \tag{1.10}$$

The difference in $\underline{A}$ between any two states at a constant composition and temperature can then be expressed as an integration of an expression for the pressure.

$$f_{\underline{A}}(T, \underline{V}_2, N_1 \ldots N_n) = f_{\underline{A}}(T, \underline{V}_1, N_1 \ldots N_n) - \int_{\underline{V}_1}^{\underline{V}_2} P d\underline{V} \tag{1.11}$$

Specifying the first state in equation 1.11 as the ideal gas state (i.e., $\underline{V}_1 \rightarrow \infty$) and dropping the subscript on the second state, equation 1.11 can be rewritten as equation 1.12.

$$f_{\underline{A}}(T, \underline{V}, N_1 \ldots N_n) = f_{\underline{A}}(T, \infty, N_1 \ldots N_n) - \int_{\infty}^{\underline{V}} P d\underline{V} \tag{1.12}$$

Equation 1.13 incorporates the ideal gas contribution to $\underline{A}$ that was used in equation 1.9.

$$f_{\underline{A}}(T,\underline{V},N_1\ldots N_n)$$
$$= f^0_{\underline{A}}(T,\underline{V},N_1\ldots N_n) + f_{\underline{A}}(T,\infty,N_1\ldots N_n) - f^0_{\underline{A}}(T,\underline{V},N_1\ldots N_n) - \int_{\infty}^{\underline{V}} P d\underline{V} \tag{1.13}$$

The ideal gas EOS, equation 1.8, applies to the first three terms on the right hand side of equation 1.13; the second term describes a state at infinite volume and the first and third terms are specifically defined as the ideal gas contribution to the Helmholtz free energy. Equation 1.13 can be simplified to equation 1.14 using equation 1.11 to calculate the difference in $\underline{A}$ between any two states at a constant temperature and composition along with equation 1.8 as an expression for the pressure of an ideal gas in terms of temperature, volume, and composition.

$$f_{\underline{A}}(T,\underline{V},N_1\ldots N_n) = f^0_{\underline{A}}(T,\underline{V},N_1\ldots N_n) - \int_{\underline{V}}^{\infty} \frac{RTN}{\underline{V}} d\underline{V} - \int_{\infty}^{\underline{V}} P d\underline{V} \tag{1.14}$$

Equations 1.9 and 1.14 can then be combined to provide an expression for the residual Helmholtz free energy.

$$f^{res}_{\underline{A}}(T,\underline{V},N_1\ldots N_n) = \underline{A}^{res} = RT\int_{\underline{V}}^{\infty}\left(\frac{P}{RT} - \frac{N}{\underline{V}}\right) d\underline{V} \tag{1.15}$$

The quantity $\underline{A}^{res}$ can then be evaluated given an appropriate EOS model for pressure as a function of temperature, volume, and composition. Equation 1.15 is an important integration because many other thermodynamic properties are accessible from the residual Helmholtz free energy. In particular, the pressure can be recovered through a volume derivative of $\underline{A}^{res}$, the entropy and enthalpy departure functions are accessible through temperature derivatives, and the fugacity can be calculated by differentiation with respect to composition.

The fugacity is an especially important value because it is used in determining phase equilibria, which is discussed in chapter 2. Derivation of the fugacity starts by differentiating equation 1.9 with respect to the molar extents; this differentiation relates $\underline{A}^{res}$ to the chemical potential.

$$\left(\frac{\partial f_{\underline{A}}}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} = \mu_j = \mu^0_j + \left(\frac{\partial \underline{A}^{res}}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} \tag{1.16}$$

Equation 1.16 equates the chemical potential to an ideal gas state chemical potential and the differential of the residual Helmholtz free energy.

The chemical potential itself is a rather abstract quantity; it represents the marginal increase in internal energy of a system as mass is added at constant volume and entropy. It is useful to separate the chemical potential into two parts as indicated in equation 1.17.

$$\mu_j = RT \ln\left(\frac{\hat{f}_j}{P^{ref}}\right) + \lambda_j(T) \tag{1.17}$$

The two parts of the chemical potential in equation 1.17 represent the fugacity and a common potential contribution that depends on only temperature. Equations 1.16 and 1.17 are combined to create equation 1.18.

$$\mu_j^0 + \left(\frac{\partial \underline{A}^{res}}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} = RT \ln\left(\frac{\hat{f}_j}{P^{ref}}\right) + \lambda_j(T) \tag{1.18}$$

As presented in equation 1.6, the partial derivative of the Gibbs free energy with respect to total volume is the pressure

$$\left(\frac{\partial \underline{G}}{\partial P}\right)_{T,N} = \underline{V} \tag{1.19}$$

In an analogous manner to equation 1.12, the difference in $\underline{G}$ between any two states at a constant composition and temperature can then be expressed as an integration of an expression for the volume.

$$f_{\underline{G}}(T,P_2,N_1 \ldots N_n) = \int_{P_1}^{P_2} \underline{V} dP + f_{\underline{G}}(T,P_1,N_1 \ldots N_n) \tag{1.20}$$

Both states in equation 1.20 are specified as ideal gas states so that the ideal gas equation can be used to evaluate the integral.

$$\begin{aligned} f_{\underline{G}}^0(T,P_2,N_1 \ldots N_n) &= \int_{P_1}^{P_2} \frac{NRT}{P} dP + f_{\underline{G}}^0(T,P_1,N_1 \ldots N_n) \\ &= NRT \ln\left(\frac{P_2}{P_1}\right) + f_{\underline{G}}^0(T,P_1,N_1 \ldots N_n) \end{aligned} \tag{1.21}$$

Equation 1.21 may be rewritten as equation 1.22 by specifying the pressure in the first state as the same reference pressure from equation 1.17 and dropping the subscript for the second state.

$$f_{\underline{G}}^{0}(T,P,N_1\ldots N_n) = NRT\ln\left(\frac{P}{P^{ref}}\right) + f_{\underline{G}}^{0}\left(T,P^{ref},N_1\ldots N_n\right) \tag{1.22}$$

Differentiating equation 1.22 with respect to the molar extents creates another relationship involving the chemical potentials.

$$\mu_j^0 = RT\ln\left(\frac{P}{P^{ref}}\right) + \mu_j^0\Big|_{P=P_{ref}} \tag{1.23}$$

Equation 1.24 modifies equation 1.23 by incorporating an entropic mixing term.

$$\begin{aligned}\mu_j^0 &= RT\ln\left(\frac{P}{P^{ref}}\right) + RT\ln\left(\frac{N_j}{N}\right) - RT\ln\left(\frac{N_j}{N}\right) + \mu_j^0\Big|_{P=P_{ref}} \\ &= RT\ln\left(\frac{N_j}{N}\frac{P}{P^{ref}}\right) + \mu_j^0\Big|_{P=P_{ref}} - RT\ln\left(\frac{N_j}{N}\right)\end{aligned} \tag{1.24}$$

Equation 1.17 serves to define both the fugacity and the common potential contribution from component j, λ_j. The partition between these two functions is subjective and may be chosen so that both functions have an intuitive physical representation. Comparing equation 1.17 with equation 1.23 suggests defining the ideal gas state fugacity as the partial pressure.

$$\hat{f}_j^0 = \frac{N_j}{N}P \tag{1.25}$$

Using the definition selected in equation 1.25, the ideal gas common potential contribution, λ_j^0, is seen to be equal to the ideal gas state chemical potential minus the entropic mixing term introduced in equation 1.24. This equality implies that the composition dependence of the ideal gas state chemical potential is exactly cancelled by the entropic mixing term; the entire expression depends only on the temperature.

$$\lambda_j^0(T) = \mu_j^0\Big|_{P=P_{ref}} - RT\ln\left(\frac{N_j}{N}\right) \tag{1.26}$$

The value λ_j^0 is equal to the ideal state chemical potential of a pure component, j. When this component is present in an ideal gas mixture, the chemical potential increases by an amount equal to $RT\ln(x_j)$. Additionally, since equation 1.25 serves to define the fugacity only in the ideal gas state, it is possible to group all non-ideal gas contributions.

in equation 1.17 together with the fugacity term so that the common potential contribution is the same in all states (i.e., $\lambda_j = \lambda_j^0$).

Equation 1.24 can be now be rewritten as equation 1.27; all states in that equation were specified as the ideal gas states, so the ideal gas EOS can be used as an expression for the pressure.

$$\mu_j^0 = RT\ln\left(N_j \frac{NRT}{\underline{V}} \frac{1}{P^{ref}}\right) + \lambda_j(T) \tag{1.27}$$

The expression from equation 1.18 may be combined with equation 1.27 to eliminate the reference pressure and create an expression for fugacity. Equation 1.28 can be used to calculate the fugacity of any substance when given a valid pressure explicit, volumetric equation of state.

$$\begin{aligned}\ln\left(\frac{\hat{f}_j N}{P N_j}\right) &= \frac{1}{RT}\left(\frac{\partial \underline{A}^{res}}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} - \ln\left(\frac{P\underline{V}}{NRT}\right) \\ &= \int_{\underline{V}}^{\infty}\left(\frac{1}{RT}\left(\frac{\partial P}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} - \frac{1}{\underline{V}}\right)d\underline{V} - \ln\left(\frac{P\underline{V}}{NRT}\right)\end{aligned} \tag{1.28}$$

Calculating fugacity in this manner serves to define it as all of the non-ideal gas state and entropic mixing contributions to the chemical potential. Pressure explicit volumetric equations of state can be used to recover several other properties from $\underline{A}^{res}$, although those derivations are not reproduced here.

The same results can be obtained for volume explicit equations of state starting from an expression for the residual Gibbs free energy.

$$\underline{G} = f_{\underline{G}}(T,\underline{V},N_1 \ldots N_n) = f_{\underline{G}}^0(T,\underline{V},N_1 \ldots N_n) + f_{\underline{G}}^{res}(T,\underline{V},N_1 \ldots N_n) \tag{1.29}$$

The derivation for $\underline{G}^{res}$ proceeds identically to the derivation for $\underline{A}^{res}$ in equations 1.12 through 1.15. The result of the derivation for $\underline{G}^{res}$ is shown in equation 1.30.

$$f_{\underline{G}}^{res}(T,\underline{V},N_1 \ldots N_n) = \underline{G}^{res} = RT\int_0^P\left(\frac{\underline{V}}{RT} - \frac{N}{P}\right)dP \tag{1.30}$$

Differentiating equation 1.30 with respect to molar extent creates a relation in chemical potential.

$$\mu_j^{res} = RT\int_0^P \left(\frac{1}{RT}\left(\frac{\partial \underline{V}}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P} \right) dP \tag{1.31}$$

The partition between ideal and residual chemical potential is also the same as for the free energies.

$$\mu_j = \mu_j^0 + \mu_j^{res} = RT \ln\left(\frac{\hat{f}_j}{P^{ref}}\right) + \lambda_j(T) \tag{1.32}$$

An expression for the ideal part of the chemical potential was described in equation 1.24, which can be combined with equations 1.31 and 1.32

$$\mu_j^{res} = RT \ln\left(\frac{\hat{f}_j N}{P N_j}\right) = RT\int_0^P \left(\frac{1}{RT}\left(\frac{\partial \underline{V}}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P} \right) dP \tag{1.33}$$

Equation 1.33 is another method for calculating fugacity that is more appropriate to volume explicit equations of state that have the pressure as an independent variable. Other properties calculable from volume explicit EOSs can be recovered from $\underline{G}^{res}$, although those expressions are not reproduced here.

1.5 – Theory of Corresponding States

Many refinements to the ideal gas EOS have been proposed, although the most significant contribution came from Johannes van der Waals and bears his name. The van der Waals EOS is presented in equation 1.34. Once again, intensive molar volumes are used in order to simplify the presentation.

$$P = \frac{RT}{V-b} - \frac{a}{V^2} \tag{1.34}$$

The two constants, *a* and *b*, introduced in the van der Waals EOS represent attractive and repulsive intermolecular forces, respectively. Neither of these EOS modifications is originally attributed to van der Waals, but incorporating both into a single model enabled simultaneous representation of both the vapor and liquid phases for the first time. The relationship between the pressure and molar volume for fluid argon as predicted by both the ideal gas and van der Waals EOSs at 140K is presented below:

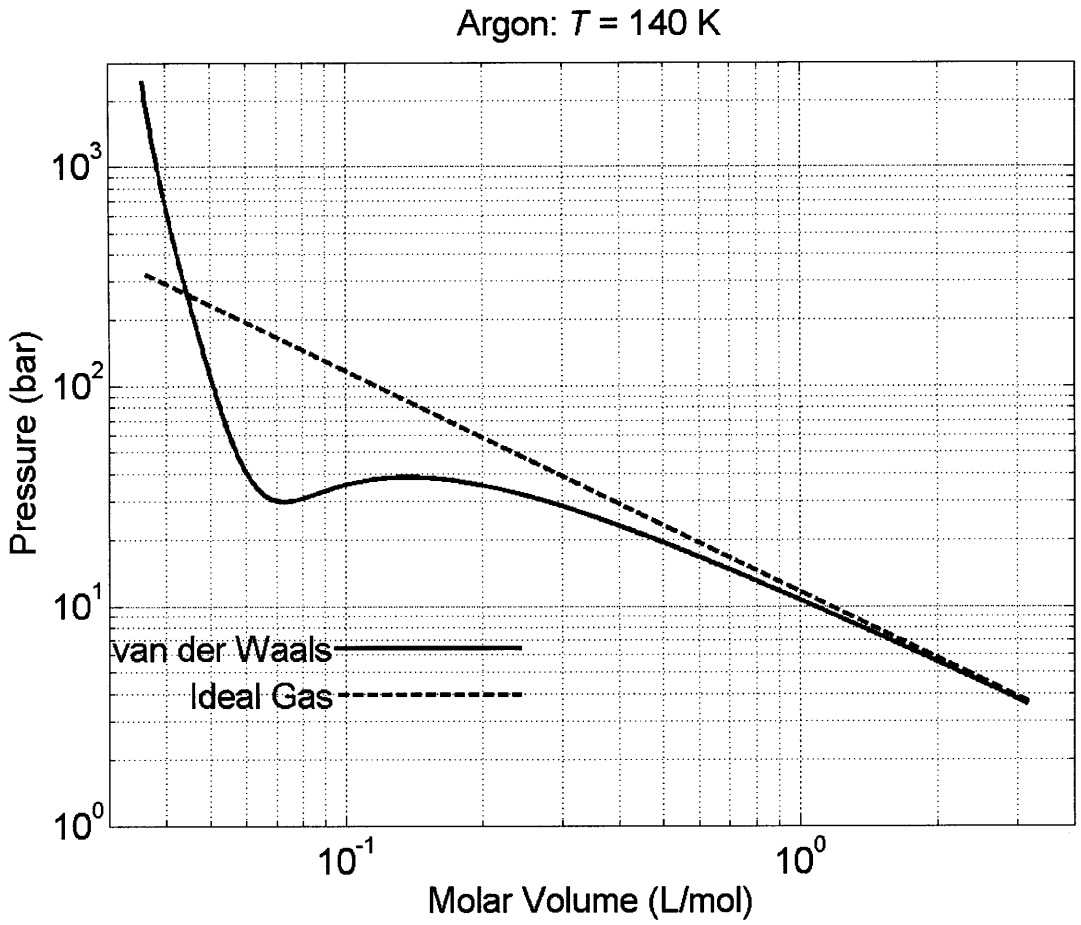

Figure 1.1: The relationship between the pressure and molar volume for fluid argon at 140K as estimated by both the ideal gas EOS and the van der Waals EOS. Parameters used for Argon in the van der Waals EOS: a = 1.36 and b = 0.0322.

The most striking feature of figure 1.1 is the multiple volume roots that can occur at a single pressure in the van der Waals EOS. The ideal gas EOS predicts only one possible molar volume at any pressure or temperature, whereas the van der Waals EOS can provide either one or three depending on the values of the *a* and *b* parameters. The smallest of these molar volume roots corresponds to a liquid phase and the largest corresponds to a vapor phase. The middle volume root always occurs along the part of an isotherm where $(\partial P/\partial V)_T > 0$, which indicates an unstable thermodynamic state as is discussed in Chapter 2. It is also important to note that the limiting behavior of the van der Waals equation at large molar volumes converges to that of an ideal gas; all EOS models need to exhibit this limiting behavior in order to be consistent with experimentally observed behavior. Van der Waals determined the values for the *a* and *b* parameters in his EOS based on his theory of corresponding states, which was potentially a greater contribution than the EOS itself. All fluids exhibit a critical point where the vapor and liquid phase become indistinguishable.

Two discontinuous phases, liquid and vapor, are experimentally observed at conditions below this critical point; only one fluid phase is observed when operating at conditions above the critical point. The theory of corresponding states hypothesizes that all fluids exhibit similar behavior that depends only on a fluid's proximity to its critical point. For example, the experimentally determined critical temperature and pressure coordinate of argon is about 150K and 50 bar. That same critical coordinate for water is about 650K and 220 bar. Argon and water are highly dissimilar fluids from a chemical perspective, yet the thermodynamic properties of argon at 75K and 25 bar are similar to those of water at 325K and 110 bar (i.e., when temperature and pressure are approximately half of their critical values).

The theory of corresponding states has proven to be remarkably accurate; it is now considered a principle of fluid behavior. In the van der Waals EOS, the values of the a and b parameters are determined using the experimentally determined critical coordinates of the fluid being modeled as indicated in equation 1.35.

$$a = \frac{27R^2T_C^2}{64P_C} \qquad b = \frac{RT_C}{8P_C} \tag{1.35}$$

The subscript C denotes the value of that property at its vapor-liquid critical point. Constraining the parameters a and b as indicated in equation 1.35 causes the van der Waals EOS to transition from predicting three molar volume roots to one molar volume root at a pressure and temperature equal to that of the specified P_C and T_C. Unfortunately, this selection of parameter values also forces the EOS predicted third critical coordinate, V_C, to be estimated as a constant value of $3b$. Actual experimental values for the critical volume vary depending on the fluid and tend to be less than $3b$, but the van der Waals EOS is constrained to match only two of the critical coordinates by virtue of only having two free parameters.

Specifying parameter values as indicated in equation 1.35 also introduces implicit composition dependence into the van der Waals EOS. As in the ideal gas equation, mole fractions do not explicitly appear in the EOS formulation, equation 1.34. However, the van der Waals model incorporates substance dependency into the two parameters, a and b, which are calculated based on fluid specific critical properties. Using the values for a and b given in equation 1.10, the van der Waals EOS can be rewritten as in equation 1.36.

$$P_r = \frac{8T_r}{3V_r - 1} - \frac{3}{V_r^2} \tag{1.36}$$

Variables with a subscript r indicate denote reduced variables; reduced variables are quantities that have been divided by their value at the critical point (e.g., $P_r = P/P_c$). The model estimated value of $3b$ has been used for V_C in equation 1.36 when determining the reduced volume.

It is possible to create generalized EOSs such as the one in equation 1.36 because of the principle of corresponding states. Only two data points, the critical temperature and critical pressure, are required to use a model that can provide thermodynamic property estimates throughout the fluid phase.

1.6 – Model Accuracy and Utility

Van der Waals' equation is significant for its novel ability to simultaneously represent both the vapor and liquid phases, but has only found use qualitatively because the accuracy of many of the estimated condensed phase properties is poor. The corresponding states approach pioneered by van der Waals has been adapted and extended by many other investigators to improve on upon condensed phase accuracy. Other coordinates besides the critical pressure and critical temperature have been proposed for use in corresponding states-based EOS models. Pitzer's acentric factor, ω, involves the saturation pressure at a reduced temperature of 0.7 is defined as indicated in equation 1.37.

$$\omega = -\log_{10}\left(P_r^{sat}\right)_{T_r=0.7} - 1 \tag{1.37}$$

The acentric factor is a popular choice for a third correlating value because it is easily accessible and can be precisely measured. Incorporating the actual critical molar volume, V_C, in corresponding states based models is not as common because of the challenge in determining it experimentally.

More than a century has passed since van der Waals proposed his equation and the theory of corresponding states, yet to this day most of the industrially useful EOS models are modified versions of his equation. The Redlich-Kwong model, presented in equation 1.38, decreases the magnitude of intermolecular attractive forces and incorporates a temperature dependency into that term.

$$P = \frac{RT}{V-b} - \frac{a}{V(V+b)\sqrt{T}} \tag{1.38}$$

The first model to find widespread use for quantitative property estimates in the condensed phase was Soave's modification to the Redlich-Kwong EOS. The combined Soave-Redlich-Kwong (SRK) model is presented in equation 1.39.

$$P = \frac{RT}{V-b} - \frac{a\cdot\alpha}{V(V+b)}$$
$$\alpha = \left(1+\left(0.480+1.574\omega-0.176\omega^2\right)\left(1-\sqrt{T_r}\right)\right)^2 \tag{1.39}$$

The Soave modification alters the temperature dependency of the intermolecular attractive forces to include a temperature dependent function, α, which is used to more accurately estimate saturation pressures. The alpha function includes an additional substance dependency that is correlated using the acentric factor introduced in equation 1.37. The alpha function is also constrained so that it does not alter the EOS predicted vapor-liquid critical point; the function is unity at the critical temperature.

Because of the modification of the van der Waals EOS, the *a* and *b* parameter values needed to reproduce the vapor-liquid critical point the in the Redlich-Kwong and SRK EOSs were altered as well. The modified values are given in equation 1.40.

$$a = \frac{R^2 T_C^2}{9\left(\sqrt[3]{2}-1\right)P_C} \qquad b = \frac{\left(\sqrt[3]{2}-1\right)RT_C}{3P_C} \tag{1.40}$$

Generally, the *a* and *b* parameters for a particular fluid are selected so that the values of P_C and T_C specified for that fluid are reproduced exactly at the vapor-liquid critical point predicted by the EOS model. The model predicted critical point of pure fluids is easily identifiable as the threshold where three distinct molar volume roots are observed to converge into one molar volume root. This threshold can be calculated by applying the criteria given in equation 1.41.

$$\left(\frac{\partial P}{\partial V}\right)_T = \left(\frac{\partial^2 P}{\partial V^2}\right)_T = 0 \tag{1.41}$$

Determining the value of *a* and *b* for fluid mixtures is more complicated because critical property data for mixtures is rarely available and the critical point criteria are more complex than what is presented in equation 1.41. One of the best features of corresponding states-based models is their ability to provide quantitatively useful property estimates with only two or three experimentally measured values: T_C, P_C, and ω. The vapor-liquid critical point of a mixture changes with composition; there is a (potentially discontinuous) line of vapor-liquid critical points even for a binary mixture. Requiring mixture critical data would severely limit the utility of any EOS model.

The saturation pressure of a fluid mixture is not well defined; a liquid mixture in equilibrium with a vapor mixture of the same composition describes an azeotrope, which is experimentally observed in only a small subset of fluid mixtures. The acentric factor described in equation 1.37 is only relevant for pure fluids. Extending EOS models to fluid mixtures and describing the composition dependency of EOS parameters is addressed in Chapter 5. Condensed phase properties can be estimated with reasonable accuracy using the SRK EOS, but substantial improvement is still possible. In particular, the highly compressible region around the vapor-liquid critical point is not estimated with an acceptable level of accuracy. An example set of molar volume predictions from the SRK EOS for Argon at 60 bar is presented in figure 1.2.

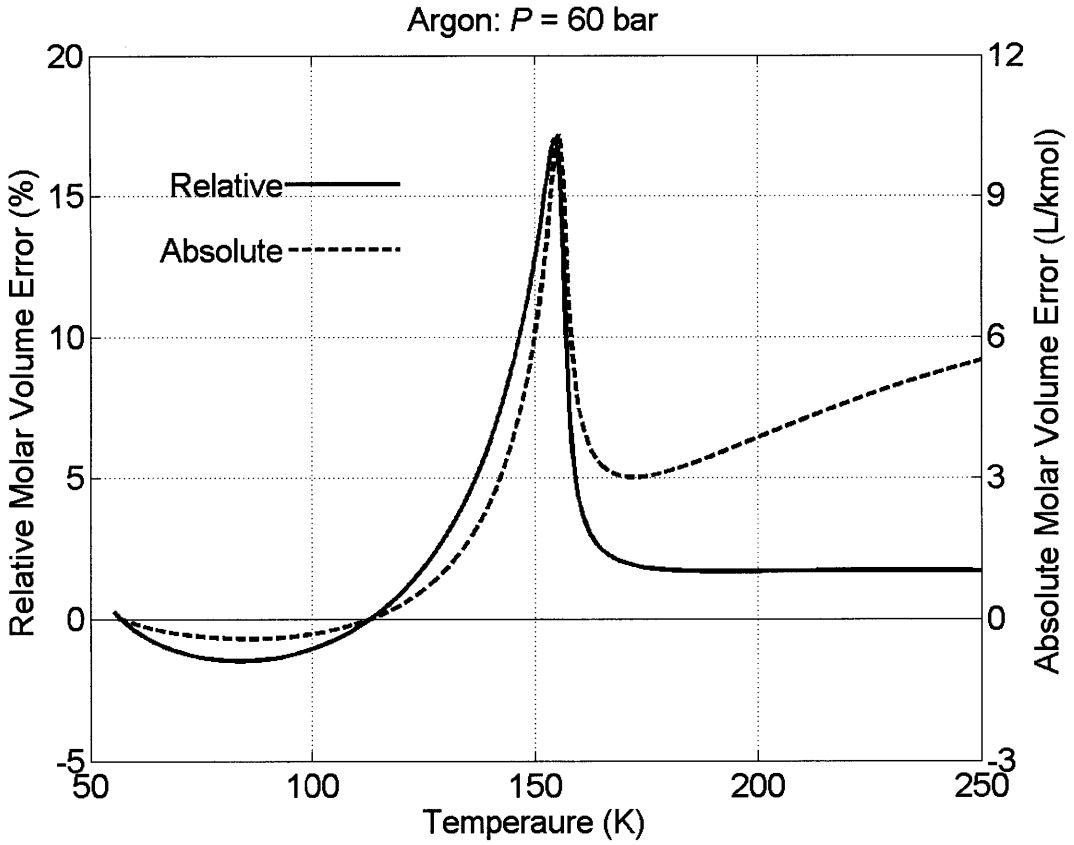

Figure 1.2: Accuracy of molar volume estimates for Argon at 60 bar using the SRK EOS. Accepted values for the molar volume of Argon were taken from Tegeler et al.

Many other EOS models have been developed. Yet despite the number of available methods, models such as the SRK continue to be used extensively because the estimations shown in figure 1.2 still represent the best available option.
Highly accurate correlations for molar volume are possible, but these correlations are very complicated, can require hundreds of regressed parameters, and may not be valid for the entire fluid region. The importance of a simple, broadly applicable model can not be overemphasized.

Several other simple models based on modifications of the van der Waals EOS are also widely used. These models, similar to the SRK EOS, are usually grouped together into a class of EOSs called cubic equations, so named because the terms in the equation are at most third order in molar volume. Cubic equations can be solved explicitly for pressure given temperature and molar volume, or explicitly for molar volume given pressure and temperature. This convenience is not as important toady as it was when the models were developed given the computational resources currently available.

However, it is important to recognize the extent of the body of work that has been done in support of cubic equations; an established database of accepted parameters is often worth more than model itself. Modified-cubic equations or generic van der Waals-type equations refer to EOS relations that have been developed based on cubic equations in order to take advantage of the large body of legacy work that has been done in this field. These modified models may not retain the explicit solution ability characteristic of the original cubic models, but are still widely applicable and simple to implement.

1.7 – Summary

A volumetric equation of state model provides an enormous amount of information about the thermodynamic properties of a system. This information can be obtained using a limited amount of experimental data through the application of the principle of corresponding states. Current EOS models can provide quantitative estimates for throughout the fluid phase, but improvements in accuracy are still needed. However, these improvements in accuracy cannot override the need for simplicity. Model selection is driven both by the accuracy of property estimation as well as the usability of the representation.

References

J Lebowitz; Studies in Statistical Mechanics Volume XIV: J.D. van der Waals: On the Continuity of the Gaseous and Liquid States. New York: North-Holland Physics, 1988.

P Mathias, H Klotz; *Chem. Eng. Prog.* 6 (1994), 67-75.

K Pitzer; *J. Am. Chem. Soc.*, 77 (1955), 3427-3433.

K Pitzer, D Lippmann, R Curl Jr., C Huggins, D Petersen; *J. Am. Chem. Soc.*, 77 (1955), 3433-3440.

O. Redlich, J. Kwong; *Chem. Rev.*, 44 (1949), 233-244.

S Sandler; Chemical and Engineering Thermodynamics. 3rd Ed. New York: John Wiley & Sons, 1999.

G Soave; *Chem. Eng. Sci.*, 27 (1972), 1197-1203.

C Tegeler, R Span, W Wagner; *J. Phys. Chem. Ref. Data*, 28 (1999), 779-850.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

Chapter 2 - Equilibrium Calculations

Any one of the instantiations of the Fundamental Equation of thermodynamics completely describes all of the stable equilibrium states of a simple system. Unfortunately, even when that equation given, determining those equilibrium states is not an easy task. This chapter describes the methods used to calculate the equilibrium states of a simple system when given an appropriate model that describes that system. For the purpose of this discussion, the accuracy of the model is not addressed.

2.1 – Equilibrium Conditions

Any of the various thermodynamic potentials can be used to formulate the criteria for equilibrium. Here, the Gibbs free energy, G, is used because the natural variables associated with it include temperature and pressure, T and P. From the perspective of process control, these two variables are typically the most precisely measured and most easily controlled. The molar quantities, N, of the n chemical species present are used in addition to temperature and pressure to complete the set of $n + 2$ independent variables required to fully describe the Gibbs free energy.

A state is a stable equilibrium state if its Gibbs free energy is at a minimum; that is, small changes in temperature, pressure, or molar quantity only increases the Gibbs free energy of a state at a stable equilibrium. Selecting the thermodynamic potential that has temperature, pressure, and the molar quantities as the independent variables is useful because these are most often the variables explicitly held constant in systems of interest. However, holding a variable constant at the system level does not imply that the variable is uniform throughout the system. As an example, it is useful to consider the internal variation of an isolated equimolar mixture of two dissimilar substances, X and Y, shown in figure 2.1.

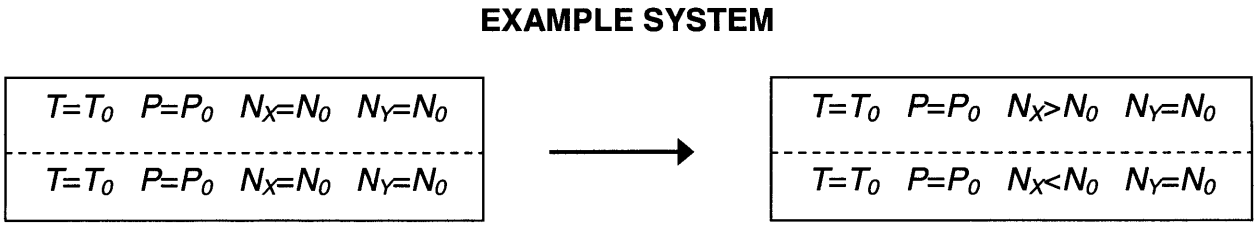

Figure 2.1: An isolated system of equal amounts of species X and species Y spontaneously undergoes a change of state: one resultant state is enriched in X and the other is depleted in X. The subscript *0* does not indicate an ideal gas property; it is used to imply an arbitrary, constant value.

The molar quantities in this example system are the same on both the left and the right, but are allowed to vary internally. The top and bottom subsystems on the left side of figure 2.1 are in the same thermodynamic state; the dashed line separating these subsystems is only hypothetical. However, these subsystems can spontaneously sample two different states in the subsystems on the right; transfer of species X across the hypothetical boundary results in two new states, one enriched in species X and one depleted in species X. The original, uniform system presented on the left in figure 2.1 is a stable equilibrium state only if neither of the substates on the right has lower Gibbs free energy.

Stable equilibrium states are readily characterized mathematically as minimums in Gibbs free energy; that is, the derivative of the Gibbs free energy is zero with respect to all of its independent variables, and the local curvature with respect to those variables is positive. The challenge of equilibrium calculations is to determine a more complete characterization of the local Gibbs free energy surface and not simply a single local minimum. A uniform system with a single stable state can spontaneously separate into multiple phases if each of the phases is also a stable state and if the total Gibbs free energy of the multiphase system is lower than that of the uniform system.

This problem is essentially unbounded because the chemical species present in a system can undergo reactions to form new substances. Mathematically, the number of chemical species, n, is arbitrarily large and those not present have their corresponding molar quantities, N_j, set to zero. In this context, determining a true global minimum in Gibbs free energy is not useful because the energetic barriers between any initial state and the true global minimum are prohibitive.

Only chemical species that are initially present in a system are considered. This approximation serves to frame the equilibrium problem in a more tractable context. Whether or not the approximation is valid, and how much error is introduced, must be dealt with for each particular system investigated.

2.2 – Consistent Variable Specification

The equilibrium problem still remains complex even after constraining the equilibrium problem to only include the species initially present. By allowing additional phases to form, additional variables are introduced that must be considered. Approaching the problem from a systems perspective is useful.

Irrespective of the number of phases, $n + 2$ independent variables are always required for a complete characterization. Selecting which variables to specify can be challenging because the introduction of additional phases can create variable dependency that is not immediately obvious.

For a system with π phases there are $(n + 1)\pi$ variables needed in addition to T and P: $x_{11} \ldots x_{1n}, \ldots, x_{\pi 1} \ldots x_{\pi n}, N_1 \ldots N_\pi$. The index i (and l as needed) is used to denote phase and the index j (and k as needed) is used to denote component. Note that in this set of variables, the index on the molar quantities, N, corresponds to total moles in a given phase and not total moles of a given component. The variables, x_{ij}, are the mole fractions of component j in phase i; their definition provides an initial set of π constraints:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1 \ldots \pi \tag{2.1}$$

An additional n relations are required for each phase beyond the first phase. A species that distributes between more than one phase must have equal chemical potential in all phases in which it is present. The $n(\pi - 1)$ constraints that arise from additional phases are given in equation 2.2:

$$\mu_{1j}(T, P, x_{11} \ldots x_{1n}) = \ldots = \mu_{\pi j}(T, P, x_{\pi 1} \ldots x_{\pi n}) \qquad \forall j = 1 \ldots n \tag{2.2}$$

There needs to be a correspondence between variables and constraining equations in any system of equations. This assignment does not need to be unique, but it does need to exist in order to determine which variables are independent. A concrete example of the variable assignment required for a single component, two phase system is given in figure 2.2.

	T	P	N_1	N_2	x_{11}	x_{21}
1*a*					×	
1*b*						×
2	×	×			×	×

Figure 2.2: An occurrence matrix showing the relationship between variables and constraints for a single component, two phase system. Each row represents a constraining equation; each column corresponds to one of the system variables.

One of the most revealing features of the correspondence shown in figure 2.2 is that the molar extents of each phase (i.e., $N_1 \ldots N_\pi$) do not appear in any of the constraints given in equations 2.1 or 2.2, so these variables must be specified. In this example, only one other independent variable is left to specify; the temperature and pressure are not independently variable quantities. Every phase that is present introduces a molar extent that is not found in any of the constraining equations, so one extensive variable must be specified for each phase present. A total of $n + 2$ independent variables are required regardless of the number of phases present, so the total number of phases possible is limited to $n + 2$ at maximum.

This result is a restatement of Gibbs' Phase rule, which imposes a limitation on the number of independently variable intensive properties in a system with a specified number of phases.

2.3 – Existence and Uniqueness of Solutions

The existence of a stable equilibrium state is guaranteed given physically reasonable values for the independent variables (i.e., positive values for molar extents, temperature, and pressure). Unfortunately, there is no way of knowing *a priori* how many phases will be present. The number of phases is bounded between 1 and the upper bound of $n + 2$ established in section 2.2. It is necessary to assume an equilibrium number of phases in order to determine the appropriate number of constraints and initialization set. The key difficulty in these calculations is that the trivial solution of $\pi - 1$ unique stable states occurring across π assumed phases is always possible, given that $\pi > 1$. The two degenerate states do not represent two physically distinguishable phases. This solution may indicate either that π phases are not present or that the dependent variables in the system of constraints from equations 2.1 and 2.2 were not initialized sufficiently close to the π state solution.

There is no way to determine if the π state solution exists without some physical intuition about the system being examined. Good initial guesses for the dependent variables are indispensable even with a perfect model for the system. In practice, good initial guesses are most acutely needed in situations when stable states are very similar in composition. Azeotropes and the vapor-liquid critical points are familiar examples. Topologically, the trajectories of two stable states (i.e., minima in Gibbs free energy) intersect on the potential energy surface. These states become arbitrarily close to one another so that the energetic barrier separating them becomes zero. In the case of an azeotrope, these minima are still distinct and the trajectories are simply crossing. For a critical point, these trajectories merge and the intersection represents a bifurcation point. If equilibrium calculations converge to a $\pi - 1$ state solution when a π state solution is believed to exist, the calculations must be repeated with different initialization of the dependent variables. The topology of the potential energy surface can vary dramatically, so the absolute difference in value between the dependent variables at initialization and at the solution may not always correspond to the likelihood of convergence.

A very simple example of this difficulty can be seen in the steady-state (i.e. long time horizon) behavior of equation 2.3.

$$\frac{dx}{dt} = rx - x^3 \tag{2.3}$$

The steady-state value of x in equation 2.3 depends on the value of the r parameter.

This dependence is shown in figure 2.3.

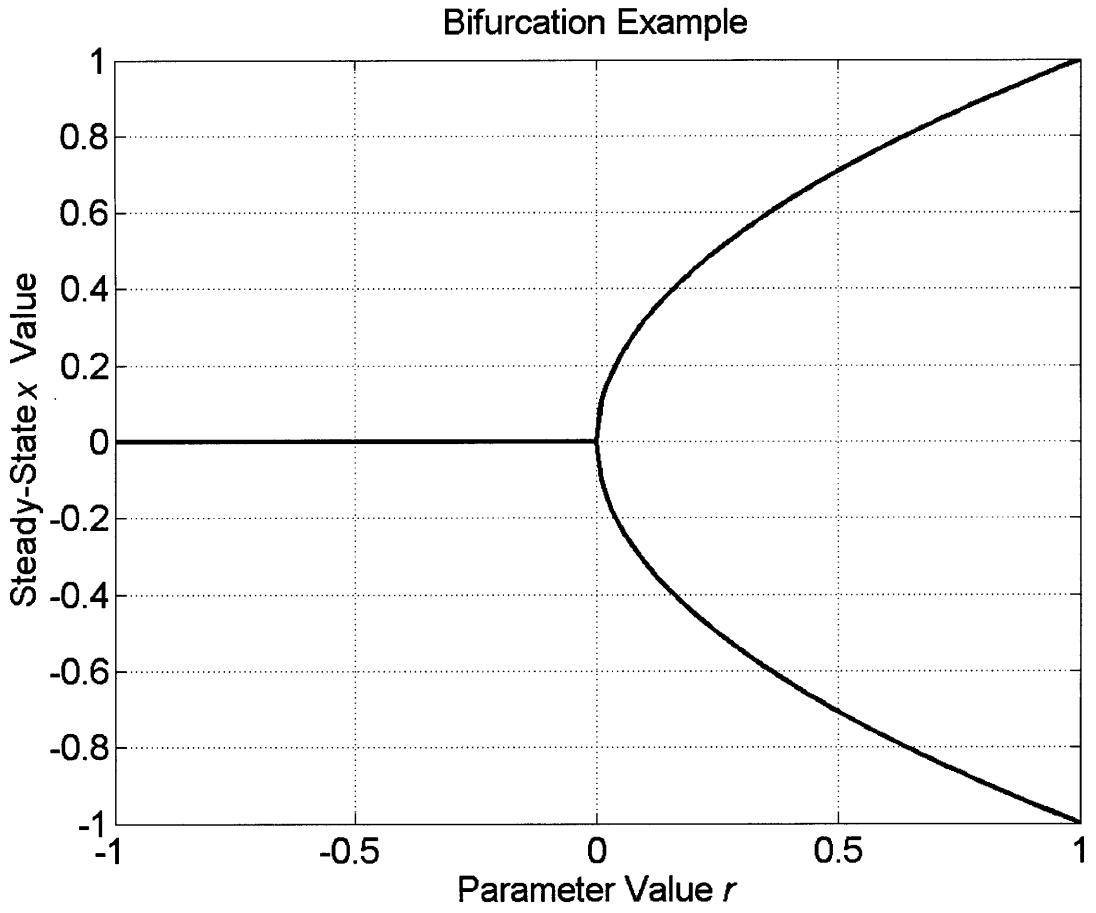

Figure 2.3: The steady-state behavior (i.e., long time horizon) of equation 2.3 depicted as a function of value selected for the r parameter.

For values of $r < 0$, the only steady state occurs at $x = 0$. However, there are two possible steady state solutions when $r > 0$. The steady-state behavior of equation 2.3 is similar to pure component vapor-liquid critical behavior in thermodynamics. At $T > T_C$ and $P > P_C$ only one possible stable equilibrium state exists. Below the critical point two possible steady-states exist; a vapor state and a liquid state. The dynamics of these equilibria is not addressed.

The behavior of multicomponent systems can be much more complex, with many more than two possible steady-states and critical parameter values that are not obvious from the model formulation. Determining these solutions by inspection is impossible and more sophisticated methods are required.

2.4 – Flash Calculations

The flash calculation is the most important equilibrium calculation that is performed; numerical simulations of a distillation column can require thousands of flash calculations. Correspondingly, the techniques for these calculations are robust and well developed.

A flash calculation is defined by its set of specified variables: two intensive variables, as well as overall compositions, are specified. The goal of the calculation is to determine the number and extent of the equilibrium phases. As indicated in section 2.1, temperature and pressure are the most common process variables to specify because of accuracy and control considerations. Molar enthalpy is also relatively common substitute for temperature because rapid processes can be approximated as isenthalpic to good accuracy. However, only isothermal flashes are considered here. The discussion of isenthalpic flashes is similar.

The chemical potential is a difficult quantity to use from a computational standpoint; the fugacity is a useful substitute. In particular the fugacity coefficient, which is equal to fugacity divided by the partial pressure as indicated in equation 2.4, will be used.

$$\mu_j - \lambda_j = RT \ln\left(\frac{\hat{f}_j}{P^{ref}}\right) = RT \ln\left(\hat{\phi}_j \frac{P}{P^{ref}} x_j\right) \tag{2.4}$$

As discussed in chapter 1, the value of λ in equation 2.4 is only a function of temperature, so it does not influence the equality of chemical potential constraints given in equation 2.2; these constraints can now be restated in terms of equifugacity conditions as in equation 2.5.

$$\hat{f}_{1j}(T, P, x_{11} \ldots x_{1n}) = \ldots = \hat{f}_{\pi j}(T, P, x_{\pi 1} \ldots x_{\pi n}) \qquad \forall j = 1 \ldots n \tag{2.5}$$

Using both the pressure and temperature as variables in the specification set has the potential to cause difficulty because of the limits on intensive specifications that were discussed in section 2.2. The one component, two phase system used to create the occurrence matrix in figure 2.2 would be underspecified without two extensive quantities. The pressure and temperature are interdependent in any system where the number of phases exceeds the number of components present. Fortunately, numerical precision helps to limit these concerns. Pressure and temperature are both continuous variables; selecting a pair of pressure and temperature coordinates that are not independent is essentially impossible under finite precision. It is possible to use this result to further limit the maximum number of phases in a flash calculation to n.
The flash calculation is fully specified using the pressure, temperature, and n overall molar extents. The total differential of the Gibbs free energy is given in equation 2.6.

Note that in this equation, the index on the molar quantities corresponds to total number of moles of a component.

$$d\underline{G} = -\underline{S}dT + \underline{V}dP + \sum_{j=1}^{n} \mu_j dN_j \tag{2.6}$$

From equation 2.6, a minimization in Gibbs free energy at constant temperature and pressure can be reposed in terms of minimizing the total chemical potential. Coupled with the definition given in equation 2.4, it is possible to work entirely in terms of fugacity or the fugacity coefficient.

The most convenient and efficient formulation of the minimization problem is given in the Rachford-Rice equation:

$$Q(\beta_1 \ldots \beta_\pi) = \sum_{i=1}^{\pi} \beta_i - \sum_{j=1}^{n} \frac{N_j}{\sum_{k=1}^{n} N_k} \ln\left(\sum_{i=1}^{\pi} \frac{\beta_i}{\hat{\phi}_{ij}} \right) \tag{2.7}$$

In equation 2.7, the objective function, Q, must be minimized with respect to the phase fractions, β_i, in order to determine the stable equilibrium state. The mole fractions within of each of the phases can be calculated as described in equation 2.8.

$$x_{ij} = \frac{\dfrac{\hat{\phi}_{i1} N_j}{\hat{\phi}_{ij} \sum_{k=1}^{n} N_k}}{1 + \sum_{l=1}^{\pi} \beta_l \left(\dfrac{\hat{\phi}_{1j}}{\hat{\phi}_{lj}} - 1 \right)} \tag{2.8}$$

Given a model of the component fugacities in the system, an initial estimate of the mole fractions, x_{ij}, can be used along with the temperature and pressure to calculate an initial estimate of the fugacity coefficients. Equation 2.7 can then be minimized to determine the converged phase fractions, β_i. The converged phase fractions and initially estimated fugacity coefficients can then be used in equation 2.8 to determine new estimates for the mole fraction x_{ij}^*. The model can be used to calculate updated fugacity coefficients based on the new estimates of the mole fractions, and the process repeated until consistent solutions have been obtained. Equation 2.7 was designed so that this process of successive substitution always converges to a solution. The phase fractions, β_i, are not constrained to physically realistic values and therefore the solution to the minimization procedure described previously may yield phase fractions less than zero or greater than one. In the event that minimization of equation 2.7 results in one or more phase fractions that are not physically realistic, the number of phases is reduced by one and the calculation is restarted.

In the event that not all components distribute uniformly throughout all phases (e.g., a pure solid phase), the fugacity coefficient of those components for the phases in which they are not present is set to an arbitrarily large value (i.e., the inverse of these fugacity coefficients is zero). The phase assigned an index of one must have all species present because of the way mole fractions are calculated in equation 2.8.
It is possible to redefine the mole fraction expression using a different reference phase for each component if there is no single phase with all components present, but that is rarely necessary. The phase with index one is typically the vapor-like or least dense phase.

Calculations for the minimization proceed as follows:

1) Evaluate the objective function Q.

2) Calculate the gradient vector, $\boldsymbol{g}$, and Hessian matrix, $\boldsymbol{H}$, of Q with respect to the phase fractions vector, β. If any of the phase fractions are equal to zero, set the respective gradient and Hessian values to zero as well. The diagonal of the Hessian matrix should be unity.

3) Using Newton's method, calculate $\Delta\beta$: $-\Delta\beta = \boldsymbol{H}^{-1}\boldsymbol{g}$. Let the initial Newton step size, α, be equal to 1. The Newton step size used here is unrelated to the alpha function described in chapter 1.

4) Calculate a new phase fractions vector $\beta^* = \beta + \alpha\Delta\beta$. Proceed with step 5 if all elements of the phase fractions vector are non-negative.
   - Choose $\alpha = \alpha_0 < 1$ such that exactly one of the previously negative elements of β^* becomes zero and all remaining values are positive.
   - Evaluate Q^* for β^*.
   - If $Q^* < Q$ then let $Q = Q^*$ and $\beta = \beta^*$. Return to step 2. Else let $\alpha = \alpha_0/2$ and repeat step 4.

5) Evaluate Q^* for β^*.

6) If $Q^* < Q$ then let $Q = Q^*$ and $\beta = \beta^*$ and proceed to step 7. Else let $\alpha = \alpha/2$ and repeat step 4.

7) If $\Delta\beta$ is sufficiently small then proceed with step 8. Else return to step 2.

8) Examine all of the elements of the gradient vector $\boldsymbol{g}$ that correspond to phase fractions equal to zero. If all of these gradient values are positive proceed to step 9. Else identify the gradient element with the largest negative value and increase its corresponding phase fraction from zero to a small positive value. Return to step 2.

9) Calculate the mole fractions in all phases, including those phases with a phase fraction equal to zero. Determine the fugacity coefficients corresponding to the new mole fractions.

This minimization procedure is repeated until the fugacity coefficients calculated in step 9 remain effectively unchanged over several iterations.
This procedure is essentially insensitive to the initial estimates for the phase fraction vector; any physically realistic set of values is sufficient. The simplest selection of phase fraction vector is equally distributing molar quantities between all phases. Unfortunately, as indicated in section 2.3, the converged solution to this procedure is enormously sensitive to the initial estimates for the dependent variables; here, the mole fractions. In practice, the mole fractions themselves are not estimated. Instead, initial values for the fugacity coefficient in each phase are used.

The execution of a flash calculation assumes that at least two phases will result. The first phase can be initialized consistently as a vapor-like phase using the ideal gas approximation; that is, all fugacity coefficients in the first phase should initially set to unity. The second phase is typically a liquid-like phase. Estimates for the fugacity components in this phase are routinely provided from the Wilson *K*-factor approximation, given in equation 2.9.

$$\ln\left(\hat{\phi}_j\right) = \ln\left(\frac{P_{c_j}}{P}\right) + 5.373\left(1+\omega_j\right)\left(1-\frac{T_{c_j}}{T}\right) \tag{2.9}$$

This approximation is a correlation of pure component phase behavior based on the critical temperature, critical pressure, and acentric factor. The estimates are reasonable initial values for multicomponent flash calculations. In absence of suitable alternatives, this expression is also extrapolated into the supercritical regime as necessary.

Estimates of additional phases require some insight into the behavior of the system under examination and are usually based on the estimates of the fugacity coefficients from the second phase. For example, a third phase is suspected (e.g., a second liquid-like phase) and physical intuition indicates that the third phase is enriched in component *j*.

In this situation the third phase fugacity coefficients can be estimated as equal to second phase coefficients for all components except component *j*. For component *j*, the third phase fugacity coefficient is specified as a larger value than what is approximated by equation 2.9. To make this separation more pronounced, the fugacity coefficient of component *j* in the second phase can be reduced by an equal amount.

If the converged calculations revert to a two phase solution despite being provided a three phase initial guess, then either the initial guess was insufficient for converging to the three phase solution, or the physical intuition was incorrect.

Unfortunately, there is no way to distinguish conclusively between the two possibilities. If repeated attempts to converge an additional phase are unsuccessful, it becomes necessary to reconsider whether that phase is present in the system.

There are no general heuristics available to initialize dependent variables for solid phases.

2.5 – Phase Envelope Calculations

A more difficult equilibrium problem is to identify the entire phase envelope for a mixture. Essentially, this relaxes the specification of the independent variables from the isothermal flash problem and requires an examination of all relevant temperatures, pressures, and compositions. For mixtures containing even a moderate number of components, this problem is very challenging because of the number of independent degrees of freedom; $n + 2$.

A subset of this problem can be addressed without resorting to exhaustive repetitions of the isothermal flash algorithm. Phase envelopes can be constructed in a step-wise fashion; building the full envelope from one-dimensional cross sections. Only the intensive coordinates are of interest in phase envelope calculations; the molar extents of each phase can be selected arbitrarily. Therefore, the complete phase envelope has $n + 2 - \pi$ dimensions. The complete phase envelope corresponds to all regions where $\pi > 1$. Fortunately, regions where $\pi = 3$ are located at the intersection of regions where $\pi = 2$, regions where $\pi = 4$ are located at the intersection of regions where $\pi = 3$, and so on. A characterization of all $\pi = 2$ regions is sufficient to reconstruct the complete envelope.

Only $\pi = 2$ equilibria are of interest, so there are only n relevant intensive variables. Because only one-dimensional cross sections are tractable, $n - 1$ intensive variables must be specified during calculations. For two component systems, typically the temperature or pressure is fixed. This specification results in *Pxy* and *Txy* binary phase diagrams, respectively.

The variable y is typically reserved for mole fractions in a vapor-like phase. Multiple iterations of the one-dimensional calculation at different fixed temperature or pressure can reconstruct the complete envelope. For systems with three components, both the temperature and pressure must be fixed. The complete envelope for three components requires iteration over both fixed variables. For more than three components, the mole fractions of the additional components need to be fixed. Three component phase envelopes are the highest dimensional phase envelopes that will be fully characterized.

The set of constraining relations for a two component, two phase equilibrium is presented in equation 2.10, and the set of constraining relations for a three component, two phase equilibrium is presented in equation 2.11.

Both of these relations define a line of phase envelope solutions (i.e., one dimension cross section of the complete envelope). Once again, the fugacity is used in place of the chemical potential, and Newton's method is used to determine a solution to the system of equations. These systems tend to be well behaved when using Newton's method; the instability of solutions on the interior of the phase envelope helps lead to rapid convergence.

$$F = 0 = \begin{matrix} \ln(K_1) + \ln(\hat{\phi}_{11}) - \ln(\hat{\phi}_{21}) \\ \ln(K_2) + \ln(\hat{\phi}_{12}) - \ln(\hat{\phi}_{22}) \\ x_{11} - x_{21}K_1 \\ x_{12} - x_{22}K_2 \\ 1 - x_{11} - x_{12} \\ 1 - x_{21} - x_{22} \\ \chi - \chi_{spec} \end{matrix} \tag{2.10}$$

$$F = 0 = \begin{matrix} \ln(K_1) + \ln(\hat{\phi}_{11}) - \ln(\hat{\phi}_{21}) \\ \ln(K_2) + \ln(\hat{\phi}_{12}) - \ln(\hat{\phi}_{22}) \\ \ln(K_3) + \ln(\hat{\phi}_{13}) - \ln(\hat{\phi}_{23}) \\ x_{11} - x_{21}K_1 \\ x_{12} - x_{22}K_2 \\ x_{13} - x_{23}K_3 \\ 1 - x_{11} - x_{12} - x_{13} \\ 1 - x_{21} - x_{22} - x_{23} \\ \chi - \chi_{spec} \end{matrix} \tag{2.11}$$

The vector of seven variables used in equation 2.10 involves x_{11}, x_{12}, x_{21}, x_{22}, K_1, K_2, and either T or P as desired. The variables K_1 and K_2 are the partition coefficients of the two components between the two phases.

The vector of nine variables used in the relations presented in equation 2.11 involves x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, K_1, K_2, and K_3. The variables K_1, K_2, and K_3 are the partition coefficients of the three components between the two phases.

In both equations 2.10 and 2.11, an entire line of solutions is described. The variable χ represents any one of the seven (as in equation 2.10) or nine (as in equation 2.11) variables desired in order to specify a location along that line of solutions.

The value of whichever variable is selected as χ is then changed slowly in order to determine a complete description of the one-dimensional phase envelope cross section. Performing a sensitivity analysis at each point along the line of solutions is very useful because it allows for the selection of the most rapidly changing variable as χ. The variable selected as χ in this manner often varies along the line of solutions.

Selecting the most sensitive variable as the control variable and moving slowly along the line of solutions is important because once again, as in section 2.4, the π - 1 solution is always a potential trivial solution to the system of equations. Here, the trivial solution is a one phase, uniform state. If the system of equations converges to the trivial solution, the calculation should be restarted at the last previous two phase solution and the specification variable χ should be changed more slowly.

Phase envelope calculations tend to be more computationally expensive than flash calculations, but they are also less problematic overall. As long as at least one two phase solution is known, the progress along the phase envelope cross section can be arbitrarily slow in order to maintain that two phase solution. Most phase envelope calculations are started from converged pure component solutions for precisely this reason. Two or three component, two phase solutions must pass through all of their constituent pure component, two phase solutions. All of the variables are both defined and finite in these states as well; fugacity coefficients and partition coefficients both have finite, non-zero values at infinite dilution.

Pure component, two phase solutions can be calculated from the set of constraints in equation 2.12.

$$F = 0 = \frac{\ln\left(\hat{\phi}_{11}\right) - \ln\left(\hat{\phi}_{21}\right)}{\chi - \chi_{spec}} \tag{2.12}$$

The vector of two variables used in the relations presented in equation 2.12 involves only T and P. All mole fractions and partition coefficients are unity and have therefore been dropped from the set of variables.

2.6 – Summary

This project has not contributed to the development or refinement of equilibrium calculations. All methodologies presented in this chapter were developed by other investigators and implemented as needed in support of creating and improved fluid models.

One of the challenges developing models is that the proposed model must accurately represent the phase behavior data. This representation involves not only accurately predicting phases that are experimentally known to exist, but also includes not predicting phase separation in regions that are known to be homogeneous. Failure to predict the existence of a phase can be either a deficiency in the model so that the phase is entirely absent, or an inaccuracy in the predicted composition of the phase such that the calculations simply do not converge as expected.

References

M Michelsen, J Mollerup; Thermodynamic Models: Fundamental & Computational Aspects. 2nd Ed. Denmark: Tie Line, 2007.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

Chapter 3 - Translation Methodology

Volume translation has been used to improve the estimation accuracy of EOS models for several decades. The translation approach is a useful methodology because it can be used to supplement existing relationships without replacing them entirely with novel EOS formulations. This chapter describes the method of volume translation as well as currently available methods. A generalized structure for developing new volume translation methods is also presented.

3.1 – Volume Translation Formalism

Volume translation was initially proposed by Joseph Martin in 1967 as a rigorous mathematical translation of the molar volume coordinate in a volumetric equation of state. Given a pressure explicit EOS, represented here as f_1, volume translation is achieved by altering the molar volume as described in equation 3.1.

$$\begin{aligned} P &= f_1(V_{UT}, T, x_1 \ldots x_n) \\ V &= V_{UT} - c \end{aligned} \tag{3.1}$$

The subscript UT is used to denote the untranslated molar volume. This value has no physical significance and is retained only for clarity. Only constant values of the parameter c were considered, so it was possible to completely eliminate V_{UT} as indicated in equation 3.2.

$$P = f_1(V + c, T, x_1 \ldots x_n) \tag{3.2}$$

By modifying a two parameter EOS to incorporate a third parameter, it was possible to satisfy all three experimentally determined critical coordinates: P_C, T_C, and V_C. Martin's translation as applied to the van der Waals EOS is given in equation 3.3.

$$P = \frac{RT}{V + c - b} - \frac{a}{(V + c)^2} \tag{3.3}$$

Parameter values for a and b remain unaffected by the translation and remain as described in Chapter 1. Given an experimentally determined value for the molar volume at the critical point, V_C, selecting a value of $3b$ - V_C for the parameter c will reproduce V_C exactly at the model predicted critical point. Unfortunately, selecting parameter values so that the van der Waals EOS reproduces all three critical coordinates exactly tends to reduce the accuracy of model estimates at other temperature and pressure conditions.

This result is not unique to the van der Waals model; applying a constant volume translation to a cubic equation of state so that the critical molar volume is reproduced accurately will underestimate the molar volume at all other pressure and temperature conditions. Martin recommended selecting a translation value so that the critical molar volume is overestimated and the translated EOS will provide a greater overall level of accuracy.

Equation 3.1 may be generalized so that other, non-constant, volume translations are possible. Equation 3.4 uses a general function, f_2, to represent the volume translation function.

$$\begin{aligned} P &= f_1(\underline{V}_{UT}, T, N_1 \ldots N_n) \\ \underline{V} &= \underline{V}_{UT} - f_2(\underline{V}_{UT}, T, N_1 \ldots N_n) \end{aligned} \tag{3.4}$$

Equation 3.4 simplifies to equation 3.1 when the volume translation function f_2 is set equal to *Nc*. Martin's volume translation followed the strict mathematical definition of molar volume translation; allowing the volume translation function to depend on the temperature or the untranslated molar volume alters extends the concept to include mathematically distorting translations of the volume coordinate as well. Using this formalism for volume translation, the system of two equations presented in equation 3.4 functions as the volumetric equation of state introduced in chapter 1. The function f_1 denotes the untranslated EOS and the function f_2 denotes the volume translation function.

Intensive formulations for an EOS, using mole fractions, x_j, instead of molar extents N_j have been used in equations 3.1 through 3.3. As discussed in chapter 1, the composition dependency of these models only appears implicitly through the selection of parameter values. This composition dependence of EOS parameters is detailed in chapter 5.

3.2 – Parameter Values Under Translation

The constant molar volume translation presented in section 3.1 did not affect the parameter values selected for *a* and *b* in the untranslated van der Waals equation. This consistency is a general result and is not limited to formulations independent of temperature and volume.

For a pure component, the EOS predicted vapor-liquid critical point occurs when the criteria presented in equation 1.36 are satisfied. These criteria are reproduced below as equation 3.5 with the added restriction that the parameters values are selected so that the criteria are satisfied for the EOS before a volume translation is applied.

$$\left(\frac{\partial P}{\partial V_{UT}}\right)_T = \left(\frac{\partial^2 P}{\partial V_{UT}^2}\right)_T = 0 \tag{3.5}$$

If the untranslated EOS satisfies these criteria at a specified pressure and temperature, then the translated EOS will as well. The relationship between $(\partial P/\partial V)_T$ and $(\partial P/\partial V_{UT})_T$ is given in equation 3.6.

$$\left(\frac{\partial P}{\partial V}\right)_T = \left(\frac{\partial P}{\partial V_{UT}}\right)_T \left(\frac{\partial V_{UT}}{\partial V}\right)_T = \left(\frac{\partial P}{\partial V_{UT}}\right)_T \left(1 - \left(\frac{\partial f_2}{\partial V_{UT}}\right)_T\right)^{-1} \tag{3.6}$$

If the untranslated EOS demonstrates a zero derivative at a point, then the translated EOS does as well, under the restriction that $(\partial f_2/\partial V_{UT})_T \neq 1$. This factor is repeated when the second derivative is examined, as shown in equation 3.7.

$$\left(\frac{\partial^2 P}{\partial V^2}\right)_T = \left(\frac{\partial^2 P}{\partial V_{UT}^2}\right)_T \left(\frac{\partial V_{UT}}{\partial V}\right)_T^2 = \left(\frac{\partial^2 P}{\partial V_{UT}^2}\right)_T \left(1 - \left(\frac{\partial f_2}{\partial V_{UT}}\right)_T\right)^{-2} \tag{3.7}$$

A translation function that is independent of the volume implies that $(\partial f_2/\partial V_{UT})_T = 0$, reducing the factor appearing in equations 3.6 and 3.7 to unity. The restriction that $(\partial f_2/\partial V_{UT})_T \neq 1$can be stated more generally as a strict inequality, presented in equation 3.8.

$$\left(\frac{\partial f_2}{\partial V_{UT}}\right)_T < 1 \quad \text{or} \quad \left(\frac{\partial f_2}{\partial \underline{V}_{UT}}\right)_{T,N} < 1 \tag{3.8}$$

In addition to ensuring that untranslated parameter values remain valid under translation, the restriction presented in equation 3.8 also has the added advantage of ensuring that additional volume roots are not introduced through the application of volume translation. Equations 3.6 and 3.7 may also be interpreted as ensuring that only those points with zero derivatives in the untranslated EOS will have zero derivatives in the translated EOS. By not introducing additional minima or maxima into the EOS model through translation, the translated model is guaranteed to have the same number of molar volume roots as the untranslated model.

3.3 – Integration of a Translated Equation of State

Chapter 1 introduced the quantity $\underline{A}^{res}$ as an important value to be calculated from a pressure explicit EOS. The method of calculating this quantity is restated in equation 3.9.

$$\underline{A}^{res} = RT\int_{\underline{V}}^{\infty}\left(\frac{P}{RT}-\frac{N}{\underline{V}}\right)d\underline{V} \tag{3.9}$$

Applying volume translation complicates this calculation because it may not be possible to work with the pressure as a function of the volume in a translated EOS formulation. Using equation 3.9 requires that the volume translation function of equation 3.4 be solved explicitly for the untranslated volume in terms of the volume. As demonstrated in equation 3.2, this reformulation is possible for constant translation values. However, translation functions that depend on the untranslated volume cannot generally be rearranged in this manner.

Working exclusively in the untranslated volume is an easier approach then trying to develop an expression for the untranslated volume in terms of the volume. The variable of integration in equation 3.9 can be transformed from the volume to the untranslated volume. This transformation results in equation 3.10, which is the preferred method of calculating the residual Helmholtz free energy from a translated EOS.

$$\begin{aligned}\underline{A}^{res} &= RT\int_{\underline{V}}^{\infty}\left(\frac{P}{RT}-\frac{N}{\underline{V}}\right)\left(\frac{\partial \underline{V}}{\partial \underline{V}_{UT}}\right)_{T,N}\left(\frac{\partial \underline{V}_{UT}}{\partial \underline{V}}\right)_{T,N}d\underline{V} \\ &= RT\int_{\underline{V}_{UT}}^{\infty}\left(\frac{f_1}{RT}-\frac{N}{\underline{V}_{UT}-f_2}\right)\left(1-\left(\frac{\partial f_2}{\partial \underline{V}_{UT}}\right)_{T,N}\right)d\underline{V}_{UT}\end{aligned} \tag{3.10}$$

This transformation makes $\underline{A}^{res}$ a function of the untranslated volume. When the pressure is specified, the untranslated EOS can be used to determine the untranslated volume, which can then be used directly in equation 3.10. Situations where the volume is specified are uncommon, although the volume translation function can be used in order to calculate a value for the untranslated volume for equation 3.10.

An important consequence of the transformation used to derive equation 3.10 is that the restriction presented in equation 3.8 arises naturally from the change in variables. This change of variables is only possible when equation 3.8 is satisfied. If for any value of the untranslated volume $(\partial f_2/\partial \underline{V}_{UT})_{T,N} = 1$, the integral in equation 3.10 becomes undefined. The fugacity or any other property that is calculated from $\underline{A}^{res}$ also becomes undefined. Any volume translation function that does not satisfy equation 3.8 cannot be used to calculate properties other than those explicitly appearing in the equation: the total volume, pressure, temperature, and molar extents. In effect, the volumetric equation of state loses its connection to the Fundamental Equation and becomes simply a correlation of molar volumes.

An analogous transformation could be attempted for $\underline{G}^{res}$. Equation 3.4 describes both pressure and volume as functions of temperature and the untranslated volume.

Working with volume or temperature as an independent variable is equally as complicated numerically. Equation 3.11 provides this alternative variable substitution.

$$\begin{aligned} \underline{G}^{res} &= RT\int_{\underline{0}}^{P}\left(\frac{\underline{V}}{RT}-\frac{N}{P}\right)dP \\ &= RT\int_{\underline{0}}^{P}\left(\frac{\underline{V}_{UT}-f_2}{RT}-\frac{N}{f_1}\right)\left(\frac{\partial P}{\partial \underline{V}_{UT}}\right)_{T,N} d\underline{V}_{UT} \end{aligned} \tag{3.11}$$

Proceeding from equation 3.11 is not possible because the quantity $(\partial P/\partial \underline{V}_{UT})_{T,N}$ must be non-zero at all conditions. As discussed in section 3.2, one of the main features of van der Waals type EOSs is their ability to predict either one or three molar volume roots, which requires the transformation in equation 3.11 to be equal to zero.

3.4 – Impact of Translation on Fugacity

Using a translation function that depends on volume also has implications for determining equilibria. Calculating the fugacity the residual Helmholtz free energy requires a differentiation at constant volume. Determining $\underline{A}^{res}$ from the untranslated volume requires additional terms to be incorporated into the expression for fugacity so that it can be calculated from the untranslated volume as well. The equation for differentiating $\underline{A}^{res}$ using the untranslated volume as an independent variable is given in equation 3.12.

$$\left(\frac{\partial \underline{A}^{res}}{\partial N_j}\right)_{T,\underline{V},N_{\neq j}} = \left(\frac{\partial \underline{A}^{res}}{\partial N_j}\right)_{T,\underline{V}_{UT},N_{\neq j}} - \left(\frac{\partial \underline{A}^{res}}{\partial \underline{V}_{UT}}\right)_{T,N}\left(\frac{\partial \underline{V}_{UT}}{\partial \underline{V}}\right)_{T,N}\left(\frac{\partial \underline{V}}{\partial N_j}\right)_{T,\underline{V}_{UT},N_{\neq j}} \tag{3.12}$$

Any volume translation will change the value of the fugacity, but changes to the fugacity may not necessarily affect the equifugacity conditions in chapter 2. This result is most easily seen when using $\underline{G}^{res}$ formulation for fugacity.

$$\ln\left(\frac{\hat{f}_j N}{P N_j}\right) = \int_{0}^{P}\left(\frac{1}{RT}\left(\frac{\partial \underline{V}}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P}\right)dP \tag{3.13}$$

The volume translation function may be substituted into equation 3.13 so that only the untranslated volume appears explicitly. Equation 3.14 describes the change in fugacity that can attributed to volume translation by separating out the contributions from the volume translation function, f_2.

$$\ln\left(\frac{\hat{f}_j N}{P N_j}\right) = \int_0^P \left(\frac{1}{RT}\left(\frac{\partial(\underline{V}_{UT} - f_2)}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P}\right) dP$$
$$= \int_0^P \left(\frac{1}{RT}\left(\frac{\partial \underline{V}_{UT}}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P}\right) dP - \int_0^P \frac{1}{RT}\left(\frac{\partial f_2}{\partial N_j}\right)_{T,P,N_{\neq j}} dP \qquad (3.14)$$

The translated fugacity is shown to be equal to the untranslated fugacity multiplied by the exponential expression shown on the second line in equation 3.15

$$\hat{f}_j = P\frac{N_j}{N}\exp\left(\int_0^P \left(\frac{1}{RT}\left(\frac{\partial \underline{V}_{UT}}{\partial N_j}\right)_{T,P,N_{\neq j}} - \frac{1}{P}\right) dP \right) \exp\left(\int_0^P \frac{-1}{RT}\left(\frac{\partial f_2}{\partial N_j}\right)_{T,P,N_{\neq j}} dP \right)$$
$$\hat{f}_j = \hat{f}_{j,UT} \exp\left(\int_0^P \frac{-1}{RT}\left(\frac{\partial f_2}{\partial N_j}\right)_{T,P,N_{\neq j}} dP \right) \qquad (3.15)$$

Phase equilibrium will not be affected by the translation when the fugacities in each phase are altered by the same amount; this situation occurs only when there is no volume dependence in the translation function and at most a linear dependence on composition. This result was first documented by Peneloux et al. in 1982. Providing a rubric for creating volume translation functions that can be applied without changing the EOS predicted phase equilibria popularized the volume translation approach as a method of improving EOS estimated molar volume accuracy.

3.5 – Available Translation Functions

Many models for the volume translation function are available. Martin and Peneloux et al. are credited for pioneering the approach; both employed translations constant in molar volume. Many other authors have employed constant translations with various methods of calculating or correlating the value of the translation. Values required for the translation will vary depending on the untranslated EOS paired with the translation function. Martin's translation was applied to the van der Waals EOS; the Peneloux translation was used with the SRK EOS.

Some translation functions may be specifically designed to complement a particular EOS, although that practice is not common. Most translation functions are intentionally robust enough so that when paired with appropriately calculated parameter values they may be used in conjunction with any van der Waals type EOS.

Table 3.1: Translation function formulations with their authors and years of publication.

Author	Year	Translation Function (f_2)
Constant		
Martin	1967	c_0
Peneloux, et al.	1982	c_0
Temperature Dependent		
Watson, et al.	1986	$c_0 + c_1 \exp\left(c_2\left(\left\|1-T_r\right\|\right)\right)$
Soreide	1989	$c_0 + \left(\left\|T_r - c_1\right\|\right)^{c_2} + c_3 \exp\left(c_4\left(T_r - 1\right)\right)$
Chou and Prausnitz	1989	$c_0 + c_1\left(\frac{c_2}{c_2+\delta}\right) \quad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_{T=T^{sat}}$
Magoulas and Tassios	1990	$c_0 + c_1 \exp\left(c_2\left(\left\|1-T_r\right\|\right)\right)$
Ungerer and Batut	1997	$c_0 + c_1 T$
Tsai and Chen	1998	$c_0\left(c_1 + c_2\left(1-T_r^{2/3}\right) + c_3\left(1-T_r^{2/3}\right)^2\right)$
Aspen	1999	$T_r \le 1: c_0 + \frac{c_1}{1+c_2-T_r}$ $T_r > 1: b + \frac{c_1\left(c_2\left(\frac{c_2}{c_1}(c_0-b)+1\right)\right)^2}{c_2^2\left(1+c_2\left(\frac{c_2}{c_1}(c_0-b)+1\right)-T_r\right)}$
Twu and Chan	2009	$\underline{V}_{UT}^{sat} - \underline{V}_{\exp}^{sat}$
Volume and Temperature Dependent		
Mathias, et al.	1989	$c_0 + c_1\left(\frac{0.41}{0.41+\delta}\right) \quad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T$
Kutney, et al.	1997	$c_0 + c_1\left(\frac{8(V_r)_{UT}T_r^{-4.5}}{(V_r)_{UT}^3 + 6.5T_r^{-6.5} + 0.5}\right)$
Laugier, et al.	2007	N/A

Table 3.1 lists of several forms of volume translation functions that have been proposed. This list identifies many of the major functional forms used for volume translation, but does not cover the various methods used to determine parameter values or any particular correspondence between proposed translation functions and equations of state.

Constant molar volume translation is the most common form of volume translation; it is a simple way of moderately improving molar volume accuracy without introducing additional complexity. Constant values are sufficient to ensure pure component molar volume estimates are accurate to within 5% at sub-critical pressures and temperatures such that $Tr < 0.8$.
Parameter values in a constant translation are usually selected so that the molar volume at ambient pressure and a reduced temperature of approximately $Tr = 0.7$ is reproduced accurately by the translated EOS.

Temperature dependence was incorporated into volume translation models in response to demand for improved accuracy and applicability to higher temperature and pressure applications. Table 3.1 is not an exhaustive list of temperature dependent formulations, but it does provide a representative sample. Many of these function use polynomial or power series terms involving the reduced temperature in order to provide flexibility. Parameter values are selected to reproduce molar volumes at whatever operating conditions are of interest. Targeting accuracy in molar volume predictions for saturated liquid volumes is the most common practice, as reflected by Twu and Chan's formulation. They recommend using Spencer and Danner's modified Rackett correlation to estimate the experimental saturated liquid molar volumes and deriving a closed form expression for the EOS saturated liquid molar volumes from whichever model is of interest.

Incorporating volume dependence into the translation function has not been a widely pursued avenue of investigation. The three functions presented in table 3.1 do represent an exhaustive list. The Mathias function was adapted by Chou and Prauznitz for wider applicability, although their modification removed the volume dependence for the sake of simplicity. Kutney's function provides excellent accuracy, but is designed specifically for a particular modified van der Waals EOS and is not appropriate for any other EOS expression. The function described by Laugier, et al. used a neural network formulation to determine the value of the volume translation function. No closed form explicit equation was provided.

3.6 – Further Translation Function Development

Using volume translation exclusively for the purpose of matching saturated liquid molar volumes is not an effective use of the volume translation methodology. Many investigators have incorporated temperature dependent function into EOS representations solely for the purpose of ensuring that the EOS reproduces those values accurately. These functions do not follow the general form of volume translation and are analogous to alpha function's representation of the saturation pressure. Chapter 2 addressed the method of calculating vapor-liquid equilibrium; for a single component system, the vapor-liquid equilibrium line can be completely described as a function of temperature.

The alpha function from chapter 1 is a temperature function with parameters regressed so that the pressure along the equilibrium line is accurately reproduced; a second function can accomplish the same thing for liquid molar volumes along the equilibrium line. A method for how or if this second correlation of pure component saturation properties should be introduced has not been established. However, it should not be introduced as a volume translation function that is only a function of temperature. The translation values required for the EOS to accurately estimate saturated molar volumes in the range $0.8 < Tr < 1.0$ are large compared to the translation values required at all other conditions.

The required molar volume translations for propane along several isotherms are shown in figure 3.1. These translation values are calculated as the difference between SRK estimated molar volumes and experimental molar volumes correlated by Miyamoto and Watanabe.

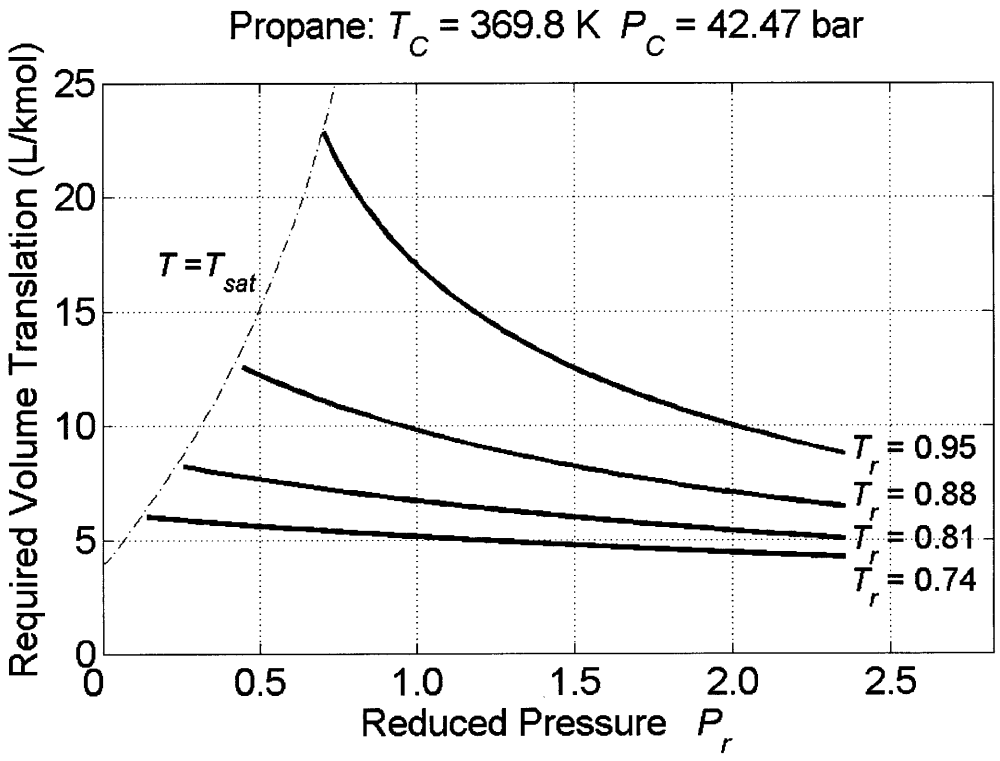

Figure 3.1: Required molar volume translations for propane along several isotherms for propane. Values are calculated as the difference between SRK estimated molar volumes and molar volumes from Miyamoto and Watanabe.

Along each isotherm the required volume translation value decreases with increasing pressure. This change corresponds to the compressibility of the fluid at the stated temperatures. At lower temperatures ($Tr < 0.7$) propane is well approximated as incompressible.

The required translation value can fall by more than half at temperatures near the critical temperature. Any function that was fit to data along the saturation line (the dashed line in figure 3.1) would cause a substantial decrease in EOS accuracy at elevated pressures. The results presented in Figure 3.1 are not unique to propane as a fluid or the SRK as an EOS model. All fluids exhibit the compressibility behavior shown as would be expected based on the principle of corresponding states. All van der Waals type cubic equations of state also demonstrate nearly identical patterns of molar volume residuals as well. Three different EOSs are contrasted in figure 3.2; the SRK presented in chapter 1, as well as the Peng-Robinson (PR) and Twu-Sim-Tassone (TST) EOSs.

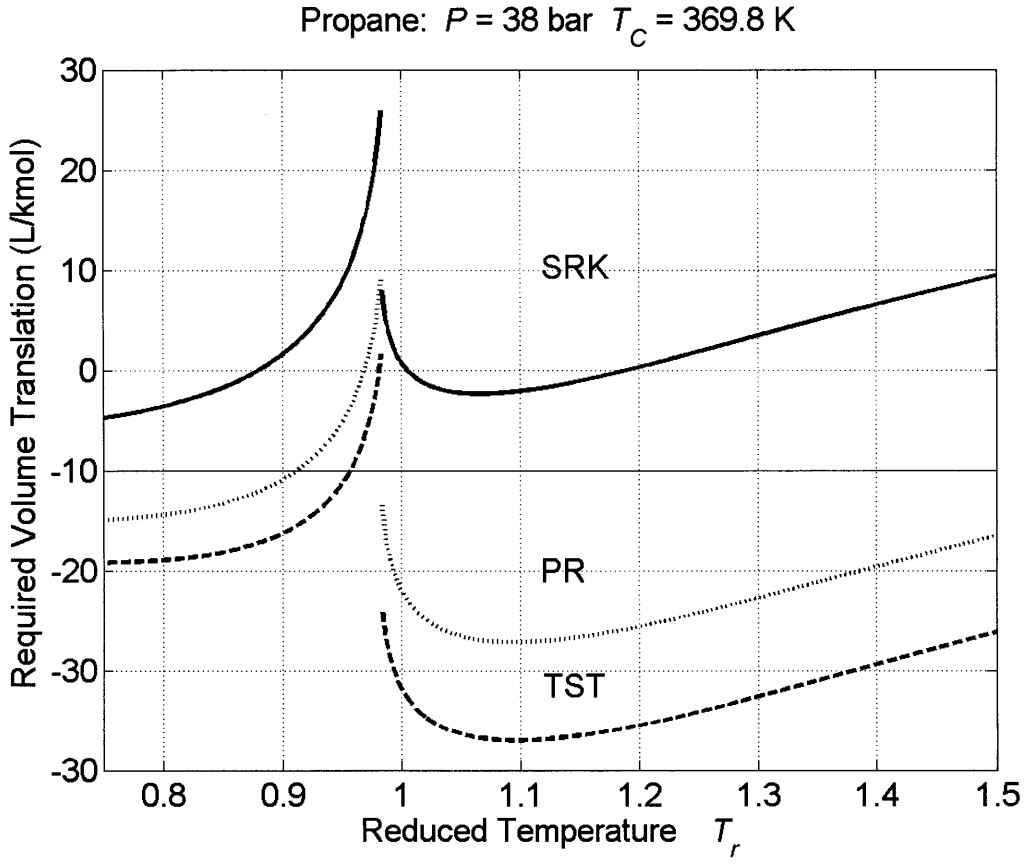

Figure 3.2: Required molar volume translations for propane as a function of temperature at a pressure of 38 bar. Values are calculated as the difference between SRK, PR, and TST estimated molar volumes and molar volumes from Miyamoto and Watanabe.

Only one pressure condition (P = 38 bar) is shown in figure 3.2, although results at all other pressure conditions are similar. Supercritical isobars would not demonstrate the discontinuity seen between the liquid (low temperature) and vapor (high temperature) states. All of the cubic models examined require roughly the same molar volume translation, to within a constant value.

The implication of this similarity is that a sufficiently robust volume translation function could be applied to any cubic equation of state by simply changing the value of the constant parameter in the volume translation function.
A dramatic example of the problems that can occur when using an inappropriate volume translation function was noted by Pfohl. He observed that using any translation function dependent on only temperature and with parameter selected to accurately reproduce saturated liquid molar volumes can cause a phenomenon called isotherm crossover. This problem is reproduced in figures 3.3a and 3.3b.

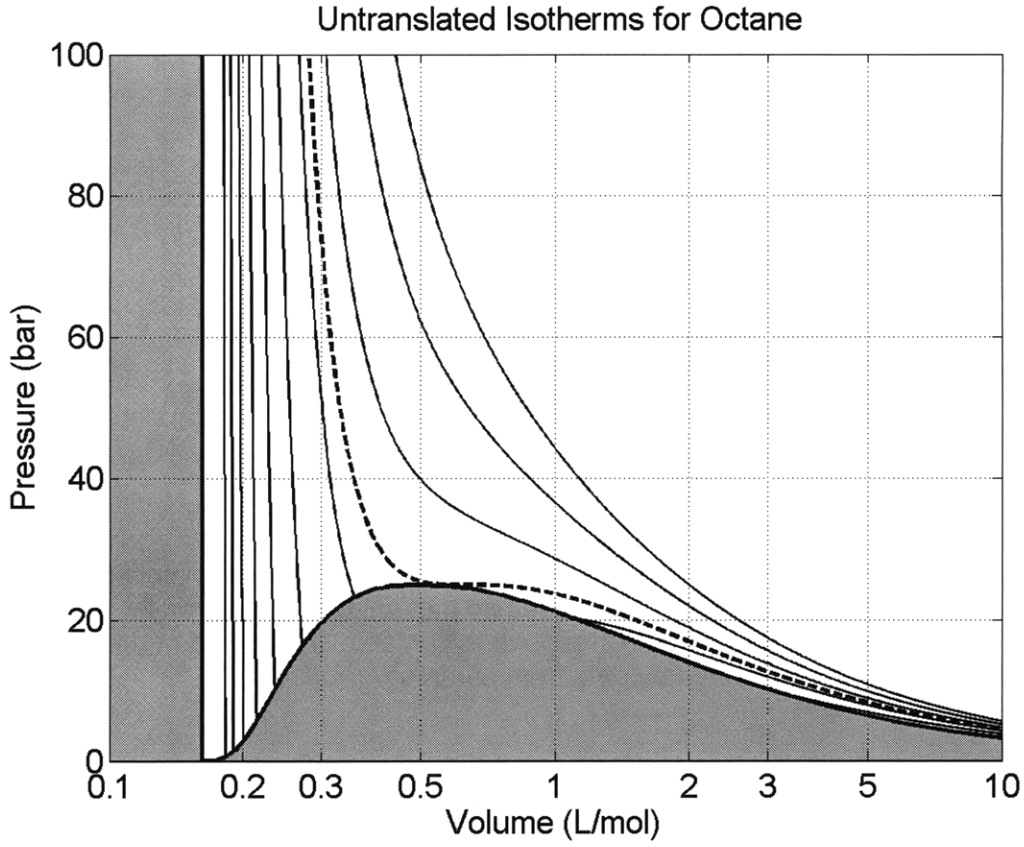

Figure 3.3a: Several isotherms for octane as modeled by the SRK EOS. The gray regions correspond to unstable or non-fluid regions of the pressure-volume diagram as determined by data from Lemmon.

Figure 3.3a depicts several sub- and supercritical isotherms for fluid octane as modeled by the SRK EOS. Regions not relevant to fluid octane on the pressure volume diagram have been grayed out. These regions correspond to unstable or solid phase conditions as indicated by data from a correlation by Lemmon. The isotherms in figure 3.3a do not necessarily accurately represent fluid octane; these are estimates provided by the SRK EOS. The volume axis is scaled logarithmically so that both the liquid and vapor phases may be included.

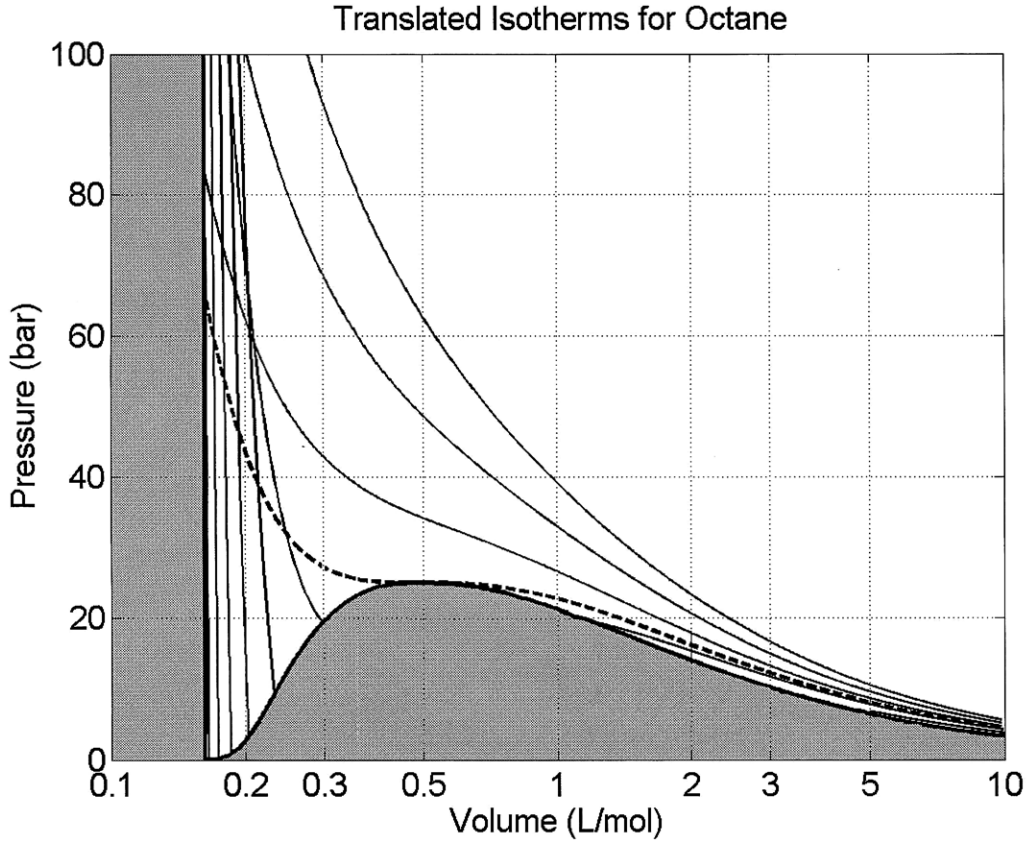

Figure 3.3b: Several isotherms for octane as modeled by the SRK EOS in conjunction with a temperature dependent translation function regressed so that saturated liquid molar volumes are accurately estimated. The gray regions correspond to unstable or non-fluid regions of the pressure-volume diagram as determined by data from Lemmon.

The isotherms presented in Figure 3.3b are calculated in the same manner as the isotherms in figure 3.3a and incorporate a temperature dependent volume translation function so that the saturated liquid molar volumes are estimated accurately. The isotherm crossover behavior noted by Pfohl occurs at liquid like molar volumes ($0.15 < V < 0.30$) and at elevated temperatures and pressures ($T > T_C$ and $P > 20$ bar). This region corresponds to the region noted in figure 3.1 as poorly modeled by volume translation functions that depend only on temperature.

The crossover behavior also effectively removes any correspondence between the volumetric equation of state model and its derivation from the Fundamental Equation. A translated EOS exhibiting isotherm crossover violates the thermodynamic construction that a state can be described by $n + 2$ independent variables, as well as exhibiting several types of non-physical behavior in the crossover region.

Volume translation methods should typically be restricted to constant valued approaches unless improved accuracy in the vicinity of the vapor liquid-critical point is necessary. Employing volume translation methods to model compressible fluids requires both temperature and volume dependence. Relying on only temperature dependence is insufficient and can severely limit the utility of the translated EOS model.

3.7 – Summary

Volume translation is a method that introduces a second relationship to supplement a pressure explicit equation of state. This second relationship describes the system volume as a function of the temperature, molar extents, and EOS estimated volume. Incorporating a volume translation function requires transforming several thermodynamic relationships so that the untranslated, EOS estimated volume may be used explicitly.

Many volume translation functions have been formulated, although a large majority of these are variations on a constant molar volume translation. Constant translation can be implemented with minimal computational cost and without changing the EOS predicted phase equilibria. However, constant translation formulations are insufficient to improve accuracy in highly compressible regions around the vapor-liquid critical point. Volume translation functions that incorporate only temperature can improve the estimates of saturated liquid molar volume properties, but are insufficient at higher pressures and cause numerical inconsistencies. Including a volume dependency in the translation function is necessary to create a robust translation that provides improved estimation accuracy in compressible regions.

References

Aspen Plus Reference Manual; Version 10. Cambridge, MA: Aspen Plus, 1999.

G Chou, J Prausnitz; *AIChE J.*, 35 (1989), 1487-1496.

K Frey, C Augustine, R Ciccolini, S Paap, M Modell, J Tester; *Fluid Phase Equilib.*, 260 (2007), 316-325.

B Jhaveri, G Youngren; *SPE Res. Eng.*, 3 (1988), 1033-1040.

M Kutney, V Dodd, K Smith, H Herzog, J Tester; *Fluid Phase Equilib.*, 128 (1997), 149-171.

S Laugier, F Rivollet, D Richon; *Fluid Phase Equilib.*, 259 (2007), 99-104.

E Lemmon, R Span; *J. Chem. Eng. Data.*, 51 (2006), 785-850.

K Magoulas, D Tassios; *Fluid Phase Equilib.*, 56 (1990), 119-140.

J Martin; *Ind. Eng. Chem.*, 59 (1967), 35-54.

J Martin; *Ind. Eng. Chem. Fundam.*, 18 (1979), 81-98.

P Mathias, T Naheiri, E Oh; *Fluid Phase Equilib.*, 47 (1989), 77-87.

M Michelsen, J Mollerup; Thermodynamic Models: Fundamental & Computational Aspects. 2nd Ed. Denmark: Tie Line, 2007.

H Miyamoto, K Watanabe, *Int. J. Thermophys.*, 21 (2000), 1045-1072.

A Peneloux, E Rauzy, R Freze; *Fluid Phase Equilib.*, 8 (1982), 7-23.

D Peng, D Robinson; *AIChE J.*, 23 (1977), 137-144.

O Pfohl; *Fluid Phase Equilib.*, 163 (1999), 157-159.

I Soreide, PhD Thesis, UNIT-NTH, Trondheim, Norway, 1989.

R Span; Multiparameter Equations of State. An Accurate Source of Thermodynamic Property Data. Berlin: Springer-Verlag, 2000.

C Spencer, R Danner; *J. Chem. Eng. Data.*, 17 (1972), 236-241.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

J Tsai, Y Chen; *Fluid Phase Equilib.*, 145 (1998), 193-215.

C Twu, H Chan; *Ind. Eng. Chem. Res.*, 48 (2009), 5901-5906.

C Twu, W Sim, V Tassone; *Chem. Eng. Prog.*, 98 (2002), 58-65.

P Ungerer, C Batut; *Rev. Inst. Fr. Pet.*, 52 (1997), 609-623.

P Watson, M Cascella, S Salerno, D Tassios; *Fluid Phase Equilib.*, 27 (1986), 35-52.

M Zabaloy, E Brignole; *Fluid Phase Equilib.*, 140 (1997), 87-95.

Chapter 4 - Novel Volume Translation: Design and Implementation

Volume translation must incorporate both temperature dependence and volume dependence in order to improve the accuracy of equation of state molar volume estimates at all process conditions. Constant molar volume translation is insufficient at temperatures and pressures near the vapor-liquid critical point; volume translation that depends only on temperature has the potential to introduce physical inconsistencies into the translated model. This chapter describes the development of a novel volume translation function and presents the results of its implementation to pure fluids.

4.1 – Proof of Concept

Chapter 1 discussed how the principle of corresponding states can be used to create an EOS model that is valid throughout the fluid region using only a few points of experimental data. Volume translation functions must also take advantage of the corresponding states principle so that translated EOSs are not prohibitively complex.

Implementing constant molar volume translation to modify an EOS so that the critical volume is accurately estimated reduces the accuracy of molar volume estimates at other temperature and pressure conditions. However, it is possible to construct a volume translation function that incorporates temperature and volume dependence so that these estimates are improved throughout the fluid regime. Cubic EOSs demonstrate a common pattern of molar volume residuals for many fluids. These residuals are calculated as the difference between the EOS predicted molar volume and accepted reference values. If a volume translation function were to represent these residuals exactly, then the translated EOS would reproduce the accepted reference molar volumes at all conditions. Using these molar volume residual values to create a volume translation value then enables representation of fluids for which comprehensive reference values are not available.

There is no guarantee that fluids without reference values behave in a similar manner. However, the set of fluids for which there is high quality reference data is very diverse; Argon, water, n-octane, and carbon dioxide all demonstrate the same behavior. Based on the principle of corresponding states, it is reasonable to expect that most other fluids will behave in the same manner. Figure 4.1 presents reduced molar volume residuals for several fluids; these residual values are calculated from the SRK EOS, although as discussed in chapter 3, all cubic equations of state will give similar results.

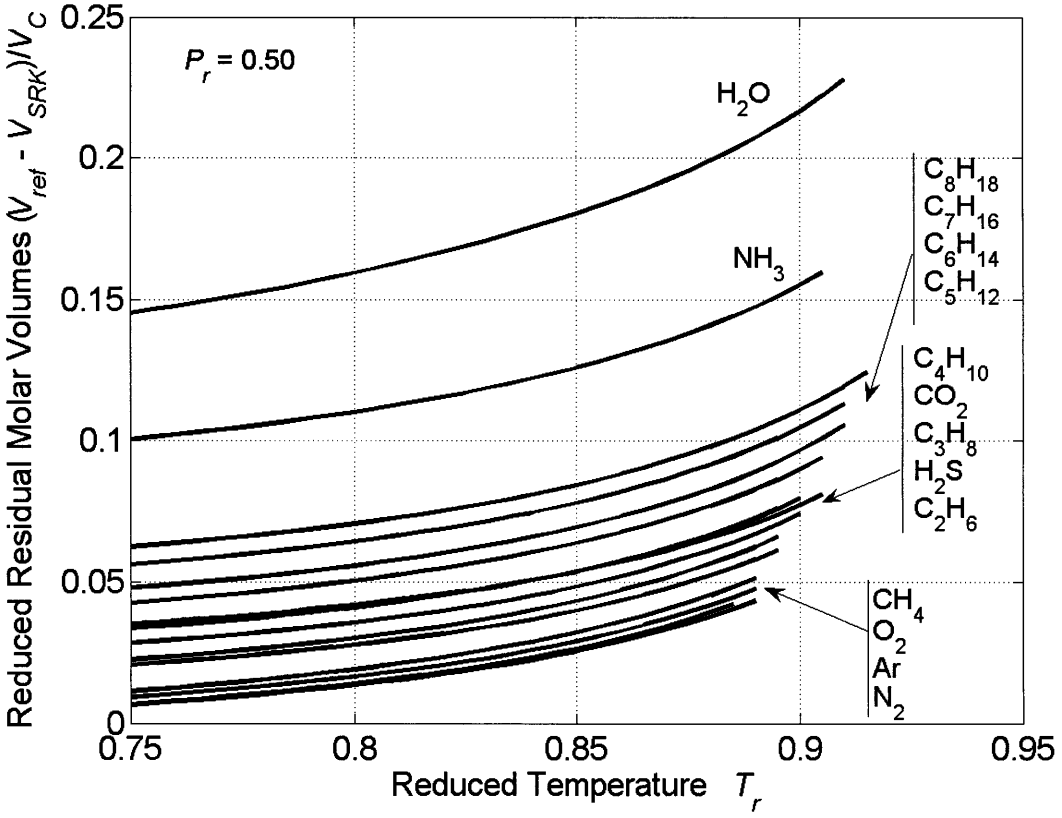

Figure 4.1: Reduced molar volume translation values for several fluids at a reduced pressure of $P_r = 0.5$. The required translation value is calculated as the difference between SRK estimated molar volume values and accepted reference values. Reference molar volumes at the critical point are used to when calculating V_r and not SRK estimated values.

A constant reduced pressure of $P_r = 0.5$ is maintained for all isobars presented in figure 4.1. Additionally, only the molar volume residuals for the liquid phase are shown. Molar volume residuals in the vapor phase are of similar absolute magnitude but are not significant. Reference values are valid are typically valid to within 1% of the true value; the relative error of EOS molar volume estimates in the vapor phase typically does not exceed 1-2%.

The pattern of residuals in figure 4.1 suggests that four reference coordinate are needed to formulate a molar volume translation: the critical pressure, critical temperature, critical molar volume, and a fourth coordinate that accounts for a substance dependent vertical shift. Molar volume residuals at a higher reduced pressure coordinate, $P_r = 0.75$, are presented in figure 4.2. The patterns shown are the same as those in figure 4.1.

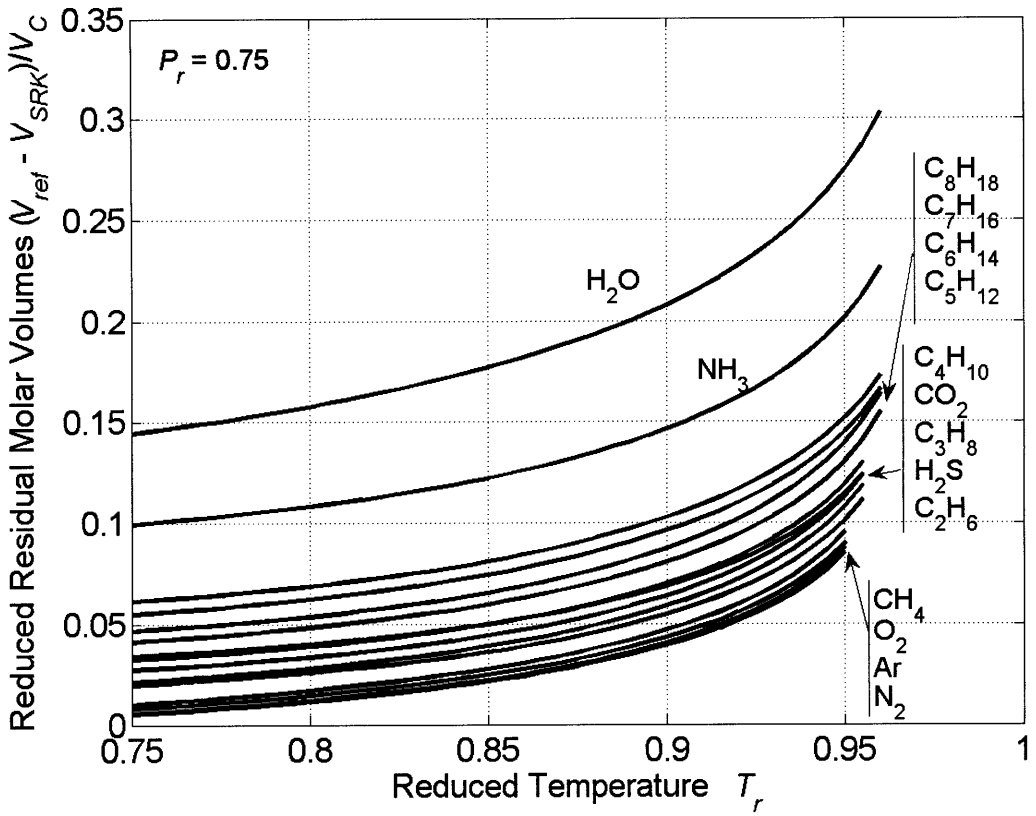

Figure 4.2: Reduced molar volume translation values for several fluids at a reduced pressure of $P_r = 0.75$. The required translation value is calculated as the difference between SRK estimated molar volume values and accepted reference values. Reference molar volumes at the critical point are used to when calculating V_r and not SRK estimated values.

Once again, only liquid molar volume residuals are shown because errors in the vapor phase are not significant. For reduced pressures greater than one (i.e., $P > P_C$ there is no discontinuity between the liquid and vapor phases. However, there is a distinct maximum along the line of molar volume residuals in the vicinity of the critical temperature. An example of this behavior was shown for Argon in figure 1.2. At temperatures above this maximum the molar volume grows sufficiently large that the relative errors in EOS estimated values are small, typically less than 3%; residual values at those temperatures do not need to be reproduced by the volume translation function.

A set of molar volume residuals at supercritical conditions are presented in figure 4.3 using a constant reduced pressure of $P_r = 2.50$. The isobars are truncated at the point of maximum relative residual molar volume error.

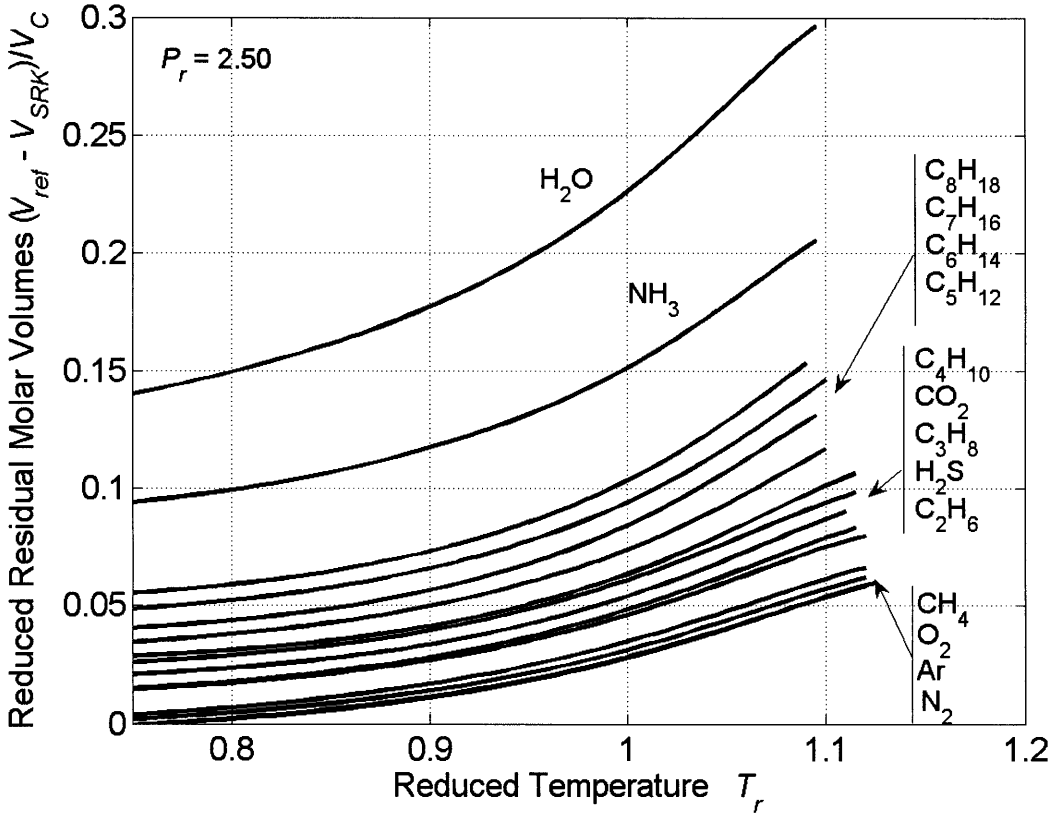

Figure 4.3: Reduced molar volume translation values for several fluids at a reduced pressure of $P_r = 2.5$. The required translation value is calculated as the difference between SRK estimated molar volume values and accepted reference values. Reference molar volumes at the critical point are used to when calculating V_r and not SRK estimated values. Isobars are truncated at the point where the reduced residual molar volume is a maximum.

Even at the high temperature and pressure depicted in figure 4.3 the pattern of molar volume residuals remains the same. Unfortunately, while it is clear from this analysis that a single volume translation function can be used to describe this pattern of residuals, no insight is provided for the functional form of this expression.

4.2 – Translation Function Design

Only three volume translation functions incorporating a dependence on the untranslated volume have been proposed; of these only two explicit expressions are provided. Mathias' expression was selected for further development. This expression is reproduced below as equation 4.1.

$$f_2 = c_0 + c_1\left(\frac{0.41}{0.41+\delta}\right) \qquad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T \tag{4.1}$$

Equation 4.1 was applied to the Peng-Robinson EOS, although a modified form of it was used by Prausnitz to improve molar volume predictions from the SRK EOS. These untranslated equations are effectively interchangeable for the purposes of volume translation design because the difference in residuals between EOSs can be accounted for entirely by the constant parameter c_0. A modified form of equation 4.1 is given below in equation 4.2. The SRK EOS has been used as the untranslated EOS model when evaluating the expression $(\partial P/\partial V_{UT})_T$, although the alpha function is omitted.

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\delta}\right) \qquad \delta = \frac{V_{UT}^2}{(V_{UT}-b)^2} - \frac{2aV_{UT}+ab}{RT(V_{UT}+b)^2} \tag{4.2}$$

Mathias described this method of volume translation as the distance parameter approach because δ, the bulk modulus, describes a dimensionless distance to the pure component vapor-liquid critical point. This quantity is zero at the critical point, as well as along the boundary of the unstable region.

4.3 – Critical Region Modifications

Equation 4.2 is able to improve the accuracy of molar volume estimates in the liquid phase, but it does not improve estimates at elevated conditions when both $P_r > 1$ and $T_r > 1$. Section 4.1 described how cubic EOSs exhibit a maximum in molar volume residual in the vicinity of the critical temperature. The maximum predicted by equation 4.2 does not coincide with the maximum calculated from EOS values. The difference between the predicted and observed maxima leads to poor accuracy in the translated molar volume estimates for temperatures that fall between these two points. Equation 4.2 was modified to include a correction in the distance parameter expression so that the translation function is more robust at supercritical conditions.

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\lambda}\right) \qquad \lambda = \delta + \frac{V_{UT} - \dfrac{3b}{4\left(\sqrt[3]{2}-1\right)}}{V_{UT}\delta + \dfrac{b}{\left(\sqrt[3]{2}-1\right)}} \qquad \delta = \frac{V_{UT}^2}{(V_{UT}-b)^2} - \frac{2aV_{UT}+ab}{RT(V_{UT}+b)^2} \tag{4.3}$$

Equation 4.3 describes a new distance parameter, λ, that is modified so that the translation function more accurately models the EOS molar volume residuals.

The untranslated SRK equation has only two adjustable parameters and incorrectly estimates the molar volume at the critical point in the same manner as the van der Waals EOS. The SRK EOS predicted critical volume, $RT_C/3P_C$ was used to modify the distance parameter in equation 4.3; that equation can be rewritten more generally as equation 4.4, which applies to an arbitrary cubic equation of state.

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\lambda}\right) \qquad \lambda = \delta + \frac{V_{UT} - \frac{3}{4}V_{C,EOS}}{V_{UT}\delta + V_{C,EOS}} \qquad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T \tag{4.4}$$

In equation 4.4, the EOS estimated critical volume, $V_{C,EOS}$, can be expressed in terms of the *a* and *b* parameters for any two constant cubic equation of state.

The modification to the distance parameter only contributes significantly near the boundary of the unstable region where when the bulk modulus, δ, goes to zero. Many different functional forms were evaluated; the expression presented in equations 4.3 and 4.4 provides the highest fidelity when reproducing EOS molar volume residuals.

4.4 – Consistency in the Unstable Region

Both the unaltered Mathias translation function, equation 4.1, and the distance parameter modified translation (DMT), equation 4.4, do not meet the thermodynamic consistency requirement presents in chapter 3 for translation functions (i.e., $(\partial f_2/\partial V_{UT})_T > 1$ for both translation functions). An additional modification is necessary so the functions can act as more than molar volume correlations.

The inconsistency arises because the bulk modulus becomes negative in the unstable region. The translation function has the potential to be undefined at points where δ, and correspondingly λ, becomes negative. An additional temperature function τ can be incorporated into expression for the bulk modulus so that it remains strictly positive at all temperature and volume conditions. This temperature expression is presented in equation 4.5.

$$\begin{aligned} \tau(T_r \le 1) &= \exp\left(5\left(1-\sqrt{T_r}\right)\right) - 1 \\ \tau(T_r > 1) &= 0 \end{aligned} \tag{4.5}$$

The expression for τ was developed for the DMT as applied to the SRK EOS. It is incorporated additively into the expression for the bulk modulus as indicated in equation 4.6.

$$\delta = \frac{V_{UT}^2}{(V_{UT}-b)^2} - \frac{2aV_{UT}+ab}{RT(V_{UT}+b)^2} + \tau \tag{4.6}$$

A different expression is required when the DMT is applied to EOSs other than the SRK. The value of the function in equation 4.5 is entirely empirical; it is determined by calculating the minimum value of the bulk modulus at each subcritical temperature. The expression for tau does not exactly reproduce these minima, instead it contributes a positive value that is greater than or equal to these minimum values. The bulk modulus is strictly positive at supercritical temperatures, so no empirical correction is necessary. Including an expression to ensure thermodynamic consistency in this manner does not change the ability of the translation function to reproduce the desired molar volume residual values.

4.5 –Saturation Pressure Estimates

The DMT volume translation function as applied to the SRK EOS is given below in equation 4.7.

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\lambda}\right) \qquad \lambda = \delta + \frac{V_{UT} - \dfrac{3b}{4\left(\sqrt[3]{2}-1\right)}}{V_{UT}\delta + \dfrac{b}{\left(\sqrt[3]{2}-1\right)}} \tag{4.7}$$

$$\delta = \frac{V_{UT}^2}{\left(V_{UT}-b\right)^2} - \frac{2aV_{UT}+ab}{RT\left(V_{UT}+b\right)^2} + \tau \qquad \begin{aligned} \tau\left(T_r \le 1\right) &= \exp\left(5\left(1-\sqrt{T_r}\right)\right) - 1 \\ \tau\left(T_r > 1\right) &= 0 \end{aligned}$$

The untranslated SRK EOS is reproduced below in equation 4.8.

$$f_1 = \frac{RT}{V-b} - \frac{a\cdot\alpha}{V\left(V+b\right)}$$

$$a = \frac{R^2T_C^2}{9\left(\sqrt[3]{2}-1\right)P_C} \qquad b = \frac{\left(\sqrt[3]{2}-1\right)RT_C}{3P_C} \tag{4.8}$$

A new alpha function is required for the untranslated SRK EOS because incorporating the volume translation function from equation 4.7 will affect estimates of the phase equilibrium. The alpha function selected is given in equation 4.9.

$$\alpha = \exp\left(\kappa_1\left(1-T_r^{\kappa_2}\right)\right) \tag{4.9}$$

Equation 4.9 has two parameters that are adjusted so that the translated EOS accurately reproduces a fluid's vapor pressure. The original SRK EOS used an alpha function that had one parameter, which was calculated from a quadratic expression in the acentric factor.

The vapor pressure of all reference fluids is reproduced accurately using the alpha function from equation 4.9 in conjunction with the DMT function applied to the SRK EOS. Maximum error occurs at the low temperature terminus of the vapor pressure curve and is between 3 and 4 percent. Average error along the entirety of the equilibrium line is less than 1 percent.

4.6 – Calculation of Fugacity

The two parameters in the alpha function presented in equation 4.9 are determined by calculating pure component phase equilibria as described in chapter 2. Appropriate expressions for the fugacity and fugacity coefficient were given in chapter 3, although integration of equations 4.7 and 4.8 is required. The integration can be performed numerically, but using an analytic expression for the fugacity coefficient reduces the computational requirement by more than an order of magnitude.

Modifications to the integral for A_{res} were described in chapter 3. Terms within this integral can be grouped together as indicated in equation 4.10.

$$\frac{\underline{A}^{res}}{RT} = \int_{\underline{V}}^{\infty}\left(\frac{P}{RT} - \frac{N}{\underline{V}}\right)d\underline{V} = \int_{\infty}^{\underline{V}}\left(\frac{N}{\underline{V}}\right)\partial\underline{V} + \int_{\underline{V}_{UT}}^{\infty}\left(\frac{f_1}{RT}\right)\partial\underline{V}_{UT} + \int_{\underline{V}_{UT}}^{\infty}\left(\frac{f_1}{RT}\right)\left(\frac{\partial f_2}{\partial\underline{V}_{UT}}\right)\partial\underline{V}_{UT} \tag{4.10}$$

This grouping of terms allows for the separation of contributions from the untranslated EOS and the volume translation function, as in equation 4.11.

$$\frac{\underline{A}^{res}}{RT} = \frac{\underline{A}^{res}_{UT}}{RT} - N\ln\left(\frac{\underline{V}_{UT} - f_2}{\underline{V}_{UT}}\right) + \int_{\underline{V}_{UT}}^{\infty}\left(\frac{f_1}{RT}\right)\left(\frac{\partial f_2}{\partial\underline{V}_{UT}}\right)\partial\underline{V}_{UT} \tag{4.11}$$

The last term on the right hand side of equation 4.11 describes the integration of a complicated expression that is derived from equations 4.7 and 4.8. A complete description of this expression is included in the Appendix; an abstraction is presented in equation 4.12.

$$\left(\frac{f_1}{RT}\right)\left(\frac{\partial f_2}{\partial\underline{V}_{UT}}\right) = \frac{p_{16}(\underline{V}_{UT})}{q_{19}(\underline{V}_{UT})} \tag{4.12}$$

The functions p_{16} and q_{19} respectively denote a sixteenth and nineteenth order polynomial in the untranslated volume. These functions also depend on temperature and composition, but for the purposes of integration, only the dependency on the untranslated volume is relevant. The existence of a closed form expression for this integral is guaranteed by means of a partial fraction expansion.

Volume translation functions that alter phase equilibrium properties and do not have closed form expressions for fugacity have severely limited utility. Rapid calculation of the fugacity is required in a majority of EOS applications. Generally, translation functions that involve exponential terms involving the untranslated volume cannot be analytically integrated. The DMT formulation, although analytic, still imposes a significant increase in computational cost because equation 4.12 includes an additional eighteen terms in the expression for A_{res}. The untranslated expression involves only two terms.

4.7 – Parameter Estimation for the Translation Function

Section 4.6 describes how the two kappa parameters in the alpha function are calculated. All other parameters in the EOS must be determined prior to evaluating the alpha function parameters because the values of a, b, c_0, and c_1 affect the estimated phase equilibrium.

The a and b parameters remain as specified in equation 4.8. One of the DMT parameters is determined by providing the experimentally determined molar volume at the critical point. At temperature T_C and pressure P_C, the untranslated SRK EOS evaluates to $V_{C,EOS}$. At these conditions, the DMT function evaluates to the expression in equation 4.13.

$$f_2\left(T_C, V_{C,EOS}\right) = c_0 + \frac{2}{3}c_1 \tag{4.13}$$

The translated EOS will exactly reproduce the experimental critical molar volume V_C when c_0 is calculated as described in equation 4.14.

$$c_0 = \frac{RT_C}{3P_C} - V_C - \frac{2}{3}c_1 \tag{4.14}$$

Simultaneously fitting both parameters in the translation function is cumbersome; the experimental critical molar volumes are at least four times larger than optimal values for the parameter c_1. The contribution of parameter c_1 in equation 4.14 has been neglected without loss of molar volume accuracy.
The first volume translation parameter is calculated using only the critical coordinates as in equation 4.15.

$$c_0 = \frac{RT_C}{3P_C} - V_C \tag{4.15}$$

All three critical coordinates are specified when applying the DMT to the SRK EOS.

However, only the critical pressure and critical temperature are reproduced exactly by the translated EOS. The EOS estimated critical volume is variable (i.e., it is not fixed in ratio with respect to the other critical parameters), but it is also not exactly equal to the value specified as the experimental critical volume.

Parameter c_1 was determined by minimizing the mean square error between the untranslated EOS molar volume residuals and the DMT expression from equation 4.7 when calculating a and b as described in equation 4.8 and c_0 as described in equation 4.15.

Table 4.1 lists the pure component parameters used in the translated EOS. All of the fluids listed in the table have detailed reference molar volume correlations available in the literature as well as precise relationships describing the saturation pressure.

Table 4.1: Pure component parameters used when applying the DMT to the SRK EOS. All fluids listed have detailed reference molar volume values available in the literature.

Fluid	T_C (K)	P_C (bar)	V_C (L/kmol)	c_1 (L/kmol)	κ_1	κ_2
CH_4	190.56	45.99	99	16.5	0.5091	1.1840
C_2H_6	305.33	48.72	146	26.5	0.7481	0.9560
C_3H_8	369.83	42.47	202	36.5	0.8980	0.8628
n-C_4H_{10}	425.13	37.96	255	50.0	0.8890	0.9915
n-C_5H_{12}	469.70	33.70	311	65.5	0.9798	0.9761
n-C_6H_{14}	507.82	30.34	370	80.5	0.9988	0.9761
n-C_7H_{16}	540.13	27.36	432	95.5	1.0890	1.0210
n-C_8H_{18}	569.32	24.97	486	120.	1.1070	1.0900
N_2	126.19	33.96	89	14.0	0.5883	1.0730
O_2	154.57	50.43	73	11.5	0.5921	1.0040
Ar	150.69	48.63	75	12.0	0.5481	1.0200
H_2S	373.10	90.00	98	16.0	0.6955	1.0580
CO_2	304.13	73.77	94	18.0	0.9391	0.9895
CF_4	227.51	37.50	141	27.5	0.7996	1.0930
H_2	33.15	12.96	71	0	0	0
NH_3	405.40	113.3	76	17.0	0.8086	1.2670
H_2O	647.10	220.6	56	19.0	0.8433	1.4760

Selecting a parameter value of zero for c_1 implies only a constant molar volume translation of c_0. Zero for both parameters recovers the untranslated EOS, albeit with the modified alpha function from equation 4.9.

Table 4.1 indicates that the recommended value for c_1 is zero for hydrogen. The value for both of the alpha function parameters should be zero when modeling hydrogen, implying that no correction for the saturation pressure is included.

Both of these recommendations are based on the atypically low critical temperature for hydrogen. The DMT formulation from equation 4.7 is still applicable at high reduced temperatures ($T_r > 3$), but it does not provide better accuracy than a constant molar volume translation. Regarding the alpha function, the saturation pressure of pure hydrogen is not a physically relevant quantity. At subcritical temperatures where there is a vapor-liquid boundary for hydrogen, pure hydrogen behaves as a binary mixture of two distinct quantum states.

Highly precise molar volume data is not available for all fluids of interest. For fluids not included in table 4.1, it is possible to estimate the value of c_1 by using the density anywhere in the liquid phase. Using the specific gravity for fluids that are liquid at ambient conditions is the simplest approach. Figure 4.4 provides a correlation for the value of c_1 in terms of the untranslated critical molar volume.

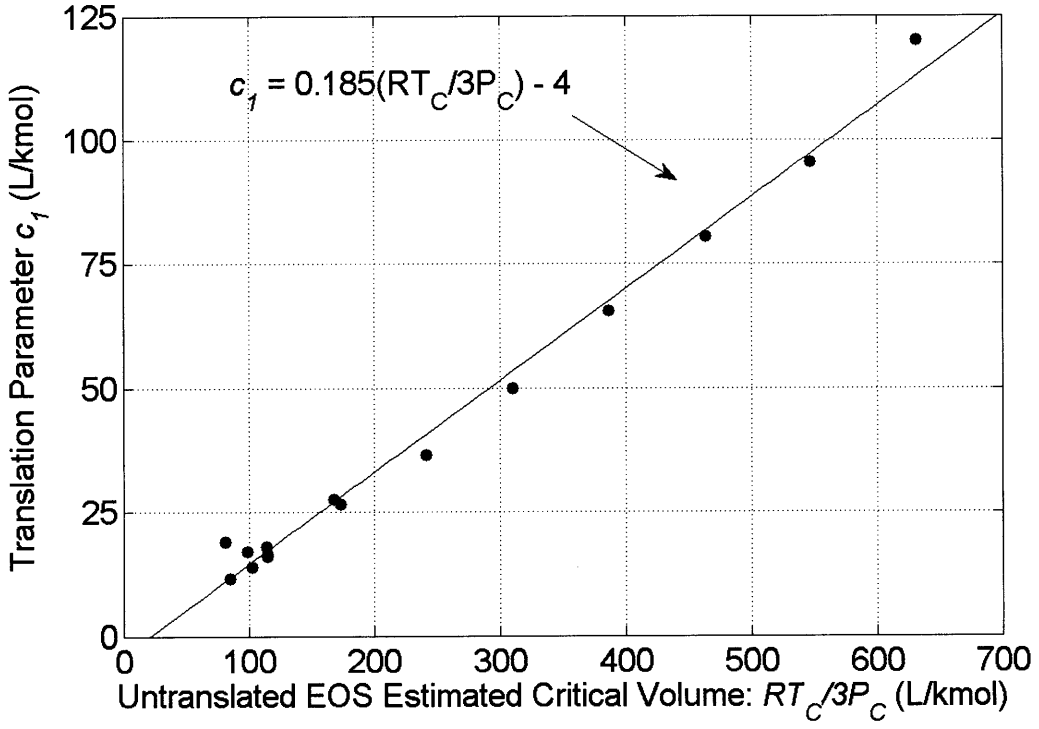

Figure 4.4: Values of the volume translation parameter c_1 in terms of the untranslated EOS estimated critical molar volume.

The linear correlation in figure 4.4 provides an initial estimate for the volume translation parameter c_1.

$$c_1[L/kmol] = 0.185\frac{RT_C}{3P_C} - 4 \tag{4.16}$$

Liquid phase data should be used to determine a more precise value. Polar fluids such as water have the highest deviations from correlation in equation 4.16.

4.8 – Accuracy of Molar Volume Estimation

The translated EOS provides highly accurate molar volume estimations throughout the fluid region. Figure 4.5 depicts the molar volume error along multiple isobars for propane, ranging from a reduced pressure of 0.5 (21 bar) to a reduced pressure of 5.0 (210 bar). Average error is less than 2% at all pressure conditions; maximum error occurs along the critical isobar and is less than 7%.

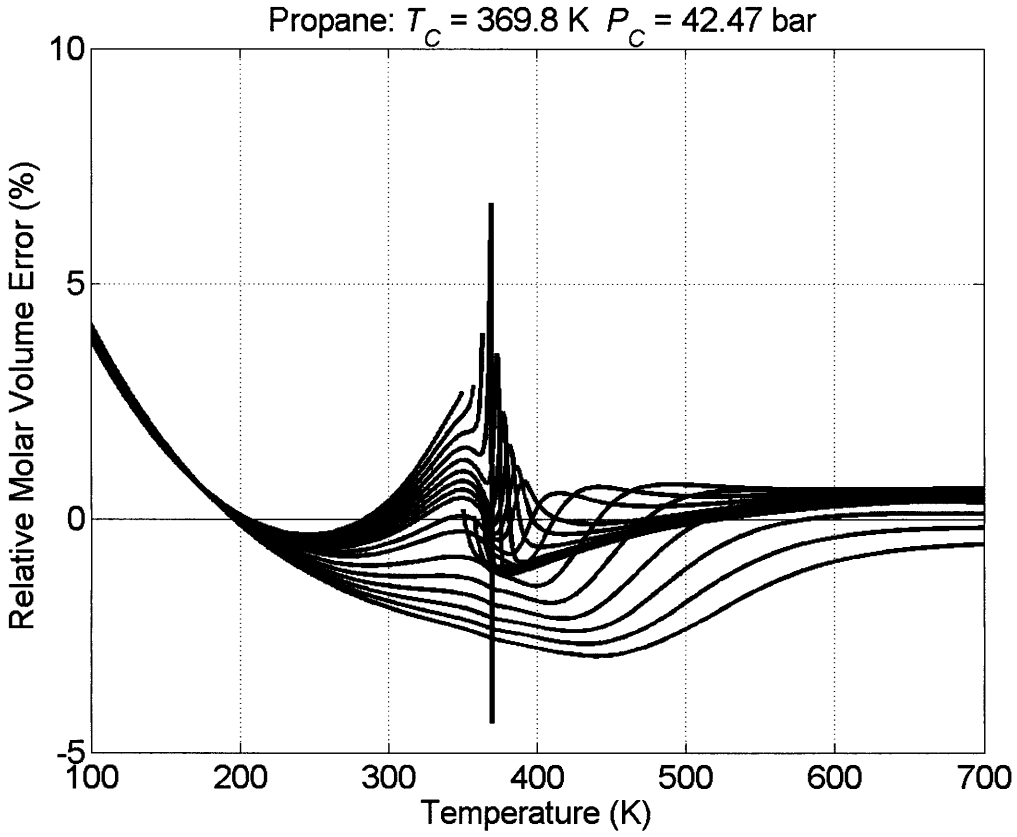

Figure 4.5: Error in molar volume estimates for propane from the DMT function applied to the SRK EOS. Isobars range from $Pr = 0.5$ to $Pr = 5.0$.

A set of isobars analogous to those in figure 4.5 is presented in figure 4.6. An optimal constant molar volume translation is used instead of the DMT function in order to determine the error in estimated molar volumes.

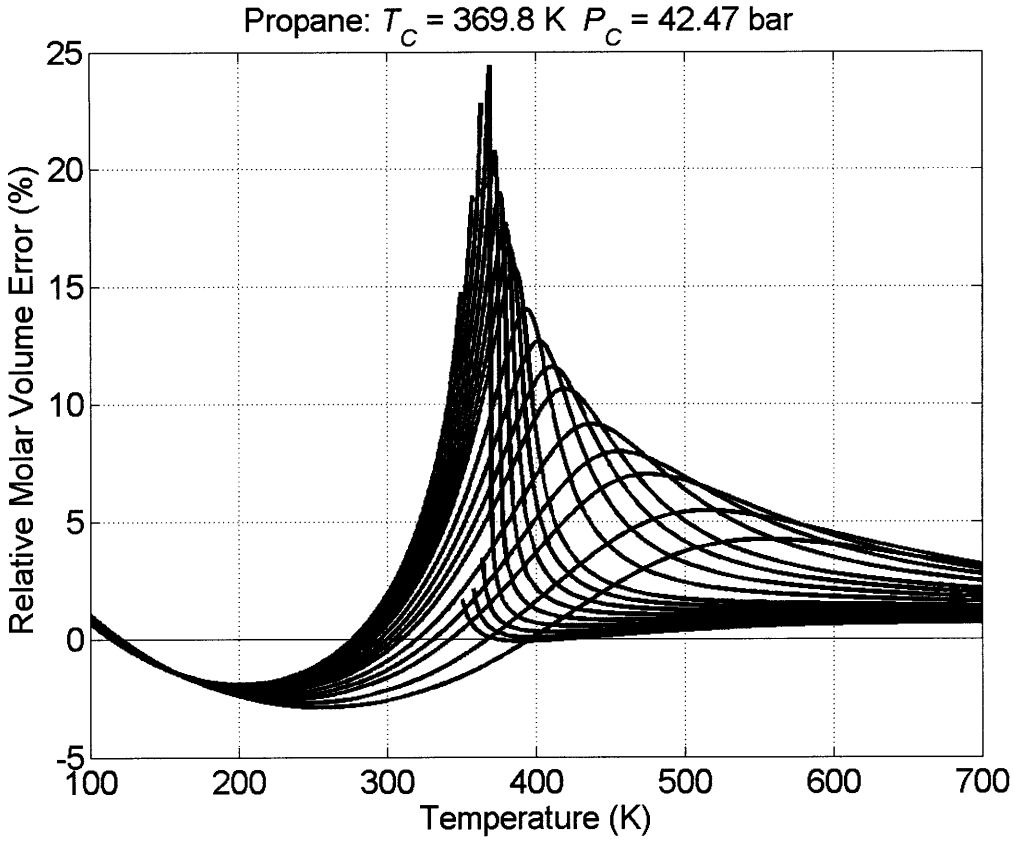

Figure 4.6: Error in molar volume estimates for propane from an optimal constant molar volume translation applied to the SRK EOS. Isobars range from $Pr = 0.5$ to $Pr = 5.0$.

The range of the vertical axis in figure 4.6 has doubled to accommodate the molar volume errors. The greatest increase in accuracy occurs around the critical temperature ($0.8 < T_r < 1.2$). At other conditions, the two models are roughly equivalent. In the vapor phase and at low temperatures in the liquid phase ($T_r < 0.7$), a constant molar volume translation provides the same level of accuracy as the modified translation function, but without the associated increase in complexity.

The density prediction accuracy of three different models is listed in table 4.2: the untranslated Soave-Redlich-Kwong EOS (SRK), the SRK EOS with an optimal constant molar volume translation (OT), and the SRK with the distance parameter modified translation (DMT). Only four fluids are included in table 4.2, although these fluids provide a good indication of the generality of the approach. The optimal constant translation value for the SRK EOS was selected to minimize the absolute average error reported for this comparison; other values may be appropriate for particular ranges of operating conditions. No translation functions that are exclusively temperature dependent were included for this comparison.

Table 4.2: Molar volume estimation accuracy provided by the Soave-Redlich-Kwong EOS (SRK), the SRK EOS with an optimal constant translation (OT) and the SRK EOS with the distance parameter modified translation (DMT).

Fluid	Pressure (bar)	Average Absolute Deviation (%)		
		SRK	OT	DMT
Methane (P_C = 46.0) OT Value = -0.3 L/kmol	20	0.63	0.51	0.66
	30	0.93	0.85	0.73
	40	1.45	1.38	0.80
	50	1.90	1.89	0.87
	60	2.33	2.33	0.89
	70	2.68	2.69	0.92
	80	2.96	2.97	0.95
Propane (P_C = 42.5) OT Value = 4.3 L/kmol	18	3.34	0.89	1.28
	27	3.62	1.14	1.24
	36	4.05	1.59	1.23
	45	4.56	2.09	1.23
	54	5.00	2.51	1.21
	63	5.38	2.85	1.23
	72	5.69	3.12	1.27
Carbon Dioxide (P_C = 73.8) OT Value = 3.6 L/kmol	30	2.03	0.83	0.41
	45	2.83	1.12	0.51
	60	3.67	1.64	0.60
	75	4.49	2.26	0.64
	90	5.29	2.87	0.60
	105	5.96	3.37	0.58
	120	6.51	3.78	0.60
Water (P_C = 221.) OT Value = 6.4 L/kmol	75	18.19	2.05	1.88
	125	18.01	2.85	2.22
	175	18.56	3.55	2.54
	225	19.41	4.14	2.83
	275	20.36	4.65	2.95
	325	21.35	5.35	2.93
	375	22.34	6.06	2.91

The temperature at each pressure condition indicated in table 4.2 ranges from the freezing temperature up to well above the critical temperature where the fluid begins to behave ideally (i.e., fluid behavior is considered ideal when the compressibility ratio asymptotes to unity; here $PV/RT > 0.95$). This choice of temperature range may introduce some bias into the results; the freezing point of carbon dioxide occurs at about $T_r = 0.7$, while the freezing point of propane occurs at about $T_r = 0.25$. The temperature interval is maintained at 1°C, which implies that the reported average error in estimate of propane molar volumes is weighted more heavily toward the liquid phase than the value for carbon dioxide. General trends remain the same regardless of the temperature interval selected.

The DMT approach is most useful for isobars near and above the critical pressure. Predictions at subcritical conditions where the fluids are mostly incompressible are also modeled accurately using the DMT function; however, constant molar volume translation provides a similar level of molar volume accuracy at these conditions without introducing as much complexity in the EOS model.

4.9 – Summary

A volume translation function dependent on both the untranslated molar volume and temperature was developed and applied to the Soave-Redlich-Kwong equation of state. This translation function is based on a distance parameter approach by Mathias; the approach was modified to improve representation in the critical region and consistency in the unstable region. Good accuracy in reproducing the vapor pressure was achieved by incorporating a two parameter alpha function.

The translation requires four parameters that can be determined from experimental critical point data, although the representation accuracy can be improved by incorporating a reference molar volume from the liquid phase as well. Estimation accuracy for molar volumes throughout the fluid phase is very good; the volume dependent translation provides the greatest advantage at temperatures near a fluid's critical temperature where constant molar volume translations are unable to provide reliable estimates.

References

K Frey, C Augustine, R Ciccolini, S Paap, M Modell, J Tester; *Fluid Phase Equilib.*, 260 (2007), 316-325.

K Frey, M Modell, J Tester; *Fluid Phase Equilib.*, 279 (2009), 56-63.

D Friend, H Ingham, J. Ely; *J. Phys. Chem. Ref. Data.*, 20 (1991), 275-347.

P Mathias, T Naheiri, E Oh; *Fluid Phase Equilib.*, 47 (1989), 77-87.

H Miyamoto, K Watanabe; *Int. J. Thermophys.*, 21 (2000), 1045-1072.

B Platzer, A Polt, G Maurer; Thermophysical properties of refrigerants, Springer-Verlag, Berlin, 1990.

B Poling, J Prausnitz, J O'Connell; The Properties of Gases and Liquids, fifth ed., McGraw-Hill, New York, 2002.

O Redlich, J Kwong; *Chem. Rev.*, 44 (1949), 233-244.

N Sakoda, M Uematsu; *Int. J. Thermophys.*, 25 (2004), 709-737.

U Setzmann, W Wagner; *J. Phys. Chem. Ref. Data.*, 20 (1991), 1061-1151.

G Soave; *Chem. Eng. Sci.*, 27 (1972), 1197-1203.

R Span, W Wagner; *Int. J. Thermophys.*, 24 (2003), 41-109.

R Span, W Wagner; *J. Phys. Chem. Ref. Data.*, 25 (1996), 1509-1596.

R Span, E Lemmon, R Jacobsen, W Wagner, A Yokozeki; *J. Phys. Chem. Ref. Data.*, 29 (2000), 1361-1433.

R Schmidt, W Wagner; *Fluid Phase Equilib.*, 19 (1985), 175-200.

R Tilner-Roth, F Harms-Watzenberg, H Baehr; *DKV Con. Rep.*, 20 (1993), 167-181.

J Valderrama; *Ind. Eng. Chem. Res.*, 42 (2003), 1603-1618.

W Wagner, A Pruss; *J. Phys. Chem. Ref. Data.*, 31 (2002), 387-535.

Chapter 5 - Implementation for Mixtures

Thermodynamic models take advantage of corresponding states methodology by formulating property relationships based on physical phenomena common to dissimilar substances, such as vapor-liquid critical points and normal boiling points. For pure substances, this approach provides good accuracy with minimal complexity. Extending this approach to mixtures while still maintaining a high degree of simplicity is challenging. This chapter describes how volumetric equation of state models developed to represent pure component properties are used to estimate mixture properties. Mixture property estimates obtained from the modified translation from chapter 4 are also presented.

5.1 – Mixture Critical Points

Volumetric equation of state parameters are determined using pure component vapor-liquid critical point data. The EOS pure component vapor-liquid critical point is determined using the stability criteria reproduced in equation 5.1.

$$\left(\frac{\partial P}{\partial V}\right)_T = \left(\frac{\partial^2 P}{\partial V^2}\right)_T = 0 \tag{5.1}$$

Equation 5.1 is a specific case of the general stability criteria for determining a critical point. As discussed in chapter 2, stable equilibrium states are defined as minima on a thermodynamic potential surface that have positive curvature. The vapor-liquid critical point is a point of bifurcation where two stable states become one state. Equations 5.2a and 5.2b are obtained when using the Gibbs free energy to formulate the generalized criteria.

$$M_1 = \begin{vmatrix} \frac{\partial^2 G}{\partial x_1^2} & \frac{\partial^2 G}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 G}{\partial x_1 \partial x_{n-1}} \\ \frac{\partial^2 G}{\partial x_2 \partial x_1} & \frac{\partial^2 G}{\partial x_2^2} & \cdots & \frac{\partial^2 G}{\partial x_2 \partial x_{n-1}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 G}{\partial x_{n-1} \partial x_1} & \frac{\partial^2 G}{\partial x_{n-1} \partial x_2} & \cdots & \frac{\partial^2 G}{\partial x_{n-1}^2} \end{vmatrix} = 0 \tag{5.2a}$$

This equation simplifies to the expression $(\partial G/\partial x_1^2) = 0$ when only one component present in the system.

Performing the appropriate variable transformations to obtain temperature and molar volume as independent variables recovers the restriction that $(\partial P/\partial V)_T = 0$ at the critical point.

$$M_2 = \begin{vmatrix} \dfrac{\partial M_1}{\partial x_1} & \dfrac{\partial M_1}{\partial x_2} & \cdots & \dfrac{\partial M_1}{\partial x_{n-1}} \\ \dfrac{\partial^2 G}{\partial x_2 \partial x_1} & \dfrac{\partial^2 G}{\partial x_2^2} & \cdots & \dfrac{\partial^2 G}{\partial x_2 \partial x_{n-1}} \\ \vdots & \vdots & & \vdots \\ \dfrac{\partial^2 G}{\partial x_{n-1} \partial x_1} & \dfrac{\partial^2 G}{\partial x_{n-1} \partial x_2} & \cdots & \dfrac{\partial^2 G}{\partial x_{n-1}^2} \end{vmatrix} = 0 \qquad (5.2b)$$

Equation 5.2a is sufficient to determine stability; equation 5.2b is only required at the vapor-liquid critical point when a description of the limiting behavior of equation 5.2a is needed. When only one component is present in the system, equation 5.2b simplifies to $(\partial^2 P/\partial V^2)_T = 0$.

Both of these equations become prohibitively complicated when additional components are incorporated into the system. Equation 5.2b involves the determinant of a matrix whose elements must be calculated as the determinants of a matrix. In practice, mixture critical points are calculated by evaluating the complete phase envelope instead of attempting to evaluate equation 5.2.

Even if it were convenient to use closed form expressions for mixture critical points, the experimental data required to describe mixture critical points are not available. The vapor-liquid critical locus for a binary system demonstrates a wide variety of behaviors and occurs at conditions that are not easily accessible experimentally. Specifying equation of state parameters through mixture critical data is not a viable option.

5.2 – Standard Parameter Mixing Rules

Parameter values and parameter composition dependency are explicitly specified for mixtures instead of relying on experimental mixture critical point data. The substance dependency of the *a* and *b* parameters suggested by van der Waals is given in equation 5.3.

$$a = \sum_j \sum_k x_j x_k \sqrt{a_j a_k} \qquad b = \sum_j x_j b_j \qquad (5.3)$$

Both the stability criteria from equation 5.2 and an estimate of experimental critical locus for mixtures are contained within a set of mixing rules.

Mixing rules are effectively models for specifying mixture critical properties, albeit indirectly. These rules are independent of the equations themselves and can be combined with any relevant EOS. The van der Waals mixing rules from equation 5.3 are extended to the SRK EOS, PR EOS, or any other EOS involving an alpha function by grouping that alpha function in with the pure component *a* parameter as in equation 5.4.

$$a = \sum_j \sum_k x_j x_k \sqrt{(a\alpha)_j (a\alpha)_k} \qquad b = \sum_j x_j b_j \tag{5.4}$$

The van der Waals mixing rules are not quantitatively accurate, but they do qualitatively represent the behavior of almost all binary mixtures examined. To account for deviations in behavior between the predicted and observed systems, tunable interaction parameters can be included into the mixing rules.

$$a = \sum_j \sum_k x_j x_k (1 - k_{jk}) \sqrt{(a\alpha)_j (a\alpha)_k} \qquad b = \sum_j x_j b_j \tag{5.5}$$

The binary interaction parameter k_{jk} is used to represent deviations from the behavior predicted in equation 5.4. It is possible to assign physical interpretation to binary interaction parameters, although most descriptions are tenuous as best. Values for k_{jk} are determined by matching experimental phase behavior data to the EOS predicted phase behavior data. These values are not constrained in any way and can be either positive or negative.

Fluid mixture of similar fluids, such as straight chain hydrocarbons with similar molecular weights, will have near zero values for the binary interaction parameter. Fluid pairs that are chemically dissimilar will tend to have larger values, and may exhibit temperature dependence in the values for the binary interaction parameter. There is no standard for representing the temperature dependence of binary interaction parameters. Typically, the required binary interaction parameter for a pair of fluids is determined at several temperature conditions and then correlated using a few polynomial terms in temperature. Including inverse terms in temperature is also common.

Unfortunately, using a single binary interaction parameter as described in equation 5.5 is insufficient to accurately represent many binary mixtures and additional tunable parameters are needed. A second binary interaction parameter is introduced in equation 5.6.

$$\begin{aligned} a &= \sum_j \sum_k x_j x_k (1 - k_{jk}) \sqrt{(a\alpha)_j (a\alpha)_k} + \sum_j x_k \left(\sum_k x_k \left(l_{jk} \sqrt{(a\alpha)_j (a\alpha)_k} \right)^{\frac{1}{3}} \right)^3 \\ b &= \sum_j x_j b_j \end{aligned} \tag{5.6}$$

Both k_{jk} and l_{jk} are symmetric parameters, that is both $k_{jk} = k_{kj}$ and $l_{jk} = l_{kj}$. When a second binary parameter is not necessary to accurate reproduce the phase envelope, setting the parameter $l_{jk} = 0$ recovers equation 5.5. The second binary interaction parameter may be temperature dependent in the same manner as the first binary interaction parameter.

Many sets of mixing rules have been developed. Based on publication frequency, developing and refining mixing rules for equation of state parameters is currently a more active area of investigation than improving the base EOS model. The mixing rules presented in equation 5.6 are attributed to Mathias, Klotz, and Prausnitz; however, for the binary case, their mixing rule is exactly equivalent to one proposed by Panagiotopoulos and Reed. The two rules give different predictions for systems of three or more components. Binary interaction parameter values are equivalent as well because these parameters are determined exclusively using two component mixture data.

The Mathias formulation was selected because it is not affected by the inconsistencies characterized by Michelsen and Kistenmacher. Specifically, many mixing rules, including the Panagiotopoulos formulation, that are suitable for binary mixtures do not behave properly for systems with three or more components. Formulating a ternary mixture of only two unique components should result in estimates equal to an equivalent binary mixture formulation.

Many mixing rules commonly used with the SRK EOS were not considered for use with the translated equation. Rules by Huron and Vidal, as well as Wong and Sandler provide good accuracy for highly asymmetric systems. Unfortunately the performance of these rules also depends on the availability of good estimates for the excess Gibbs or Helmholtz free energy. These properties are readily available from activity coefficient models, which was a motivating factor for designing the rules to require these data. Unfortunately, activity coefficient models are not valid for near critical and compressible fluid behavior, which is the area of particular interest when using the DMT function.

5.3 – Volume Translation Parameter Mixing Rules

Linear mixing rules were selected for the two volume translation parameters in the DMT function.

$$c_0 = \sum_j x_j c_{0j} \qquad c_1 = \sum_j x_j c_{1j} \tag{5.7}$$

All volume translation functions use linear mixing rules to take advantage of the invariance of the phase equilibrium when volume independent translations are paired with linear mixing rules.

Selecting zero values for the c_1 parameter will result in a constant molar volume translation, so using linear mixing rules for c_0 retains that utility in the even that volume dependence is not applied. Using linear mixing rules for both parameters is a consequence of no other mixing rules being available in the literature. Developing a new mixing rule for a volume translation parameter is not in the scope of this study.

Equation 5.6 includes a method of determining a mixture expression for the b parameter in the DMT function. A method for applying the empirical temperature function, tau, must also be described to complete the extension of the DMT to mixture systems.

Determining a mixture formulation for tau is difficult because of its piecewise definition and discontinuous derivative at the critical temperature. No equivalent expressions have been developed by other investigators so there is not a convenient reference for appropriate mixing rules. A linear mixing rule results in several points with discontinuous derivatives, one for each component in the mixture. Specifying a pseudo-critical temperature to use in the tau function still results in a discontinuous derivative at that specified temperature, but only a single discontinuity occurs and all of the derivatives of the DMT function with respect to composition are smooth.

$$T_{C,tau} = \sum_j x_j T_{Cj} \tag{5.8}$$

This temperature is not equal to the mixture critical temperature predicted by the translated EOS and is only used when calculating the value of the tau function for mixtures.

5.4 – Translated Binary Phase Envelope Estimates

Including volume dependence into the DMT function affects the phase envelopes predicted by the translated EOS. Chapter 4 incorporated a different alpha function to be used when the DMT was applied to the SRK EOS in order to account for difference in pure component predictions. All binary mixtures are bounded by these pure component predictions; that is, accuracy is ensured by pure component alpha functions at both ends of the binary phase diagram where the mole fraction goes to either zero or unity. At temperatures where both binary components are subcritical the phase envelope cross section reaches both extremes of the binary diagram and the volume dependence of the DMT function does not significantly change the estimated phase equilibrium.

Figure 5.1 depicts a constant temperature phase envelope cross section of the carbon dioxide - propane system at 70°F. A single binary interaction parameter, $k_{jk} = 0.14$ as described in equation 5.6, was used in both the translated and untranslated estimates in order to match the experimental data.

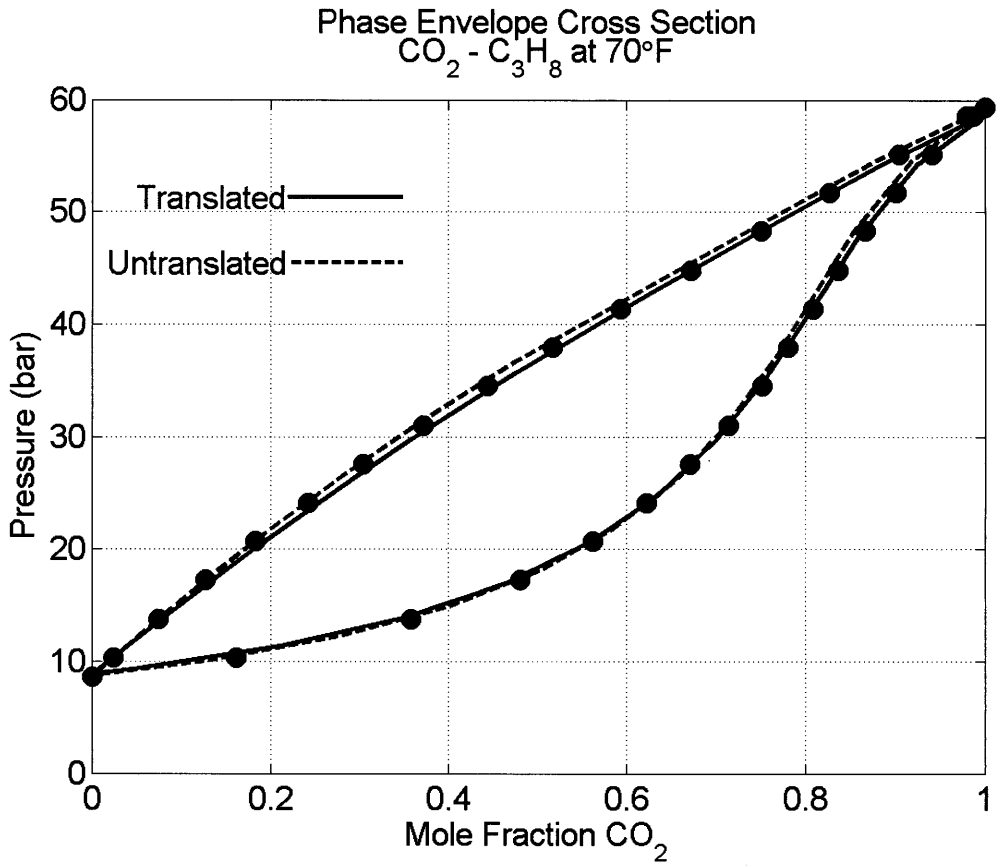

Figure 5.1: Phase envelope cross section for the binary carbon dioxide – propane system at a constant temperature of 70°F. Untranslated estimates are from the SRK EOS using a binary interaction parameter $k_{jk} = 0.14$ as described in equation 5.6. Translated estimates are from the DMT modification to the SRK EOS and use the same interaction parameter values. Experimental data are from Reamer et al.

At conditions where both binary components in a mixture are subcritical, the phase envelope predictions from the DMT modified SRK EOS are equivalent to those from the untranslated SRK EOS. Large effects were not expected at most conditions because the volume translation provides the largest correction in the critical region; however, the phase envelope is not significantly affected even near pure component critical points. Figure 5.1 depicts the binary carbon dioxide – propane system at a temperature where $T_r > 0.95$ for carbon dioxide. The ability to use the same binary parameters both before and after translation is valuable because a large database of binary parameter values is already available for use in the untranslated equation. Even if additional refinement of the representation is desired, the untranslated interaction value provides an excellent initial estimate.

Higher temperature binary phase envelopes, where one of the pure fluids is above its critical temperature, provide a more stringent test for the effects of the translation on phase envelop predictions.

Figure 5.2 depicts three constant temperature phase envelope cross sections of the carbon dioxide – methane system at 230K, 250K, and 270K.

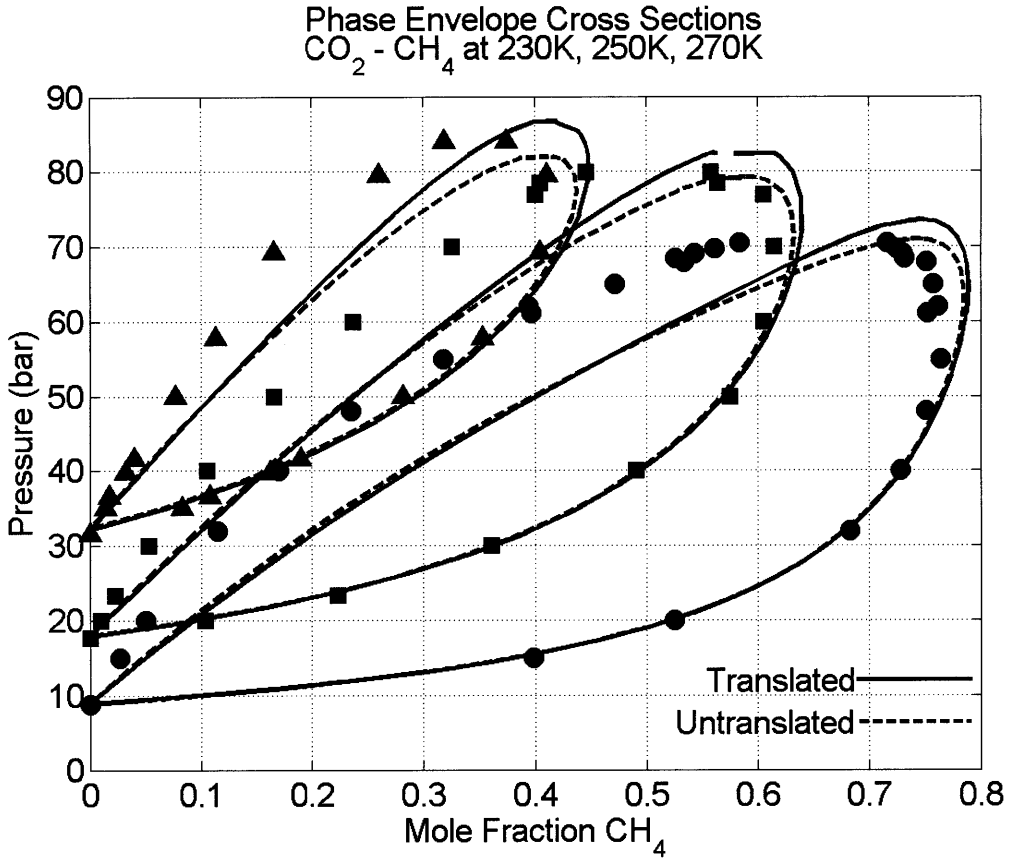

Figure 5.2: Phase envelope cross sections for the binary carbon dioxide – methane system at constant temperatures of 230K, 250K, 270K. Untranslated estimates are from the SRK EOS, translated estimates are from the DMT modification to the SRK EOS. No interaction parameters are used. Experimental data are from Davalos et al.

The estimates in figure 5.2 reveal both the limited impact from volume dependent translation and the inadequacy of the mixing rules for representation of data near the mixture critical point. Both the translated and untranslated equations provide adequate representation for one branch of the phase envelope, but are not accurate for the compositions along both simultaneously. It is not possible to improve the representation shown in figure 5.2 using the mixing rules from equation 5.6. Adjusting the binary interaction parameter k_{jk} can improve the representation in the liquid phase (i.e., high pressure) branch of the phase envelope, but it simultaneously decreases the accuracy in the vapor phase (i.e., low pressure). The optimal parameter value is subjective because the minimum in mean square error between the model and data is very broad. A zero value is near this minimum and was selected to simplify the representation.

Including a non-zero value for the second binary interaction parameter l_{jk} does not improve the accuracy of the model estimates near the mixture critical point. The influence of the parameter l_{jk} is most prominent at conditions where there is a strong phase asymmetry; near the mixture critical point, where the composition of both phases is similar, changing the value of l_{jk} alters phase envelope predictions in the same manner as changing the value of k_{jk}.

The second binary interaction parameter is most useful for systems that separate into two highly dissimilar phases. Figure 5.3 depicts the water – carbon dioxide at 15°C; a sufficiently low temperature for vapor-liquid-liquid equilibrium to occur.

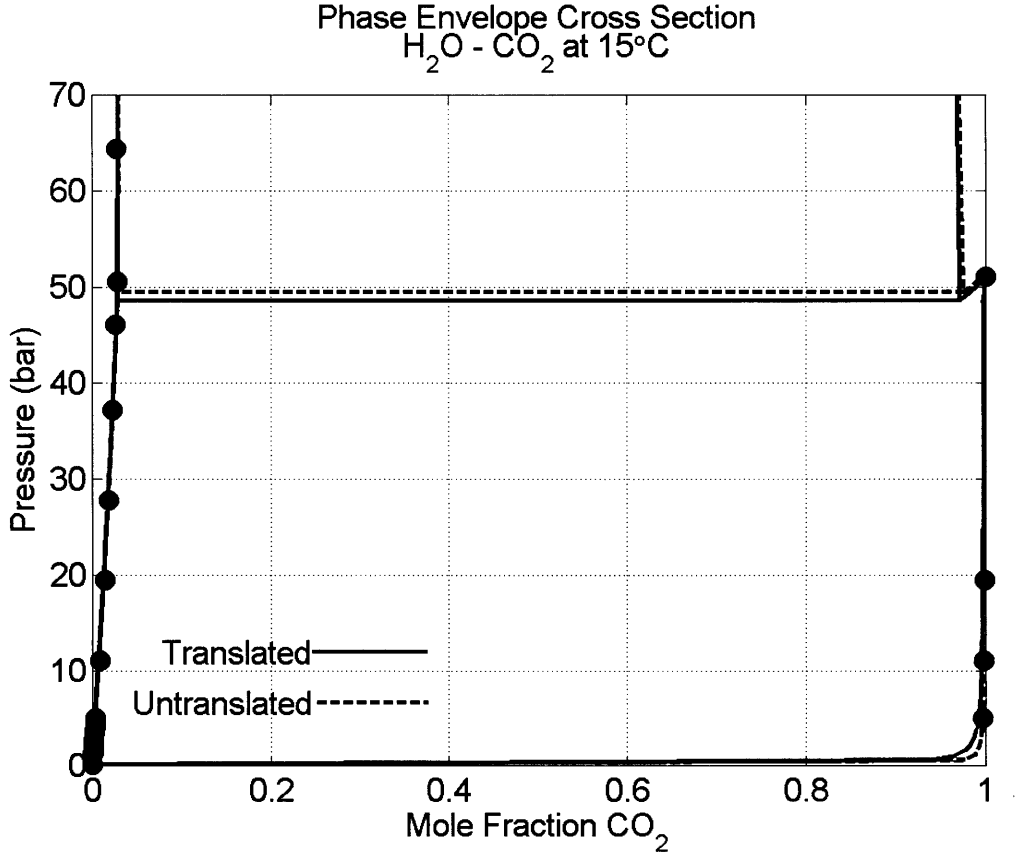

Figure 5.3: Phase envelope cross sections for the binary carbon dioxide – water system at a constant temperature of 15°C. Untranslated estimates are from the SRK EOS, translated estimates are from the DMT modification to the SRK EOS. Interaction parameter values for both models are k_{jk} = -0.12 and l_{jk} = 0.09. Data are taken from Anderson and King et al.

Applying the modified translation function to binary systems involving a third phase or requiring two interaction parameters does not significantly affect the predicted phase equilibrium.

Incorporating a volume dependent translation function alters phase envelope estimates, but these effects are not significant at subcritical conditions. Effects near the mixture critical point are more pronounced, but do not influence the accuracy of the model representation in any consistent manner. Table 5.1 lists the average disagreement in phase envelope estimates between the translated and untranslated models. These differences were calculated without using binary interaction parameters.

Table 5.1: Average differences between the untranslated SRK EOS phase equilibria estimates and the estimates from the DMT applied to the SRK equation. The entire phase envelope cross section was evaluated at the temperatures indicated. Binary interaction parameters were not used.

System	Temperature Range (K)	Average Difference	System	Temperature Range (K)	Average Difference
CO_2 - CH_4	210 - 290	2.8	H_2O - NH_4 **	300 - 600	14.1
CO_2 - C_3H_8	210 - 350	4.8	H_2O - CH_4	300 - 500	6.1
CO_2 - nC_8H_{18}	230 - 470	4.4	H_2O - C_3H_8 **	300 - 600	9.4
CO_2 - H_2S	210 - 360	3.5	H_2O - CO_2 **	300 - 600	13.2
H_2S – CH_4	200 - 350	3.6	N_2 - C_3H_8	200 - 350	3.8
H_2S - C_3H_8	200 - 350	2.3	N_2 - nC_8H_{18}	200 - 450	5.5
H_2S - nC_8H_{18}	200 - 500	5.1	N_2 - CO_2	210 - 290	3.5
CO_2 - H_2S	210 - 290	4.2	** Temperature range for system includes three phase equilibriua		

The differences indicated in table 5.1 are calculated with respect to the translated and untranslated models; they do not involve experimentally obtained data. The averages presented represent predictions over wide temperature ranges. As indicated previously, the differences at temperatures below the critical temperatures of both binary components tend to be small. Differences increase with temperature and are the greatest near the mixture critical point. The three systems that exhibit three phase equilibria also present the greatest differences between the translated and untranslated representation.

The purpose of this comparison is to provide a measure of the influence volume dependent translation has on phase equilibria estimates when interaction parameters are held constant.
The abilities of the translated and untranslated EOSs to represent binary phase equilibria are equivalent and determined primarily by the selection of mixing rules. Using

parameters from the untranslated EOS to obtain accurate estimates from the translated EOS without additional regression is useful. The untranslated SRK EOS has been widely used for several decades and many systems have been evaluated.

5.5 – Translated Binary Molar Volume Estimates

Molar volumes estimated by the translated EOS are significantly more accurate than those estimated by the untranslated EOS. Figure 5.4 depicts the saturated molar volume predictions for the carbon dioxide - propane system at 70°F. No binary parameters were adjusted to match the data in figure 5.4; binary parameters for the carbon dioxide - propane system were determined from phase equilibrium data as in figure 5.1.

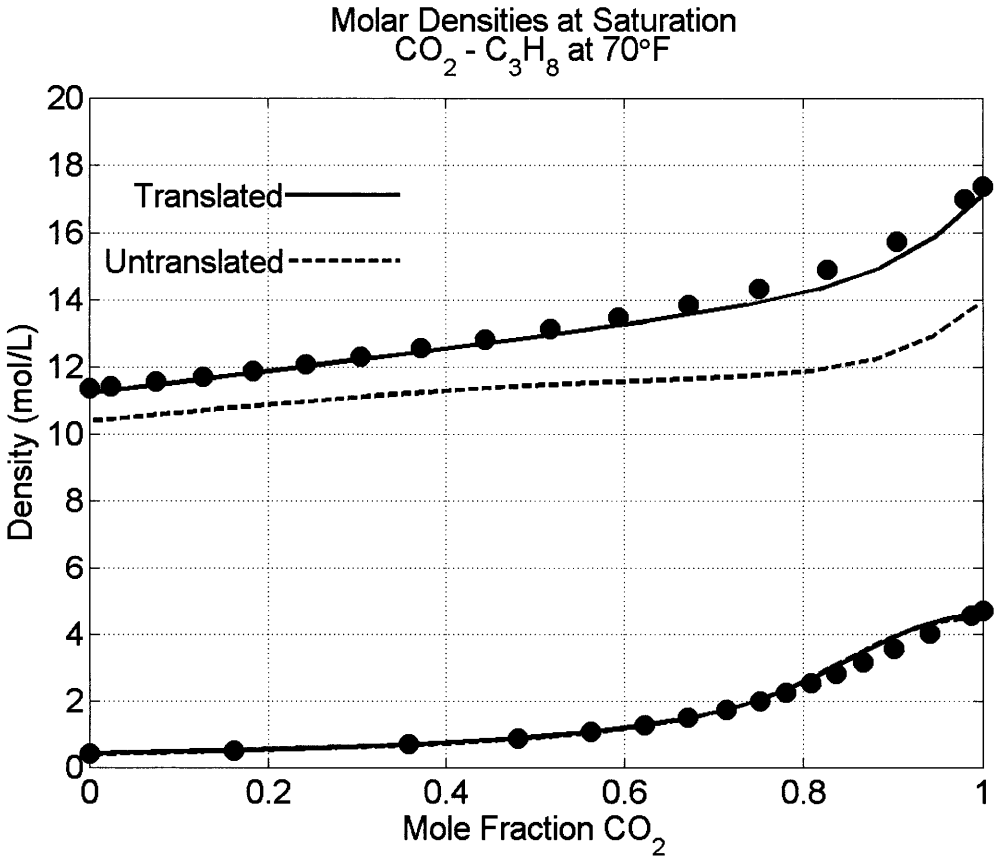

Figure 5.4: Saturated molar volume estimates for the binary carbon dioxide - propane system at a constant temperature of 70°F. Experimental data are from Reamer et al. Both models use a binary interaction parameter $k_{ik} = 0.14$.

The improvement in the molar volume accuracy of the mixture can be attributed in large part to accurate molar volumes for both pure components.

The untranslated EOS predicts the correct qualitative relationship between molar volume and mole fraction, but overestimates the volume by 10 to 20%. By correcting the pure component volumes at all pressure and temperature conditions, the mixture volumes are improved as well.

The accuracy of saturated molar volumes near the mixture critical point depends primarily on the accuracy of predicting the location of that point. There is a dearth of experimental density data near mixture critical points because of the difficulty in precisely measuring densities near a critical point. Figure 5.5 depicts the molar densities at saturation for the carbon dioxide - acetone binary system at 40°C.

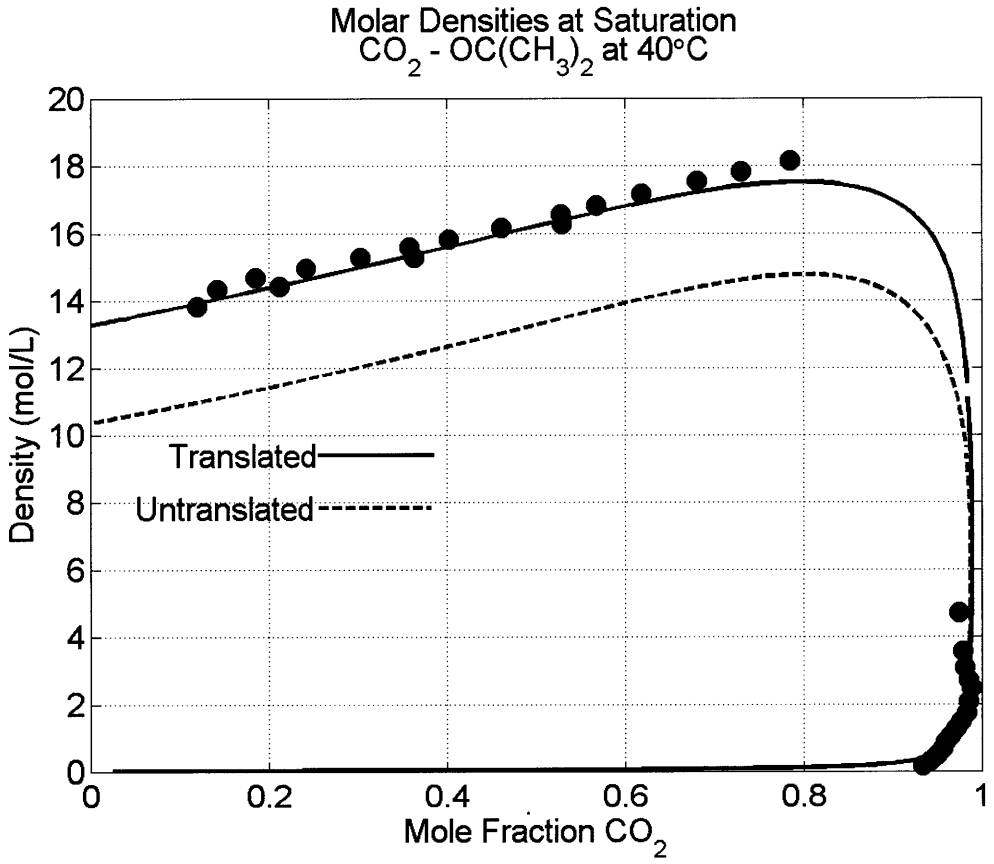

Figure 5.5: Saturated molar volume estimates for the binary carbon dioxide - acetone system at a constant temperature of 40°C. Experimental data are from Stievano and Day et al. Both models do not use binary interaction parameters.

Acetone was not included in the list of pure components examined in chapter 4 because there are no comprehensive references available for the molar volume behavior of pure acetone; it tends to associate and form dimers.

The associating behavior of acetone was not taken into account when estimating parameters for the DMT function. Parameters for acetone are given in table 5.2. The translation parameter c_1 was estimated using the specific gravity of acetone; the two alpha function parameters were calculated using Antoine data.

Table 5.2: Pure component parameters used for acetone when applying the DMT to the SRK EOS. Critical constants, Antoine coefficients, and specific gravity were taken from Poling et al.

Fluid	T_C (K)	P_C (bar)	V_C (L/kmol)	c_1 (L/kmol)	κ_1	κ_2
$OC(CH_3)_2$	508.1	47.01	209	72.0	0.8619	1.372

The value estimated for the translation parameter c_1 is about 25% greater than what would be predicted using the correlation in equation 4.16 for acetone. Some of this discrepancy may be due to the associating behavior of acetone, resulting in a denser than expected fluid.

The estimates in figure 5.5 once again show that the DMT function provides excellent accuracy in the liquid and vapor phase, although data for comparison near the critical point are absent. The EOS estimated mixture critical point occurs at a mole fraction of carbon dioxide that is approximately 0.98, independent of the selected model. If molar volume accuracy in this region is lacking, the primary means of improvement is refining the predicted mixture critical point.

The molar volumes of highly asymmetric mixtures were not examined. In these systems, both phases in nearly pure fluids and will have good accuracy because of the pure component DMT function.

5.6 – Ternary and Multicomponent Estimates

Extending the results of binary system representation to ternary and higher order systems is not complicated. However, demonstrating the ability to provide meaningful estimates is important because no parameters specific to ternary and higher order systems are used in either the translated or untranslated EOSs.

Three phase systems are difficult to visualize because of the number of independent variables involved. Figures 5.1 through 5.3 are all constant temperature cross-sections of a complete binary phase envelope, which has three independent coordinates. A complete ternary phase envelope requires four coordinates for a complete description, so a two dimensional cross section hold two of those coordinates constant. A cross section of the water - carbon dioxide - propane ternary system is presented in figure 5.6.

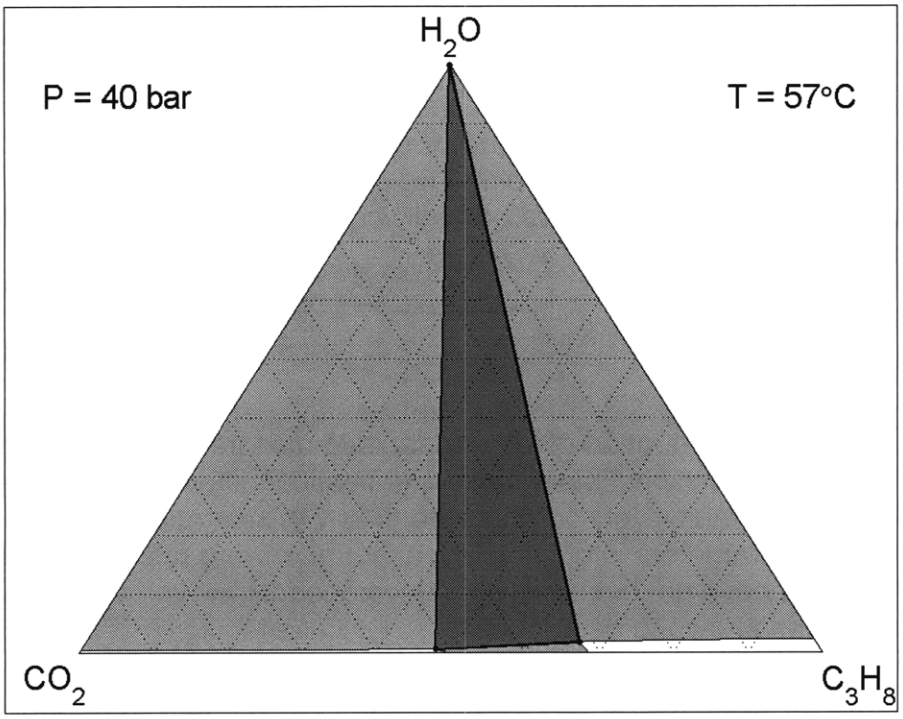

Figure 5.6: Phase envelope cross section for the ternary propane - carbon dioxide - water system at a constant temperature of 57°C and a constant pressure of 40 bar. Translated and untranslated estimates are indistinguishable in this presentation.

The upper point of the triangle diagram in figure 5.6 represents pure water, the other two vertices represent pure carbon dioxide and pure propane. Unshaded regions are single phase fluid, lightly shaded regions represent two-phase coexistence regions, and the central darkly shaded region represents a three phase coexistence region. The single phase regions that are rich in water (liquid) and carbon dioxide (vapor) in figure 5.6 are small but present. The single phase region rich in propane (liquid) is more easily visible.

Experimental data for the ternary system in figure 5.6 is available from Gil et al., although it was not included in the figure for the sake of clarity. Both the translated and untranslated EOSs provide equivalent estimates at these conditions; the difference in accuracy between the two models is not significant with regard to the compositions at equilibrium.

The translated EOS provides superior molar volume accuracy, although the improvement for ternary and higher order systems is not as pronounced as for binary systems. In particular, using an optimal constant translation value as described in chapter 4 achieves nearly the same level of accuracy as using the DMT function. The DMT function was designed to provide superior accuracy in the critical region; multicomponent mixtures typically have only one or two of their constitutive components at near critical conditions. Additionally, accuracy at the mixture critical point is influenced primarily by accurate estimation of that point. The greatest advantage of the DMT function is observed for mixtures that are primarily composed of one or two components at conditions near their respective critical points.

5.7 – Summary

Extending pure component equation of state models to mixture systems involves using mixing rule models to indirectly estimate mixture critical conditions. Directly calculating mixture critical points is prohibitively complicated. Many mixing rules are available for the standard *a* and *b* parameters in cubic EOS, although mixing rules for volume translation parameters have not been similarly developed. However, simple, linear mixing rules have been shown to be adequate.

The phase equilibrium estimates for the translated EOS are largely the same as estimates from the untranslated EOS despite the volume-dependent translation function affecting the equilibrium conditions. Differences are most notable near the mixture critical point, although predictions at these conditions are much more sensitive to the selection of mixing rules than volume translation.

Molar volume estimates for binary mixtures are substantially improved using the DMT function. Additionally, translation parameters for new fluids can be easily estimated for this function to provide this increase in accuracy without a detailed regression. Multicomponent systems do not demonstrate the same increase in accuracy when modeled with the DMT because constant molar volume translations are typically sufficient at most conditions. Volume dependent translation is most useful in systems with one or two dominant components that are near their respective critical points.

References

G Anderson; *J. Chem. Eng. Data.*, 47 (2002), 219-222.

J Davalos, W Anderson, R Phelps, A Kidnay; *J. Chem. Eng. Data.*, 21 (1976), 81-85.

C Day, C Chang, C Chen; *J. Chem. Eng. Data.,* 41 (1996), 839-843.

M Huron, J Vidal; *Fluid Phase Equilib.*, 3 (1979), 255-271.

L Gil, S Avila, P Garcia-Gimenez, S Blanco, C Berro, S Otin, I Velasco; *Ind. Eng. Chem. Res.*, 45 (2006) 3974-3980.

M King, A Mubarak, J Kim, T Bott; *J. Supercrit. Fluids.*, 5 (1992), 296-302.

P Mathias, H Klotz, J Prausnitz; *Fluid Phase Equilib.*, 67 (1991), 31-44.

M Michelsen, H Kistenmacher; *Fluid Phase Equlib.*, 59 (1990), 229-230.

M Michelsen, J Mollerup; Thermodynamic Models: Fundamental & Computational Aspects. 2nd Ed. Denmark: Tie Line, 2007.

A Panagiotopoulos, R Reid; *ACS Symp. Ser.*, 300 (1986), 571-582.

D Peng, D Robinson; *AIChE J.*, 23 (1977), 137-144.

B Poling, J Prausnitz, J O'Connell; The Properties of Gases and Liquids. 5th Ed. New York: McGraw-Hill, 2002.

H Reamer, B Sage, W Lacey; *Ind. Eng. Chem.*, 43 (1951), 2515-2520.

S Sandler; Chemical and Engineering Thermodynamics. 3rd Ed. New York: John Wiley & Sons, 1999.

M Stievano, N Elvassore; *J. Supercrit. Fluids*, 33 (2005), 7-14.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

P VanKonynenburg, R Scott; *Phil. Trans. Roy. Soc. Lon.*, 298 (1980), 495-540.

S Wong, S Sandler; *AIChE J.*, 38 (1992), 671-680.

Chapter 6 - Applications for the Translation Function

Volume translations incorporating both temperature and volume dependency are able to provide molar volume estimates that are accurate to within five percent at all fluid conditions. This consistency is in contract to constant molar volume translation functions that can provide molar volume estimates accurate to within five percent at most conditions, but are not accurate at temperatures in the vicinity of the critical point. Untranslated equation of state models to not generally provide quantitatively useful molar volume estimates other than in the vapor region. This chapter addresses several applications where the superior accuracy of the modified translation function is useful.

6.1 – Estimating Reversible Work Requirements

Pressure changing operations are ubiquitous throughout the chemical industry. The distinction between compressors and pumps depends on the fluid and operating conditions involved. Liquids often can be accurately approximated as incompressible fluids while gases cannot. This distinction is less clear at elevated temperatures or at supercritical conditions where all fluids are compressible to some extent.

When calculating the work requirement for a pressure changing operation, both the total work done on the fluid as well as the work required moving that fluid into and out of the pressure changing device (i.e., mass flow work) must be calculated. The total work can be expressed as the integral of the pressure over the change in the volume, while the mass flow work involves the upstream and downstream conditions (e.g., the upstream the fluid state is P_1 and V_1, while the conditions downstream are P_2 and V_2). The reversible work required in this situation is described by equation 6.1.

$$W_{rev} = -\int_{V_1}^{V_2} PdV + (P_2V_2 - P_1V_1) = \int_{P_1}^{P_2} VdP \qquad (6.1)$$

The total work and the mass flow work were combined into a single integral expression involving the molar volume. Equation 6.1 describes the reversible work; it was necessary to assume that the combined entropy change of the system and surroundings is equal to zero in order to arrive at this expression.

Any type of pressure change that causes a difference in volume involves mechanical work, although this volume change can be accomplished in different ways. Rapid volume change restricts the amount of heat that can be transferred between the system and its environment. Doing work on the system to rapidly decrease its volume will tend increase the system temperature. If the volume change occurs slowly, then the system can remain in thermal equilibrium with its environment and the temperature remains close to constant.

Actual fluid compression operations are neither adiabatic (i.e., no heat transfer) or isothermal; they tend to fall somewhere between these two extremes.

Improved molar volume accuracy provides increased confidence in estimating work requirements for pressure changing operations. An isotherm for pure carbon dioxide was selected for use in this comparison because accurate reference values are available. Additionally, the compression of pure, or nearly pure, carbon dioxide has particular relevance to the development of the modified translation. Carbon sequestration and tertiary oil recovery both involve the compression of mixtures predominantly composed of carbon dioxide from ambient conditions to high pressure, subsurface conditions. The near ambient critical temperature of carbon dioxide, about 31°C, makes property estimation in this region difficult.

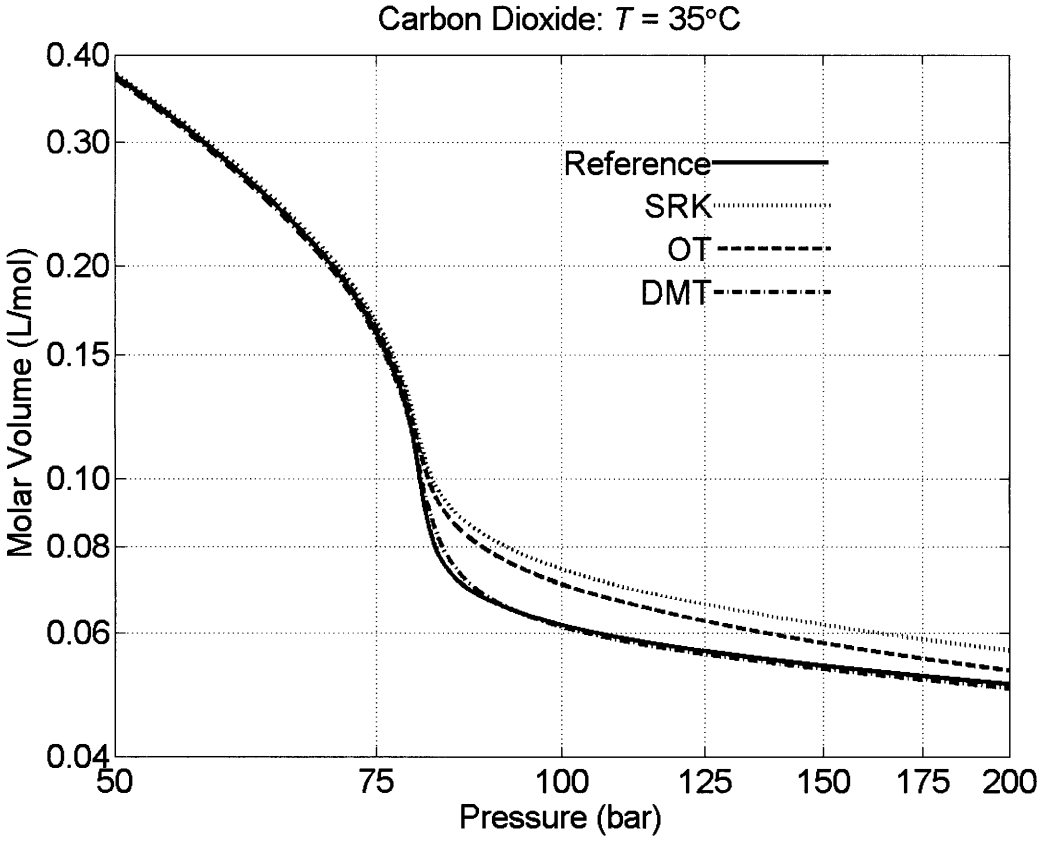

Figure 6.1: Isotherms for carbon dioxide at 35°C as estimated by the SRK EOS, the SRK EOS with an optimal constant translation (OT) and the SRK EOS with the modified translation (DMT). Reference values are taken from Span and Wagner.

Figure 6.1 depicts several isotherms for carbon dioxide at 35°C; selecting this temperature avoids a first order phase transition from vapor to liquid during the compression.

An adiabatic compression would involve continuously varying temperature between the two states. The optimal constant translation selected for carbon dioxide is equal to 3.6 L/kmol, which is the same value used in chapter 4. The isotherms in figure 6.1 reveal that this constant translation value is not optimal at 35°C; larger positive values for the constant translation would improve accuracy at this temperature, but would be inappropriate at lower temperatures.

The reference value for the work required to perform an isothermal compression of carbon dioxide from 50 bar to 200 bar at 35°C is 31.9 kJ/kg. This value was calculated by integrating the 'Reference' isotherm in figure 6.1 from 50 bar to 200 bar. The work requirement estimated by the untranslated SRK is 34.8 kJ/kg; an optimal constant translation results in an estimate of 33.6 kJ/kg, and the DMT estimates 31.7 kJ/kg. The DMT provides the most accurate estimate; the relative errors from the three models are only 9.1%, 5.3%, and 0.6% respectively.

The results presented for the isothermal compression indicate that using the DMT function to estimate volumes provides a useful increase in accuracy. These conditions were selected in order to emphasize the advantages in using the DMT function for calculations in the vicinity of the vapor-liquid critical point. At other conditions, the difference in accuracy is less pronounced. The reversible work requirement for the isothermal compression of carbon dioxide from 1 bar to 50 bar is 214 kJ/kg. All three models provide accurate estimates of this value to within working precision (i.e., approximately 0.4%).

Although all operations will involve additional energetic losses due to friction and other irreverabilities, calculation of the reversible work required for a pressure changing operation establishes a baseline that can be used to decide on the viability or expense of a process.

Volume dependent translations provide a notable accuracy advantage over constant molar volume translations when operating in the vicinity of the vapor liquid critical point. The tradeoff between increased model complexity and increased accuracy at other conditions is more subjective. Generally, the DMT function is most useful when fluid operations occur between a reduced temperature of 0.8 and 1.2. Simpler models provide sufficient representation outside of this temperature range.

6.2 –Efficiency of Pressure Changing Operations

Pressure changing operations do not operate reversibly. The isothermal compression described in section 6.1 would need to be performed arbitrarily slowly, so that the system was never out of equilibrium, in order to be reversible.

The actual amount of work required relative to the calculated reversible work needed is defined as the stage efficiency, etc.

$$\eta_t = \frac{W}{W_{rev}} \tag{6.2}$$

The reversible work from equation 6.2 is calculated according to equation 6.1. The example calculations from section 6.1 involved an isothermal pressure change; this selection was made to simplify calculations, any path between initial and final states may be used. The subscript *t* on eta in equation 6.2 denotes the total efficiency of the pressure changing operation. In practice, there are physical limitations to the compression ratio that can be achieved by a single operation. Large pressure changes require several smaller pressure changes performed in sequence. Internal combustion engines may have compression ratios in excess of 10, while turbines typically operate with ratios near 7, and most compressors involve ratios under 5. The lower ratios on compressors are the result of increased heat dissipation requirements. To maintain the desired working temperature, the compressed fluid usually needs to be cooled between stages of the overall pressure changing operation.

The actual efficiency of compressors and pumps is highly dependent on the specific methods and geometries used, although some generalizations can be made; an in depth treatment is well beyond the scope of this study. Accurate values for the efficiency of a pressure changing operation are important for design purposes; several operations are usually combined in series, resulting in an overall efficiency that is the product of the efficiencies of individual stages.

Machines that are geometrically similar, have similar flow velocity profiles, exhibit the same ratio of gravitational to inertial forces, and operate with fluids that have the same thermodynamic properties will demonstrate similar efficiencies. This principle is used to identify dimensionless quantities that are used to estimate efficiencies. Employing similarity in this manner is analogous to using reduced coordinates to represent fluid models in thermodynamics (e.g., T_r, P_r, V_r), although the physical interpretation is not as obvious. The four most important dimensionless parameters with regard to efficiency estimations are the dimensionless specific speed, n_s, and dimensionless specific diameter, d_s, the Reynolds number, *Re*, and the Mach number, *Ma*. These quantities are defined below.

$$n_s = \frac{\omega\sqrt{\dot{V}}}{(H_{ad}g)^{3/4}} \qquad d_s = \frac{D(H_{ad}g)^{1/4}}{\sqrt{\dot{V}}} \qquad Re = \frac{vl}{\upsilon} \qquad Ma = \frac{v}{c_s} \tag{6.3}$$

In equation 6.3, ω is the angular velocity of rotation in the pressure changing device, *v* is the tip speed of the rotating part, and D is the characteristic diameter *l* is the characteristic length. None of these quantities are influenced by the fluid properties. However, the remaining four quantities are dependent on volumetric accuracy.

The expression $H_{ad}g$ refers to the product of the adiabatic head with the acceleration due to gravity. This quantity is exactly equal to the reversible work performed by the operation calculated by equation 6.1. Also include in both quantities defined in equation 6.3 is the square root of the volumetric flow rate through the pressure changing device. The volumetric flow rate is affected by density estimates because device specifications often involve mass flow rates and not volumetric flow rates. Volumetric flow rates are easier to measure in practice, but are not preferred from a design perspective because of their variability with pressure and temperature conditions.

The kinematic viscosity, υ, is linearly proportional to the specific volume in the same manner as the volumetric flow; kinematic viscosity is equal to the absolute fluid viscosity divided by the density. The Mach number in equation 6.3 is the ratio between the tip speed, v, and the speed of sound, c_s. The speed of sound in a fluid can is calculates as indicated in equation 6.4. The modified translation function impacts both the estimated value of the specific volume as well as the pressure derivative in equation 6.4.

$$c_s = -V\sqrt{\left(\frac{\partial P}{\partial V}\right)_S} \tag{6.4}$$

Fortunately, errors in volume do tend of offset somewhat when calculating the parameters in equation 6.3. Untranslated EOS models tend to overestimate the specific volume, which corresponds to overestimations of both the reversible work requirement and the volumetric flow rate. Overestimating both of these values by 5 percent would only result in a 3 percent overestimation of specific speed. The explicit dependence of the efficiency on the four dimensionless parameters from equation 6.3 is complicated and will not be reproduced here. Improved molar volume accuracy improves the accuracy of these parameters and correspondingly leads to better estimates of efficiency.

6.3 – Process Modeling and Optimization

Most process simulators do not use EOS estimated molar volumes because of their suspect accuracy. A popular substitute is the Rackett molar volume correlation. This correlation was developed specifically to correlate saturated molar volumes, although in practice it has been used extensively for all molar volumes at sub-critical temperatures. This correlation is presented in equation 6.2.

$$V = \frac{RT_C}{P_C} Z_{RA}^{\left(1+(1-T_r)^{\frac{2}{7}}\right)} \tag{6.5}$$

Using equation 6.5 requires correlations for determining the critical temperature and critical pressure for mixtures of interest.

Recommended correlations for these values require estimates of the critical volume as well. The value Z_{RA} is the Rackett compressibility factor, which is adjustable for each pure component.

The level of complexity of the correlation in equation 6.2 is equivalent to that of the DMT function. Figure 6.2 depicts estimates from the untranslated, translated, and Rackett models for an isobar of pure carbon dioxide.

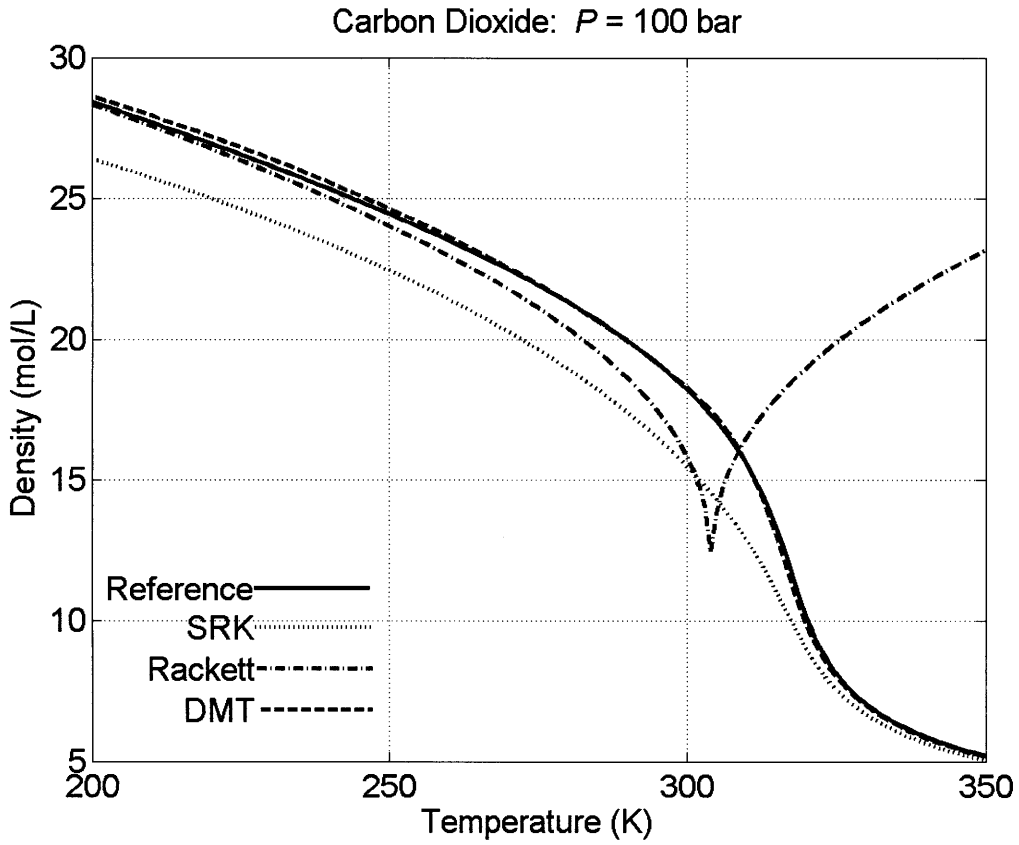

Figure 6.2: Density estimates for pure carbon dioxide at 100 bar. Predictions are from the untranslated SRK EOS, DMT modification to the SRK EOS, and the Rackett correlation. Reference values at these conditions are taken from Span and Wagner.

The Rackett correlation generally provides better accuracy for saturated molar volumes than the DMT function. However, the Rackett equation does not account for compressibility effects and is only valid at temperatures below the estimated critical temperature. Compressibility effects are significant at pressures above a fluid's critical pressure; the Rackett equation was never intended for extension away from saturated conditions. Additionally, the transition between the Rackett model and an EOS model for higher temperature estimates is problematic.

A popular choice is to use the Rackett model at subcritical temperatures ($T < T_C$) and the SRK model at higher temperatures. Unfortunately, as can be seen in figure 6.2, this would introduces a discontinuity into molar volume estimates (i.e., at approximately 304K, the transition between the Rackett and SRK models involves a step change).

The DMT function can be used as a substitute for the Rackett correlation when performing process modeling and optimization to avoid the difficulties that occur when transferring between two models. The absolute error that occurs because of this transition is typically not large, as discussed in section 6.1, although the discontinuity has the potential to lead to numerical errors and erroneous results. The discontinuity that exists at elevated pressure conditions when switching between methods of molar volume estimation effectively introduces a first-order transition (e.g., phase change) at conditions where only a uniform fluid should be present.

6.4 – Process Development at Supercritical Conditions

Designing an EOS modification specifically applicable to elevated pressure and temperature conditions supports process development at those conditions. Many design heuristics recommend selecting operating conditions such that near critical behavior is avoided; fluid properties in this region change rapidly and can be difficult to control. However, many operations either cannot avoid near critical conditions or are designed specifically to take advantage of rapidly changing fluid properties.

The example of carbon dioxide compression for sequestration purposes or enhanced oil recovery was cited in section 6.1. Pure fluids are never encountered in practice, typically because separation costs are too high or introduction of additional components is unavoidable. Subsurface operation inevitably introduces water as a mixture component. Binary water – carbon dioxide systems are abstractions because these two components readily react to form varying amount of carbonic acid. For the purposes of this study, reacting equilibria have been neglected. Figure 6.3 is an analogous to figure 5.3 from chapter five, although densities at saturation are presented instead of pressures. Only the carbon dioxide rich phase has been presented for the sake of clarity.

The density estimates from the DMT function presented in figure 6.3 are in excellent agreement with experimentally reported values. It is important to emphasize once again that the binary interaction parameters were determined using only the saturation pressure; the molar volume estimates in figure 6.3 are a consequence of good representation of the phase envelope. These estimates are for a carbon dioxide rich phase that is slightly above its vapor-liquid critical point, near a critical end point. The critical locus for the water – carbon dioxide is discontinuous, with a second set of critical points occurring at much higher pressures and temperatures.

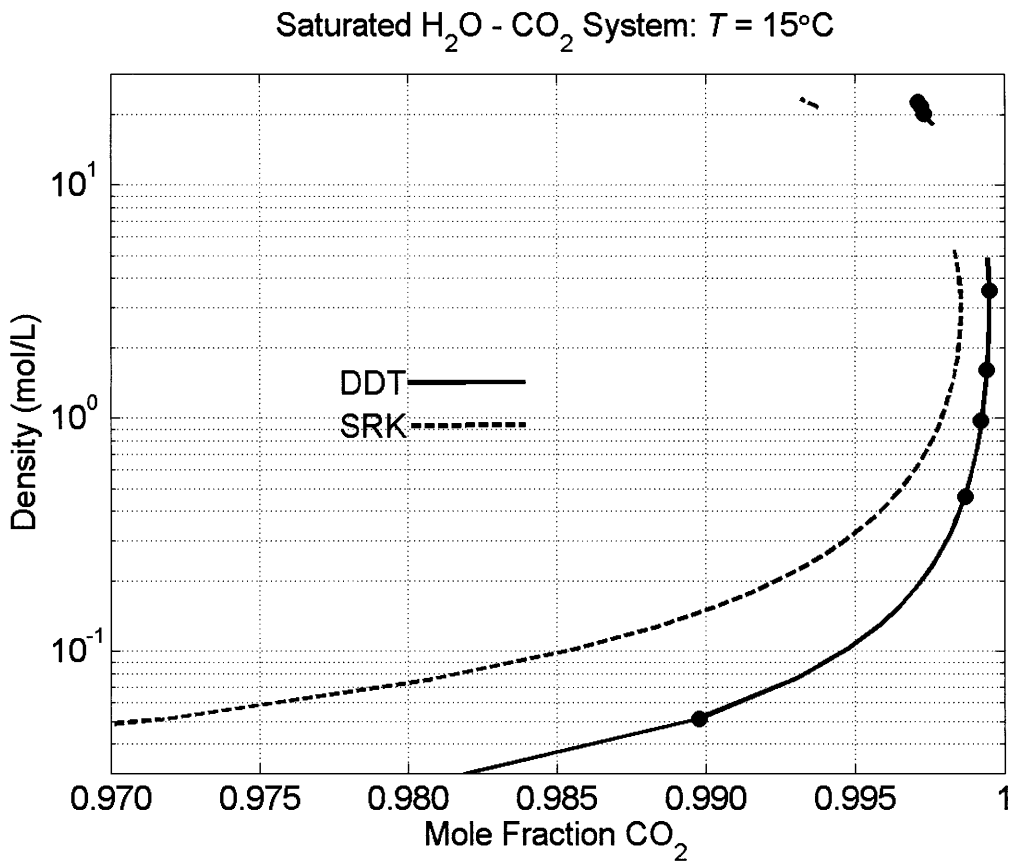

Figure 6.3: Saturated density estimates for the carbon dioxide rich phase in a water - carbon dioxide system at 15°C. Binary interaction parameters are the same as presented in chapter 5; both models use k_{jk} = -0.12 and l_{jk} = 0.09. Data are taken from Anderson and King et al.

One of the major considerations for operating in the vicinity of a critical point is the dearth of experimental data available. When using a volume translation function based on a corresponding states approach, this minimal amount of data can be leveraged into accurate estimates over a large range of conditions.

It is beneficial to operate near a critical point for some applications. Figure 6.4 depicts the pressure at which a copolymer precipitates out of acetone when near critical carbon dioxide is used as an anti-solvent. The mechanical challenges incurred when using a near critical fluid are offset in this instance by the benign nature of carbon dioxide and acetone. Other organic solvents used in recrystalization can be significantly more hazardous from a chemical standpoint. The tradeoffs between mechanical and chemical concerns depend in large part on the specific system of interest; having good physical property estimates can help remove much of the uncertainty involved with near critical fluid operation.

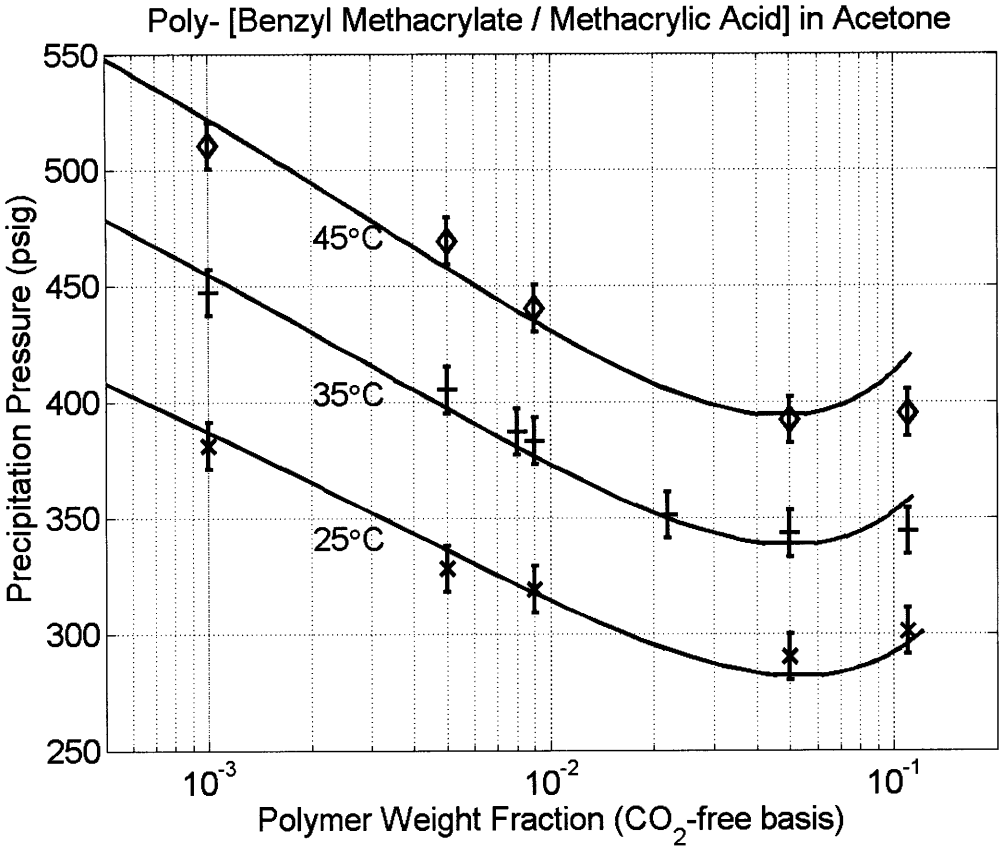

Figure 6.4: Pressure at which an 85% benzyl methacrylate / 15% methacrylic acid copolymer precipitates from acetone when carbon dioxide is introduced as an antisolvent. This pressure is shown as a function of the polymer weight fraction in solution prior to the addition of carbon dioxide. Three different temperature conditions are shown: 25°C, 35°C, and 45°C. Data are from Paap.

The DMT function does not provide improved estimates of phase behavior when compared to the untranslated SRK EOS. Interaction parameters were adjusted as necessary to represent the data in figure 6.4. The fugacity of the solid polymer phase was approximated as equal to the fugacity of the polymer in an EOS predicted subcooled liquid phase; no EOS can simultaneously represent both the fluid and solid phases.

Figure 6.4 is included here to illustrate interesting behavior that occurs at elevated pressure and temperature conditions. The solubility of the copolymer demonstrates a minimum in pressure at the conditions depicted. Systems at these conditions are not routinely investigated because of the difficulty in modeling physical properties near the vapor-liquid critical point. Improved models such as the DMT function contribute towards higher confidence in operating at these conditions.

6.5 – Summary

Incorporating volume dependence into a translation function allows for significantly improved molar volume accuracy, although this improved accuracy comes with increased complexity as well. Applications such as compressor design can take advantage of the increased accuracy for estimates of both the total reversible work requirement as well as for determining the efficiency of the pressure changing device.

An alternative application for the translation function is to provide a single, continuous model for molar volume estimates at supercritical conditions. Other models, such as the Rackett equation are available, although these models were not intended for use in the vicinity of the vapor-liquid critical point. Providing improved models for supercritical process conditions contributes toward developing new applications in this region.

References

Aspen Plus Reference Manual; Version 10. Cambridge, MA: Aspen Plus, 1999.

O Balje; Turbomachines - A Guide to Design, Selection, and Theory. New York: John Wiley & Sons, 1981.

S Paap; *PhD Thesis*. MIT, 2009.

H Rackett; *J. Chem. Eng. Data.*, 15 (1970), 514-517.

R Span, W Wagner; *J. Phys. Chem. Ref. Data.*, 25 (1996), 1509-1596.

C Spencer, R Danner; *J. Chem. Eng. Data*, 17 (1972), 236-241.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

Chapter 7 - Conclusions and Recommendations

A general rubric for constructing volume translation functions was developed and used to create a novel volume translation function applied to the Soave-Redlich-Kwong equation of state. This chapter summarizes the incorporation of the volume translation function into a cubic EOS framework and the results of its application to pure fluids and mixtures. Additionally, comments on the applicability of the function as well as suggestions for additional thermodynamic model development have been provided.

7.1 – Project Summary

Chemical thermodynamics is an elegant and comprehensive structure for correlating and predicting thermophysical properties. Volumetric equations of state are the most versatile of the many thermodynamic models available because they incorporate most of the information available in the Fundamental Equation and can be expressed in terms of experimentally observable properties. A volumetric equation of state in conjunction with a model for the ideal gas state heat capacity is sufficient to completely recover the Fundamental Equation. The principle of corresponding states enables the development of robust, yet simple models that are valid throughout the entire fluid region. However, the most widely used of these models, cubic equations of state, do not provide good accuracy when estimating molar volume.

The method of volume translation has been used by many investigators to improve the liquid molar volume accuracy of cubic equations of state. Constant molar volume translations provide an increase in accuracy at reduced temperatures less than 0.7 without increasing model complexity, but cannot be used to accurately predict molar volumes along the entire pure component saturation curve. Temperature dependent translation functions provide accuracy for the entire saturation curve, but introduce serious inconsistencies at pressures above the critical pressure. Incorporating both temperature dependency and volume dependency into the translation function was shown to be necessary for improving or maintaining molar volume estimates at all conditions throughout the fluid region.

Formulating a simple expression for the required volume translation function was possible through a corresponding states approach; the pattern of molar volume residuals exhibited by current cubic equations of state was correlated using only two adjustable parameters. Both of these parameters can be estimated or calculated directly from the critical molar volume, although additional density data enabled a superior fit. Unfortunately, creating a translation function that depends on volume affects the predicted phase equilibria. This effect is minor for pure components, especially at temperatures below the vapor-liquid critical temperature. An alternative alpha function, selected from the literature, was used to account for changes in predicted phase equilibria.

Standard mixing rules were used to extend the volume dependent translation function to fluid mixtures. Both of the parameters in the volume translation function were formulated for mixtures using linear combination rules. The effect of the translation function on mixture phase envelope calculations was limited; the greatest change was seen in the vicinity of the mixture critical point. These changes did not provide any consistent increase or decrease in prediction accuracy for the mixtures. Mixture critical point inaccuracy was mostly attributed to selection of the mixing rules and not the volume dependency of the translation function.

Incorporating volume dependence in the translation function provided a high level of consistency in molar volume estimates. Pure fluid molar volumes were accurate to within approximately five percent at all conditions; mixture volumes were also accurate to within five percent given appropriate composition specifications. This accuracy can be useful for applications where density information is important. Compressor work requirements for pure fluids in the vicinity of the critical point were estimated accurately to within one percent. The translation function may be most useful by providing a single, smooth model for use in process simulation and optimization.

7.2 – Analysis of the Volume Translation Approach

The volume translation approach is not highly regarded in the literature; it has been described it as "excessively empirical". Much of the criticism can be attributed to volume translation functions that have been used to increase accuracy at some conditions while simultaneously decreasing it at others. The most successful engineering models for property estimation either do not use volume translation or use a constant molar volume translation. This study has presented a structure for developing volume translation functions that incorporate both temperature and density dependence, which is a necessary to avoid introducing inconsistencies through translation. A translation based on a modification of Mathias' distance parameter approach was formulated, applied to the Soave-Redlich-Kwong model, and shown to provide excellent molar volume accuracy throughout the fluid phase.

This project was motivated by the need for computationally simple and robust models that can provide good accuracy at high temperatures and pressures for predicting both volumetric properties and equilibrium conditions. The reference correlations cited in chapter 4 when developing the modified translation are too complicated for routine use; they involve hundreds of regressed parameters and do not provide any physical insight into the systems they represent. It was hoped that a translation function that provided good representational accuracy would also provide some insight into fluid behavior and not simply be an empirical correlation of the molar volumes. Unfortunately, despite the success of the modified translation in improving accuracy for molar volume estimates, other properties remained generally unaffected. Phase equilibrium estimates were altered slightly, as described in chapters 4 and 5, but these changes were typically small and did not consistently increase of decrease the accuracy of estimates.

Most of the improvement in mixture accuracy was a direct consequence of the increase in pure component accuracy. Most mixture volumes can be approximated as linear combinations of the volumes of their constituent components. This approximation tends to be accurate at most conditions, although it breaks down in the neighborhood of the mixture vapor-liquid critical point. It was not possible to discern between inadequacies in the translation function and inadequacies in the mixing rules at mixture critical conditions. The corresponding states based approach used in the formulation of the translation function predicts fluid properties based on that fluid's proximity to its vapor liquid critical point. When the critical point is not accurate, then no statement about the quality of the model can be made.

Contrasting a constant molar volume translation with the modified translation developed in this study is the most equitable comparison for the abilities of the translation function. The constant molar volume translation is significantly simpler because it does not influence the model predicted phase equilibrium. It also provides equivalent molar volume accuracy at most conditions, except for in the region of reduced temperature from $0.8 < T_r < 1.2$. For molar volumes, it is not clear if the improvement in accuracy in this region merits the additional complexity introduced by the volume dependence in the translation function. Using the fugacity coefficient translated by a volume dependent function is not recommended because the computational requirement for equilibrium calculations is increased without any improvement in accuracy.

Incorporating volume dependency into the translation function is a valid method for improving molar volume accuracy, but not for any other property calculated form the equation of state. The DMT function developed in this study is useful primarily as a tool for predicting molar volumes when more accurate tools are not available.

7.3 – Direct Correlation of Residual Potentials

An alternative approach to the volumetric equation of state approach is one where models are constructed for a residual potential; most commonly A^{res}, but occasionally G^{res} as well. The most accurate correlation for the properties of water is defined piecewise using five separate representations of G^{res}, each of which is valid for different ranges of pressures and temperatures.

These correlations involve tens or hundreds of parameters are usually created only for pure fluids because sufficiently extensive mixture data is not available. However, recent work has been done on creating a model of this type for mixtures, although it involves in excess of fifteen hundred of parameters. The mathematical expression of these models is often a series of polynomial or exponential terms that have been selected for their ability to represent the data and not any physical significance. A few models have even been extended to the complex domain and involve imaginary terms; although this practice may improve accuracy, there does not seem any way for those terms to be physically relevant.

The volume dependent translation function did not incorporate any physical insight into its mathematical representation, although it is possible that one of the results of this study can be used to guide development in this area. Integration of the expression presented in equation 4.12 results in a series of eighteen logarithmic terms in the untranslated volume. The untranslated expression for A^{res} calculated from the SRK EOS involves three logarithmic terms in the untranslated volume. None of the reference equations, expressed as models of A^{res}, use terms involving the logarithm of the volume. Several different basis set terms have been investigated, including square well terms and hard sphere terms, but these terms have not been found to improve the representation. Incorporating logarithmic terms may allow for improved accuracy using fewer adjusted parameters.

Other residual potential models do not involve direct regression of observable data; expressions for intermolecular forces are postulated and then residual potential models are derived from these postulated forces. In most instances, this derivation involves volume integrals that cannot be explicitly evaluated and approximations must be used. Models developed in this manner tend to provide property estimates of roughly the same accuracy as the modified cubic equations used in this study; a three parameter first principles model will provide the same accuracy as a three parameter modified cubic equation. Additionally, these first principles models are usually extended to mixtures using mixing rules similar to those applied to cubic EOSs

7.4 – Additional Mixing Rule Development

The need for improved mixing rules was indicated in sections 7.2 and 7.3. Mixing rules are simple correlations for estimating mixture critical properties (or potentially other mixture parameters) and are indispensable when using a corresponding states methodology, which depends on accurate values of the vapor-liquid critical point. Mixing rule development is a much more active area of investigation than either volume translation or any other aspect of thermophysical property modeling. The quality of mixing rule models is usually determined by their ability to represent binary phase envelopes, which has led to consistency issues for ternary and higher order mixtures as described in chapter 5. The motivation for mixing rules such as the Huron-Vidal rules or Wong-Sandler rules was the inadequacy of the simpler rules for complex binary mixtures. The more detailed formulations unquestionably provide more accurate estimates, but also involve increased complexity as well.

Mixture density estimates from the DMT function depend in large part on the mixing rules selected for the *a* and *b* parameters, and not the two translation function parameters. Incorporating interaction parameters on the translation function parameters does not have a significant impact on mixture properties. Volume translation results can be improved by refining the mixing rules used in the untranslated EOS.

7.5 – Recommendations for Property Modeling

A cubic equation of state with a constant molar volume translation is sufficient for most applications. There does not appear to be any benefit when incorporating a volume dependent translation function for estimating properties other than the molar volume. Employing the empirical temperature correction, tau, to ensure thermodynamic consistency in the unstable region of the DMT function enables it to be used for estimating A_{res}, although the translated value of this property was not found to be consistently more accurate than its untranslated value. Molar volume estimates are not affected by the tau function.

The formulation of the DMT function was accomplished by evaluating many candidate expressions for representing molar volume residuals and retaining expressions that improved the quality of the representation. This procedure could be automated and extended to expressions for the residual Helmholtz free energy so that other properties in addition to the molar volumes could be included when determining the quality of the representation.

A hypothetically perfect corresponding states model is one involving only a few parameters that can all be determined from pure component critical properties or other easily accessible parameters such as the acentric factor. However, even with this hypothetically perfect model, mixture representations still must be formulated using auxiliary models (i.e., mixing rules). The models needed to formulate the mixture representation are as important to the overall quality of the representation as the pure component EOS model.

The data requirement for validating mixture thermodynamic models is also proportionately higher than the requirement for pure component models. The mixture critical region is very data poor; it is impossible to make broad statements about model quality without some experimental data for comparison. The ultimate goal of modeling is to provide accurate property estimations without the need for extensive experimental data, but for many systems insufficient data are available to even validate a proposed model. Additional data are needed, particularly in the mixture critical region.

References

J Ahlers, J Gmehling; *Fluid Phase Equilib.*, 191 (2001), 177-188.

G Dahlquist, A Bjorck, N Anderson; Numerical Methods, New Jersey: Prentice-Hall, 1974.

A Elhassan, R. Craven, K deReuck; *Fluid Phase Equilib.*, 130 (1997), 167-187.

R Feistel, W Wagner; *J. Phys. Chem. Ref. Data.*, 35 (2006), 1021-1046.

O Kunz, R Klimeck, W Wagner, M Jaeschke; *GERG Technical Monograph 15*, 2007.

J Prausnitz; *AIChE J.*, 50 (2004), 739-761.

R Span, W Wagner; *Int. J. Thermophys.*, 24 (2003), 1-39.

R Span, W Wagner, E Lemmon, R Jacobsen; *Fluid Phase Equilib.*, 183-184 (2001), 1-20.

J Tester, M Modell; Thermodynamics and Its Applications. 3rd Ed. New Jersey: Prentice Hall, 1997.

J Valderrama; *Ind. Eng. Chem. Res.*, 42 (2003), 1603-1618.

W Wagner, A Pruss; *J. Phys. Chem. Ref. Data.*, 31 (2002), 387-535.

J Wisniak; *J. Chem. Edu.*, 74, 1997, 546-548.

Appendix A.1 - DMT Function Implementation for Molar Volumes

The volume translation methodology used in this study is presented below in equations A.1 and A.2.

$$P = f_1(\underline{V}_{UT}, T, N_1 \ldots N_n) \tag{A.1}$$

$$\underline{V} = \underline{V}_{UT} - f_2(\underline{V}_{UT}, T, N_1 \ldots N_n) \tag{A.2}$$

The novel modified translation function based on a modified distance parameter approach (DMT) is presented in equation A.3.

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\lambda}\right) \qquad \lambda = \delta + \frac{V_{UT} - \frac{3}{4}V_{C,EOS}}{V_{UT}\delta + V_{C,EOS}} \qquad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T \tag{A.3}$$

Equation A.3 is valid for any two parameter cubic equation of state. Two parameter cubic equations of state have constant values for the critical compressibility (i.e., the EOS predicted critical volume $V_{C,EOS}$ can be expressed in terms of the other two EOS parameters). When the SRK EOS model (i.e., equations A.4 and A.5) is used as the untranslated relationship, lambda and delta from equation A.2 are expressed as presented in equation A.6.

$$f_1 = \frac{RT}{V_{UT} - b} - \frac{a \cdot \alpha}{V_{UT}(V_{UT} + b)} \qquad a = \frac{R^2 T_C^2}{9\left(\sqrt[3]{2} - 1\right)P_C} \qquad b = \frac{\left(\sqrt[3]{2} - 1\right)RT_C}{3P_C} \tag{A.4}$$

$$\alpha = \left(1 + \left(0.480 + 1.574\omega - 0.176\omega^2\right)\left(1 - \sqrt{T_r}\right)\right)^2 \tag{A.5}$$

$$f_2 = c_0 + c_1\left(\frac{1}{1+2\lambda}\right) \qquad \lambda = \delta + \frac{V_{UT} - \frac{3b}{4\left(\sqrt[3]{2} - 1\right)}}{V_{UT}\delta + \frac{b}{\left(\sqrt[3]{2} - 1\right)}} \qquad \delta = \frac{V_{UT}^2}{(V_{UT} - b)^2} - \frac{2aV_{UT} + ab}{RT(V_{UT} + b)^2} \tag{A.6}$$

A different alpha function may be substituted for the relation presented in equation A.5 without affecting the estimation accuracy of the volume translation function. The original SRK alpha function is presented here for simplicity.

The constant volume translation parameter, c_0, is calculated from equation A.7.

$$c_0 = \frac{RT_C}{3P_C} - V_C \tag{A.7}$$

Recommended values for implementing the DMT function applied to the SRK EOS are presented in table A.1 below.

Table A.1: Pure component parameters used when applying the DMT to the SRK EOS.

Fluid	T_C (K)	P_C (bar)	V_C (L/kmol)	ω	c_1 (L/kmol)
CH_4	190.56	45.99	99	0.011	16.5
C_2H_6	305.33	48.72	146	0.105	26.5
C_3H_8	369.83	42.47	202	0.152	36.5
n-C_4H_{10}	425.13	37.96	255	0.211	50.0
n-C_5H_{12}	469.70	33.70	311	0.251	65.5
n-C_6H_{14}	507.82	30.34	370	0.299	80.5
n-C_7H_{16}	540.13	27.36	432	0.349	95.5
n-C_8H_{18}	569.32	24.97	486	0.393	120.
N_2	126.19	33.96	89	0.039	14.0
O_2	154.57	50.43	73	0.021	11.5
Ar	150.69	48.63	75	-0.002	12.0
H_2S	373.10	90.00	98	0.100	16.0
CO_2	304.13	73.77	94	0.225	18.0
CF_4	227.51	37.50	141	0.179	27.5
H_2	33.15	12.96	71	-0.219	0
$OC(CH_3)_2$	508.10	47.01	209	0.307	72.0
NH_3	405.40	113.3	76	0.256	17.0
H_2O	647.10	220.6	56	0.344	19.0

The value used for the volume translation parameter c_1 should be determined by matching DMT predictions to available liquid phase data. When such data is not available, the correlation presented in equation A.8 can provide an estimate of this value.

$$c_1\left[\frac{L}{kmol}\right] = 0.185\frac{RT_C}{3P_C} - 4 \tag{A.8}$$

This correlation is appropriate for non-polar and non-associating fluids, but tends to underestimate appropriate values for fluids such as water, acetone, or ammonia. For example; the value estimate by equation A.8 for acetone is 52 L/kmol, while the recommended value (as determined from the specific gravity of acetone) is 72 L/kmol.

Underestimating the appropriate value for the translation parameter c_1 results in underestimating the molar volume (i.e., overestimating density) at low temperatures (T_r < 0.7) in the liquid phase. Near critical and vapor phase densities are not affected by underestimating the translation parameter c_1.

Mixture values for the two untranslated EOS parameters *a* and *b* may determined using whatever mixing rule is desired. The mixture values for the parameters *a* and *b* in the untranslated EOS should be used for those parameters when they appear in the translation function.

Mixture values for the two parameters unique to the translation function are calculated as indicated in equation A.9.

$$c_0 = \sum_j x_j c_{0j} \qquad c_1 = \sum_j x_j c_{1j} \tag{A.9}$$

No binary interaction parameters are used or needed when determining mixture values for the two volume translation parameters.

Appendix A.2 - DMT Function Implementation for Fugacity

Using the DMT function to calculate properties other than the molar volume is not recommended. Translated estimates of dew and bubble point pressures are not more accurate than their untranslated equivalents. Phase envelope predictions are determined almost exclusively by the selection of the mixing rules for the untranslated EOS parameters.

If the DMT function is to be used in equilibrium calculations, additional modifications are required. The expression for delta from equation A.3 must incorporate the temperature dependent function, tau.

$$\delta_{old} = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T \qquad \delta = \frac{-V_{UT}^2}{RT}\left(\frac{\partial P}{\partial V_{UT}}\right)_T + \tau \tag{A.10}$$

For the DMT function applied to the SRK EOS, the modified delta is given in equation A.11, and the expression for tau is given in equation A.12.

$$\delta = \frac{V_{UT}^2}{(V_{UT}-b)^2} - \frac{2aV_{UT}+ab}{RT(V_{UT}+b)^2} + \tau \tag{A.11}$$

$$\tau(T_r \le 1) = \exp\left(5\left(1-\sqrt{T_r}\right)\right) - 1$$
$$\tau(T_r > 1) = 0 \tag{A.12}$$

The expression for tau was selected so that the modified delta function in equation A.11 greater than or equal to zero for all possible values of V_{UT} and T. When using the DMT function to calculate mixture fugacities, an estimate of the mixture critical temperature is necessary to determine the value of T_r that appears in the expression for tau. This mixture critical temperature estimate may be determined from equation A.13.

$$T_{C,tau} = \sum_j x_j T_{Cj} \tag{A.13}$$

Equation A.13 should only be used for determining an estimate of the mixture critical temperature for use in the tau function; it is not equivalent to the mixture critical temperature predicted by either the translated or untranslated EOS.

The alpha function in the untranslated EOS must also be changed to account for the influence of the volume dependency in the translation function on the fugacity. A recommended expression for the revised alpha function in the untranslated EOS is presented in equation A.14.

$$\alpha = \exp\left(\kappa_1\left(1-T_r^{\kappa_2}\right)\right) \tag{A.14}$$

The alpha function in equation A.14 is used in place of the alpha function in equation A.5 when calculating fugacities from the SRK EOS when translated by the DMT function. The two kappa parameters in equation A.14 are determined by matching the translated EOS predicted pure component vapor pressures with reference values for the pure component vapor pressure (e.g., Antoine parameters). Table A.2 provides recommended values of these kappa parameters for several fluids.

Table A.2: Pure component parameters for kappa; used when applying the DMT to the SRK EOS and calculating fugacity from the translated EOS.

Fluid	κ_1	κ_2	Fluid	κ_1	κ_2
CH_4	0.5091	1.1840	O_2	0.5921	1.0040
C_2H_6	0.7481	0.9560	Ar	0.5481	1.0200
C_3H_8	0.8980	0.8628	H_2S	0.6955	1.0580
n-C_4H_{10}	0.8890	0.9915	CO_2	0.9391	0.9895
n-C_5H_{12}	0.9798	0.9761	CF_4	0.7996	1.0930
n-C_6H_{14}	0.9988	0.9761	H_2	0	0
n-C_7H_{16}	1.0890	1.0210	$OC(CH_3)_2$	0.8619	1.372
n-C_8H_{18}	1.1070	1.0900	NH_3	0.8086	1.2670
N_2	0.5883	1.0730	H_2O	0.8433	1.4760

Correlations for the kappa parameters presented in table A.2 are not available. Default values of unity should be used if no pure component vapor pressure data are available.

The fugacity coefficient is calculated according to equation A.15. In this study, numerical differentiation was used to evaluate the partial differential with respect to molar extent that appears in equation A.15.

$$\ln\left(\hat{\phi}_i\right) = \frac{\partial}{\partial N_j}\left(\frac{\underline{A}^{res}}{RT}\right) - \ln\left(\frac{P\underline{V}}{NRT}\right) \tag{A.15}$$

The differential in equation A.15 was evaluated numerically, which allowed rapid substitution of different mixing rules without reformulating the expression for mixture fugacity. The method of central differences was used, as indicated in equation A.16.

$$\ln\left(\hat{\phi}_i\right) = \left(\frac{\underline{A}^{res}\Big|_{N_j=N_j+1e-6} - \underline{A}^{res}\Big|_{N_j=N_j-1e-6}}{2RT\cdot 10^{-6}}\right) - \ln\left(\frac{P\underline{V}}{NRT}\right) \tag{A.16}$$

Sufficient precision is used during calculations that this approximation is not a source of error. For some values of N_j, the expression N_j - $1 \cdot 10^{-6}$ may become negative. Negative molar extents are not physically realistic, but these values can be used in thermodynamic models without loss of

The total volume used in equations A.15 and A.16 is the translated volume. An expression for the residual Helmholtz free energy is presented in equation A.17.

$$\frac{\underline{A}^{res}}{RT} = N\ln\left(\frac{\underline{V}_{UT}}{\underline{V}_{UT} - bN}\right) + \frac{a\alpha N}{RTb}\ln\left(\frac{\underline{V}_{UT}}{\underline{V}_{UT} - bN}\right) - N\ln\left(\frac{\underline{V}}{\underline{V}_{UT}}\right) + \int_{\underline{V}_{UT}}^{\infty} \frac{p_{16}(\underline{V}_{UT})}{q_{19}(\underline{V}_{UT})} \partial \underline{V}_{UT} \tag{A.17}$$

The first two terms on the right hand side or equation A.17 incorporate the contribution of the untranslated EOS toward the residual Helmholtz free energy. These two terms dominate at most conditions; the remaining terms contribute significantly in the critical region. The integral in the fourth term involves a sixteenth and nineteenth order polynomial in the untranslated volume. The nineteenth order polynomial q_{19} has no constant term and can be expressed as indicated in equation A.18.

$$q_{19} = \underline{V}_{UT}\left(Q_9\underline{V}_{UT}^9 + Q_8\underline{V}_{UT}^8 + \ldots + Q_1\underline{V}_{UT} + Q_0\right) \tag{A.18}$$

All of the solutions to the expression $q_{19} = 0$ should be calculated using standard root finding methods. One root occurs at $V_{UT} = 0$, the other nine unique roots are denoted R_n where n indicates the index of the root.

The sixteenth order polynomial cannot be simplified. It is not necessary to calculate the roots of this polynomial.

$$p_{16} = P_{16}\underline{V}_{UT}^{16} + P_{15}\underline{V}_{UT}^{15} + \ldots + P_1\underline{V}_{UT} + P_0 \tag{A.19}$$

It is convenient to represent these polynomials in vector form as in equation A.20.

$$\boldsymbol{P} = \begin{bmatrix} P_{16} \\ P_{15} \\ \vdots \\ P_1 \\ P_0 \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} Q_9 \\ Q_8 \\ \vdots \\ Q_1 \\ Q_0 \end{bmatrix} \tag{A.20}$$

Both $\boldsymbol{P}$ and $\boldsymbol{Q}$ are understood to represent polynomials. Standard polynomial operations apply to these values. For example, $\boldsymbol{P'}$ has only sixteen non-zero elements and represents a fifteenth order polynomial, while $\boldsymbol{Q}^2$ has nineteen non-zero elements and represents an eighteenth order polynomial.

The expression $\boldsymbol{Q}(R_n)$ indicates the polynomial $\boldsymbol{Q}$ evaluated at the value R_n. The expression $\boldsymbol{Q}(R_n)$ should equal zero because of the definition of R_n.

$$\boldsymbol{P'} = \begin{bmatrix} 16P_{16} \\ 15P_{15} \\ \vdots \\ P_1 \\ 0 \end{bmatrix} \qquad \boldsymbol{Q^2} = \begin{bmatrix} Q_9^2 \\ 2Q_9Q_8 \\ \vdots \\ 2Q_1Q_0 \\ Q_0^2 \end{bmatrix} \qquad \boldsymbol{Q}(R_n) = 0 \tag{A.21}$$

Using the method of partial fractions, the integral from equation A.17 can be evaluated as indicated in equation A.22.

$$\int_{\underline{V}_{UT}}^{\infty} \frac{p_{16}(\underline{V}_{UT})}{q_{19}(\underline{V}_{UT})} \partial \underline{V}_{UT} = NA_1 \ln\left(\frac{\underline{V}_{UT}}{N}\right) - \sum_{n=1}^{9}\left(\frac{NB_n}{\underline{V}_{UT} - R_n}\right) + \sum_{n=1}^{9} NC_n \ln(\underline{V}_{UT} - R_n) \tag{A.22}$$

The values for A_1, B_n, and C_n can be calculated from the vectors of polynomial coefficients $\boldsymbol{P}$ and $\boldsymbol{Q}$.

$$A_1 = \frac{P_0}{Q_0^2} \tag{A.23}$$

Using the value for A_1 from equation A.23, an intermediate polynomial expression $\boldsymbol{S}$ can be defined.

$$\boldsymbol{S} = \frac{\boldsymbol{P} - A_1\boldsymbol{Q^2}}{\underline{V}_{UT}} \tag{A.24}$$

The expression $\boldsymbol{P} - A_1\boldsymbol{Q^2}$ represents an eighteenth order polynomial in the untranslated volume; however, calculating A_1 according to equation A.23 ensures that there is no constant term in this expression.
The division by $\underline{V}_{UT}$ in equation A.24 creates a seventeenth order polynomial in the untranslated volume.

$$B_n = \frac{\boldsymbol{S}(R_n)}{\left(\boldsymbol{Q'}(R_n)\right)^2} \tag{A.25}$$

$$C_n = \frac{\boldsymbol{S'}(R_n)}{\boldsymbol{Q'}(R_n)} - B_n \frac{\boldsymbol{Q''}(R_n)}{\boldsymbol{Q'}(R_n)} \tag{A.26}$$

Expressions for the two polynomials ***P*** and ***Q*** are given below. The value *Vc_eff* is shorthand notation for the untranslated EOS parameter *b* divided by the cube root of two minus one, as indicated in equation A.6.

P_{16} = 2*C1*R^3*T^3*(-7*T*R*Vc_eff + 16*T*R*b*tau + 8*T*R*b*tau^2 - 3*T*R*tau*Vc_eff - 16*a*tau - 8*a*tau^2)

P_{15} = 2*C1*R^2*T^2*(32*R^2*T^2*tau^2*b^2 + 9*T^2*R^2*b*Vc_eff + 96*R^2*T^2*tau*b^2 + 13*T^2*R^2*b*tau*Vc_eff + 32*R^2*T^2*b^2 - 32*R*T*a*b*tau - 16*T*R*a*tau*Vc_eff + 7*T*R*(a*alpha)*Vc_eff - 64*R*T*a*b + 3*T*R*(a*alpha)*tau*Vc_eff - 16*T*R*a*Vc_eff - 8*(a*alpha)*R*T*b*tau^2 - 16*(a*alpha)*R*T*b*tau + 16*R*T*a*b*tau^2 + 32*a^2 + 16*(a*alpha)*a*tau + 8*(a*alpha)*a*tau^2 + 32*a^2*tau)

P_{14} = 2*C1*R*T*(64*T*R*(a*alpha)*a*b + 16*T*R*(a*alpha)*a*tau*Vc_eff - 32*a^3 + 208*R^3*T^3*b^3 - 8*R^2*T^2*a*Vc_eff^2 - 160*R^2*T^2*b^2*a + 32*R^2*T^2*b*a*tau*Vc_eff + 8*R^3*T^3*b*Vc_eff^2 + 85*R^3*T^3*tau*Vc_eff*b^2 + 32*R^2*T^2*b^2*a*tau^2 - 23*R^2*T^2*b*a*Vc_eff + 32*R*T*a^2*Vc_eff + 208*R^3*T^3*b^3*tau - 112*R*T*a^2*b*tau + 16*R^3*T^3*b^3*tau^2 - 16*R*T*a^2*b + 154*R^3*T^3*Vc_eff*b^2 - 32*(a*alpha)*a^2 - 32*T^2*R^2*(a*alpha)*b^2 - 19*T^2*R^2*(a*alpha)*b*tau*Vc_eff - 16*T^2*R^2*(a*alpha)*tau^2*b^2 - 23*T^2*R^2*(a*alpha)*b*Vc_eff - 64*T^2*R^2*(a*alpha)*tau*b^2 - 32*T*R*(a*alpha)*a*b*tau^2 + 16*T*R*(a*alpha)*a*Vc_eff - 32*(a*alpha)*a^2*tau)

P_{13} = -2*C1*(80*R^3*T^3*a*b^3*tau^2 - 16*(a*alpha)*b^2*a*tau^2*R^2*T^2 - 32*(a*alpha)*b^2*R^2*T^2*a + 8*(a*alpha)*R^3*T^3*b*Vc_eff^2 - 8*(a*alpha)*a*Vc_eff^2*R^2*T^2 + 53*(a*alpha)*R^3*T^3*tau*Vc_eff*b^2 + 48*(a*alpha)*R^3*T^3*b^3*tau - 32*(a*alpha)*R^3*T^3*b^3*tau^2 + 32*(a*alpha)*a^2*Vc_eff*R*T - 290*R^4*T^4*b^3*Vc_eff + 122*(a*alpha)*R^3*T^3*Vc_eff*b^2 - 16*R^3*T^3*b*a*Vc_eff^2 + 15*R^3*T^3*a*Vc_eff*b^2 - 64*R^3*T^3*a*b^3*tau - 32*R^2*T^2*a^2*tau*b^2 - 64*R^3*T^3*a*tau*Vc_eff*b^2 - 160*R*T*a^3*b + 144*(a*alpha)*R^3*T^3*b^3 + 32*(a*alpha)*b^2*R^2*T^2*a*tau + 64*R^3*T^3*a*b^3 + 64*R^2*T^2*b^2*a^2 - 176*(a*alpha)*a^2*b*tau*R*T - 80*(a*alpha)*R*T*a^2*b + 9*(a*alpha)*b*R^2*T^2*a*Vc_eff - 544*R^4*T^4*b^4 + 96*R^4*T^4*tau^2*b^4 + 64*(a*alpha)*b*a*tau*R^2*T^2*Vc_eff - 128*R^4*T^4*b^4*tau - 32*(a*alpha)*a^3 - 53*R^4*T^4*b^3*tau*Vc_eff - 32*R^4*T^4*b^2*Vc_eff^2 + 112*R^2*T^2*a^2*b*Vc_eff)

P_{12} = -2*C1*b*(40*R^3*T^3*a*b^3*tau^2 - 96*(a*alpha)*b^2*a*tau^2*R^2*T^2 + 128*(a*alpha)*b^2*R^2*T^2*a + 16*(a*alpha)*R^3*T^3*b*Vc_eff^2 + 32*(a*alpha)*a*Vc_eff^2*R^2*T^2 - 85*(a*alpha)*R^3*T^3*tau*Vc_eff*b^2 - 128*(a*alpha)*R^3*T^3*b^3*tau - 80*(a*alpha)*R^3*T^3*b^3*tau^2 - 176*(a*alpha)*a^2*Vc_eff*R*T + 13*R^4*T^4*b^3*Vc_eff + 14*(a*alpha)*R^3*T^3*Vc_eff*b^2 - 32*R^3*T^3*b*a*Vc_eff^2 - 22*R^3*T^3*a*Vc_eff*b^2 - 144*R^3*T^3*a*b^3*tau - 304*R^2*T^2*a^2*tau*b^2 + 160*R^3*T^3*a*tau*Vc_eff*b^2 + 264*R*T*a^3*b + 192*(a*alpha)*R^3*T^3*b^3 + 32*(a*alpha)*b^2*R^2*T^2*a*tau - 104*R^3*T^3*a*b^3 - 40*R^2*T^2*b^2*a^2 + 320*(a*alpha)*a^2*b*tau*R*T + 32*(a*alpha)*R*T*a^2*b - (a*alpha)*b*R^2*T^2*a*Vc_eff - 696*R^4*T^4*b^4 + 136*R^4*T^4*tau^2*b^4 - 32*(a*alpha)*b*a*tau*R^2*T^2*Vc_eff + 224*R^4*T^4*b^4*tau + 224*(a*alpha)*a^3 + 255*R^4*T^4*b^3*tau*Vc_eff - 16*R^4*T^4*b^2*Vc_eff^2 - 32*R^2*T^2*a^2*b*Vc_eff)

P_{11} = 2*C1*b^2*(160*R^3*T^3*a*b^3*tau^2 - 136*(a*alpha)*b^2*a*tau^2*R^2*T^2 -
40*(a*alpha)*b^2*R^2*T^2*a + 32*(a*alpha)*R^3*T^3*b*Vc_eff^2 +
16*(a*alpha)*a*Vc_eff^2*R^2*T^2 + 223*(a*alpha)*R^3*T^3*tau*Vc_eff*b^2 +
224*(a*alpha)*R^3*T^3*b^3*tau - 40*(a*alpha)*R^3*T^3*b^3*tau^2 - 320*(a*alpha)*a^2*Vc_eff*R*T
- 485*R^4*T^4*b^3*Vc_eff + 317*(a*alpha)*R^3*T^3*Vc_eff*b^2 - 80*R^3*T^3*b*a*Vc_eff^2 +
222*R^3*T^3*a*Vc_eff*b^2 + 64*R^3*T^3*a*b^3*tau - 320*R^2*T^2*a^2*tau*b^2 -
80*R^3*T^3*a*tau*Vc_eff*b^2 + 48*R*T*a^3*b + 40*(a*alpha)*R^3*T^3*b^3 -
48*(a*alpha)*b^2*R^2*T^2*a*tau + 176*R^3*T^3*a*b^3 - 32*R^2*T^2*b^2*a^2 +
48*(a*alpha)*a^2*b*tau*R*T - 72*(a*alpha)*R*T*a^2*b - 38*(a*alpha)*b*R^2*T^2*a*Vc_eff +
384*R^4*T^4*b^4 + 64*R^4*T^4*tau^2*b^4 + 192*(a*alpha)*b*a*tau*R^2*T^2*Vc_eff -
448*R^4*T^4*b^4*tau + 648*(a*alpha)*a^3 - 335*R^4*T^4*b^3*tau*Vc_eff -
96*R^4*T^4*b^2*Vc_eff^2 + 304*R^2*T^2*a^2*b*Vc_eff)

P_{10} = 2*C1*b^3*(-64*(a*alpha)*b^2*a*tau^2*R^2*T^2 - 32*(a*alpha)*b^2*R^2*T^2*a +
80*(a*alpha)*R^3*T^3*b*Vc_eff^2 + 96*(a*alpha)*a*Vc_eff^2*R^2*T^2 -
143*(a*alpha)*R^3*T^3*tau*Vc_eff*b^2 - 160*(a*alpha)*R^3*T^3*b^3*tau^2 +
48*(a*alpha)*a^2*Vc_eff*R*T - 320*R^4*T^4*b^3*Vc_eff + 155*(a*alpha)*R^3*T^3*Vc_eff*b^2 -
40*R^3*T^3*b*a*Vc_eff^2 + 139*R^3*T^3*a*Vc_eff*b^2 - 256*R^3*T^3*a*b^3*tau -
224*R^2*T^2*a^2*tau*b^2 + 320*R^3*T^3*a*tau*Vc_eff*b^2 + 336*R*T*a^3*b +
272*(a*alpha)*R^3*T^3*b^3 + 128*(a*alpha)*b^2*R^2*T^2*a*tau + 272*R^3*T^3*a*b^3 -
32*R^2*T^2*b^2*a^2 + 576*(a*alpha)*a^2*b*tau*R*T + 144*(a*alpha)*R*T*a^2*b -
162*(a*alpha)*b*R^2*T^2*a*Vc_eff + 224*R^4*T^4*tau^2*b^4 -
272*(a*alpha)*b*a*tau*R^2*T^2*Vc_eff - 224*R^4*T^4*b^4*tau - 960*(a*alpha)*a^3 +
233*R^4*T^4*b^3*tau*Vc_eff - 136*R^4*T^4*b^2*Vc_eff^2 - 320*R^2*T^2*a^2*b*Vc_eff)

P_9 = 2*C1*b^4*(-40*b*(a*alpha)*R^3*T^3*Vc_eff^2 + 245*b*(a*alpha)*R^2*T^2*a*Vc_eff +
112*b^3*(a*alpha)*R^3*T^3 + 224*b^2*(a*alpha)*a*tau^2*R^2*T^2 - 672*b*(a*alpha)*a^2*tau*R*T -
425*b^2*(a*alpha)*R^3*T^3*tau*Vc_eff + 128*b^4*R^4*T^4*tau + 553*b^3*R^4*T^4*tau*Vc_eff +
448*b^2*R^2*T^2*a^2*tau - 256*b^3*R^3*T^3*a*tau - 224*b*R^2*T^2*a^2*Vc_eff +
160*b*R^3*T^3*a*Vc_eff^2 - 421*b^2*R^3*T^3*a*Vc_eff + 672*(a*alpha)*a^3 +
144*b^3*R^4*T^4*Vc_eff + 64*b^2*R^4*T^4*Vc_eff^2 - 336*b*R*T*a^3 - 160*b^3*R^3*T^3*a*tau^2
+ 576*(a*alpha)*a^2*Vc_eff*R*T + 64*b^4*R^4*T^4*tau^2 + 160*b^2*R^2*T^2*a^2 -
136*(a*alpha)*a*Vc_eff^2*R^2*T^2 + 112*b^3*R^3*T^3*a - 224*b^3*(a*alpha)*R^3*T^3*tau -
144*b*(a*alpha)*R*T*a^2 - 128*b*(a*alpha)*a*tau*R^2*T^2*Vc_eff -
320*b^2*(a*alpha)*R^3*T^3*Vc_eff - 64*b^2*(a*alpha)*R^2*T^2*a +
192*b^2*(a*alpha)*R^2*T^2*a*tau)

P_8 = -2*C1*b^5*R*T*(-111*T^3*R^3*b^3*Vc_eff + 136*T^3*R^3*b^4*tau^2 - 224*T^3*R^3*b^2*Vc_eff^2 -
23*T^3*R^3*b^3*tau*Vc_eff + 128*T^2*R^2*b^3*(a*alpha)*tau - 160*T^2*R^2*b^3*(a*alpha)*tau^2
+ 160*T^2*R^2*b*(a*alpha)*Vc_eff^2 + 112*T^2*R^2*b^3*(a*alpha) + 224*T^2*R^2*b^3*a -
16*T*R*b^2*a^2 - 105*T^2*R^2*b^2*(a*alpha)*tau*Vc_eff + 316*T^2*R^2*b^2*a*Vc_eff -
144*T^2*R^2*b^3*a*tau + 144*T^2*R^2*b^2*(a*alpha)*Vc_eff - 40*T^2*R^2*b^3*a*tau^2 +
320*T^2*R^2*b^2*a*tau*Vc_eff + 64*T*R*(a*alpha)*a*Vc_eff^2 + 64*T*R*b^2*(a*alpha)*a*tau^2 +
192*T*R*b^2*(a*alpha)*a*tau - 448*T*R*b*a^2*Vc_eff - 448*T*R*b*(a*alpha)*a*tau*Vc_eff -
224*T*R*b^2*(a*alpha)*a + 32*T*R*b^2*a^2*tau - 315*T*R*b*(a*alpha)*a*Vc_eff -
208*T^3*R^3*b^4*tau - 144*T^3*R^3*b^4 + 48*b*(a*alpha)*a^2 + 672*(a*alpha)*a^2*Vc_eff)

P_7 = -2*C1*b^6*(-480*b*(a*alpha)*a^2*tau*R*T - 80*b^2*R^3*T^3*a*tau*Vc_eff +
420*b*(a*alpha)*R^2*T^2*a*Vc_eff + 208*b^2*(a*alpha)*R^2*T^2*a*tau -
425*b^2*(a*alpha)*R^3*T^3*tau*Vc_eff + 136*b^2*(a*alpha)*a*tau^2*R^2*T^2 -
96*b^4*R^4*T^4*tau + 96*b^4*R^4*T^4*tau^2 - 96*b^4*R^4*T^4 - 48*b^3*(a*alpha)*R^3*T^3*tau -
64*b^2*R^4*T^4*Vc_eff^2 - 144*b*R*T*a^3 + 336*(a*alpha)*a^3 + 49*b^3*R^4*T^4*Vc_eff +
160*b*R^3*T^3*a*Vc_eff^2 - 324*b^2*R^3*T^3*a*Vc_eff - 224*(a*alpha)*a*Vc_eff^2*R^2*T^2 -
192*b*(a*alpha)*R*T*a^2 + 32*b^3*(a*alpha)*R^3*T^3 + 176*b^3*R^3*T^3*a +
128*b^2*R^2*T^2*a^2 - 80*b^3*R^3*T^3*a*tau^2 + 128*b*(a*alpha)*a*tau*R^2*T^2*Vc_eff -
224*b^3*R^3*T^3*a*tau - 177*b^2*(a*alpha)*R^3*T^3*Vc_eff - 40*b^3*(a*alpha)*R^3*T^3*tau^2 -
112*b^2*(a*alpha)*R^2*T^2*a + 377*b^3*R^4*T^4*tau*Vc_eff + 32*b*R^2*T^2*a^2*Vc_eff +
224*b^2*R^2*T^2*a^2*tau)

P_6 = 2*C1*b^7*(-288*b*(a*alpha)*a^2*tau*R*T + 160*b^2*R^3*T^3*a*tau*Vc_eff -
220*b*(a*alpha)*R^2*T^2*a*Vc_eff + 128*b^2*(a*alpha)*R^2*T^2*a*tau -
25*b^2*(a*alpha)*R^3*T^3*tau*Vc_eff + 96*b^2*(a*alpha)*a*tau^2*R^2*T^2 + 16*b^4*R^4*T^4*tau
+ 16*b^4*R^4*T^4*tau^2 - 16*b^4*R^4*T^4 + 64*b^3*(a*alpha)*R^3*T^3*tau -
136*b^2*R^4*T^4*Vc_eff^2 - 48*b*R*T*a^3 + 192*(a*alpha)*a^3 + 480*(a*alpha)*a^2*Vc_eff*R*T +
66*b^3*R^4*T^4*Vc_eff + 40*b*R^3*T^3*a*Vc_eff^2 + 239*b^2*R^3*T^3*a*Vc_eff -
64*(a*alpha)*a*Vc_eff^2*R^2*T^2 - 48*b*(a*alpha)*R*T*a^2 + 80*b^3*(a*alpha)*R^3*T^3 +
16*b^3*R^3*T^3*a - 16*b^2*R^2*T^2*a^2 - 32*b^3*R^3*T^3*a*tau^2 -
272*b*(a*alpha)*a*tau*R^2*T^2*Vc_eff - 17*b^2*(a*alpha)*R^3*T^3*Vc_eff -
80*b^3*(a*alpha)*R^3*T^3*tau^2 - 160*b^2*(a*alpha)*R^2*T^2*a - 129*b^3*R^4*T^4*tau*Vc_eff -
224*b*R^2*T^2*a^2*Vc_eff + 160*b*(a*alpha)*R^3*T^3*Vc_eff^2 + 80*b^2*R^2*T^2*a^2*tau)

P_5 = 2*C1*b^8*R*T*(-16*b*(a*alpha)*a^2*tau + 106*T^3*R^3*b^3*Vc_eff + 32*T^3*R^3*b^4*tau^2 -
96*T^3*R^3*b^2*Vc_eff^2 - 16*b*a^3 + 95*T^3*R^3*b^3*tau*Vc_eff +
16*T^2*R^2*b^3*(a*alpha)*tau - 32*T^2*R^2*b^3*(a*alpha)*tau^2 +
40*T^2*R^2*b*(a*alpha)*Vc_eff^2 + 32*T^2*R^2*b^3*(a*alpha) + 16*T^2*R^2*b^3*a +
32*T*R*b^2*a^2 + 80*T^2*R^2*b*a*Vc_eff^2 - 223*T^2*R^2*b^2*(a*alpha)*tau*Vc_eff -
105*T^2*R^2*b^2*a*Vc_eff - 64*T^2*R^2*b^3*a*tau - 98*T^2*R^2*b^2*(a*alpha)*Vc_eff -
16*T^2*R^2*b^3*a*tau^2 - 64*T^2*R^2*b^2*a*tau*Vc_eff - 136*T*R*(a*alpha)*a*Vc_eff^2 +
16*T*R*b^2*(a*alpha)*a*tau^2 + 96*T*R*b^2*(a*alpha)*a*tau + 80*T*R*b*a^2*Vc_eff +
192*T*R*b*(a*alpha)*a*tau*Vc_eff - 32*T*R*b^2*(a*alpha)*a + 32*T*R*b^2*a^2*tau +
305*T*R*b*(a*alpha)*a*Vc_eff - 32*T^3*R^3*b^4 - 64*b*(a*alpha)*a^2 - 288*(a*alpha)*a^2*Vc_eff)

P_4 = 2*C1*b^9*(64*b*(a*alpha)*a^2*tau*R*T - 32*b^2*R^3*T^3*a*tau*Vc_eff +
39*b*(a*alpha)*R^2*T^2*a*Vc_eff - 32*b^2*(a*alpha)*R^2*T^2*a*tau -
b^2*(a*alpha)*R^3*T^3*tau*Vc_eff - 32*b^2*(a*alpha)*a*tau^2*R^2*T^2 + 8*b^4*R^4*T^4*tau^2 -
8*b^4*R^4*T^4 + 16*b^2*R^4*T^4*Vc_eff^2 + 8*b*R*T*a^3 - 32*(a*alpha)*a^3 -
16*(a*alpha)*a^2*Vc_eff*R*T + 5*b^3*R^4*T^4*Vc_eff - 32*b*R^3*T^3*a*Vc_eff^2 -
58*b^2*R^3*T^3*a*Vc_eff + 96*(a*alpha)*a*Vc_eff^2*R^2*T^2 + 16*b*(a*alpha)*R*T*a^2 -
16*b^3*(a*alpha)*R^3*T^3 - 8*b^3*R^3*T^3*a + 8*b^2*R^2*T^2*a^2 + 8*b^3*R^3*T^3*a*tau^2 +
32*b*(a*alpha)*a*tau*R^2*T^2*Vc_eff - 16*b^3*R^3*T^3*a*tau + 58*b^2*(a*alpha)*R^3*T^3*Vc_eff
+ 16*b^3*(a*alpha)*R^3*T^3*tau^2 + 32*b^2*(a*alpha)*R^2*T^2*a + 43*b^3*R^4*T^4*tau*Vc_eff +
32*b*R^2*T^2*a^2*Vc_eff - 80*b*(a*alpha)*R^3*T^3*Vc_eff^2 - 16*b^2*R^2*T^2*a^2*tau)

P_3 = -2*C1*b^10*(-64*(a*alpha)*a^2*Vc_eff*R*T - 8*(a*alpha)*a^3 + 16*R^3*T^3*b*a*Vc_eff^2 + 19*R^4*T^4*b^3*Vc_eff - 8*(a*alpha)*R*T*a^2*b - 16*R^3*T^3*a*tau*Vc_eff*b^2 - 43*(a*alpha)*R^3*T^3*Vc_eff*b^2 + 16*R^2*T^2*a^2*b*Vc_eff + 86*(a*alpha)*b*R^2*T^2*a*Vc_eff - 8*(a*alpha)*b^2*a*tau^2*R^2*T^2 + 16*(a*alpha)*a^2*b*tau*R*T + 16*(a*alpha)*b^2*R^2*T^2*a*tau - 14*R^3*T^3*a*Vc_eff*b^2 - 16*(a*alpha)*a*Vc_eff^2*R^2*T^2 + 8*(a*alpha)*b^2*R^2*T^2*a + 64*(a*alpha)*b*a*tau*R^2*T^2*Vc_eff - 8*(a*alpha)*R^3*T^3*b^3*tau^2 - 32*R^4*T^4*b^2*Vc_eff^2 + 5*R^4*T^4*b^3*tau*Vc_eff + 8*(a*alpha)*R^3*T^3*b^3 - 53*(a*alpha)*R^3*T^3*tau*Vc_eff*b^2 + 32*(a*alpha)*R^3*T^3*b*Vc_eff^2)

P_2 = 2*C1*b^11*Vc_eff*R*T*(3*R^3*T^3*b^3*tau + 4*R^3*T^3*b^3 + 8*R^3*T^3*b^2*Vc_eff + 16*T^2*R^2*Vc_eff*b*(a*alpha) - 3*R^2*T^2*a*b^2 - 5*T^2*R^2*b^2*(a*alpha)*tau - 19*b^2*(a*alpha)*R^2*T^2 + 8*R^2*T^2*a*Vc_eff*b - 32*T*R*Vc_eff*(a*alpha)*a + 14*b*(a*alpha)*a*R*T + 16*T*R*b*(a*alpha)*a*tau - 16*(a*alpha)*a^2)

P_1 = 2*C1*b^12*Vc_eff*R^2*T^2*(4*b^3*R^2*T^2 + 3*T^2*R^2*b^3*tau - 3*T*R*b^2*a - 3*T*R*b^2*(a*alpha)*tau + 8*T*R*Vc_eff*b*(a*alpha) - 4*b^2*(a*alpha)*R*T + 3*b*(a*alpha)*a + 8*Vc_eff*(a*alpha)*a)

P_0 = 2*C1*b^14*(a*alpha)*R^2*T^2*Vc_eff*(4*T*R*b + 3*T*R*b*tau - 3*a)

Q_9 = 10*R^2*T^2*tau + 4*R^2*T^2*tau^2 + 10*R^2*T^2

Q_8 = 16*tau*b*R^2*T^2 + 3*R^2*T^2*Vc_eff + 20*b*R^2*T^2 + 4*tau*R^2*T^2*Vc_eff - 20*a*R*T - 16*a*tau*R*T

Q_7 = -8*a*R*T*Vc_eff + 16*a^2-2*a*b*R*T + 8*b*R^2*T^2*Vc_eff - 16*tau^2*b^2*R^2*T^2 + 6*b^2*R^2*T^2 - 16*tau*b^2*R^2*T^2+24*a*tau*b*R*T

Q_6 = -48*b*a^2 + 32*a*tau*b^2*R*T + 12*a*b*R*T*Vc_eff - 16*tau*b^2*R^2*T^2*Vc_eff + 40*a*b^2*R*T + 8*b^3*R^2*T^2 - 32*tau*b^3*R^2*T^2

Q_5 = 36*b^2*a^2 + 26*b^4*R^2*T^2 - 56*a*tau*b^3*R*T + 4*tau*b^4*R^2*T^2 + 2*a*b^3*R*T + 24*tau^2*b^4*R^2*T^2 - 16*b^3*R^2*T^2*Vc_eff + 16*a*b^2*R*T*Vc_eff

Q_4 = -28*b^3*a*R*T*Vc_eff - 20*a*b^4*R*T + 24*tau*b^4*R^2*T^2*Vc_eff - 10*b^4*R^2*T^2*Vc_eff + 4*b^5*R^2*T^2 + 16*b^3*a^2 + 16*tau*b^5*R^2*T^2 - 16*a*tau*b^4*R*T

Q_3 = -16*tau^2*b^6*R^2*T^2 + 2*a*b^5*R*T - 8*a*b^4*Vc_eff*R*T + 40*a*tau*b^5*R*T - 24*b^4*a^2 - 14*b^6*R^2*T^2 + 8*b^5*Vc_eff*R^2*T^2

Q_2 = -16*tau*b^6*R^2*T^2*Vc_eff + 20*a*b^5*R*T*Vc_eff + 8*b^6*R^2*T^2*Vc_eff

Q_1 = 4*b^6*a^2 - 2*a*b^7*R*T - 8*a*tau*b^7*R*T + 2*tau*b^8*R^2*T^2 + 4*b^8*R^2*T^2 + 4*tau^2*b^8*R^2*T^2

Q_0 = -4*a*b^7*R*T*Vc_eff - b^8*R^2*T^2*Vc_eff + 4*tau*b^8*R^2*T^2*Vc_eff

Appendix A.3 - DMT Development Environment

MatLab was used to create a development environment for the DMT function. The script below "builder" returns two temperature vectors, two vectors of volumetric residuals, two vectors of proposed translations, two vectors of reference molar volumes, and two vectors of EOS estimated molar volumes.

The script requires a reduced pressure condition and fluid index as arguments. The first set of vectors is only populated when a discrete phase transition occurs (i.e., $P_r < 1.0$). This script was the main tool in developing the DMT function; various functional forms were evaluated in an effort to match the correction vectors, *c1* and *c2*, to the molar volume residual vectors, *e1* and *e2*.

This script also has the ability to estimate molar volumes from the BWRS EOS and the PC-SAFT EOS.

```
%*****************************************

function [T1,T2,e1,e2,c1,c2,V1,V2,Vx1,Vx2] = builder(Pr,index)

BWRS_flag = 0; PCSAFT_flag = 0;

ans_set = prop_data(index,6);
Pc = ans_set.Pc; Tc = ans_set.Tc; Vmin = ans_set.Vmin; Tt = ans_set.Tt;
R = 8.31451e-2; nsteps = 1200;
T = [linspace(Tt+1,Tc-0.5,nsteps/2),linspace(Tc+0.5,Tc*2.0,nsteps/2)];
HPL = (2^(1/3)-1)*R*Tc/Pc/3;

P = Pr*Pc; cursor = 0; V = zeros(nsteps,1); Vx = zeros(nsteps,1);
c = zeros(nsteps,1);

for j = 1:nsteps
    if(T(j)<Tc)
        [Psat,VLsat,VVsat] = sat_data(T(j),index);
        [Vlims] = eq_lims(T(j),index);
        if(P>Psat)
            cursor = cursor + 1;
            limit1 = [HPL+1e-6,min(Vlims)];
            limit2 = [Vmin,1.05*VLsat];
        else
            limit1 = [max(Vlims),1e5];
            limit2 = [0.95*VVsat,1e5];
        end
    else
        limit1 = [HPL+1e-6,1e5];
        limit2 = [Vmin,1e5];
    end
```

```
    cursor = (P<Pc)*cursor;

    Vx(j) = fzero(@(V) eq_data(V,T(j),index) -P, limit1);
    V(j)  = fzero(@(V) vol_data(V,T(j),index)-P, limit2);

    if(BWRS_flag)
        Vx(j) = fzero(@(V) BWRS_data(V,T(j),index)-P, limit2);
        V(j)  = fzero(@(V) vol_data(V,T(j),index)-P,  limit2);
    end

    if(PCSAFT_flag)
        Vx(j) = fzero(@(V) PCSAFT_data(V,T(j),index)-P, limit2);
        V(j)  = fzero(@(V) vol_data(V,T(j),index)-P,  limit2);
    end

    [Px,c(j)] = eq_data(Vx(j),T(j),index);
end

T1    = T(1:cursor);       T2   = T((cursor+1):length(T));
V1    = V(1:cursor);       V2   = V((cursor+1):length(T));
Vx1   = Vx(1:cursor);      Vx2  = Vx((cursor+1):length(T));
c1    = c(1:cursor);       c2   = c((cursor+1):length(T));
e1    = Vx1- V1;           e2   = Vx2 - V2;

%*******************************************

function [P,c] = eq_data(V,T,index)

R = 8.31451e-2;

ans_set = prop_data(index,5); Pc = ans_set.Pc;
Tc = ans_set.Tc; kap = ans_set.kap; M = ans_set.M; Vc = ans_set.Vc;

alpha = exp(kap(1)*(1-(T/Tc).^kap(2)));
ac = R*R*Tc*Tc/Pc/9/(2^(1/3)-1);
a = ac*alpha;
b = (2^(1/3)-1)*R*Tc/Pc/3;

tau = (exp(5*(1-sqrt(T/Tc)))-1)*(T<Tc);
X = V*V/(V-b)/(V-b) - (2*ac*V+ac*b)/R/T/(V+b)/(V+b) + tau;
delt  = X + (V - R*Tc/Pc/4)./(V*X + R*Tc/Pc/3);
c = (R*Tc/Pc/3 - Vc) + M*(1./(1 + 2*delt) - 1);

P = R*T/(V-b)-a/V/(V+b);

return

%*******************************************

function [V] = eq_lims(T,index)

R = 8.31451e-2;

ans_set = prop_data(index,5); Pc = ans_set.Pc;
Tc = ans_set.Tc; kap = ans_set.kap;
```

```
alpha = exp(kap(1)*(1-(T/Tc).^kap(2)));
ac = R*R*Tc*Tc/Pc/9/(2^(1/3)-1);
a = ac*alpha;
b = (2^(1/3)-1)*R*Tc/Pc/3;

CFS(1) = -R*T;
CFS(2) = 2*a-2*R*T*b;
CFS(3) = -R*T*b^2-3*a*b;
CFS(4) = 0;
CFS(5) = a*b^3;

V = roots(CFS); V = V(~imag(V));
V = nonzeros(V.*(V>b));

return

%*****************************************

function [P] = BWRS_data(V,T,index)

% ** Starling 1975 **

switch(index)
    case 2 % Carbon Dioxide
        B0 = 0.394117;     A0 = 6592.03;        C0 = 2.95902e9;
        D0 = 4.09151e11;   E0 = 1.02898e10;      b = 0.971443;
         a = 5632.85;       d =   5.99297e4; alpha = 0.395525;
         c = 2.74668e9; gamma = 1.64916;
    case 3 % Methane
        B0 = 0.723251;      A0 = 7520.29;       C0 = 2.71092e8;
        D0 = 1.07737e10;    E0 = 3.01122e10;     b = 0.925404;
         a = 2574.89;        d = 47489.1;    alpha = 0.468828;
         c = 4.37222e8;  gamma = 1.48640;
    case 4 % Ethane
        B0 = 0.826059;      A0 = 13439.3;       C0 = 2.95195e9;
        D0 = 2.57477e11;    E0 = 1.46819e13;     b = 3.11206;
         a = 22404.5;        d = 702189;     alpha = 0.909681;
         c = 6.81826e9;  gamma = 2.99656;
    case 5 % Propane
        B0 = 0.964762;      A0 = 18634.7;       C0 = 7.96178e9;
        D0 = 4.53708e11;    E0 = 2.56053e13;     b = 5.462480;
         a = 40066.40;       d = 1.50520e7;  alpha = 2.014020;
         c = 2.74461e10; gamma = 4.56182;
    case 8 % Nitrogen
        B0 = 0.677022;      A0 = 4185.05;       C0 = 1.37936e8;
        D0 = 1.95183e10;    E0 = 1.21648e12;     b = 0.833470;
         a = 1404.59;        d = 31189.4;    alpha = 0.302696;
         c = 8.44317e7;  gamma = 1.10011;
    case 11 % Hydrogen Sulfide
        B0 = 0.297508;      A0 = 10586.3;       C0 = 2.11496e9;
        D0 = 4.86518e10;    E0 = 3.93226e10;     b = 2.53315;
         a = 20511.0;        d = 19973.1;    alpha = 0.165961;
         c = 4.36132e9;  gamma = 1.20447;
    case 16 % Octane
```

```
        B0 = 4.86965;       A0 = 81690.6;       C0 = 9.96546e10;
        D0 = 7.90575e12;    E0 = 3.46419e13;     b = 10.590700;
         a = 131646.00;      d = 1.85906e8;  alpha = 34.512400;
         c = 6.42053e11; gamma = 21.98880;
end

T = T*9/5;
R = 10.7335;
V = V*0.0353146667/0.00220462262;
rhom = 1/V;

P = rhom*R*T + (B0*R*T-A0-C0/T^2+D0/T^3-E0/T^4)*rhom^2 + ...
    (b*R*T-a-d/T)*rhom^3 + alpha*(a+d/T)*rhom^6 + ...
    c*rhom^3/T^2*(1+gamma*rhom^2)*exp(-gamma*rhom^2);
P = P/14.5037738;

return

% %*****************************************

function [P] = PCSAFT_data(V,T,index)

% ** Sadowski 2001 **

ans_set = prop_data(index,4);
m = ans_set.m; sig = ans_set.sig; eok = ans_set.eok;

Nav = 6.0221415e+23; kbz = 1.3806503e-23; N = 1; rho = (N/V)*Nav/1e27;
d  = sig.*(1-0.12*exp(-3*eok/T)); Z = 1; mbar = m;
squig0 = rho*pi*(Z/N*(m.*d.^0))/6; squig1 = rho*pi*(Z/N*(m.*d.^1))/6;
squig2 = rho*pi*(Z/N*(m.*d.^2))/6; squig3 = rho*pi*(Z/N*(m.*d.^3))/6;
eta    = squig3;

sig_mat = sig; eok_mat = eok;

prefac = [1+0*mbar,(mbar-1)./mbar,(mbar-1).*(mbar-2)./mbar./mbar ];
aval = [  0.9105631445,   -0.3084016918,  -0.0906148351;
          0.6361281449,    0.1860531159,   0.4527842806;
          2.6861347891,   -2.5030047259,   0.5962700728;
        -26.5473624910,   21.4197936290,  -1.7241829131;
         97.7592087840,  -65.2558853300,  -4.1302112531;
       -159.5915408700,   83.3186804810,  13.7766318700;
         91.2977740840,  -33.7469229300,  -8.6728470368];
bval = [  0.7240946941,   -0.5755498075,   0.0976883116;
          2.2382791861,    0.6995095521,  -0.2557574982;
         -4.0025849485,    3.8925673390,  -9.1558561530;
        -21.0035768150,  -17.2154716480,  20.6420759740;
         26.8556413630,  192.6722644700, -38.8044300520;
        206.5513384100, -161.8264616500,  93.6267740770;
       -355.6023561200, -165.2076934600, -29.6669055850];

aval = aval.*((eta.^(0:6)')*ones(1,3));
bval = bval.*((eta.^(0:6)')*ones(1,3)); I2 = sum(bval*prefac');
daval = aval.*(((1:7)')*ones(1,3)); dI1 = sum(daval*prefac');
dbval = bval.*(((1:7)')*ones(1,3)); dI2 = sum(dbval*prefac');
```

```
Zhs = squig3/(1-squig3) + 3*squig1*squig2/squig0/(1-squig3)^2 + ...
      (3*squig2^3-squig3*squig2^3)/squig0/(1-squig3)^3;
ghs = [d.^0,d/2,(d/2).^2]*...
         [1/(1-squig3);3*squig2/(1-squig3)^2;2*squig2^2/(1-squig3)^3];
dgh = [d.^0,d/2,(d/2).^2]*...
      [squig3/(1-squig3)^2;3*squig2/(1-squig3)^2 + 6*squig2*squig3/(1-
squig3)^3;
       4*squig2^2/(1-squig3)^3+6*squig2^2*squig3/(1-squig3)^4];
Zhc = mbar*Zhs - Z/N*((m-1)./ghs.*dgh);

C1 = 1/(1 + mbar*(8*eta-2*eta^2)/(1-eta)^4 + ...
     (1-mbar)*(20*eta-27*eta^2+12*eta^3-2*eta^4)/(1-eta)^2/(2-eta)^2);
C2 = -(C1^2)*(mbar*(-4*eta^2+20*eta+8)/(1-eta)^5 + ...
             (1-mbar)*(2*eta^3+12*eta^2-48*eta+40)/(1-eta)^3/(2-eta)^3);
m2e1s3 = sum(sum((m*m)*((eok_mat/T).^1).*sig_mat.^3));
m2e2s3 = sum(sum((m*m)*((eok_mat/T).^2).*sig_mat.^3));
Zdisp = -2*pi*rho*dI1*m2e1s3 - pi*rho*mbar*(dI2*C1+C2*eta*I2)*m2e2s3;

Ztot = 1 + Zhc + Zdisp;
P = Ztot*kbz*T*rho*1e25;           % bar

return

%*****************************************
```

Appendix A.4 - Pure Component Calculation Examples

An example output from the "builder" script has been included below. This output was used to calculate several of the averages presented for carbon dioxide in table 4.2. For the seven cases presented in the table for carbon dioxide, the arguments to the "builder" script included the index for carbon dioxide (i.e., 2) and the reduced pressure conditions (i.e., 60/73.8).

Table A.3: Output from the "builder" script using the arguments 60/73.8 and 2.

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
217.59	0.037105	0.039665	0.002839	0.002559	295.15	0.20922	0.21588	0.008039	0.006657
217.74	0.037121	0.039682	0.002841	0.002561	295.3	0.21052	0.21702	0.008018	0.006503
217.88	0.037137	0.0397	0.002844	0.002563	295.44	0.21178	0.21814	0.007998	0.006362
218.02	0.037152	0.039717	0.002846	0.002565	295.58	0.213	0.21923	0.007979	0.006233
218.17	0.037168	0.039735	0.002848	0.002567	295.73	0.21419	0.22031	0.00796	0.006114
218.31	0.037184	0.039753	0.002851	0.002569	295.87	0.21536	0.22136	0.007942	0.006003
218.45	0.037199	0.039771	0.002853	0.002571	296.02	0.21649	0.2224	0.007925	0.0059
218.6	0.037215	0.039788	0.002855	0.002573	296.16	0.21761	0.22341	0.007908	0.005805
218.74	0.037231	0.039806	0.002857	0.002575	296.3	0.2187	0.22441	0.007892	0.005715
218.88	0.037246	0.039824	0.00286	0.002578	296.45	0.21976	0.22539	0.007876	0.005631
219.03	0.037262	0.039842	0.002862	0.00258	296.59	0.22081	0.22636	0.007861	0.005553
219.17	0.037278	0.03986	0.002864	0.002582	296.73	0.22184	0.22731	0.007846	0.005479
219.32	0.037294	0.039878	0.002867	0.002584	296.88	0.22284	0.22825	0.007831	0.00541
219.46	0.03731	0.039896	0.002869	0.002586	297.02	0.22383	0.22918	0.007817	0.005344
219.6	0.037326	0.039914	0.002872	0.002588	297.16	0.22481	0.23009	0.007803	0.005283
219.75	0.037341	0.039932	0.002874	0.00259	297.31	0.22577	0.23099	0.007789	0.005224
219.89	0.037357	0.03995	0.002876	0.002593	297.45	0.22671	0.23188	0.007776	0.005169
220.03	0.037373	0.039968	0.002879	0.002595	297.6	0.22764	0.23276	0.007763	0.005116
220.18	0.037389	0.039986	0.002881	0.002597	297.74	0.22856	0.23363	0.007751	0.005067
220.32	0.037405	0.040005	0.002884	0.002599	297.88	0.22947	0.23449	0.007738	0.00502
220.46	0.037421	0.040023	0.002886	0.002602	298.03	0.23036	0.23533	0.007726	0.004975
220.61	0.037437	0.040041	0.002888	0.002604	298.17	0.23124	0.23617	0.007714	0.004932
220.75	0.037453	0.040059	0.002891	0.002606	298.31	0.23211	0.237	0.007703	0.004892
220.9	0.03747	0.040078	0.002893	0.002608	298.46	0.23297	0.23782	0.007692	0.004853
221.04	0.037486	0.040096	0.002896	0.002611	298.6	0.23382	0.23863	0.00768	0.004816
221.18	0.037502	0.040115	0.002898	0.002613	298.74	0.23465	0.23943	0.00767	0.004781
221.33	0.037518	0.040133	0.002901	0.002615	298.89	0.23548	0.24023	0.007659	0.004748
221.47	0.037534	0.040152	0.002903	0.002618	299.03	0.2363	0.24102	0.007648	0.004716
221.61	0.03755	0.040171	0.002906	0.00262	299.18	0.23711	0.2418	0.007638	0.004685
221.76	0.037567	0.040189	0.002908	0.002623	299.32	0.23791	0.24257	0.007628	0.004656
221.9	0.037583	0.040208	0.002911	0.002625	299.46	0.23871	0.24334	0.007618	0.004628
222.04	0.037599	0.040227	0.002913	0.002627	299.61	0.23949	0.2441	0.007608	0.004602
222.19	0.037616	0.040245	0.002916	0.00263	299.75	0.24027	0.24485	0.007599	0.004576
222.33	0.037632	0.040264	0.002918	0.002632	299.89	0.24104	0.24559	0.007589	0.004552
222.48	0.037648	0.040283	0.002921	0.002635	300.04	0.24181	0.24633	0.00758	0.004529
222.62	0.037665	0.040302	0.002923	0.002637	300.18	0.24256	0.24707	0.007571	0.004507
222.76	0.037681	0.040321	0.002926	0.00264	300.32	0.24331	0.2478	0.007562	0.004486

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
222.91	0.037698	0.04034	0.002929	0.002642	300.47	0.24405	0.24852	0.007553	0.004465
223.05	0.037714	0.040359	0.002931	0.002645	300.61	0.24479	0.24924	0.007544	0.004446
223.19	0.037731	0.040378	0.002934	0.002648	300.76	0.24552	0.24995	0.007536	0.004427
223.34	0.037747	0.040397	0.002936	0.00265	300.9	0.24624	0.25065	0.007527	0.00441
223.48	0.037764	0.040416	0.002939	0.002653	301.04	0.24696	0.25135	0.007519	0.004393
223.62	0.03778	0.040436	0.002942	0.002655	301.19	0.24767	0.25205	0.007511	0.004376
223.77	0.037797	0.040455	0.002944	0.002658	301.33	0.24838	0.25274	0.007503	0.004361
223.91	0.037814	0.040474	0.002947	0.002661	301.47	0.24908	0.25343	0.007495	0.004346
224.06	0.03783	0.040493	0.00295	0.002663	301.62	0.24978	0.25411	0.007487	0.004332
224.2	0.037847	0.040513	0.002952	0.002666	301.76	0.25047	0.25478	0.007479	0.004318
224.34	0.037864	0.040532	0.002955	0.002669	301.9	0.25115	0.25546	0.007471	0.004305
224.49	0.03788	0.040552	0.002958	0.002671	302.05	0.25183	0.25613	0.007464	0.004293
224.63	0.037897	0.040571	0.00296	0.002674	302.19	0.25251	0.25679	0.007456	0.004281
224.77	0.037914	0.040591	0.002963	0.002677	302.34	0.25318	0.25745	0.007449	0.00427
224.92	0.037931	0.040611	0.002966	0.00268	302.48	0.25385	0.2581	0.007442	0.004259
225.06	0.037948	0.04063	0.002969	0.002683	302.62	0.25451	0.25876	0.007435	0.004248
225.2	0.037965	0.04065	0.002971	0.002685	302.77	0.25517	0.2594	0.007427	0.004239
225.35	0.037981	0.04067	0.002974	0.002688	302.91	0.25582	0.26005	0.00742	0.004229
225.49	0.037998	0.040689	0.002977	0.002691	303.05	0.25647	0.26069	0.007413	0.00422
225.64	0.038015	0.040709	0.00298	0.002694	303.2	0.25711	0.26132	0.007407	0.004212
225.78	0.038032	0.040729	0.002982	0.002697	303.34	0.25775	0.26196	0.0074	0.004204
225.92	0.038049	0.040749	0.002985	0.0027	303.48	0.25839	0.26259	0.007393	0.004196
226.07	0.038067	0.040769	0.002988	0.002703	303.63	0.25902	0.26321	0.007386	0.004189
226.21	0.038084	0.040789	0.002991	0.002706	304.63	0.26333	0.26748	0.007344	0.004149
226.35	0.038101	0.040809	0.002994	0.002709	305.14	0.26545	0.26959	0.007326	0.004134
226.5	0.038118	0.040829	0.002997	0.002712	305.64	0.26753	0.27166	0.007308	0.004123
226.64	0.038135	0.04085	0.002999	0.002715	306.15	0.26958	0.27369	0.00729	0.004115
226.78	0.038152	0.04087	0.003002	0.002718	306.66	0.27159	0.2757	0.007273	0.00411
226.93	0.038169	0.04089	0.003005	0.002721	307.16	0.27357	0.27768	0.007257	0.004107
227.07	0.038187	0.04091	0.003008	0.002724	307.67	0.27552	0.27963	0.007242	0.004106
227.22	0.038204	0.040931	0.003011	0.002727	308.18	0.27744	0.28155	0.007227	0.004108
227.36	0.038221	0.040951	0.003014	0.00273	308.68	0.27934	0.28345	0.007212	0.004112
227.5	0.038239	0.040972	0.003017	0.002733	309.19	0.28121	0.28532	0.007198	0.004117
227.65	0.038256	0.040992	0.00302	0.002736	309.7	0.28305	0.28717	0.007185	0.004124
227.79	0.038274	0.041013	0.003023	0.002739	310.2	0.28487	0.289	0.007172	0.004133
227.93	0.038291	0.041033	0.003026	0.002742	310.71	0.28667	0.29081	0.007159	0.004143
228.08	0.038308	0.041054	0.003029	0.002746	311.22	0.28844	0.2926	0.007146	0.004154
228.22	0.038326	0.041075	0.003032	0.002749	311.72	0.2902	0.29436	0.007134	0.004167
228.36	0.038343	0.041096	0.003035	0.002752	312.23	0.29193	0.29611	0.007123	0.004181
228.51	0.038361	0.041116	0.003038	0.002755	312.74	0.29365	0.29785	0.007111	0.004196
228.65	0.038379	0.041137	0.003041	0.002759	313.25	0.29535	0.29956	0.0071	0.004211
228.8	0.038396	0.041158	0.003044	0.002762	313.75	0.29703	0.30126	0.00709	0.004228
228.94	0.038414	0.041179	0.003047	0.002765	314.26	0.29869	0.30294	0.007079	0.004245
229.08	0.038432	0.0412	0.00305	0.002769	314.77	0.30034	0.3046	0.007069	0.004264
229.23	0.038449	0.041221	0.003053	0.002772	315.27	0.30197	0.30626	0.007059	0.004283
229.37	0.038467	0.041242	0.003056	0.002775	315.78	0.30359	0.30789	0.007049	0.004303
229.51	0.038485	0.041263	0.003059	0.002779	316.29	0.30519	0.30952	0.00704	0.004323

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
229.66	0.038503	0.041285	0.003062	0.002782	316.79	0.30678	0.31113	0.00703	0.004344
229.8	0.03852	0.041306	0.003065	0.002786	317.3	0.30836	0.31272	0.007021	0.004366
229.94	0.038538	0.041327	0.003068	0.002789	317.81	0.30992	0.31431	0.007012	0.004388
230.09	0.038556	0.041349	0.003072	0.002792	318.31	0.31147	0.31588	0.007003	0.004411
230.23	0.038574	0.04137	0.003075	0.002796	318.82	0.313	0.31744	0.006995	0.004434
230.38	0.038592	0.041392	0.003078	0.0028	319.33	0.31453	0.31899	0.006987	0.004458
230.52	0.03861	0.041413	0.003081	0.002803	319.83	0.31604	0.32052	0.006978	0.004482
230.66	0.038628	0.041435	0.003084	0.002807	320.34	0.31754	0.32205	0.00697	0.004506
230.81	0.038646	0.041456	0.003087	0.00281	320.85	0.31903	0.32356	0.006963	0.004531
230.95	0.038664	0.041478	0.003091	0.002814	321.36	0.32051	0.32507	0.006955	0.004556
231.09	0.038682	0.0415	0.003094	0.002817	321.86	0.32198	0.32656	0.006947	0.004582
231.24	0.038701	0.041522	0.003097	0.002821	322.37	0.32344	0.32805	0.00694	0.004608
231.38	0.038719	0.041544	0.0031	0.002825	322.88	0.32489	0.32953	0.006933	0.004634
231.52	0.038737	0.041565	0.003104	0.002828	323.38	0.32633	0.33099	0.006926	0.004661
231.67	0.038755	0.041587	0.003107	0.002832	323.89	0.32776	0.33245	0.006919	0.004687
231.81	0.038774	0.041609	0.00311	0.002836	324.4	0.32918	0.3339	0.006912	0.004714
231.96	0.038792	0.041632	0.003114	0.00284	324.9	0.3306	0.33534	0.006905	0.004742
232.1	0.03881	0.041654	0.003117	0.002843	325.41	0.332	0.33677	0.006898	0.004769
232.24	0.038829	0.041676	0.00312	0.002847	325.92	0.3334	0.33819	0.006892	0.004797
232.39	0.038847	0.041698	0.003124	0.002851	326.42	0.33478	0.33961	0.006885	0.004825
232.53	0.038866	0.04172	0.003127	0.002855	326.93	0.33616	0.34102	0.006879	0.004853
232.67	0.038884	0.041743	0.00313	0.002859	327.44	0.33754	0.34242	0.006873	0.004881
232.82	0.038903	0.041765	0.003134	0.002863	327.95	0.3389	0.34381	0.006867	0.004909
232.96	0.038921	0.041788	0.003137	0.002867	328.45	0.34026	0.3452	0.006861	0.004938
233.1	0.03894	0.04181	0.003141	0.00287	328.96	0.34161	0.34657	0.006855	0.004966
233.25	0.038958	0.041833	0.003144	0.002874	329.47	0.34295	0.34794	0.006849	0.004995
233.39	0.038977	0.041856	0.003148	0.002878	329.97	0.34428	0.34931	0.006843	0.005024
233.54	0.038996	0.041878	0.003151	0.002882	330.48	0.34561	0.35067	0.006837	0.005053
233.68	0.039015	0.041901	0.003155	0.002886	330.99	0.34693	0.35202	0.006832	0.005083
233.82	0.039033	0.041924	0.003158	0.002891	331.49	0.34825	0.35336	0.006826	0.005112
233.97	0.039052	0.041947	0.003162	0.002895	332	0.34956	0.3547	0.006821	0.005141
234.11	0.039071	0.04197	0.003165	0.002899	332.51	0.35086	0.35603	0.006816	0.005171
234.25	0.03909	0.041993	0.003169	0.002903	333.01	0.35216	0.35736	0.00681	0.0052
234.4	0.039109	0.042016	0.003172	0.002907	333.52	0.35345	0.35868	0.006805	0.00523
234.54	0.039128	0.042039	0.003176	0.002911	334.03	0.35473	0.35999	0.0068	0.00526
234.68	0.039147	0.042062	0.003179	0.002915	334.53	0.35601	0.3613	0.006795	0.005289
234.83	0.039166	0.042085	0.003183	0.00292	335.04	0.35728	0.3626	0.00679	0.005319
234.97	0.039185	0.042109	0.003187	0.002924	335.55	0.35855	0.3639	0.006785	0.005349
235.12	0.039204	0.042132	0.00319	0.002928	336.06	0.35981	0.36519	0.00678	0.005379
235.26	0.039223	0.042155	0.003194	0.002932	336.56	0.36107	0.36648	0.006775	0.005409
235.4	0.039242	0.042179	0.003198	0.002937	337.07	0.36232	0.36776	0.006771	0.005439
235.55	0.039262	0.042202	0.003201	0.002941	337.58	0.36357	0.36904	0.006766	0.005469
235.69	0.039281	0.042226	0.003205	0.002945	338.08	0.36481	0.37031	0.006761	0.005499
235.83	0.0393	0.04225	0.003209	0.00295	338.59	0.36605	0.37157	0.006757	0.005529
235.98	0.039319	0.042273	0.003213	0.002954	339.1	0.36728	0.37284	0.006752	0.00556
236.12	0.039339	0.042297	0.003216	0.002959	339.6	0.3685	0.37409	0.006748	0.00559
236.26	0.039358	0.042321	0.00322	0.002963	340.11	0.36972	0.37534	0.006743	0.00562

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
236.41	0.039378	0.042345	0.003224	0.002968	340.62	0.37094	0.37659	0.006739	0.00565
236.55	0.039397	0.042369	0.003228	0.002972	341.12	0.37215	0.37783	0.006735	0.00568
236.7	0.039417	0.042393	0.003231	0.002977	341.63	0.37336	0.37907	0.006731	0.005711
236.84	0.039436	0.042417	0.003235	0.002981	342.14	0.37457	0.38031	0.006726	0.005741
236.98	0.039456	0.042441	0.003239	0.002986	342.65	0.37577	0.38154	0.006722	0.005771
237.13	0.039475	0.042466	0.003243	0.00299	343.15	0.37696	0.38276	0.006718	0.005801
237.27	0.039495	0.04249	0.003247	0.002995	343.66	0.37815	0.38398	0.006714	0.005832
237.41	0.039515	0.042514	0.003251	0.003	344.17	0.37934	0.3852	0.00671	0.005862
237.56	0.039534	0.042539	0.003255	0.003004	344.67	0.38052	0.38641	0.006706	0.005892
237.7	0.039554	0.042563	0.003259	0.003009	345.18	0.3817	0.38762	0.006702	0.005922
237.84	0.039574	0.042588	0.003263	0.003014	345.69	0.38287	0.38883	0.006698	0.005952
237.99	0.039594	0.042613	0.003267	0.003019	346.19	0.38404	0.39003	0.006694	0.005983
238.13	0.039614	0.042637	0.003271	0.003023	346.7	0.38521	0.39122	0.006691	0.006013
238.28	0.039634	0.042662	0.003275	0.003028	347.21	0.38637	0.39242	0.006687	0.006043
238.42	0.039654	0.042687	0.003279	0.003033	347.71	0.38753	0.39361	0.006683	0.006073
238.56	0.039674	0.042712	0.003283	0.003038	348.22	0.38869	0.39479	0.00668	0.006103
238.71	0.039694	0.042737	0.003287	0.003043	348.73	0.38984	0.39597	0.006676	0.006133
238.85	0.039714	0.042762	0.003291	0.003048	349.23	0.39099	0.39715	0.006672	0.006163
238.99	0.039734	0.042787	0.003295	0.003053	349.74	0.39213	0.39833	0.006669	0.006193
239.14	0.039754	0.042812	0.003299	0.003058	350.25	0.39328	0.3995	0.006665	0.006223
239.28	0.039775	0.042837	0.003303	0.003063	350.76	0.39441	0.40067	0.006662	0.006253
239.42	0.039795	0.042863	0.003307	0.003068	351.26	0.39555	0.40183	0.006658	0.006283
239.57	0.039815	0.042888	0.003312	0.003073	351.77	0.39668	0.40299	0.006655	0.006313
239.71	0.039836	0.042914	0.003316	0.003078	352.28	0.39781	0.40415	0.006651	0.006343
239.86	0.039856	0.042939	0.00332	0.003083	352.78	0.39893	0.40531	0.006648	0.006373
240	0.039876	0.042965	0.003324	0.003088	353.29	0.40006	0.40646	0.006645	0.006403
240.14	0.039897	0.04299	0.003329	0.003093	353.8	0.40117	0.40761	0.006641	0.006433
240.29	0.039917	0.043016	0.003333	0.003099	354.3	0.40229	0.40875	0.006638	0.006462
240.43	0.039938	0.043042	0.003337	0.003104	354.81	0.4034	0.4099	0.006635	0.006492
240.57	0.039959	0.043068	0.003341	0.003109	355.32	0.40451	0.41103	0.006632	0.006522
240.72	0.039979	0.043094	0.003346	0.003115	355.82	0.40562	0.41217	0.006629	0.006551
240.86	0.04	0.04312	0.00335	0.00312	356.33	0.40672	0.4133	0.006625	0.006581
241	0.040021	0.043146	0.003354	0.003125	356.84	0.40782	0.41443	0.006622	0.00661
241.15	0.040042	0.043172	0.003359	0.003131	357.34	0.40892	0.41556	0.006619	0.00664
241.29	0.040062	0.043198	0.003363	0.003136	357.85	0.41002	0.41669	0.006616	0.006669
241.44	0.040083	0.043225	0.003368	0.003141	358.36	0.41111	0.41781	0.006613	0.006698
241.58	0.040104	0.043251	0.003372	0.003147	358.87	0.4122	0.41893	0.00661	0.006728
241.72	0.040125	0.043278	0.003377	0.003152	359.37	0.41329	0.42004	0.006607	0.006757
241.87	0.040146	0.043304	0.003381	0.003158	359.88	0.41437	0.42116	0.006604	0.006786
242.01	0.040167	0.043331	0.003386	0.003164	360.39	0.41545	0.42227	0.006601	0.006815
242.15	0.040188	0.043358	0.00339	0.003169	360.89	0.41653	0.42338	0.006598	0.006844
242.3	0.04021	0.043384	0.003395	0.003175	361.4	0.41761	0.42448	0.006596	0.006874
242.44	0.040231	0.043411	0.003399	0.00318	361.91	0.41868	0.42559	0.006593	0.006902
242.58	0.040252	0.043438	0.003404	0.003186	362.41	0.41975	0.42669	0.00659	0.006931
242.73	0.040273	0.043465	0.003409	0.003192	362.92	0.42082	0.42778	0.006587	0.00696
242.87	0.040295	0.043492	0.003413	0.003198	363.43	0.42189	0.42888	0.006584	0.006989
243.02	0.040316	0.043519	0.003418	0.003203	363.93	0.42295	0.42997	0.006582	0.007018

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
243.16	0.040337	0.043547	0.003423	0.003209	364.44	0.42402	0.43106	0.006579	0.007047
243.3	0.040359	0.043574	0.003427	0.003215	364.95	0.42508	0.43215	0.006576	0.007075
243.45	0.04038	0.043601	0.003432	0.003221	365.46	0.42613	0.43324	0.006573	0.007104
243.59	0.040402	0.043629	0.003437	0.003227	365.96	0.42719	0.43432	0.006571	0.007132
243.73	0.040424	0.043656	0.003442	0.003233	366.47	0.42824	0.4354	0.006568	0.007161
243.88	0.040445	0.043684	0.003446	0.003239	366.98	0.42929	0.43648	0.006565	0.007189
244.02	0.040467	0.043712	0.003451	0.003245	367.48	0.43034	0.43756	0.006563	0.007217
244.16	0.040489	0.04374	0.003456	0.003251	367.99	0.43138	0.43863	0.00656	0.007246
244.31	0.040511	0.043768	0.003461	0.003257	368.5	0.43243	0.4397	0.006558	0.007274
244.45	0.040532	0.043796	0.003466	0.003263	369	0.43347	0.44077	0.006555	0.007302
244.6	0.040554	0.043824	0.003471	0.003269	369.51	0.43451	0.44184	0.006553	0.00733
244.74	0.040576	0.043852	0.003476	0.003275	370.02	0.43555	0.4429	0.00655	0.007358
244.88	0.040598	0.04388	0.003481	0.003282	370.52	0.43658	0.44397	0.006548	0.007386
245.03	0.04062	0.043908	0.003486	0.003288	371.03	0.43761	0.44503	0.006545	0.007414
245.17	0.040643	0.043937	0.003491	0.003294	371.54	0.43865	0.44609	0.006543	0.007441
245.31	0.040665	0.043965	0.003496	0.003301	372.04	0.43967	0.44714	0.00654	0.007469
245.46	0.040687	0.043994	0.003501	0.003307	372.55	0.4407	0.4482	0.006538	0.007497
245.6	0.040709	0.044023	0.003506	0.003313	373.06	0.44173	0.44925	0.006535	0.007524
245.74	0.040732	0.044051	0.003511	0.00332	373.57	0.44275	0.4503	0.006533	0.007552
245.89	0.040754	0.04408	0.003516	0.003326	374.07	0.44377	0.45135	0.006531	0.007579
246.03	0.040776	0.044109	0.003521	0.003333	374.58	0.44479	0.4524	0.006528	0.007607
246.17	0.040799	0.044138	0.003526	0.003339	375.09	0.44581	0.45344	0.006526	0.007634
246.32	0.040821	0.044167	0.003532	0.003346	375.59	0.44682	0.45448	0.006524	0.007661
246.46	0.040844	0.044196	0.003537	0.003352	376.1	0.44784	0.45553	0.006521	0.007688
246.61	0.040867	0.044226	0.003542	0.003359	376.61	0.44885	0.45656	0.006519	0.007715
246.75	0.040889	0.044255	0.003548	0.003366	377.11	0.44986	0.4576	0.006517	0.007742
246.89	0.040912	0.044284	0.003553	0.003372	377.62	0.45087	0.45864	0.006515	0.007769
247.04	0.040935	0.044314	0.003558	0.003379	378.13	0.45187	0.45967	0.006512	0.007796
247.18	0.040958	0.044344	0.003564	0.003386	378.63	0.45288	0.4607	0.00651	0.007823
247.32	0.040981	0.044373	0.003569	0.003393	379.14	0.45388	0.46173	0.006508	0.00785
247.47	0.041004	0.044403	0.003574	0.0034	379.65	0.45488	0.46276	0.006506	0.007876
247.61	0.041027	0.044433	0.00358	0.003407	380.16	0.45588	0.46379	0.006504	0.007903
247.75	0.04105	0.044463	0.003585	0.003413	380.66	0.45688	0.46481	0.006502	0.007929
247.9	0.041073	0.044493	0.003591	0.00342	381.17	0.45788	0.46583	0.006499	0.007956
248.04	0.041096	0.044523	0.003596	0.003427	381.68	0.45887	0.46685	0.006497	0.007982
248.19	0.041119	0.044554	0.003602	0.003435	382.18	0.45986	0.46787	0.006495	0.008008
248.33	0.041142	0.044584	0.003608	0.003442	382.69	0.46086	0.46889	0.006493	0.008034
248.47	0.041166	0.044615	0.003613	0.003449	383.2	0.46184	0.46991	0.006491	0.008061
248.62	0.041189	0.044645	0.003619	0.003456	383.7	0.46283	0.47092	0.006489	0.008087
248.76	0.041213	0.044676	0.003625	0.003463	384.21	0.46382	0.47193	0.006487	0.008112
248.9	0.041236	0.044707	0.00363	0.00347	384.72	0.4648	0.47294	0.006485	0.008138
249.05	0.04126	0.044737	0.003636	0.003478	385.22	0.46579	0.47395	0.006483	0.008164
249.19	0.041283	0.044768	0.003642	0.003485	385.73	0.46677	0.47496	0.006481	0.00819
249.33	0.041307	0.044799	0.003648	0.003492	386.24	0.46775	0.47596	0.006479	0.008215
249.48	0.041331	0.044831	0.003653	0.0035	386.74	0.46873	0.47697	0.006477	0.008241
249.62	0.041355	0.044862	0.003659	0.003507	387.25	0.46971	0.47797	0.006475	0.008266
249.77	0.041379	0.044893	0.003665	0.003515	387.76	0.47068	0.47897	0.006473	0.008292

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
249.91	0.041402	0.044925	0.003671	0.003522	388.27	0.47166	0.47997	0.006471	0.008317
250.05	0.041426	0.044956	0.003677	0.00353	388.77	0.47263	0.48097	0.006469	0.008342
250.2	0.041451	0.044988	0.003683	0.003537	389.28	0.4736	0.48197	0.006467	0.008368
250.34	0.041475	0.04502	0.003689	0.003545	389.79	0.47457	0.48296	0.006465	0.008393
250.48	0.041499	0.045052	0.003695	0.003553	390.29	0.47554	0.48396	0.006463	0.008418
250.63	0.041523	0.045084	0.003701	0.00356	390.8	0.47651	0.48495	0.006461	0.008443
250.77	0.041547	0.045116	0.003707	0.003568	391.31	0.47747	0.48594	0.006459	0.008467
250.91	0.041572	0.045148	0.003713	0.003576	391.81	0.47844	0.48693	0.006458	0.008492
251.06	0.041596	0.04518	0.00372	0.003584	392.32	0.4794	0.48792	0.006456	0.008517
251.2	0.041621	0.045212	0.003726	0.003592	392.83	0.48036	0.4889	0.006454	0.008541
251.35	0.041645	0.045245	0.003732	0.0036	393.33	0.48132	0.48989	0.006452	0.008566
251.49	0.04167	0.045278	0.003738	0.003608	393.84	0.48228	0.49087	0.00645	0.00859
251.63	0.041694	0.04531	0.003745	0.003616	394.35	0.48324	0.49185	0.006448	0.008615
251.78	0.041719	0.045343	0.003751	0.003624	394.85	0.48419	0.49283	0.006447	0.008639
251.92	0.041744	0.045376	0.003757	0.003632	395.36	0.48515	0.49381	0.006445	0.008663
252.06	0.041769	0.045409	0.003764	0.00364	395.87	0.4861	0.49479	0.006443	0.008687
252.21	0.041794	0.045442	0.00377	0.003649	396.38	0.48706	0.49577	0.006441	0.008711
252.35	0.041819	0.045476	0.003777	0.003657	396.88	0.48801	0.49674	0.00644	0.008735
252.49	0.041844	0.045509	0.003783	0.003665	397.39	0.48896	0.49772	0.006438	0.008759
252.64	0.041869	0.045542	0.00379	0.003673	397.9	0.48991	0.49869	0.006436	0.008783
252.78	0.041894	0.045576	0.003796	0.003682	398.4	0.49085	0.49966	0.006434	0.008807
252.93	0.04192	0.04561	0.003803	0.00369	398.91	0.4918	0.50063	0.006433	0.00883
253.07	0.041945	0.045644	0.00381	0.003699	399.42	0.49275	0.5016	0.006431	0.008854
253.21	0.04197	0.045678	0.003816	0.003707	399.92	0.49369	0.50257	0.006429	0.008877
253.36	0.041996	0.045712	0.003823	0.003716	400.43	0.49463	0.50353	0.006428	0.008901
253.5	0.042021	0.045746	0.00383	0.003725	400.94	0.49557	0.5045	0.006426	0.008924
253.64	0.042047	0.04578	0.003837	0.003733	401.44	0.49651	0.50546	0.006424	0.008947
253.79	0.042073	0.045815	0.003844	0.003742	401.95	0.49745	0.50642	0.006423	0.00897
253.93	0.042098	0.045849	0.00385	0.003751	402.46	0.49839	0.50739	0.006421	0.008993
254.07	0.042124	0.045884	0.003857	0.00376	402.97	0.49933	0.50835	0.006419	0.009016
254.22	0.04215	0.045919	0.003864	0.003769	403.47	0.50027	0.5093	0.006418	0.009039
254.36	0.042176	0.045953	0.003871	0.003778	403.98	0.5012	0.51026	0.006416	0.009062
254.51	0.042202	0.045988	0.003878	0.003787	404.49	0.50213	0.51122	0.006414	0.009085
254.65	0.042228	0.046024	0.003885	0.003796	404.99	0.50307	0.51217	0.006413	0.009107
254.79	0.042254	0.046059	0.003893	0.003805	405.5	0.504	0.51313	0.006411	0.00913
254.94	0.042281	0.046094	0.0039	0.003814	406.01	0.50493	0.51408	0.00641	0.009152
255.08	0.042307	0.04613	0.003907	0.003823	406.51	0.50586	0.51503	0.006408	0.009175
255.22	0.042333	0.046165	0.003914	0.003832	407.02	0.50679	0.51598	0.006406	0.009197
255.37	0.04236	0.046201	0.003922	0.003842	407.53	0.50771	0.51693	0.006405	0.009219
255.51	0.042386	0.046237	0.003929	0.003851	408.03	0.50864	0.51788	0.006403	0.009241
255.65	0.042413	0.046273	0.003936	0.00386	408.54	0.50956	0.51883	0.006402	0.009263
255.8	0.04244	0.046309	0.003944	0.00387	409.05	0.51049	0.51977	0.0064	0.009285
255.94	0.042466	0.046346	0.003951	0.003879	409.55	0.51141	0.52072	0.006399	0.009307
256.09	0.042493	0.046382	0.003959	0.003889	410.06	0.51233	0.52166	0.006397	0.009329
256.23	0.04252	0.046419	0.003966	0.003898	410.57	0.51325	0.52261	0.006396	0.009351
256.37	0.042547	0.046455	0.003974	0.003908	411.08	0.51417	0.52355	0.006394	0.009372
256.52	0.042574	0.046492	0.003981	0.003918	411.58	0.51509	0.52449	0.006393	0.009394

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
256.66	0.042601	0.046529	0.003989	0.003928	412.09	0.51601	0.52543	0.006391	0.009416
256.8	0.042629	0.046566	0.003997	0.003937	412.6	0.51693	0.52637	0.00639	0.009437
256.95	0.042656	0.046603	0.004004	0.003947	413.1	0.51784	0.5273	0.006388	0.009458
257.09	0.042683	0.046641	0.004012	0.003957	413.61	0.51876	0.52824	0.006387	0.009479
257.23	0.042711	0.046678	0.00402	0.003967	414.12	0.51967	0.52917	0.006385	0.009501
257.38	0.042738	0.046716	0.004028	0.003977	414.62	0.52059	0.53011	0.006384	0.009522
257.52	0.042766	0.046753	0.004036	0.003987	415.13	0.5215	0.53104	0.006383	0.009543
257.67	0.042794	0.046791	0.004044	0.003998	415.64	0.52241	0.53197	0.006381	0.009564
257.81	0.042822	0.046829	0.004052	0.004008	416.14	0.52332	0.53291	0.00638	0.009584
257.95	0.042849	0.046868	0.00406	0.004018	416.65	0.52423	0.53384	0.006378	0.009605
258.1	0.042877	0.046906	0.004068	0.004029	417.16	0.52514	0.53476	0.006377	0.009626
258.24	0.042905	0.046944	0.004076	0.004039	417.67	0.52605	0.53569	0.006375	0.009646
258.38	0.042934	0.046983	0.004085	0.004049	418.17	0.52695	0.53662	0.006374	0.009667
258.53	0.042962	0.047022	0.004093	0.00406	418.68	0.52786	0.53755	0.006373	0.009687
258.67	0.04299	0.047061	0.004101	0.004071	419.19	0.52876	0.53847	0.006371	0.009708
258.81	0.043019	0.0471	0.00411	0.004081	419.69	0.52967	0.5394	0.00637	0.009728
258.96	0.043047	0.047139	0.004118	0.004092	420.2	0.53057	0.54032	0.006369	0.009748
259.1	0.043076	0.047178	0.004127	0.004103	420.71	0.53147	0.54124	0.006367	0.009768
259.25	0.043104	0.047218	0.004135	0.004114	421.21	0.53237	0.54216	0.006366	0.009788
259.39	0.043133	0.047257	0.004144	0.004125	421.72	0.53328	0.54308	0.006364	0.009808
259.53	0.043162	0.047297	0.004152	0.004136	422.23	0.53418	0.544	0.006363	0.009828
259.68	0.043191	0.047337	0.004161	0.004147	422.73	0.53507	0.54492	0.006362	0.009848
259.82	0.04322	0.047377	0.00417	0.004158	423.24	0.53597	0.54584	0.00636	0.009867
259.96	0.043249	0.047418	0.004179	0.004169	423.75	0.53687	0.54676	0.006359	0.009887
260.11	0.043278	0.047458	0.004188	0.00418	424.25	0.53777	0.54767	0.006358	0.009907
260.25	0.043307	0.047499	0.004196	0.004191	424.76	0.53866	0.54859	0.006357	0.009926
260.39	0.043337	0.047539	0.004205	0.004203	425.27	0.53956	0.5495	0.006355	0.009945
260.54	0.043366	0.04758	0.004215	0.004214	425.78	0.54045	0.55041	0.006354	0.009965
260.68	0.043396	0.047621	0.004224	0.004226	426.28	0.54134	0.55133	0.006353	0.009984
260.83	0.043425	0.047663	0.004233	0.004237	426.79	0.54224	0.55224	0.006351	0.010003
260.97	0.043455	0.047704	0.004242	0.004249	427.3	0.54313	0.55315	0.00635	0.010022
261.11	0.043485	0.047746	0.004251	0.004261	427.8	0.54402	0.55406	0.006349	0.010041
261.26	0.043515	0.047787	0.004261	0.004272	428.31	0.54491	0.55497	0.006348	0.01006
261.4	0.043545	0.047829	0.00427	0.004284	428.82	0.5458	0.55588	0.006346	0.010079
261.54	0.043575	0.047871	0.004279	0.004296	429.32	0.54669	0.55678	0.006345	0.010097
261.69	0.043605	0.047913	0.004289	0.004308	429.83	0.54757	0.55769	0.006344	0.010116
261.83	0.043636	0.047956	0.004298	0.00432	430.34	0.54846	0.55859	0.006343	0.010135
261.97	0.043666	0.047998	0.004308	0.004332	430.84	0.54935	0.5595	0.006341	0.010153
262.12	0.043697	0.048041	0.004318	0.004344	431.35	0.55023	0.5604	0.00634	0.010172
262.26	0.043727	0.048084	0.004327	0.004357	431.86	0.55112	0.56131	0.006339	0.01019
262.41	0.043758	0.048127	0.004337	0.004369	432.36	0.552	0.56221	0.006338	0.010208
262.55	0.043789	0.048171	0.004347	0.004382	432.87	0.55288	0.56311	0.006336	0.010226
262.69	0.04382	0.048214	0.004357	0.004394	433.38	0.55377	0.56401	0.006335	0.010244
262.84	0.043851	0.048258	0.004367	0.004407	433.89	0.55465	0.56491	0.006334	0.010262
262.98	0.043882	0.048301	0.004377	0.004419	434.39	0.55553	0.56581	0.006333	0.01028
263.12	0.043913	0.048345	0.004387	0.004432	434.9	0.55641	0.56671	0.006332	0.010298
263.27	0.043945	0.04839	0.004398	0.004445	435.41	0.55729	0.5676	0.00633	0.010316

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
263.41	0.043976	0.048434	0.004408	0.004458	435.91	0.55817	0.5685	0.006329	0.010334
263.55	0.044008	0.048479	0.004418	0.004471	436.42	0.55904	0.5694	0.006328	0.010351
263.7	0.04404	0.048523	0.004429	0.004484	436.93	0.55992	0.57029	0.006327	0.010369
263.84	0.044071	0.048568	0.004439	0.004497	437.43	0.5608	0.57119	0.006326	0.010386
263.99	0.044103	0.048613	0.00445	0.00451	437.94	0.56168	0.57208	0.006325	0.010404
264.13	0.044135	0.048659	0.00446	0.004523	438.45	0.56255	0.57297	0.006323	0.010421
264.27	0.044167	0.048704	0.004471	0.004537	438.95	0.56343	0.57386	0.006322	0.010438
264.42	0.0442	0.04875	0.004482	0.00455	439.46	0.5643	0.57475	0.006321	0.010455
264.56	0.044232	0.048796	0.004493	0.004564	439.97	0.56517	0.57564	0.00632	0.010472
264.7	0.044265	0.048842	0.004503	0.004577	440.48	0.56605	0.57653	0.006319	0.010489
264.85	0.044297	0.048888	0.004514	0.004591	440.98	0.56692	0.57742	0.006318	0.010506
264.99	0.04433	0.048935	0.004526	0.004605	441.49	0.56779	0.57831	0.006317	0.010523
265.13	0.044363	0.048981	0.004537	0.004619	442	0.56866	0.5792	0.006316	0.01054
265.28	0.044396	0.049028	0.004548	0.004633	442.5	0.56953	0.58009	0.006314	0.010556
265.42	0.044429	0.049075	0.004559	0.004647	443.01	0.5704	0.58097	0.006313	0.010573
265.57	0.044462	0.049123	0.004571	0.004661	443.52	0.57127	0.58186	0.006312	0.01059
265.71	0.044495	0.04917	0.004582	0.004675	444.02	0.57214	0.58274	0.006311	0.010606
265.85	0.044529	0.049218	0.004594	0.004689	444.53	0.573	0.58363	0.00631	0.010622
266	0.044562	0.049266	0.004605	0.004704	445.04	0.57387	0.58451	0.006309	0.010639
266.14	0.044596	0.049314	0.004617	0.004718	445.54	0.57474	0.58539	0.006308	0.010655
266.28	0.04463	0.049363	0.004629	0.004733	446.05	0.5756	0.58627	0.006307	0.010671
266.43	0.044664	0.049411	0.004641	0.004748	446.56	0.57647	0.58715	0.006306	0.010687
266.57	0.044698	0.04946	0.004652	0.004762	447.06	0.57733	0.58803	0.006305	0.010703
266.71	0.044732	0.049509	0.004665	0.004777	447.57	0.57819	0.58891	0.006304	0.010719
266.86	0.044766	0.049558	0.004677	0.004792	448.08	0.57906	0.58979	0.006302	0.010735
267	0.044801	0.049608	0.004689	0.004807	448.59	0.57992	0.59067	0.006301	0.010751
267.15	0.044835	0.049658	0.004701	0.004822	449.09	0.58078	0.59155	0.0063	0.010766
267.29	0.04487	0.049708	0.004714	0.004838	449.6	0.58164	0.59243	0.006299	0.010782
267.43	0.044905	0.049758	0.004726	0.004853	450.11	0.5825	0.5933	0.006298	0.010797
267.58	0.04494	0.049808	0.004739	0.004868	450.61	0.58336	0.59418	0.006297	0.010813
267.72	0.044975	0.049859	0.004751	0.004884	451.12	0.58422	0.59505	0.006296	0.010828
267.86	0.04501	0.04991	0.004764	0.0049	451.63	0.58508	0.59593	0.006295	0.010843
268.01	0.045046	0.049961	0.004777	0.004915	452.13	0.58594	0.5968	0.006294	0.010859
268.15	0.045081	0.050012	0.00479	0.004931	452.64	0.5868	0.59767	0.006293	0.010874
268.29	0.045117	0.050064	0.004803	0.004947	453.15	0.58766	0.59855	0.006292	0.010889
268.44	0.045153	0.050116	0.004816	0.004963	453.65	0.58851	0.59942	0.006291	0.010904
268.58	0.045189	0.050168	0.004829	0.00498	454.16	0.58937	0.60029	0.00629	0.010919
268.73	0.045225	0.05022	0.004843	0.004996	454.67	0.59023	0.60116	0.006289	0.010934
268.87	0.045261	0.050273	0.004856	0.005012	455.18	0.59108	0.60203	0.006288	0.010949
269.01	0.045297	0.050326	0.00487	0.005029	455.68	0.59193	0.6029	0.006287	0.010963
269.16	0.045334	0.050379	0.004883	0.005045	456.19	0.59279	0.60377	0.006286	0.010978
269.3	0.04537	0.050433	0.004897	0.005062	456.7	0.59364	0.60463	0.006285	0.010992
269.44	0.045407	0.050486	0.004911	0.005079	457.2	0.59449	0.6055	0.006284	0.011007
269.59	0.045444	0.05054	0.004925	0.005096	457.71	0.59535	0.60637	0.006283	0.011021
269.73	0.045481	0.050594	0.004939	0.005113	458.22	0.5962	0.60723	0.006282	0.011036
269.87	0.045519	0.050649	0.004953	0.00513	458.72	0.59705	0.6081	0.006281	0.01105
270.02	0.045556	0.050704	0.004967	0.005147	459.23	0.5979	0.60896	0.00628	0.011064

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
270.16	0.045594	0.050759	0.004982	0.005165	459.74	0.59875	0.60983	0.006279	0.011078
270.31	0.045632	0.050814	0.004996	0.005182	460.24	0.5996	0.61069	0.006278	0.011092
270.45	0.045669	0.050869	0.005011	0.0052	460.75	0.60045	0.61156	0.006277	0.011106
270.59	0.045708	0.050925	0.005026	0.005218	461.26	0.6013	0.61242	0.006276	0.01112
270.74	0.045746	0.050981	0.00504	0.005236	461.76	0.60215	0.61328	0.006275	0.011134
270.88	0.045784	0.051038	0.005055	0.005254	462.27	0.60299	0.61414	0.006275	0.011148
271.02	0.045823	0.051095	0.00507	0.005272	462.78	0.60384	0.615	0.006274	0.011161
271.17	0.045862	0.051152	0.005086	0.00529	463.29	0.60469	0.61586	0.006273	0.011175
271.31	0.0459	0.051209	0.005101	0.005309	463.79	0.60553	0.61672	0.006272	0.011189
271.45	0.04594	0.051267	0.005117	0.005327	464.3	0.60638	0.61758	0.006271	0.011202
271.6	0.045979	0.051324	0.005132	0.005346	464.81	0.60722	0.61844	0.00627	0.011216
271.74	0.046018	0.051383	0.005148	0.005365	465.31	0.60807	0.6193	0.006269	0.011229
271.89	0.046058	0.051441	0.005164	0.005383	465.82	0.60891	0.62015	0.006268	0.011242
272.03	0.046098	0.0515	0.00518	0.005403	466.33	0.60976	0.62101	0.006267	0.011255
272.17	0.046138	0.051559	0.005196	0.005422	466.83	0.6106	0.62187	0.006266	0.011268
272.32	0.046178	0.051619	0.005212	0.005441	467.34	0.61144	0.62272	0.006265	0.011281
272.46	0.046218	0.051678	0.005228	0.005461	467.85	0.61228	0.62358	0.006264	0.011294
272.6	0.046259	0.051739	0.005245	0.00548	468.35	0.61313	0.62443	0.006263	0.011307
272.75	0.046299	0.051799	0.005261	0.0055	468.86	0.61397	0.62529	0.006263	0.01132
272.89	0.04634	0.05186	0.005278	0.00552	469.37	0.61481	0.62614	0.006262	0.011333
273.03	0.046381	0.051921	0.005295	0.00554	469.87	0.61565	0.62699	0.006261	0.011346
273.18	0.046422	0.051982	0.005312	0.00556	470.38	0.61649	0.62785	0.00626	0.011358
273.32	0.046464	0.052044	0.005329	0.00558	470.89	0.61733	0.6287	0.006259	0.011371
273.47	0.046505	0.052106	0.005347	0.005601	471.4	0.61816	0.62955	0.006258	0.011383
273.61	0.046547	0.052169	0.005364	0.005621	471.9	0.619	0.6304	0.006257	0.011396
273.75	0.046589	0.052232	0.005382	0.005642	472.41	0.61984	0.63125	0.006256	0.011408
273.9	0.046632	0.052295	0.0054	0.005663	472.92	0.62068	0.6321	0.006255	0.01142
274.04	0.046674	0.052358	0.005418	0.005684	473.42	0.62152	0.63295	0.006255	0.011433
274.18	0.046717	0.052422	0.005436	0.005705	473.93	0.62235	0.6338	0.006254	0.011445
274.33	0.04676	0.052486	0.005454	0.005727	474.44	0.62319	0.63464	0.006253	0.011457
274.47	0.046803	0.052551	0.005473	0.005748	474.94	0.62402	0.63549	0.006252	0.011469
274.61	0.046846	0.052616	0.005491	0.00577	475.45	0.62486	0.63634	0.006251	0.011481
274.76	0.046889	0.052681	0.00551	0.005792	475.96	0.62569	0.63719	0.00625	0.011493
274.9	0.046933	0.052747	0.005529	0.005814	476.46	0.62653	0.63803	0.006249	0.011504
275.05	0.046977	0.052813	0.005548	0.005836	476.97	0.62736	0.63888	0.006249	0.011516
275.19	0.047021	0.05288	0.005567	0.005859	477.48	0.62819	0.63972	0.006248	0.011528
275.33	0.047066	0.052947	0.005587	0.005881	477.99	0.62903	0.64057	0.006247	0.011539
275.48	0.04711	0.053014	0.005606	0.005904	478.49	0.62986	0.64141	0.006246	0.011551
275.62	0.047155	0.053082	0.005626	0.005927	479	0.63069	0.64225	0.006245	0.011562
275.76	0.0472	0.05315	0.005646	0.00595	479.51	0.63152	0.6431	0.006244	0.011574
275.91	0.047245	0.053219	0.005666	0.005973	480.01	0.63236	0.64394	0.006244	0.011585
276.05	0.047291	0.053288	0.005687	0.005997	480.52	0.63319	0.64478	0.006243	0.011596
276.19	0.047337	0.053357	0.005707	0.00602	481.03	0.63402	0.64562	0.006242	0.011608
276.34	0.047383	0.053427	0.005728	0.006044	481.53	0.63485	0.64647	0.006241	0.011619
276.48	0.047429	0.053497	0.005749	0.006068	482.04	0.63568	0.64731	0.00624	0.01163
276.63	0.047476	0.053568	0.00577	0.006093	482.55	0.6365	0.64815	0.00624	0.011641
276.77	0.047522	0.053639	0.005791	0.006117	483.05	0.63733	0.64899	0.006239	0.011652

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
276.91	0.047569	0.053711	0.005813	0.006142	483.56	0.63816	0.64982	0.006238	0.011663
277.06	0.047617	0.053783	0.005835	0.006166	484.07	0.63899	0.65066	0.006237	0.011673
277.2	0.047664	0.053856	0.005857	0.006191	484.57	0.63982	0.6515	0.006236	0.011684
277.34	0.047712	0.053929	0.005879	0.006217	485.08	0.64064	0.65234	0.006236	0.011695
277.49	0.04776	0.054002	0.005901	0.006242	485.59	0.64147	0.65318	0.006235	0.011705
277.63	0.047808	0.054076	0.005924	0.006268	486.1	0.6423	0.65401	0.006234	0.011716
277.77	0.047857	0.05415	0.005946	0.006293	486.6	0.64312	0.65485	0.006233	0.011726
277.92	0.047906	0.054225	0.00597	0.00632	487.11	0.64395	0.65569	0.006232	0.011737
278.06	0.047955	0.054301	0.005993	0.006346	487.62	0.64477	0.65652	0.006232	0.011747
278.21	0.048004	0.054377	0.006016	0.006372	488.12	0.6456	0.65736	0.006231	0.011758
278.35	0.048054	0.054453	0.00604	0.006399	488.63	0.64642	0.65819	0.00623	0.011768
278.49	0.048104	0.05453	0.006064	0.006426	489.14	0.64725	0.65903	0.006229	0.011778
278.64	0.048155	0.054608	0.006088	0.006453	489.64	0.64807	0.65986	0.006228	0.011788
278.78	0.048205	0.054686	0.006113	0.006481	490.15	0.64889	0.66069	0.006228	0.011798
278.92	0.048256	0.054764	0.006137	0.006508	490.66	0.64972	0.66152	0.006227	0.011808
279.07	0.048307	0.054843	0.006162	0.006536	491.16	0.65054	0.66236	0.006226	0.011818
279.21	0.048359	0.054923	0.006187	0.006564	491.67	0.65136	0.66319	0.006225	0.011828
279.35	0.048411	0.055003	0.006213	0.006592	492.18	0.65218	0.66402	0.006225	0.011838
279.5	0.048463	0.055084	0.006239	0.006621	492.69	0.653	0.66485	0.006224	0.011847
279.64	0.048515	0.055165	0.006265	0.00665	493.19	0.65383	0.66568	0.006223	0.011857
279.79	0.048568	0.055247	0.006291	0.006679	493.7	0.65465	0.66651	0.006222	0.011866
279.93	0.048621	0.05533	0.006317	0.006708	494.21	0.65547	0.66734	0.006222	0.011876
280.07	0.048675	0.055413	0.006344	0.006738	494.71	0.65629	0.66817	0.006221	0.011885
280.22	0.048729	0.055497	0.006371	0.006768	495.22	0.65711	0.669	0.00622	0.011895
280.36	0.048783	0.055581	0.006399	0.006798	495.73	0.65792	0.66983	0.006219	0.011904
280.5	0.048837	0.055666	0.006426	0.006829	496.23	0.65874	0.67066	0.006219	0.011913
280.65	0.048892	0.055751	0.006454	0.006859	496.74	0.65956	0.67148	0.006218	0.011923
280.79	0.048947	0.055838	0.006483	0.00689	497.25	0.66038	0.67231	0.006217	0.011932
280.93	0.049003	0.055925	0.006511	0.006922	497.75	0.6612	0.67314	0.006216	0.011941
281.08	0.049059	0.056012	0.00654	0.006953	498.26	0.66201	0.67396	0.006216	0.01195
281.22	0.049115	0.0561	0.00657	0.006985	498.77	0.66283	0.67479	0.006215	0.011959
281.37	0.049172	0.056189	0.006599	0.007017	499.27	0.66365	0.67562	0.006214	0.011968
281.51	0.049229	0.056279	0.006629	0.00705	499.78	0.66446	0.67644	0.006214	0.011977
281.65	0.049287	0.056369	0.006659	0.007083	500.29	0.66528	0.67727	0.006213	0.011985
281.8	0.049344	0.05646	0.00669	0.007116	500.8	0.6661	0.67809	0.006212	0.011994
281.94	0.049403	0.056552	0.006721	0.007149	501.3	0.66691	0.67891	0.006211	0.012003
282.08	0.049462	0.056644	0.006752	0.007183	501.81	0.66773	0.67974	0.006211	0.012011
282.23	0.049521	0.056738	0.006784	0.007217	502.32	0.66854	0.68056	0.00621	0.01202
282.37	0.04958	0.056832	0.006816	0.007251	502.82	0.66936	0.68138	0.006209	0.012028
282.51	0.04964	0.056926	0.006848	0.007286	503.33	0.67017	0.68221	0.006209	0.012037
282.66	0.049701	0.057022	0.006881	0.007321	503.84	0.67098	0.68303	0.006208	0.012045
282.8	0.049762	0.057118	0.006914	0.007357	504.34	0.6718	0.68385	0.006207	0.012054
282.95	0.049823	0.057215	0.006947	0.007393	504.85	0.67261	0.68467	0.006206	0.012062
283.09	0.049885	0.057313	0.006981	0.007429	505.36	0.67342	0.68549	0.006206	0.01207
283.23	0.049947	0.057412	0.007016	0.007465	505.86	0.67423	0.68631	0.006205	0.012078
283.38	0.05001	0.057512	0.007051	0.007502	506.37	0.67505	0.68713	0.006204	0.012086
283.52	0.050073	0.057612	0.007086	0.00754	506.88	0.67586	0.68795	0.006204	0.012094

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
283.66	0.050137	0.057714	0.007121	0.007577	507.38	0.67667	0.68877	0.006203	0.012102
283.81	0.050201	0.057816	0.007157	0.007615	507.89	0.67748	0.68959	0.006202	0.01211
283.95	0.050265	0.057919	0.007194	0.007654	508.4	0.67829	0.69041	0.006202	0.012118
284.09	0.050331	0.058023	0.007231	0.007693	508.91	0.6791	0.69123	0.006201	0.012126
284.24	0.050396	0.058129	0.007268	0.007732	509.41	0.67991	0.69204	0.0062	0.012133
284.38	0.050463	0.058235	0.007306	0.007772	509.92	0.68072	0.69286	0.0062	0.012141
284.53	0.05053	0.058342	0.007344	0.007812	510.43	0.68153	0.69368	0.006199	0.012149
284.67	0.050597	0.05845	0.007383	0.007853	510.93	0.68234	0.69449	0.006198	0.012156
284.81	0.050665	0.058559	0.007423	0.007894	511.44	0.68315	0.69531	0.006198	0.012164
284.96	0.050733	0.058669	0.007463	0.007935	511.95	0.68396	0.69613	0.006197	0.012171
285.1	0.050803	0.05878	0.007503	0.007977	512.45	0.68476	0.69694	0.006196	0.012179
285.24	0.050872	0.058892	0.007544	0.00802	512.96	0.68557	0.69776	0.006196	0.012186
285.39	0.050943	0.059006	0.007585	0.008063	513.47	0.68638	0.69857	0.006195	0.012193
285.53	0.051014	0.05912	0.007627	0.008106	513.97	0.68719	0.69939	0.006194	0.0122
285.67	0.051085	0.059236	0.00767	0.00815	514.48	0.68799	0.7002	0.006194	0.012207
285.82	0.051158	0.059352	0.007713	0.008195	514.99	0.6888	0.70101	0.006193	0.012215
285.96	0.05123	0.05947	0.007757	0.00824	515.5	0.68961	0.70183	0.006192	0.012222
286.1	0.051304	0.059589	0.007801	0.008285	516	0.69041	0.70264	0.006192	0.012229
286.25	0.051378	0.05971	0.007846	0.008332	516.51	0.69122	0.70345	0.006191	0.012236
286.39	0.051453	0.059832	0.007892	0.008378	517.02	0.69202	0.70427	0.00619	0.012242
286.54	0.051529	0.059955	0.007938	0.008426	517.52	0.69283	0.70508	0.00619	0.012249
286.68	0.051605	0.060079	0.007985	0.008473	518.03	0.69363	0.70589	0.006189	0.012256
286.82	0.051683	0.060204	0.008032	0.008522	518.54	0.69444	0.7067	0.006188	0.012263
286.97	0.051761	0.060331	0.008081	0.008571	519.04	0.69524	0.70751	0.006188	0.012269
287.11	0.051839	0.06046	0.00813	0.008621	519.55	0.69605	0.70832	0.006187	0.012276
287.25	0.051919	0.06059	0.008179	0.008671	520.06	0.69685	0.70913	0.006187	0.012282
287.4	0.051999	0.060721	0.00823	0.008722	520.56	0.69765	0.70994	0.006186	0.012289
287.54	0.05208	0.060854	0.008281	0.008774	521.07	0.69846	0.71075	0.006185	0.012295
287.68	0.052162	0.060988	0.008333	0.008826	521.58	0.69926	0.71156	0.006185	0.012302
287.83	0.052245	0.061124	0.008385	0.008879	522.08	0.70006	0.71237	0.006184	0.012308
287.97	0.052329	0.061262	0.008439	0.008933	522.59	0.70086	0.71318	0.006183	0.012314
288.12	0.052414	0.061401	0.008493	0.008987	523.1	0.70167	0.71399	0.006183	0.012321
288.26	0.0525	0.061542	0.008548	0.009043	523.61	0.70247	0.71479	0.006182	0.012327
288.4	0.052586	0.061685	0.008604	0.009099	524.11	0.70327	0.7156	0.006182	0.012333
288.55	0.052674	0.06183	0.008661	0.009156	524.62	0.70407	0.71641	0.006181	0.012339
288.69	0.052763	0.061976	0.008719	0.009213	525.13	0.70487	0.71722	0.00618	0.012345
288.83	0.052852	0.062124	0.008777	0.009272	525.63	0.70567	0.71802	0.00618	0.012351
288.98	0.052943	0.062274	0.008837	0.009331	526.14	0.70647	0.71883	0.006179	0.012357
289.12	0.053035	0.062427	0.008897	0.009392	526.65	0.70727	0.71963	0.006179	0.012363
289.26	0.053128	0.062581	0.008959	0.009453	527.15	0.70807	0.72044	0.006178	0.012368
289.41	0.053222	0.062737	0.009021	0.009515	527.66	0.70887	0.72124	0.006177	0.012374
289.55	0.053318	0.062896	0.009085	0.009578	528.17	0.70967	0.72205	0.006177	0.01238
289.7	0.053414	0.063056	0.00915	0.009642	528.67	0.71047	0.72285	0.006176	0.012385
289.84	0.053512	0.06322	0.009215	0.009707	529.18	0.71127	0.72366	0.006176	0.012391
289.98	0.053611	0.063385	0.009282	0.009774	529.69	0.71207	0.72446	0.006175	0.012396
290.13	0.053712	0.063553	0.00935	0.009841	530.2	0.71286	0.72527	0.006174	0.012402
290.27	0.053814	0.063723	0.009419	0.009909	530.7	0.71366	0.72607	0.006174	0.012407

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
290.41	0.053917	0.063896	0.00949	0.009979	531.21	0.71446	0.72687	0.006173	0.012413
290.56	0.054022	0.064072	0.009561	0.01005	531.72	0.71526	0.72767	0.006173	0.012418
290.7	0.054128	0.06425	0.009634	0.010122	532.22	0.71605	0.72848	0.006172	0.012423
290.84	0.054236	0.064431	0.009709	0.010195	532.73	0.71685	0.72928	0.006171	0.012429
290.99	0.054346	0.064616	0.009784	0.01027	533.24	0.71765	0.73008	0.006171	0.012434
291.13	0.054457	0.064803	0.009861	0.010346	533.74	0.71844	0.73088	0.00617	0.012439
291.28	0.05457	0.064993	0.00994	0.010424	534.25	0.71924	0.73168	0.00617	0.012444
291.42	0.054684	0.065187	0.010019	0.010503	534.76	0.72004	0.73248	0.006169	0.012449
291.56	0.054801	0.065384	0.010101	0.010583	535.26	0.72083	0.73329	0.006168	0.012454
291.71	0.05492	0.065585	0.010184	0.010666	535.77	0.72163	0.73409	0.006168	0.012459
291.85	0.05504	0.06579	0.010268	0.010749	536.28	0.72242	0.73489	0.006167	0.012464
291.99	0.055163	0.065998	0.010355	0.010835	536.78	0.72322	0.73568	0.006167	0.012468
292.14	0.055287	0.06621	0.010443	0.010923	537.29	0.72401	0.73648	0.006166	0.012473
292.28	0.055414	0.066426	0.010532	0.011012	537.8	0.72481	0.73728	0.006166	0.012478
292.42	0.055544	0.066647	0.010624	0.011103	538.31	0.7256	0.73808	0.006165	0.012483
292.57	0.055676	0.066872	0.010717	0.011197	538.81	0.72639	0.73888	0.006164	0.012487
292.71	0.05581	0.067102	0.010813	0.011292	539.32	0.72719	0.73968	0.006164	0.012492
292.86	0.055947	0.067337	0.01091	0.01139	539.83	0.72798	0.74048	0.006163	0.012496
293	0.056087	0.067577	0.01101	0.01149	540.33	0.72877	0.74127	0.006163	0.012501
293.14	0.056229	0.067822	0.011111	0.011593	540.84	0.72957	0.74207	0.006162	0.012505
293.29	0.056375	0.068074	0.011215	0.011698	541.35	0.73036	0.74287	0.006162	0.01251
293.43	0.056524	0.068331	0.011321	0.011806	541.85	0.73115	0.74367	0.006161	0.012514
293.57	0.056677	0.068594	0.01143	0.011918	542.36	0.73194	0.74446	0.006161	0.012518
293.72	0.056833	0.068864	0.011541	0.012032	542.87	0.73274	0.74526	0.00616	0.012522
293.86	0.056992	0.069142	0.011654	0.012149	543.37	0.73353	0.74605	0.006159	0.012527
294	0.057156	0.069426	0.01177	0.01227	543.88	0.73432	0.74685	0.006159	0.012531
294.15	0.057324	0.069719	0.011889	0.012395	544.39	0.73511	0.74764	0.006158	0.012535
294.29	0.057497	0.07002	0.012011	0.012523	544.89	0.7359	0.74844	0.006158	0.012539
294.44	0.057674	0.07033	0.012136	0.012656	545.4	0.73669	0.74923	0.006157	0.012543
294.58	0.057856	0.07065	0.012263	0.012794	545.91	0.73748	0.75003	0.006157	0.012547
294.72	0.058044	0.07098	0.012394	0.012936	546.42	0.73827	0.75082	0.006156	0.012551
294.87	0.058238	0.071321	0.012529	0.013083	546.92	0.73906	0.75162	0.006156	0.012554
295.01	0.058438	0.071674	0.012667	0.013236	547.43	0.73985	0.75241	0.006155	0.012558
					547.94	0.74064	0.7532	0.006155	0.012562
					548.44	0.74143	0.754	0.006154	0.012566
					548.95	0.74222	0.75479	0.006153	0.012569
					549.46	0.74301	0.75558	0.006153	0.012573
					549.96	0.7438	0.75637	0.006152	0.012577
					550.47	0.74459	0.75717	0.006152	0.01258
					550.98	0.74537	0.75796	0.006151	0.012583
					551.48	0.74616	0.75875	0.006151	0.012587
					551.99	0.74695	0.75954	0.00615	0.01259
					552.5	0.74774	0.76033	0.00615	0.012594
					553.01	0.74853	0.76112	0.006149	0.012597
					553.51	0.74931	0.76191	0.006149	0.0126
					554.02	0.7501	0.7627	0.006148	0.012603
					554.53	0.75089	0.76349	0.006148	0.012606

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
					555.03	0.75167	0.76428	0.006147	0.01261
					555.54	0.75246	0.76507	0.006147	0.012613
					556.05	0.75324	0.76586	0.006146	0.012616
					556.55	0.75403	0.76665	0.006146	0.012619
					557.06	0.75482	0.76744	0.006145	0.012622
					557.57	0.7556	0.76823	0.006145	0.012624
					558.07	0.75639	0.76901	0.006144	0.012627
					558.58	0.75717	0.7698	0.006144	0.01263
					559.09	0.75796	0.77059	0.006143	0.012633
					559.59	0.75874	0.77138	0.006143	0.012636
					560.1	0.75953	0.77216	0.006142	0.012638
					560.61	0.76031	0.77295	0.006142	0.012641
					561.12	0.76109	0.77374	0.006141	0.012643
					561.62	0.76188	0.77452	0.006141	0.012646
					562.13	0.76266	0.77531	0.00614	0.012648
					562.64	0.76344	0.7761	0.00614	0.012651
					563.14	0.76423	0.77688	0.006139	0.012653
					563.65	0.76501	0.77767	0.006139	0.012656
					564.16	0.76579	0.77845	0.006138	0.012658
					564.66	0.76658	0.77924	0.006138	0.01266
					565.17	0.76736	0.78002	0.006137	0.012662
					565.68	0.76814	0.78081	0.006137	0.012665
					566.18	0.76892	0.78159	0.006136	0.012667
					566.69	0.7697	0.78237	0.006136	0.012669
					567.2	0.77049	0.78316	0.006135	0.012671
					567.71	0.77127	0.78394	0.006135	0.012673
					568.21	0.77205	0.78472	0.006134	0.012675
					568.72	0.77283	0.78551	0.006134	0.012677
					569.23	0.77361	0.78629	0.006133	0.012679
					569.73	0.77439	0.78707	0.006133	0.012681
					570.24	0.77517	0.78785	0.006132	0.012682
					570.75	0.77595	0.78864	0.006132	0.012684
					571.25	0.77673	0.78942	0.006131	0.012686
					571.76	0.77751	0.7902	0.006131	0.012688
					572.27	0.77829	0.79098	0.00613	0.012689
					572.77	0.77907	0.79176	0.00613	0.012691
					573.28	0.77985	0.79254	0.006129	0.012692
					573.79	0.78063	0.79332	0.006129	0.012694
					574.29	0.78141	0.7941	0.006129	0.012695
					574.8	0.78219	0.79488	0.006128	0.012697
					575.31	0.78297	0.79566	0.006128	0.012698
					575.82	0.78374	0.79644	0.006127	0.0127
					576.32	0.78452	0.79722	0.006127	0.012701
					576.83	0.7853	0.798	0.006126	0.012702
					577.34	0.78608	0.79878	0.006126	0.012703
					577.84	0.78686	0.79956	0.006125	0.012705
					578.35	0.78763	0.80034	0.006125	0.012706

T1	e1	c1	V1	Vx1	T2	e2	c2	V2	Vx2
					578.86	0.78841	0.80112	0.006124	0.012707
					579.36	0.78919	0.8019	0.006124	0.012708
					579.87	0.78996	0.80267	0.006124	0.012709
					580.38	0.79074	0.80345	0.006123	0.01271
					580.88	0.79152	0.80423	0.006123	0.012711
					581.39	0.79229	0.80501	0.006122	0.012712
					581.9	0.79307	0.80578	0.006122	0.012713
					582.4	0.79385	0.80656	0.006121	0.012714
					582.91	0.79462	0.80734	0.006121	0.012715
					583.42	0.7954	0.80811	0.00612	0.012715
					583.93	0.79617	0.80889	0.00612	0.012716
					584.43	0.79695	0.80967	0.006119	0.012717
					584.94	0.79772	0.81044	0.006119	0.012717
					585.45	0.7985	0.81122	0.006119	0.012718
					585.95	0.79927	0.81199	0.006118	0.012719
					586.46	0.80005	0.81277	0.006118	0.012719
					586.97	0.80082	0.81354	0.006117	0.01272
					587.47	0.8016	0.81432	0.006117	0.01272
					587.98	0.80237	0.81509	0.006116	0.012721
					588.49	0.80315	0.81587	0.006116	0.012721
					588.99	0.80392	0.81664	0.006116	0.012721
					589.5	0.80469	0.81742	0.006115	0.012722
					590.01	0.80547	0.81819	0.006115	0.012722
					590.52	0.80624	0.81896	0.006114	0.012722
					591.02	0.80701	0.81974	0.006114	0.012723
					591.53	0.80779	0.82051	0.006113	0.012723
					592.04	0.80856	0.82128	0.006113	0.012723
					592.54	0.80933	0.82206	0.006113	0.012723
					593.05	0.8101	0.82283	0.006112	0.012723
					593.56	0.81088	0.8236	0.006112	0.012723
					594.06	0.81165	0.82437	0.006111	0.012723
					594.57	0.81242	0.82514	0.006111	0.012723
					595.08	0.81319	0.82592	0.00611	0.012723
					595.58	0.81396	0.82669	0.00611	0.012723
					596.09	0.81474	0.82746	0.00611	0.012723
					596.6	0.81551	0.82823	0.006109	0.012723
					597.1	0.81628	0.829	0.006109	0.012722
					597.61	0.81705	0.82977	0.006108	0.012722
					598.12	0.81782	0.83054	0.006108	0.012722
					598.63	0.81859	0.83131	0.006108	0.012722
					599.13	0.81936	0.83208	0.006107	0.012721
					599.64	0.82013	0.83285	0.006107	0.012721
					600.15	0.8209	0.83362	0.006106	0.01272
					600.65	0.82167	0.83439	0.006106	0.01272
					601.16	0.82244	0.83516	0.006105	0.012719
					601.67	0.82321	0.83593	0.006105	0.012719
					602.17	0.82398	0.8367	0.006105	0.012718

The deviations for the untranslated SRK EOS were calculated as e1/V1 and e2/V2; the deviations for the SRK EOS with optimal constant translation (OT) were calculated as (e1-0.036)/V1 and (e2-0.036)/V2; the deviations from the SRK with the modified distrance parameter translation (DMT) were calculated as (e1-c1)/V1 and (e2-c2)/V2. These values are depicted below in figure A.1.

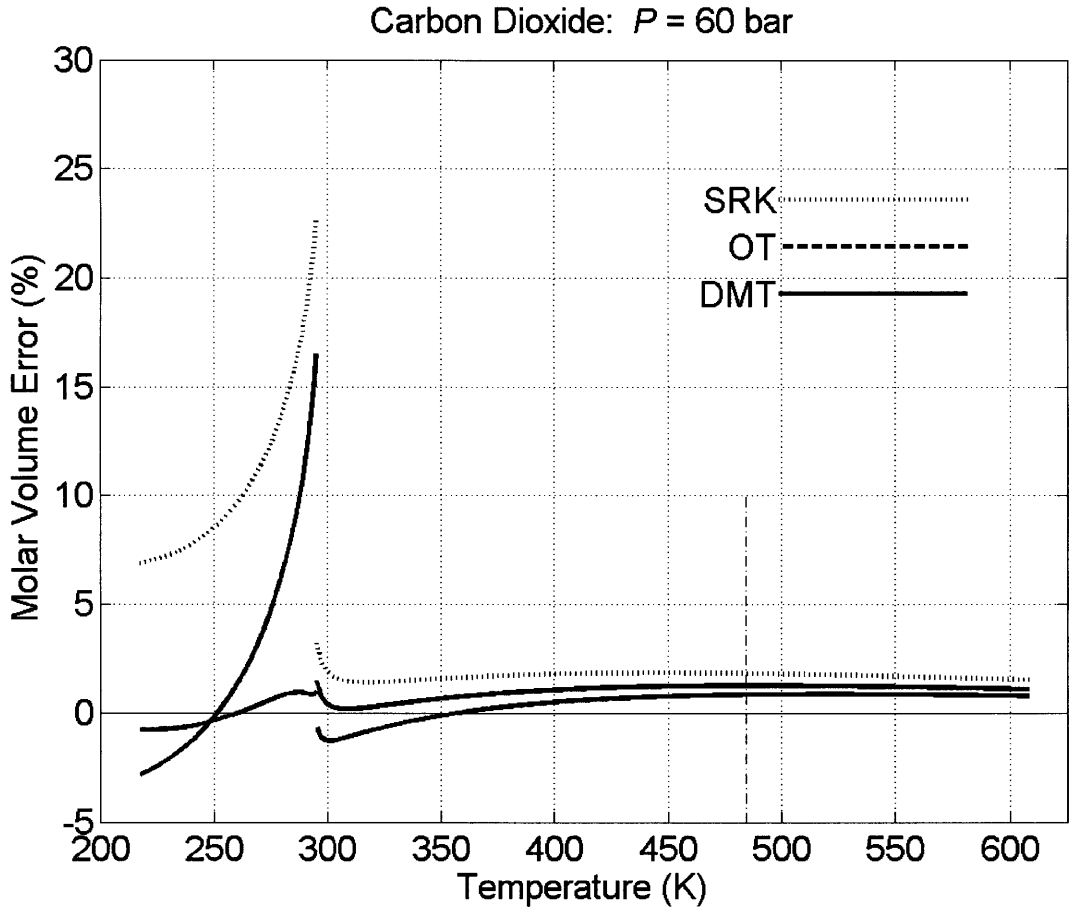

Figure A.1: Error in molar volume estimates for carbon dioxide at P = 60 bar. The vertical line at 484.5K indicates the temperature threshold above which the fluid is characterized as behaving ideally.

The absolute values of the molar volume errors depicted in figure A.1 were averaged for temperature where carbon dioxide behaves non-ideally. Ideal gas behavior is characterized here as conditions where the compressibility approaches unity. For this data set, when T > 484.5 K the compressibility is asymptotes from 1.05 to 1.00. This threshold is depicted by the vertical line at 484.5 K in figure A.1.

The absolute percent errors at T < 484.5 were averaged and reported in table 4.2 as the deviations for carbon dioxide.

Appendix A.5 - Pure Component Parameters

Pure component parameters for many fluids were collected from the literature. The script "prop_data" provides these parameters based on the index of the fluid selected. The fluid index used in other scripts is definied here.

This script provides various sets of parameters depending on the data set specified in the datflag argument.

```
function [ans_set] = prop_data(index,datflag)

% ************************************************
% Parameters for Thermodynamic Models
%
% Pc:    Critical pressure (bar)
% Tc:    Critical temperature (K)
% Vc:    Critical volume (L/mol)
% M:     DMT volume translation parameter (L/mol)
% kap:   DMT alpha function parameters
% Tt:    Triple point temperature (K)
% Vmin:  Close packed limit for references (L/mol)
% omega: Acentric factor
% m:     PC-SAFT segment number
% sig:   PC-SAFT interaction parameter (angstrom)
% eok:   PC-SAFT energetic parameter (K)
% rZc:   Rackett correlation value
%
% ************************************************

ncomps = length(index);

m = zeros(ncomps,1); sig = zeros(ncomps,1); eok = zeros(ncomps,1);
Tc = zeros(ncomps,1); Pc = zeros(ncomps,1); Vc = zeros(ncomps,1);
M = zeros(ncomps,1); kap = zeros(ncomps,2); Tt = zeros(ncomps,1);
Vmin = zeros(ncomps,1); omega = zeros(ncomps,1); rZc = zeros(ncomps,1);

for j1 = 1:ncomps
    switch index(j1)
        case 1  % Water
            Tc(j1) = 647.096; Pc(j1) = 220.64; Vc(j1) = 0.05595;
            M(j1) = 0.0190; kap(j1,:) = [0.8433,1.4760];
            Tt(j1) = 273.16; Vmin(j1) = 1.30e-2; omega(j1) = 0.344;
            m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
            rZc(j1) = 0.2294;
        case 2  % Carbon Dioxide
            Tc(j1) = 304.1282; Pc(j1) = 73.773; Vc(j1) = 0.09412;
            M(j1) = 0.0180; kap(j1,:) = [0.9391,0.9895];
            Tt(j1) = 216.592; Vmin(j1) = 3.50e-2; omega(j1) = 0.225;
            m(j1) = 2.0729; sig(j1) = 2.7852; eok(j1) = 169.21;
            rZc(j1) = 0.2700;
```

```
case 3  % Methane
    Tc(j1) = 190.564; Pc(j1) = 45.992; Vc(j1) = 0.09863;
    M(j1) = 0.0165; kap(j1,:) = [0.5091,1.1840];
    Tt(j1) = 90.6941; Vmin(j1) = 3.20e-2; omega(j1) = 0.011;
    m(j1) = 1.0000; sig(j1) = 3.7039; eok(j1) = 150.03;
    rZc(j1) = 0.2876;
case 4  % Ethane
    Tc(j1) = 305.33; Pc(j1) = 48.718; Vc(j1) = 0.1456;
    M(j1) = 0.0265; kap(j1,:) = [0.7481,0.9560];
    Tt(j1) = 90.352; Vmin(j1) = 4.40e-2; omega(j1) = 0.105;
    m(j1) = 1.6069; sig(j1) = 3.5206; eok(j1) = 191.42;
    rZc(j1) = 0.2789;
case 5  % Propane
    Tc(j1) = 369.825; Pc(j1) = 42.4709; Vc(j1) = 0.2018;
    M(j1) = 0.0365; kap(j1,:) = [0.8980,0.8628];
    Tt(j1) = 85.48; Vmin(j1) = 5.72e-2; omega(j1) = 0.1524;
    m(j1) = 2.0020; sig(j1) = 3.6184; eok(j1) = 208.11;
    rZc(j1) = 0.2736;
case 6  % n-Butane
    Tc(j1) = 425.125; Pc(j1) = 37.96; Vc(j1) = 0.2551;
    M(j1) = 0.0500; kap(j1,:) = [0.8890,0.9915];
    Tt(j1) = 134.87; Vmin(j1) = 7.51e-2; omega(j1) =  0.211;
    m(j1) = 2.3316; sig(j1) = 3.7086; eok(j1) = 222.88;
    rZc(j1) = 0.2728;
case 7 % Ammonia
    Tc(j1) = 405.40; Pc(j1) = 113.33; Vc = 0.07569;
    M(j1) = 0.0170; kap(j1,:) = [0.8086,1.2670];
    Tt(j1) = 195.495; Vmin(j1) = 2.22e-2; omega(j1) =  0.256;
    m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
    rZc(j1) = 0.2465;
case 8 % Nitrogen
    Tc(j1) = 126.192; Pc(j1) = 33.958; Vc(j1) = 0.08941;
    M(j1) = 0.0140; kap(j1,:) = [0.5883,1.0730];
    Tt(j1) = 63.151; Vmin(j1) = 2.93e-2; omega(j1) =  0.039;
    m(j1) = 1.2053; sig(j1) = 3.3130; eok(j1) = 90.96;
    rZc(j1) = 0.2905;
case 9 % Oxygen
    Tc(j1) = 154.581; Pc(j1) = 50.430; Vc(j1) = 0.07337;
    M(j1) = 0.0115; kap(j1,:) = [0.5921,1.0040];
    Tt(j1) = 54.361; Vmin(j1) = 2.05e-2; omega(j1) =  0.021;
    m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
    rZc(j1) = 0.2909;
case 10 % Argon
    Tc(j1) = 150.687; Pc(j1) = 48.630; Vc(j1) = 0.07459;
    M(j1) = 0.0120; kap(j1,:) = [0.5481,1.0200];
    Tt(j1) = 54.361; Vmin(j1) = 2.10e-2; omega(j1) = -0.002;
    m(j1) = 0.9285; sig(j1) = 3.4784; eok(j1) = 122.23;
    rZc(j1) = 0.2912;
case 11 % Hydrogen Sulfide
    Tc(j1) = 373.1; Pc(j1) = 90.000; Vc(j1) = 0.09814;
    M(j1) = 0.0160; kap(j1,:) = [0.6955,1.0580];
    Tt(j1) = 187.7; Vmin(j1) = 3.2e-2; omega(j1) =  0.100;
    m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
    rZc(j1) = 0.2851;
```

```
case 12 % Methanol
    Tc(j1) = 512.6; Pc(j1) = 81.035; Vc(j1) = 0.1163;
    M(j1) = 0; kap(j1,:) = [0.0000,0.0000];
    Tt(j1) = 175.61; Vmin(j1) = 0; omega(j1) =  0.5625;
    m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
    rZc(j1) = 0.2318;
case 13 % n-Pentane
    Tc(j1) = 469.7; Pc(j1) = 33.700; Vc(j1) = 0.3110;
    M(j1) = 0.0655; kap(j1,:) = [0.9798,0.9761];
    Tt(j1) = 143.4; Vmin(j1) = 9.2e-2; omega(j1) =  0.251;
    m(j1) = 2.6896; sig(j1) = 3.7729; eok(j1) = 231.20;
    rZc(j1) = 0.2685;
case 14 % n-Hexane
    Tc(j1) = 507.82; Pc(j1) = 30.340; Vc(j1) = 0.3696;
    M(j1) = 0.0805; kap(j1,:) = [0.9988,1.0490];
    Tt(j1) = 177.83; Vmin(j1) = 11.2e-2; omega(j1) =  0.299;
    m(j1) = 3.0576; sig(j1) = 3.7983; eok(j1) = 236.77;
    rZc(j1) = 0.2635;
case 15 % n-Heptane
    Tc(j1) = 540.13; Pc(j1) = 27.360; Vc(j1) = 0.4319;
    M(j1) = 0.0955; kap(j1,:) = [1.0890,1.0210];
    Tt(j1) = 182.55; Vmin(j1) = 12.51e-2; omega(j1) =  0.349;
    m(j1) = 3.4831; sig(j1) = 3.8049; eok(j1) = 238.40;
    rZc(j1) = 0.2611;
case 16 % n-Octane
    Tc(j1) = 569.32; Pc(j1) = 24.970; Vc(j1) = 0.4863;
    M(j1) = 0.1200; kap(j1,:) = [1.1070,1.0900];
    Tt(j1) = 216.37; Vmin(j1) = 14.10e-2; omega(j1) =  0.393;
    m(j1) = 3.8176; sig(j1) = 3.8373; eok(j1) = 242.78;
    rZc(j1) = 0.2567;
case 17 % Hydrogen
    Tc(j1) = 33.145; Pc(j1) = 12.964; Vc(j1) = 0.07086;% 0.06448;
    M(j1) = 0.0105; kap(j1,:) = [0.1718,1.247];
    M(j1) = 0;      kap(j1,:) = [0,1];
    Tt(j1) = 13.957; Vmin(j1) = 2.0e-2; omega(j1) = -0.219;
    m(j1) = 1.0000; sig(j1) = 2.9860; eok(j1) = 19.2775;
    rZc(j1) = 0.3199;
case 18 % Acetone
    Tc(j1) = 508.1; Pc(j1) = 47.01; Vc(j1) = 0.2090;
    M(j1) = 0.072; kap(j1,:) = [0.8619,1.372];
    Tt(j1) = 178.45; Vmin(j1) = 0; omega(j1) =  0.307;
    m(j1) = 2.8912; sig(j1) = 3.2279; eok(j1) = 247.42;
    rZc(j1) = 0.2459;
case 19 % Tetrafluoromethane
    Tc(j1) = 227.51; Pc(j1) = 37.5; Vc(j1) = 0.1407;
    M(j1) = 0.0275; kap(j1,:) = [0.7996,1.0930];
    Tt(j1) = 98.94; Vmin(j1) = 4.2e-2; omega(j1) =  0.1785;
    m(j1) = 1.0000; sig(j1) = 1.0000; eok(j1) = 100.00;
    rZc(j1) = 0.2789;
case 201 % Poly benzyl methacrylate 85% 5451 MW
    Tc(j1) = 1200; Pc(j1) = 1; Vc(j1) = 1;
    M(j1) = 1; kap(j1,:) = [1,1];
    Tt(j1) = 1; Vmin(j1) =  1; omega(j1) =  1;
    m(j1) = 112.3; sig(j1) = 3.77; eok(j1) = 296.3;
    rZc(j1) = 0.3333;
```

```
        case 202 % Poly benzyl methacrylate 80% 5895 MW
            Tc(j1) = 1200; Pc(j1) = 1; Vc(j1) = 1;
            M(j1) = 1; kap(j1,:) = [1,1];
            Tt(j1) = 1; Vmin(j1) =  1; omega(j1) =  1;
            m(j1) = 122.6; sig(j1) = 3.76; eok(j1) = 293.5;
            rZc(j1) = 0.3333;
        case 203 % Poly benzyl methacrylate 75% 5686 MW
            Tc(j1) = 1200; Pc(j1) = 1; Vc(j1) = 1;
            M(j1) = 1; kap(j1,:) = [1,1];
            Tt(j1) = 1; Vmin(j1) =  1; omega(j1) =  1;
            m(j1) = 119.4; sig(j1) = 3.76; eok(j1) = 290.8;
            rZc(j1) = 0.3333;

        case 21 % Naphthalene
            Tc(j1) = 748.4; Pc(j1) = 40.5; Vc(j1) = 0.4070;
            M(j1) = 1; kap(j1,:) = [1,1];
            Tt(j1) = 1; Vmin(j1) =  1; omega(j1) =  1;
            m(j1) = 3.0915; sig(j1) = 3.8333; eok(j1) = 348.40;
            rZc(j1) = 0.2650;
        case 22 % Toluene
            Tc(j1) = 591.75; Pc(j1) = 41.08; Vc(j1) = 0.3160;
            M(j1) = 0.08; kap(j1,:) = [1,1];
            Tt(j1) = 178.0; Vmin(j1) =  1; omega(j1) =  0.2570;
            m(j1) = 2.8149; sig(j1) = 3.7169; eok(j1) = 285.69;
            rZc(j1) = 0.2640;
    end
end

switch(datflag)
    case {1}
        ans_set.Pc = Pc; ans_set.Tc = Tc; ans_set.omega = omega;
    case {2}
        ans_set.Tc = Tc; ans_set.Tt = Tt; ans_set.kap = kap;
    case {3}
        ans_set.Pc = Pc; ans_set.Tc = Tc; ans_set.Vc = Vc;
        ans_set.rZc = rZc;
    case {4}
        ans_set.m = m; ans_set.sig = sig; ans_set.eok = eok;
    case {5}
        ans_set.Pc = Pc; ans_set.Tc = Tc; ans_set.kap = kap;
        ans_set.M = M; ans_set.Vc = Vc;
    case {6}
        ans_set.Pc = Pc; ans_set.Tc = Tc; ans_set.Vmin = Vmin;
        ans_set.Tt = Tt;
end
```

Appendix A.6 - Pure Component Saturation Property Reference Data

This script "sat_data" is a compilation of saturation pressure and molar volume data for various fluids. Arguments required for this script include the fluid index and temperature of interest.

```
function [P,VL,VV] = sat_data(T,substance)
% ************************************************************************
% P (bar):          Vapor pressure.
% VL (m^3/kmol):    Volume of the saturated liquid.
% VV (m^3/kmol):    Volume of the saturated vapor.
% T (K):            Temperature selected substance.
% substance:        Index of substance. [Min (K)    Max (K) ]
%                       1 = H2O         [273.16     647.096 ]
%                       2 = CO2         [216.592    304.1282]
%                       3 = CH4         [ 90.6941   190.564 ]
%                       4 = C2H6        [ 90.352    305.33  ]
%                       5 = C3H8        [ 85.48     369.825 ]
%                       6 = n-C4H10     [134.87     425.125 ]
%                       7 = NH3         [195.495    405.40  ]
%                       8 = N2          [ 63.151    126.192 ]
%                       9 = O2          [ 54.361    154.571 ]
%                      10 = Ar          [ 83.8058   150.687 ]
%                      11 = H2S         [187.67     373.37  ]
%                      12 = CH3OH       [175.61     512.6   ]
%                      13 = n-C5H12     [143.4      469.70  ]
%                      14 = n-C6H14     [177.83     507.82  ]
%                      15 = n-C7H16     [182.55     540.13  ]
%                      16 = n-C8H18     [216.37     569.32  ]
%                      17 = H2          [ 13.957     33.145 ]
%                      18 = CO(CH3)2    [178.45     508.10  ]
%                      19 = CF4         [ 98.94     227.51  ]
%                      20 = Polymer     []
%                      21 = Naphthalene []
%                      22 = Toluene     [178.0      591.75  ]
% ************************************************************************

switch (substance)

    case 1 % Water
        Tc = 647.096; Pc = 220.64; Vc = 1/17.873728; Tr = T/Tc;

        a = [-7.85951783,1.84408259,-11.7866497,...
              22.6807411,-15.9618719,1.80122503];
        t = [1;1.5;3;3.5;4;7.5];
        P = Pc*exp(a*((1-Tr).^t./Tr));

        a = [1,1.99274064,1.0996534,-0.510839303, ...
             -1.75493479,-45.5170352,-674694.450];
```

```
    t = [0;1/3;2/3;5/3;16/3;43/3;110/3];
    VL = Vc./(a*((1-Tr).^t));

    a = [-2.0315040,-2.68302940,-5.38626492,...
         -17.2991605,-44.7586581,-63.9201063];
    t = [2/6;4/6;8/6;18/6;37/6;71/6];
    VV = Vc./exp(a*((1-Tr).^t));

case 2 % Carbon Dioxide
    Tc = 304.1282; Pc = 73.773; Vc = 1/10.6249; Tr = T/Tc;

    a = [-7.0602087,1.9391218,-1.6463597,-3.2995634];
    t = [1.0;1.5;2.0;4.0];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [1.9245108, -0.62385555, -0.32731127, 0.39245142];
    t = [0.34;0.5;10/6;11/6];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.7074879,-0.82274670,-4.6008549,-10.111178,-29.742252];
    t = [0.34;0.5;1;7/3;14/3];
    VV = Vc./exp(a*((1-Tr).^t));

case 3 % Methane
    Tc = 190.564; Pc = 45.992; Vc = 1/10.139; Tr = T/Tc;

    a = [-6.036219,1.409353,-0.4945199,-1.443048];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [1.9906389,-0.78756197,0.036976723];
    t = [0.354;0.5;2.5];
    VL = Vc./exp(a*(1-Tr).^t);

    a = [-1.880284,-2.8526531,-3.0006480,...
         -5.2511690,-13.191859,-37.553961];
    t = [0.354;5/6;1.5;2.5;25/6;47/6];
    VV = Vc./exp(a*((1-Tr).^t));

case 4 % Ethane
    Tc = 305.33; Pc = 48.718; Vc = 1/6.87; Tr = T/Tc;

    a = [-7.955315,1.532027,14.78068,-13.43179,4.704891];
    t = [1;1;1.9;2;3];
    Tx = (1-Tr).^t; Tx(1) = Tx(1)/Tr;
    P = Pc*exp(a*Tx);

    a = [1.930740,-0.6539856,0.8141362,-0.3397430,-0.3838141];
    t = [0.355;2;3;0.645;4]; Tx = 1-Tr;
    VL = Vc./(1+(a(1)*Tx^t(1)+a(2)*Tx^t(2)+a(3)*Tx^t(3)+a(5)*Tx^t(5))...
         /(1+a(4)*Tx^t(4)));

    a = [-0.7483719,-1.372895,-1.192597,1.861505,1.313649];
    Tx = 1-Tr;  R = 8.31451e-2; tau = 1/Tr;
    rhov = P/R/T*(1+P*tau^8/Pc*(Pc*Vc/R/Tc-1)*(1+(a(1)*Tx^0.355+ ...
```

```
            a(2)*Tx^0.71+a(3)*(Tx+Tx^4)+a(4)*Tx^2)/(1+a(5)*Tx)))^(-1);
    VV = 1/rhov;

case 5 % Propane
    Tc = 369.825; Pc = 42.4709; Vc = 1/4.95514; Tr = T/Tc;

    a = [-6.741653,1.455497,-1.312986,-2.111039];
    t = [1.0;1.5;2.5;4.5];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [0.2758388,1.810924,-0.8907309,0.1273854];
    t = [0.2;0.4;0.6;1.8];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [0.5312985,-1.702073,-4.998449,-12.18881,-42.75035,-107.8777];
    t = [0.1;0.2;0.8;2.4;5.8;13.9];
    VV = Vc./exp(a*((1-Tr).^t));

case 6 % n-Butane
    Tc = 425.125; Pc = 37.960; Vc = 58.1222/227.84; Tr = T/Tc;

    a = [-7.246998,2.899084,-2.506508,-2.460527];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [1.378063,-0.4003577,0.3561708];
    t = [0.3;1.3;1.7];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.903334,-4.860578,-11.11094,-28.93348,-73.96275];
    t = [0.33;0.9;2.4;4.9;10];
    VV = Vc./exp(a*((1-Tr).^t));

case 7 % Ammonia
    Tc = 405.40; Pc = 113.330; Vc = 1/13.2118; Tr = T/Tc;

    a = [-7.239, 1.478,-0.561,-1.169,-1.677];
    t = [1;1.5;2;3;4];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [ 1.976,-0.8309, 0.5402,-0.4238, 0.2037];
    t = [0.333;0.667;1.5;2.5;4.5];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-0.4048,-4.963, 0.8474,-4.848,-9.325,-15.37,-36.26];
    t = [0.2;0.6;1;1.5;2.5;4.5;6.5];
    VV = Vc./exp(a*((1-Tr).^t));

case 8 % Nitrogen
    Tc = 126.192; Pc = 33.958; Vc = 1/313.3*28.01348; Tr = T/Tc;

    a = [-6.12445284,1.26327220,-0.765910082,-1.77570564];
    t = [1.0;1.5;2.5;5];
    P = Pc*exp(a*((1-Tr).^t./Tr));
```

```
    a = [1.48654237,-0.280476066,0.0894143085,-0.119879866];
    t = [0.3294;4/6;16/6;35/6];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.70127164,-3.70402649,1.29859383,-0.561424977,-2.68505381];
    t = [0.34;5/6;7/6;13/6;28/6];
    VV = Vc./exp(a*((1-Tr).^t./Tr));

case 9 % Oxygen
    Tc = 154.581; Pc = 50.430; Vc = 1/13.63; Tr = T/Tc;

    a = [-6.031,1.149,-0.4011,-1.602];
    t = [1.0;1.5;2.5;4.5];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [1.509,-0.3097,0.05805,0.01578];
    t = [1/3;2/3;7/3;4];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.703,-3.599,1.179,-0.3737,-2.515];
    t = [0.34;5/6;7/6;13/6;28/6];
    VV = Vc./exp(a*((1-Tr).^t./Tr));

case 10 % Argon
    Tc = 150.687; Pc = 48.63; Vc = 1/13.4074; Tr = T/Tc;

    a = [-5.9409785,1.3553888,-0.46497607,-1.5399043];
    t = [1.0;1.5;2;4.5];
    P = Pc*exp(a*((1-Tr).^t./Tr));

    a = [1.5004262,-0.31381290,0.086461622,-0.041477525];
    t = [0.334;2/3;7/3;4];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.70695656,-4.02739448,1.55177558,-2.30683228];
    t = [0.345;5/6;1;13/3];
    VV = Vc./exp(a*((1-Tr).^t./Tr));

case 11 % Hydrogen Sulfide
    Tc = 373.37; Pc = 89.6291; Vc = 1/10.20; Tr = T/Tc;

    a = [-6.423889,1.699405,-1.211219,-2.217591];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [2.122841,-0.8907727,0.1148276];
    t = [0.354;1/2;5/2];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-2.001663,-3.339645,-0.6781599,-13.33131,-3.988066,-75.23041];
    t = [0.354;5/6;3/2;5/2;25/6;47/6];
    VV = Vc./exp(a*((1-Tr).^t));

case 12 % Methanol
    Tc = 512.6; Pc = 81.035; Vc = 1/8.71; Tr = T/Tc;
```

```
    a = [-8.8738823,2.3698322,-10.852598,-0.12446396];
    t = [1.0; 1.5; 2.0; 2.5];
    P = Pc*exp((a*(((Tc-T)./T).^t)).*Tr);

    a = [1.0000, 1.8145364, 1.2703194, -1.0182657, 2.7562720, ...
         -1.8958692];
    t = [0; 1/3; 2/3; 4/3; 11/3; 13/3];
    VL = Vc./(a*((1-Tr).^t));

    a = [-9.74320519e0,6.27207077e1,-7.11162934e2,-1.28732378e4, ...
          9.54090636e3,...
         -8.06229292e4,8.90380974e4,-2.68827787e4, 1.53392797e4,-...
          1.39186128e4,...
         -8.77428382e3,9.25853159e3,-7.36156195e3];
    t = [1/6;1/3;2/3;7/6;3/2;5/2;8/3;7/2;5;6;15;21;1];
    Tx = (1-Tr).^t; Tx(13) = log(Tr);
    VV = Vc./exp(a*Tx);

case 13 % n-Pentane
    Tc = 469.70; Pc = 33.70; Vc = 72.150/232.00; Tr = T/Tc;

    a = [-7.472,2.849,-2.564,-3.093];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [1.4,-0.433,0.3496,0.03852];
    t = [0.3;1.3;1.7;2.5];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.911,-4.888,-1.78,-14.07,-16.1,-14.02,-79.47];
    t = [0.33;0.9;1.4;2.9;4.6;6.2;10];
    VV = Vc./exp(a*((1-Tr).^t));

case 14 % n-Hexane
    Tc = 507.82; Pc = 30.34; Vc = 86.177/233.18; Tr = T/Tc;

    a = [-7.686,3.001,-2.963,-3.264];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [1.247,0.7429,-2.672,3.069,-1.089];
    t = [0.3;0.6;1.2;1.7;2.5];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.727,-6.998,2.046,-24.97,12.78,-55.39,-45.12];
    t = [0.33;0.9;1.4;2.9;4.6;6.2;10];
    VV = Vc./exp(a*((1-Tr).^t));

case 15 % n-Heptane
    Tc = 540.13; Pc = 27.36; Vc = 100.204/232.00; Tr = T/Tc;

    a = [-7.972,3.243,-3.322,-3.909];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));
```

```
    a = [1.131,1.136,-3.071,3.01,-0.845];
    t = [0.3;0.6;1.2;1.7;2.5];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.805,-6.177,-0.6594,-16.99,-17.34,-18.71,-87.45];
    t = [0.33;0.9;1.4;2.9;4.6;6.2;10];
    VV = Vc./exp(a*((1-Tr).^t));

case 16 % n-Octane
    Tc = 569.32; Pc = 24.97; Vc = 114.231/234.90; Tr = T/Tc;

    a = [-8.194,3.428,-3.713,-4.338];
    t = [1;1.5;2;4.5];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [0.781,2.315,-6.12,6.53,-2.271];
    t = [0.3;0.6;1.2;1.7;2.5];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-1.736,-8.341,4.202,-33.7,33.9,-94.21,-17.91];
    t = [0.33;0.9;1.4;2.9;4.6;6.2;10];
    VV = Vc./exp(a*((1-Tr).^t));

case 17 % Hydrogen
    Tc = 33.145; Pc = 12.964; Vc = 1/15.508; Tr = T/Tc;

    a = [-4.348, 2.763, -1.538, 0.021];
    t = [0.9713; 1.99; 1.824; 0.6145];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [-2.809, 4.977, -0.0219, -1.126];
    t = [0.4093; 0.3304; 0.9672; 0.2301];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [-23, -10.65, -5.479, -2.915];
    t = [9.964;4.521;1.654;0.4608];
    VV = Vc./exp(a*((1-Tr).^t));

case 18 % Acetone
    Tc =  508.1; Pc = 47.01; Vc = 1/15.508; Tr = T/Tc;

    a = [-7.265,0.9567,-0.6738,-3.037];
    t = [1;1.5;2;4];
    P = Pc*exp(a*(((1-Tr).^t)./Tr));

    a = [0, 0, 0, 0];
    t = [0; 0; 0; 0];
    VL = Vc./exp(a*((1-Tr).^t));

    a = [0, 0, 0, 0];
    t = [0; 0; 0; 0];
    VV = Vc./exp(a*((1-Tr).^t));

case 19 % Tetrafluoromethane
```

```
        Tc =  227.51; Pc = 37.50; Vc = 88.010/625.7; Tr = T/Tc; tau = 1-T/Tc;
        MW = 88.010; rho_c = 625.7;

        a = [-6.769991,1.116643,-1.879338,-2.466954];
        t = [2;3;6;12]/2;
        P = Pc*exp((Tc/T)*(a*tau.^t));

        a = [1.576968,0.8385832,0.2698069];
        t = [1;2;9]/3;
        VL = 1/((rho_c*(a*tau.^t)+rho_c)/88.010);

        a = [-1.704,-6.273,2.337,-27.53,42.63,-101.4,36.21];
        t = [0.33;0.9;1.4;2.9;4.6;6.2;10];
        VV = Vc./exp(a*((1-Tr).^t));

    case 22 % Toluene
        Tc =  591.75; Pc = 41.08; Vc = 0.3160; Tr = T/Tc;

        a = [-6.931,0.7013,-0.121,-4.205,-0.06422];
        t = [1;1.5;2;4;0.666];
        P = Pc*exp(a*(((1-Tr).^t)./Tr));

        a = [0, 0, 0, 0];
        t = [0; 0; 0; 0];
        VL = Vc./exp(a*((1-Tr).^t));

        a = [0, 0, 0, 0];
        t = [0; 0; 0; 0];
        VV = Vc./exp(a*((1-Tr).^t));
end

% ************************************************************************
```

Appendix A.7 - Pure Component Molar Volume Reference Data

This script "vol_data" is a compilation and molar volume data for various fluids. Arguments required for this script include the fluid index, volume, and temperature of interest. Most of these reference correlations are formulated as explicit in the residual Helmholtz free energy, so the pressure is returned as a function of the temperature and volume.

```
function [P,H,S,U] = vol_data(V,T,substance)
% ***********************************************************************
% P (bar):           Pressure of selected substance.
% H (kJ/mol):        Enthalpy of selected substance. (U = 0 @ 273.16 K in L_sat)
% Cp(J/mol/K):       Constant pressure heat capacity.
% V (m^3/kmol):      Volume of selected substance.
% T (K):             Temperature selected substance.
% substance:                Min (        K /    Tr)   Max (   K /    Tr)   Max (   bar /    Pr )
%               1 = H2O          273.16   / 0.423         1273 /  1.967         10 000 /   45.32
%               2 = CO2          216.592  / 0.712         1500 /  4.932          8 000 /  108.44
%               3 = CH4           90.6941 / 0.476          625 /  3.280         10 000 /  217.42
%               4 = C2H6          90.352  / 0.296          625 /  2.05             700 /   14.368
%               5 = C3H8          85.48   / 0.231          643 /  1.739          1 030 /   24.25
%               6 = n-C4H10      134.87   / 0.317          589 /  1.385            690 /   18.18
%               7 = NH3          195.495  / 0.482          750 /  1.850          5 000 /   44.11
%               8 = N2            63.151  / 0.500         1000 /  7.924         22 000 /  647.86
%               9 = O2            54.361  / 0.352          300 /  1.941            818 /   16.22
%              10 = Ar            83.8058 / 0.556          700 /  4.645         10 000 /  205.63
%              11 = H2S          187.67   / 0.503          760 /  2.04           1 700 /   18.97
%              12 = CH3OH        175.61   / 0.342          620 /  1.208          8 000 /   98.72
%              13 = n-C5H12      143.4    / 0.305          573 /  1.220            690 /   20.47
%              14 = n-C6H14      263.     / 0.518          548 /  1.079            920 /   30.32
%              15 = n-C7H16      182.6    / 0.338          523 /  1.283           1000 /   36.55
%              16 = n-C8H18      258.     / 0.453          548 /  0.963            960 /   38.45
%              17 = H2            13.957  / 0.453         1000 / 30.17          20 000 / 1542.73
%              18 = CO(CH3)2      --      /   --            -- /    --              -- /   --
%              19 = CF4           98.94   / 0.435          623 /  2.739            507 /   13.52
% ***********************************************************************
```

```
switch (substance)
    case 1 % Water
        MW = 18.01527; rho = 1/(V)*MW; R = 0.46151805;
        Pc = 22.0640; rhoc = 322; Tc = 647.096;
        Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

        gam0 = [1.28728967; 3.53734222; 7.74073708; 9.24437796; 27.5075105];
        a0   = [-8.32044648201, 6.6832105268, 3.00632, 0.012436, 0.97315, 1.27950, 0.96956, 0.24873];

        d1 = [1;1;1;2;2;3;4]; t1 = [-0.5;0.875;1;0.5;0.75;0.375;1];
        a1 = [ 0.12533547935523e-1, 0.78957634722828e1  ,-0.87803203303561e1 ,0.31802509345418e0,...
              -0.26145533859358e0 ,-0.78199751687981e-2 ,0.88089493102134e-2];

        d2 = [1;1;1;2;2;3;4;4;5;7;9;10;11;13;15;1;2;2;2;3;4;4;4;5;6;6;...
              7;9;9;9;9;9;10;10;12;3;4;4;5;14;3;6;6;6];
        t2 = [4;6;12;1;5;4;2;13;9;3;4;11;4;13;1;7;1;9;10;10;3;7;10;10;...
              6;10;10;1;2;3;4;8;6;9;8;16;22;23;23;10;50;44;46;50];
        c2 = [1;1;1;1;1;1;1;1;1;1;1;1;1;1;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;3;3;3;3;4;6;6;6;6];
        a2 = [-0.66856572307965e0 , 0.20433810950965e0  ,-0.66212605039687e-4,-0.19232721156002e0 ,...
              -0.25709043003438e0 , 0.16074868486251e0  ,-0.40092828925807e-1, 0.39343422603254e-6,...
              -0.75941377088144e-5, 0.56250979351888e-3 ,-0.15608652257135e-4, 0.11537996422951e-8,...
               0.36582165144204e-6,-0.13251180074668e-11,-0.62639586912454e-9,-0.10793600908932e0 ,...
               0.17611491008752e-1, 0.22132295167546e0  ,-0.40247669763528e0 , 0.58083399985759e0 ,...
               0.49969146990806e-2,-0.31358700712549e-1 ,-0.74315929710341e0 , 0.47807329915480e0 ,...
               0.20527940895948e-1,-0.13636435110343e0  , 0.14180634400617e-1, 0.83326504880713e-2,...
              -0.29052336009585e-1, 0.38615085574206e-1 ,-0.20393486513704e-1,-0.16554050063734e-2,...
               0.19955571979541e-2, 0.15870308324157e-3 ,-0.16388568342530e-4, 0.43613615723811e-1,...
               0.34994005463765e-1,-0.76788197844621e-1 , 0.22446277332006e-1,-0.62689710414685e-4,...
              -0.55711118565645e-9,-0.19905718354408e0  ,0.31777497330738e0  ,-0.11841182425981e0];

        d3 = [3;3;3]; eta3 = [20;20;20]; beta3 = [150;150;250];
        t3 = [0;1;4]; gamma3 = [1.21;1.21;1.25];
        a3 = [-0.31306260323435e2 , 0.31546140237781e2  ,-0.25213154341695e4];

        alpha4 = [3.5;3.5]; beta4 = [0.3;0.3]; b4 = [0.85;0.95];
        A4 = [0.32;0.32]; B4 = [0.2;0.2]; C4 = [28;32]; D4 = [700;800];
        a4 = [-0.14874640856724e0,0.31806110878444e0];

        THETA = (1-tau)+A4.*((delta-1)*(delta-1)).^(1./(2*beta4));
```

```
DELTA = THETA.^2 + B4.*((delta-1).*(delta-1)).^(alpha4);
PSI   = exp(-C4.*((delta-1)*(delta-1))-D4.*((tau-1)*(tau-1)));

delDELTAdelta      = (delta-1)*(A4.*THETA.*2./beta4.*(((delta-1)*(delta-1)).^...
                     ((1./(2*beta4))-1)) + ...
                     2*B4.*alpha4.*(((delta-1).*(delta-1)).^(alpha4-1)));
delDELTAdeltadelta = 1./(delta-1).*delDELTAdelta+(delta-1).^2.*(4*B4.*alpha4.*...
      (alpha4-1).*((delta-1).^2).^(alpha4-2) + ...
                          2*A4.*A4./beta4./beta4.*(((delta-1).^2).^(1/2./beta4-1)).^2 + ...
                          A4.*THETA.*4./beta4.*(1/2./beta4-1).*((delta-1).^2).^(1/2./beta4-2));

delPSIdelta = -2*C4*(delta-1).*PSI;    delDELTAbdelta = b4.*(DELTA.^(b4-1)).*delDELTAdelta;
delPSItau   = -2*D4*(tau-1).*PSI;      delDELTAbtau   = -2*THETA.*b4.*(DELTA.^(b4-1));

delPSIdeltatau    = 4*C4.*D4.*(delta-1).*(tau-1).*PSI;
delPSItautau      = (2*D4.*(tau-1).^2-1).*2.*D4.*PSI;
delPSIdeltadelta  = (2*C4.*(delta-1).^2-1).*2.*C4.*PSI;

delDELTAbdeltatau   = -A4.*b4*2./beta4.*DELTA.^(b4-1).*(delta-1).*...
                      ((delta-1).^2).^(1./(2.*beta4)-1) - ...
                      2*THETA.*b4.*(b4-1).*DELTA.^(b4-2).*delDELTAdelta;
delDELTAbtautau     = 2*b4.*DELTA.^(b4-1)+4*THETA.*THETA.*b4.*(b4-1).*DELTA.^(b4-2);
delDELTAbdeltadelta = b4.*(DELTA.^(b4-1).*delDELTAdeltadelta+(b4-1).*DELTA.^(b4-...
                      2).*(delDELTAdelta).^2);


phir  = a1*((delta.^d1).*(tau.^t1)) + ...
        a2*(exp(-delta.^c2).*(tau.^t2).*(delta.^d2)) + ...
        a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2))) + ...
        a4*((DELTA.^b4).*PSI.*delta);

phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
        a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1))) + ...
        a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
           (d3./delta - 2*eta3*(delta-1))) + ...
        a4*((DELTA.^b4).*(PSI+delta*delPSIdelta)+delta*PSI.*delDELTAbdelta);

phirt = a1*(t1.*tau.^(t1-1).*(delta.^d1)) + ...
        a2*(exp(-delta.^c2).*t2.*(delta.^d2).*(tau.^(t2-1))) + ...
        a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
```

```
          (t3./tau-2*beta3.*(tau-gamma3))) + ...
        a4*(delta.*((DELTA.^b4).*delPSItau+PSI.*delDELTAbtau));

phinot  = log(delta) + a0(1) + a0(2)*tau + a0(3)*log(tau) + a0(4:8)*log(1-exp(-gam0*tau));

phinottt =  - a0(3)/tau/tau - a0(4:8)*(gam0.*gam0.*exp(-gam0*tau).*((1-exp(-gam0*tau)).^-2));

phinott  = a0(2) + a0(3)/tau     + a0(4:8)*(gam0.*((1-exp(-gam0.*tau)).^(-1)-1));

phirtt = a1*(t1.*(t1-1).*delta.^d1.*tau.^(t1-2)) + ...
         a2*(t2.*(t2-1).*delta.^d2.*tau.^(t2-2).*exp(-delta.^c2)) + ...
         a3*(delta.^d3.*tau.^t3.*exp(-eta3.*(delta-1).^2-beta3.*(tau-gamma3).^2).* ...
             ((t3./tau-2*beta3.*(tau-gamma3)).^2-t3./tau./tau-2*beta3)) + ...
         a4*(delta.*(delDELTAbtautau.*PSI+2*delDELTAbtau.*delPSItau+DELTA.^b4.*delPSItautau));

phirdt = a1*(d1.*t1.*(tau.^(t1-1)).*(delta.^(d1-1))) + ...
         a2*(t2.*(tau.^(t2-1)).*(delta.^(d2-1)).*(d2-c2.*delta.^c2).*exp(-delta.^c2)) + ...
         a3*(delta.^d3.*tau.^t3.*exp(-eta3.*(delta-1).^2-beta3.*(tau-gamma3).^2).* ...
             (d3./delta-2*eta3.*(delta-1)).*(t3./tau-2*beta3.*(tau-gamma3))) + ...
         a4*(DELTA.^b4.*(delPSItau+delta.*delPSIdeltatau)+delta.*delDELTAbdelta.*delPSItau+ ...
             delDELTAbtau.*(PSI+delta.*delPSIdelta)+delDELTAbdeltatau.*delta.*PSI);

phirdd = a1*(d1.*(d1-1).*delta.^(d1-2).*tau.^t1) + ...
         a2*(exp(-delta.^c2).*(delta.^(d2-2).*tau.^t2.*((d2-c2.*delta.^c2).*(d2-1-...
             c2.*delta.^c2)-c2.*c2.*delta.^c2))) + ...
         a3*(tau.^t3.*exp(-eta3.*(delta-1).^2-beta3.*(tau-gamma3).^2).* ...
             (-2.*eta3.*delta.^d3+4.*eta3.^2.*delta.^d3.*(delta-1).^2- ...
             4.*d3.*eta3.*delta.^(d3-1).*(delta-1) +d3.*(d3-1).*delta.^(d3-2))) + ...
         a4*(DELTA.^b4.*(2*delPSIdelta+delta.*delPSIdeltadelta) + ...
             2*delDELTAbdelta.*(PSI+delta.*delPSIdelta) + ...
             delDELTAbdeltadelta.*delta.*PSI);

U = tau*(phinott+phirt)*MW*R*T/1000;
S = tau*(phinott+phirt)*MW*R - phinot*R*MW - phir*R*MW;
P  = (1         + delta*phird)*rho*T*R/100;
H  = (1 + tau*phinott + tau*phirt + delta*phird)*R*T*MW/1000;
Cp = (-tau*tau*(phinottt + phirtt) + ...
     ((1+delta*phird-delta*tau*phirdt)^2)/(1+2*delta*phird+delta*delta*phirdd))*R*MW;
```

```
case 2 % Carbon Dioxide
    MW = 44.0095; rho = 1/(V)*MW; R = 0.1889241;
    Pc = 7.3773; rhoc = 467.6; Tc = 304.1282;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;1;2;2;3]; t1 = [0;0.75;1;2;0.75;2;0.75];
    a1 = [ 0.38856823203161e0 , 0.29385475942740e1 ,-0.55867188534934e1 ,-0.76753199592477e0, ...
           0.31729005580416e0 , 0.54803315897767e0 , 0.12279411220335e0];

    d2 = [1;2;4;5;5;5;6;6;6;1;1;4;4;4;7;8;2;3;3;5;5;6;7;8;10;4;8];
    t2 = [1.5;1.5;2.5;0;1.5;2;0;1;2;3;6;3;6;8;6;0;7;12;16;22;24;16;24;8;2;28;14];
    c2 = [1;1;1;1;1;1;1;1;1;2;2;2;2;2;2;2;3;3;3;4;4;4;4;4;5;6];
    a2 = [ 0.21658961543220e1 , 0.15841735109724e1 ,-0.23132705405503e0 , 0.58116916431436e-1,...
          -0.55369137205382e0 , 0.48946615909422e0 ,-0.24275739843501e-1, 0.62494790501678e-1,...
          -0.12175860225246e0 ,-0.37055685270086e0 ,-0.16775879700426e-1,-0.11960736637987e0 ,...
          -0.45619362508778e-1, 0.35612789270346e-1,-0.74427727132052e-2,-0.17395704902432e-2,...
          -0.21810121289527e-1, 0.24332166559236e-1,-0.37440133423463e-1, 0.14338715756878e0 ,...
          -0.13491969083286e0 ,-0.23151225053480e-1, 0.12363125492901e-1, 0.21058321972940e-2,...
          -0.33958519026368e-3, 0.55993651771592e-2,-0.30335118055646e-3];

    d3 = [2;2;2;3;3]; eta3 = [25;25;25;15;20]; beta3 = [325;300;300;275;275];
    t3 = [1;0;1;3;3]; gamma3 = [1.16;1.19;1.19;1.25;1.22];
    a3 = [-0.21365488688320e3 , 0.26641569149272e5 ,-0.24027212204557e5 ,-0.28341603423999e3 ,...
           0.21247284400179e3];

    alpha4 = [3.5;3.5;3]; beta4 = [0.3;0.3;0.3]; b4 = [0.875;0.925;0.875];
    A4 = [0.7;0.7;0.7]; B4 = [0.3;0.3;1]; C4 = [10;10;12.5]; D4 = [275;275;275];
    a4 = [-0.66642276540751e0,0.72608632349897e0,0.55068668612842e-1];

    THETA = (1-tau)+A4.*((delta-1)*(delta-1)).^(1./(2*beta4));
    DELTA = THETA.^2 + B4.*((delta-1).*(delta-1)).^(alpha4);
    PSI = exp(-C4.*((delta-1)*(delta-1))-D4.*((tau-1)*(tau-1)));
    delDELTAdelta = (delta-1)*(A4.*THETA.*2./beta4.*(((delta-1)*(delta-1)).^((1./(2*beta4))-1)) ...
                    + 2*B4.*alpha4.*(((delta-1).*(delta-1)).^(alpha4-1)));
    delPSIdelta = -2*C4*(delta-1).*PSI; delDELTAbdelta = b4.*(DELTA.^(b4-1)).*delDELTAdelta;
    delPSItau   = -2*D4*(tau-1).*PSI;   delDELTAbtau   = -2*THETA.*b4.*(DELTA.^(b4-1));

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
```

```
            (d3./delta - 2*eta3*(delta-1))) + ...
          a4*((DELTA.^b4).*(PSI+delta*delPSIdelta)+delta*PSI.*delDELTAbdelta);
    phirt = a1*(t1.*(delta.^d1).*tau.^(t1-1)) + ...
          a2*(exp(-delta.^c2).*t2.*(delta.^d2).*(tau.^(t2-1))) + ...
          a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
            (t3./tau-2*beta3.*(tau-gamma3))) + ...
          a4*(delta.*((DELTA.^b4).*delPSItau+PSI.*delDELTAbtau));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

case 3 % Methane
    MW = 16.0426; rho = 1/(V)*MW; R = 0.5182705;
    Pc = 4.5992; rhoc = 162.66; Tc = 190.564;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;2;2;2;3;4;4;8;9;10]; t1 = [-0.5;0.5;1;0.5;1;1.5;4.5;0;1;3;1;3;3];
    a1 = [ 0.4367901028e-1, 0.6709236199e0 ,-0.1765577859e1 , 0.8582330241e0 ,...
          -0.1206513052e1 , 0.5120467220e0 ,-0.4000010791e-3,-0.1247842423e-1,...
           0.3100269701e-1, 0.1754748522e-2,-0.3171921605e-5,-0.2240346840e-5,...
           0.2947056156e-6];

    d2 = [1;1;1;2;4;5;6;1;2;3;4;4;3;5;5;8;2;3;4;4;4;5;6];
    t2 = [0;1;2;0;0;2;2;5;5;5;2;4;12;8;10;10;10;14;12;18;22;18;14];
    c2 = [1;1;1;1;1;1;1;2;2;2;2;2;3;3;3;3;4;4;4;4;4;4;4];
    a2 = [ 0.1830487909e0 , 0.1511883679e0 ,-0.4289363877e0 , 0.6894002446e-1,...
          -0.1408313996e-1,-0.3063054830e-1,-0.2969903708e-1,-0.1932040831e-1,...
          -0.1105739959e0 , 0.9952548995e-1, 0.8548437825e-2,-0.6150555662e-1,...
          -0.4291792423e-1,-0.1813207290e-1, 0.3445904760e-1,-0.2385919450e-2,...
          -0.1159094939e-1, 0.6641693602e-1,-0.2371549590e-1,-0.3961624905e-1,...
          -0.1387292044e-1, 0.3389489599e-1,-0.2927378753e-2];

    d3 = [2;0;0;0]; eta3 = [20;40;40;40]; beta3 = [200;250;250;250];
    t3 = [2;0;1;2]; gamma3 = [1.07;1.11;1.11;1.11];
    a3 = [ 0.9324799946e-4,-0.6287171518e1 , 0.1271069467e2 ,-0.6423953466e1];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
          a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1))) + ...
          a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
            (d3./delta - 2*eta3.*(delta-1)));
```

```
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*t2.*(delta.^d2).*(tau.^(t2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
               (t3./tau-2*beta3.*(tau-gamma3)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

case 4 % Ethane
    MW = 30.070; rho = 1/(V); R = 8.314510;
    Pc = 4.8718; rhoc = 6.87; Tc = 305.33;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;2;2;3;3;3;6;7;7;8]; t1 = [0;1.5;2.5;-0.5;1.5;2;0;1;2.5;0;2;5;2];
    a1 = [ 0.46215430560e0 ,-0.19236936387e1 , 0.39878604003e0 , 0.16054532372e-1, ...
           0.12895242219e0 , 0.35458320491e-1, 0.34927844540e-1,-0.11306183380e-1, ...
          -0.39809032779e-1, 0.83031936834e-3, 0.45921575183e-3, 0.17530287917e-6, ...
          -0.70919516126e-4];

    d2 = [1;1;2;2;3;3;5;6;7;8;10;2;3;3;4;4;5;5;5];
    t2 = [5;6;3.5;5.5;3;7;6;8.5;4;6.5;5.5;22;11;18;11;23;17;18;23];
    c2 = [2;2;2;2;2;2;2;2;2;2;2;4;4;4;4;4;4;4;4];
    a2 = [-0.23436162249e0 , 0.84574697645e-1, 0.14861052010e0 ,-0.10016857867e0 , ...
          -0.59264824388e-1,-0.41263514217e-1, 0.21855161869e-1,-0.74552720958e-4, ...
          -0.98859085572e-2, 0.10208416499e-2,-0.52189655847e-3, 0.98592162030e-4, ...
           0.46865140856e-1,-0.19558011646e-1,-0.46557161651e-1, 0.32877905376e-2, ...
           0.13572090185e0 ,-0.10846471455e0 ,-0.67502836903e-2];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 5 % Propane
    MW = 44.09562; rho = 1/(V)*MW; R = 8.314472/MW;
    Pc = 4.24709; rhoc = 218.5; Tc = 369.825;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;
```

```
    d1 = [1;1;2;2;3;5;8;8]; t1 = [-0.25;1.5;-0.75;0;1.25;1.5;0.5;2.5];
    a1 = [ 2.698378e-1,-1.339252e0 ,-2.273858e-2, 2.414973e-1,-3.321461e-2,...
           2.203323e-3, 5.935588e-5,-1.137457e-6];

    d2 = [3;3;8;5;6;1;5;7;2;3;15];
    t2 = [1.5;1.75;-0.25;3;3;4;2;-1;2;19;5];
    c2 = [1;1;1;1;1;2;2;2;3;3;3];
    a2 = [-2.379299e0 , 2.337373e0 , 1.242344e-3,-7.352787e-3, 1.965751e-3,...
          -1.402666e-1,-2.093360e-2,-2.475221e-4,-1.482723e-2,-1.303038e-2,...
           3.634670e-5];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

case 6 % n-Butane
    MW = 58.1222; rho = 1/(V)*MW; R = 8.314472;
    Pc = 3.796; rhoc = 227.84; Tc = 425.125;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;2;2;3;5;8;8]; t1 = [-0.25;1.5;-0.75;0;1.25;1.5;0.5;2.5];
    a1 = [ 2.952054e-1,-1.326360e0 ,-2.031317e-3, 2.240301e-1, ...
          -3.635425e-2, 1.905841e-3, 7.409154e-5,-1.401175e-6];

    d2 = [3;3;8;5;6;1;5;7;2;3;15];
    t2 = [1.5;1.75;-0.25;3;3;4;2;-1;2;19;5];
    c2 = [1;1;1;1;1;2;2;2;3;3;3];
    a2 = [-2.492172e0 , 2.386920e0 , 1.424009e-3,-9.393388e-3, ...
           2.616590e-3,-1.977323e-1,-3.809534e-2, 1.523948e-3, ...
          -2.391345e-2,-9.535229e-3, 3.928384e-5];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));
```

```
    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 7 % Ammonia
    MW = 17.0302; rho = 1/(V)*MW; R = 8.313956/MW;
    Pc = 11.333; rhoc = 225; Tc = 405.40;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [2;1;4;1;15]; t1 = [-1/2;1/2;1;3/2;3];
    a1 = [ 0.4554431e-1, 0.7238548e0 , 0.1229470e-1,-0.1858814e1 , 0.2141882e-10];

    d2 = [3;3;1;8;2;1;8;1;2;3;2;4;3;1;2;4];
    t2 = [0;3;4;4;5;3;5;6;8;8;10;10;5;15/2;15;30];
    c2 = [1;1;1;1;1;2;2;2;2;2;2;2;3;3;3;3];
    a2 = [-0.1430020e-1, 0.3441324e0 ,-0.2873571e0 , 0.2352589e-4,-0.3497111e-1,...
           0.2397852e-1, 0.1831117e-2,-0.4085375e-1, 0.2379275e0 ,-0.3548972e-1,...
          -0.1823729e0 , 0.2281556e-1,-0.6663444e-2,-0.8847486e-2, 0.2272635e-2,...
          -0.5588655e-3];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

case 8 % Nitrogen
    MW = 28.01348; rho = 1/(V); R = 8.31451;
    Pc = 3.3958; rhoc = 11.1839; Tc = 126.192;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;2;2;3;3]; t1 = [0.25;0.875;0.5;0.875;0.375;0.75];
    a1 = [ 0.924803575275e0, -0.492448489428e0, 0.661883336938e0, -0.192902649201e1, ...
          -0.622469309629e-1, 0.349943957581e0];

    d2 = [1;1;1;3;3;4;6;6;7;7;8;8;1;2;3;4;5;8;4;5;5;8;3;5;6;9];
    t2 = [0.5;0.75;2.0;1.25;3.5;1.0;0.5;3.0;0.0;2.75;0.75;2.5; ...
          4.0;6.0;6.0;3.0;3.0;6.0;16.0;11.0;15.0;12.0;12.0;7.0;4.0;16.0];
```

```
    c2 = [1;1;1;1;1;1;1;1;1;1;1;2;2;2;2;2;2;3;3;3;3;4;4;4;4];
    a2 = [ 0.564857472498e0 ,-0.161720005987e1 ,-0.481395031883e0 , 0.421150636384e0 , ...
          -0.161962230825e-1, 0.172100994165e0 , 0.735448924933e-2, 0.168077305479e-1, ...
          -0.107626664179e-2,-0.137318088513e-1, 0.635466899859e-3, 0.304432279419e-2, ...
          -0.435762336045e-1,-0.723174889316e-1, 0.389644315272e-1,-0.212201363910e-1, ...
           0.408822981509e-2,-0.551990017984e-4,-0.462016716479e-1,-0.300311716011e-2, ...
           0.368825891208e-1,-0.255856846220e-2, 0.896915264558e-2,-0.441513370350e-2, ...
           0.133722924858e-2, 0.264832491957e-3];

    d3 = [1;1;3;2]; eta3 = [20;20;15;25]; beta3 = [325;325;300;275];
    t3 = [0.0;1.0;2.0;3.0]; gamma3 = [1.16;1.16;1.13;1.25];
    a3 = [ 0.196688194015e2 ,-0.209115600730e2 , 0.167788306989e-1, 0.262767566274e4 ];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
               (d3./delta - 2*eta3*(delta-1)));
    phirt = a1*(t1.*tau.^(t1-1).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*t2.*(delta.^d2).*(tau.^(t2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.*((tau-gamma3).^2)).* ...
               (t3./tau-2*beta3.*(tau-gamma3)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 9 % Oxygen
    MW = 31.9988; rho = 1/(V); R = 8.31434;
    Pc = 5.0430; rhoc = 13.63; Tc = 154.581;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;2;2;3;3;3;6;7;7;8]; t1 = [0;1.5;2.5;-0.5;1.5;2.0;0.0;1.0;2.5;0.0;2.0;5.0;2.0];
    a1 = [ 0.3983768749e0 ,-0.1846157454e1 , 0.4183473197e0 , 0.2370620711e-1, ...
           0.9771730573e-1, 0.3017891294e-1, 0.2273353212e-1, 0.1357254086e-1, ...
          -0.4052698943e-1, 0.5454628515e-3, 0.5113182277e-3, 0.2953466883e-6, ...
          -0.8687645072e-4];

    d2 = [1;1;2;2;3;3;5;6;7;8;10;2;3;3;4;4;5;5;5];
```

```
    t2 = [5.0;6.0;3.5;5.5;3.0;7.0;6.0;8.5;4.0;6.5;5.5;22;11;18;11;23;17;18;23];
    c2 = [2;2;2;2;2;2;2;2;2;2;2;4;4;4;4;4;4;4;4];
    a2 = [-0.2127082589e0 , 0.8735941958e-1, 0.1275509190e0 ,-0.9067701064e-1, ...
          -0.3540084206e-1,-0.3623278059e-1, 0.1327699290e-1,-0.3254111865e-3, ...
          -0.8313582932e-2, 0.2124570559e-2,-0.8325206232e-3,-0.2626173276e-4, ...
           0.2599581482e-2, 0.9984649663e-2, 0.2199923153e-2,-0.2591350486e-1, ...
          -0.1259630848e0 , 0.1478355637e0 ,-0.1011251078e-1];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 10 % Argon
    MW = 39.948; rho = 1/(V)*MW; R = 8.314472/MW;
    Pc = 4.8630; rhoc = 535.6; Tc = 150.687;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;1;1;2;2;2;2;3;3;4]; t1 = [0;0.25;1;2.75;4;0;0.25;0.75;2.75;0;2;0.75];
    a1 = [ 0.88722304990011e-1, 0.70514805167298e0 ,-0.16820115654090e1 ,-0.14909014431486e0 ,...
          -0.12024804600940e0 ,-0.12164978798599e0 , 0.40035933626752e0 ,-0.27136062699129e0 ,...
           0.24211924579645e0 , 0.57889583185570e-2,-0.41097335615341e-1, 0.24710761541614e-1];

    d2 = [1;1;3;4;4;5;7;10;10;2;2;4;4;8;3;5;5;6;6;7;7;8;9;5;6];
    t2 = [3;3.50;1;2;4;3;0;0.50;1;1;7;5;6;6;10;13;14;11;14;8;14;6;7;24;22];
    c2 = [1;1;1;1;1;1;1;1;1;2;2;2;2;2;3;3;3;3;3;3;3;3;3;4;4];
    a2 = [-0.32181391750702e0 , 0.33230017695794e0 , 0.31019986287345e-1,-0.30777086002437e-1,...
           0.93891137419581e-1,-0.90643210682031e-1,-0.45778349276654e-3,-0.82659729025197e-4,...
           0.13013415603147e-3,-0.11397840001996e-1,-0.24455169960535e-1,-0.64324067175955e-1,...
           0.58889471093674e-1,-0.64933552112965e-3,-0.13889862158435e-1, 0.40489839296910e0 ,...
          -0.38612519594749e0 ,-0.18817142332233e0 , 0.15977647596482e0 , 0.53985518513856e-1,...
          -0.28953417958014e-1,-0.13025413381384e-1, 0.28948696775778e-2,-0.22647134304796e-2,...
           0.17616456196368e-2];

    d3 = [2;1;2;3]; eta3 = [20;20;20;20]; beta3 = [250;375;300;225];
    t3 = [3;1;0;0]; gamma3 = [1.11;1.14;1.17;1.11];
    a3 = [ 0.58552454482774e-2,-0.69251908270028e0 , 0.15315490030516e1 ,-0.27380447449783e-2];
```

```
    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.* ...
               ((tau-gamma3).^2)).*(d3./delta - 2*eta3.*(delta-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1))) + ...
            a3*((delta.^d3).*(tau.^t3).*exp(-eta3.*((delta-1).^2)-beta3.* ...
               ((tau-gamma3).^2)).*(t3./tau-2*beta3.*(tau-gamma3)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T*MW/1000;

case 11 % Hydrogen Sulfide
    MW = 34.0818; rho = 1/(V); R = 8.314472;
    Pc = 8.96291; rhoc = 10.20; Tc = 373.37;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;2;2;3;4;8;9;10]; t1 = [0.241;0.705;1;0.626;1.12;1.63;0.21;3.08;0.827;3.05;3.05];
    a1 = [ 0.1545780e0 ,-0.1717693e1 ,-0.1595211e1 , 0.2046589e1 , ...
          -0.1690358e1 , 0.9483623e0 ,-0.6800772e-1, 0.4372273e-2, ...
           0.3788552e-4,-0.3680980e-4, 0.8710726e-5];

    d2 = [1;1;1;2;5;1;4;4;3;8;2;3];
    t2 = [0.11;1.07;1.95;0.142;2.13;4.92;1.75;3.97;11.8;10;9.83;14.2];
    c2 = [1;1;1;1;1;2;2;2;3;3;4;4];
    a2 = [ 0.6886876e0 , 0.2751922e1 ,-0.1492558e1 , 0.9202832e0 , ...
          -0.2103469e0 , 0.1084359e-2, 0.3754723e-1,-0.5885793e-1, ...
          -0.2329265e-1,-0.1272600e-3,-0.1336824e-1, 0.1053057e-1];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T/1000;
```

```
case 12 % Methanol
    % **** Here be dragons ****
    MW = 32.04216; rho = 1/(V); R = 8.31448;
    Pc = 81.035; rhoc = 8.78517; Tc = 513.380;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;1;2;2;2;2;2;3;3;3;4;4;5;6;7]; t1 = [0;1;2;3;1;2;3;4;6;0;3;4;0;7;1;6;7];
    a1 = [-2.80062505988e0, 1.25636372418e1,-1.30310563173e1,3.26593134060e0,-4.11425343805e0 ,...
           3.46397741254e0,-8.36443967590e-2,-3.69240098923e-1,3.13180842152e-3,6.03201474111e-1,...
          -2.31158593638e-1,1.06114844945e-1,-7.92228164995e-2,-4.22419150975e-5,7.58196739214e-3,...
           -2.44617434701e-5,1.15080328802e-6];

    d2 = [1;1;1;1;2;2;2;2;3;4;5;5;5;5;6;9;6;6;4];    f1 = 1.008630309990;
    t2 = [1;2;3;4;1;2;3;5;1;2;1;2;4;5;2;5;9;14;19]; f2 = 0.998480657603;
    c2 = [2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;4;4;6];
    a2 = [-1.25099747447e1,2.73092835391e1,-2.12070717086e1, 6.32799472270e0 , 1.43687921636e1 ,...
          -2.87450766617e1,1.85397216068e1,-3.88720372879e0 ,-4.16602487963e0 , 5.29665875982e0 ,...
           5.09360272812e-1,-3.30257604839e0,-3.11045210826e-1, 2.73460830583e-1,5.18916583979e-1,...
           -2.27570803104e-3, 2.11658196182e-2,-1.14335123221e-2, 2.49860798459e-3];

    d3 = [1;1;1;1;1;3;3;3]; e3 = [3.9;3.9;3.9;3.9;3.9;23.1;23.1;23.1];
    c3 = [2;3;2;4;2;3;2;4]; beta3 = [2;2;3;3;4;3;4;4]; gamma3 = [6;6;6;6;6;25;25;25];
    a3 =[-8.19291988442e0,4.78601004557e-1,-4.44161392885e-1,1.79621810410e-1,-6.87602278259e-1,...
           2.40459848295e0 ,-6.88463987466e0 , 1.13992982501e0 ];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-(f1.*delta).^c2).*(d2-c2.*((f1.*delta).^c2)).*(tau.^t2).*(delta.^(d2-1)))+ ...
            a3*((delta.^(d3-1)).*exp(e3.*f2.*tau-gamma3-(beta3.*f1.*delta).^c3).*(d3-...
                 c3.*(beta3.*f1.*delta).^c3));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-(f1.*delta).^c2).*(delta.^d2).*t2.*(tau.^(t2-1))) + ...
            a3*(e3.*f2.*(delta.^d3).*exp(e3.*f2.*tau-gamma3-(beta3.*f1.*delta).^c3));

    P = (1         + delta*phird)*rho*T*R/100;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 13 % n-Pentane
    MW = 72.150; rho = 1/(V)*MW; R = 8.314510;
    Pc = 3.370; rhoc = 232.00; Tc = 469.70;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;
```

```
    d1 = [1;1;1;2;3;7]; t1 = [0.25;1.125;1.5;1.375;0.25;0.875];
    a1 = [0.10968643e1 ,-0.29988888e1 , 0.99516887e0,-0.16170709e0 , ...
          0.11334460e0 , 0.26760595e-3];

    d2 = [2;5;1;4;3;4];
    t2 = [0.625;1.75;3.625;3.625;14.5;12.0];
    c2 = [1;1;2;2;3;3];
    a2 = [ 0.40979882e0 ,-0.40876423e-1 ,-0.38169482e0 ,-0.10931957e0 , ...
          -0.32073223e-1, 0.16877016e-1];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 14 % n-Hexane
    MW = 86.177; rho = 1/(V)*MW; R = 8.314510;
    Pc = 3.034; rhoc = 233.18; Tc = 507.82;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;3;7]; t1 = [0.25;1.125;1.5;1.375;0.25;0.875];
    a1 = [ 0.10553238e1 ,-0.26120616e1 , 0.76613883e0 ,-0.29770321e0 , ...
           0.11879908e0 , 0.27922861e-3];

    d2 = [2;5;1;4;3;4];
    t2 = [0.625;1.75;3.625;3.625;14.5;12.0];
    c2 = [1;1;2;2;3;3];
    a2 = [ 0.46347590e0 , 0.11433197e-1,-0.48256969e0 ,-0.93750559e-1, ...
          -0.67273247e-2,-0.51141584e-2];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));
    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;
```

```
case 15 % n-Heptane
    MW = 100.204; rho = 1/(V)*MW; R = 8.314510;
    Pc = 2.736; rhoc = 232.00; Tc = 540.13;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;3;7]; t1 = [0.25;1.125;1.5;1.375;0.25;0.875];
    a1 = [0.10543748e1 ,-0.26500682e1 , 0.81730048e0 ,-0.30451391e0 , ...
          0.12253869e0 , 0.27266473e-3];

    d2 = [2;5;1;4;3;4];
    t2 = [0.625;1.75;3.625;3.625;14.5;12.0];
    c2 = [1;1;2;2;3;3];
    a2 = [ 0.49865826e0 ,-0.71432815e-3,-0.54236896e0 ,-0.13801822e0, ...
          -0.61595287e-2, 0.48602510e-3];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 16 % n-Octane
    MW = 114.231; rho = 1/(V)*MW; R = 8.314510;
    Pc = 2.497; rhoc = 234.90; Tc = 569.32;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d1 = [1;1;1;2;3;7]; t1 = [0.25;1.125;1.5;1.375;0.25;0.875];
    a1 = [ 0.10722545e1 ,-0.24632951e1 , 0.65386674e0 ,-0.36324974e0, ...
           0.12713270e0 , 0.30713573e-3];

    d2 = [2;5;1;4;3;4];
    t2 = [0.625;1.75;3.625;3.625;14.5;12.0];
    c2 = [1;1;2;2;3;3];
    a2 = [ 0.52656857e0 , 0.19362863e-1,-0.58939427e0 ,-0.14069964e0 , ...
          -0.78966331e-2, 0.33036598e-2];

    phird = a1*(d1.*(delta.^(d1-1)).*(tau.^t1)) + ...
            a2*(exp(-delta.^c2).*(d2-c2.*(delta.^c2)).*(tau.^t2).*(delta.^(d2-1)));
```

```
    phirt = a1*(t1.*(tau.^(t1-1)).*(delta.^d1)) + ...
            a2*(exp(-delta.^c2).*(delta.^d2).*t2.*(tau.^(t2-1)));

    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;

case 17 % Hydrogen
    MW = 2.0158; rho = 1/(V)*MW; R = 8.314510;
    Pc = 1.2964; rhoc = 15.508*MW; Tc = 33.145;
    Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

    d0 = [1; 4; 1; 1; 2; 2; 3];
    t0 = [0.6844; 1; 0.989; 0.489; 0.803; 1.1444; 1.409];
    a0 = [-6.93643,    0.01,       2.1101,    4.52059,    0.732564,...
          -1.34086,    0.130985];

    p1 = [1; 1]; d1 = [1; 3]; t1 = [1.754; 1.311]; a1 = [-0.777414,  0.351944 ];

    d2 = [2; 1; 3; 1; 1]; t2 = [4.187; 5.646; 0.791; 7.249; 2.986];
    a2 = [-0.0211716,  0.0226312,  0.032187, -0.0231752,  0.0557346];

    phi2   = [-1.685 ;  -0.489;  -0.103;  -2.506;  -1.607];
    beta2  = [-0.171 ; -0.2245; -0.1304; -0.2785; -0.3967];
    gamma2 = [ 0.7164;  1.3444;  1.4517;  0.7204;  1.5445];
    D2     = [ 1.506;   0.156;   1.736;   0.67;    1.662];

    phird = a0*(d0.*(delta.^(d0-1)).*(tau.^t0)) + ...
            a1*(exp(-delta.^p1).*(d1-p1.*(delta.^p1)).*(tau.^t1).*(delta.^(d1-1))) + ...
            a2*((delta.^d2).*(tau.^t2).*exp(phi2.*((delta-D2).^2)+beta2.*((tau-gamma2).^2)).* ...
               (d2./delta + 2*phi2.*(delta-D2)));

    phirt = a0*(t0.*tau.^(t0-1).*(delta.^d0)) + ...
            a1*(exp(-delta.^p1).*t1.*(delta.^d1).*(tau.^(t1-1))) + ...
            a2*((delta.^d2).*(tau.^t2).*exp(phi2.*((delta-D2).^2)+beta2.*((tau-gamma2).^2)).* ...
               (t2./tau + 2*beta2.*(tau-gamma2)));

    P = (1         + delta*phird)*rho*T*R/100/MW;
    H = (tau*phirt + delta*phird)*R*T/1000;
```

```
    case 18 % Acetone
        MW = 58.08;    rho = 1/(V)*MW;        R = 8.314510;
        Pc = 47.01;   rhoc = 1/(0.2090)*MW;  Tc = 508.1;

        P = R*T/V;
        H = 0;

    case 19 % Tetrafluoromethane
        MW = 88.010; rho = 1/(V); R = 8.314510;
        Pc = 37.5; rhoc = 625.7; Tc = 227.51;
        Tr = T/Tc; rhor = rho/rhoc; tau = 1/Tr; delta = rhor;

        a = [ 1.216275210e+00,  6.539895439e+02, -4.840432848e+04,  1.032226559e+07, ...
             -4.768213696e+08, -9.653189466e-02,  5.372779367e+01, -7.262485069e+02, ...
              1.572693239e-02, -6.176388369e+00, -3.350887856e-04,  1.643358038e-01, ...
              3.181774174e-03, -2.031922965e+06,  1.147536891e+09, -1.704558604e+11, ...
              1.456391292e+04, -1.246846225e+07,  2.785225717e+09,  1.975168825e-02];

        B = a(1) - a(2)/T - a(3)/T^2 - a(4)/T^3 - a(5)/T^4;
        C = a(6) + a(7)/T + a(8)/T^2; D = a(9) + a(10)/T;
        E = a(11) + a(12)/T; F = a(13)/T;
        G = a(14)/T^3 + a(15)/T^4 + a(16)/T^5;
        H = a(17)/T^3 + a(18)/T^4 + a(19)/T^5;

        P = rho*T*(R+B*rho+C*rho^2+D*rho^3+E*rho^4+F*rho^5+(G+H*rho^2)*rho^2*exp(-a(20)*rho^2))/1e2;
        H = 0;
end

% ************************************************************************
```

Appendix A.8 - Fugacity Coefficient Calculation

The fugacity coefficient was calculated by a script named "thermo". This script provided the fugacity coefficient from the SRK (1), PR (2), DMT (3), and PC-SAFT EOS (4). The desired model was selected by changing the value of *data_set* in line three. This script requires the pressure, temperature, composition, and fluid indicies of the desired mixture. Additionally, the most stable (0), liquid (1), or vapor (2) molar volume root could be selected through the argument *phase*.

The sixth argument, *alpha_bns*, was optional and only used when regressing parameters for the alpha corrections.

```
function [phi,gibbsE,molarV] = thermo(P,T,Z,index,phase,alpha_bns)

data_set = 3;

if(nargin <6); alpha_bns = 0; end
if(data_set==4);[phi,gibbsE,molarV] = poly_thermo(P,T,Z,index,phase); return; end;
X = FugInt1(P,T,Z,index,data_set,alpha_bns);V = FugInt3(X,T,Z,index,data_set);
phi = zeros(length(X),length(index)); R = 8.31451e-2; Nstep = 1e-6; Xstep = 1e-5*X;

for j1 = 1:length(V)
    for j2 = 1:length(index)
        Zup=Z; Zup(j2)=Zup(j2)+Nstep/2; Zdn=Z; Zdn(j2)=Zdn(j2)-Nstep/2;
        Xup=X(j1); Xup=Xup+Xstep(j1)/2; Xdn=X(j1); Xdn=Xdn-Xstep(j1)/2;
        phi(j1,j2)  = exp((FugInt2(X(j1),T,Zup,index,data_set,alpha_bns)-...
                           FugInt2(X(j1),T,Zdn,index,data_set,alpha_bns))/Nstep          - ...
                          ((FugInt2(Xup  ,T,Z  ,index,data_set,alpha_bns)-FugInt2(Xdn  ,T,Z  ...
                            ,index,data_set,alpha_bns))/Xstep(j1)) *     ...
                          ((FugInt3(X(j1),T,Zup,index,data_set)-...
                                FugInt3(X(j1),T,Zdn,index,data_set))/Nstep)      /     ...
                          ((FugInt3(Xup  ,T,Z  ,index,data_set)-FugInt3(Xdn  ,T,Z  ...
                               ,index,data_set))/Xstep(j1))    - ...
                          log(P*(V(j1))/sum(Z)/R/T)                                          );
    end
end
```

```
if(phase == 0)
    [gibbsE,phase] = min((R*T*log(phi))*(Z'/sum(Z))); molarV = V(phase);
elseif(phase == 1)
    [molarV,phase] = min(V);
elseif(phase == 2)
    [molarV,phase] = max(V);
end
phi = phi(phase,:);
gibbsE = ((R*T*log(phi))*(Z'/sum(Z)));

return;

% ************************************************

function F = FugInt2(X,T,Z,index,data_set,alpha_bns)

R = 8.31451e-2; N = sum(Z);

switch(data_set)
    case 1  % RKS Equation
        V = FugInt3(X,T,Z,index,data_set);
        ans_set = prop_data(index,1);
        Pc = ans_set.Pc; Tc = ans_set.Tc; omega = ans_set.omega;
        ac = R*R*Tc.*Tc./Pc/9/(2^(1/3)-1);
        m = (0.480+1.574*omega-0.176*omega.^2);
        alpha = (1+m.*(1-sqrt(T./Tc))).^2;
        param.a = ac.*(alpha + alpha_bns);
        param.b = (2^(1/3)-1)*R*Tc./Pc/3; param.T = T;
        ans_set = mix_rule(Z,param,11);
        a_mix = ans_set.a_mix; b_mix = ans_set.b_mix;
        F = N*log(V/N)-N*log(X/N-b_mix)+a_mix*N/R/T/b_mix*(log(X)-log(X+b_mix*N));
    case 2  % PR Equation
        V = FugInt3(X,T,Z,index,data_set);
        ans_set = prop_data(index,1);
        Pc = ans_set.Pc; Tc = ans_set.Tc; omega = ans_set.omega;
        ac = R*R*Tc.*Tc./Pc*0.45724;
        m = (0.37464+1.5422*omega-0.26992*omega.^2);
        alpha = (1+m.*(1-sqrt(T./Tc))).^2;
        param.a = ac.*(alpha + alpha_bns);
        param.b = R*Tc./Pc*0.07780; param.T = T;
```

```
    ans_set = mix_rule(Z,param,12);
    a_mix = ans_set.a_mix; b_mix = ans_set.b_mix;
    F = N*log(V/N)-N*log(X/N-b_mix)+sqrt(2)*a_mix*N/4/R/T/b_mix*...
        (log(X+(1-sqrt(2))*b_mix*N)-log(X+(1+sqrt(2))*b_mix*N));
case 3  % DMT Equation
    V = FugInt3(X,T,Z,index,data_set);
    ans_set = prop_data(index,5);
    Pc = ans_set.Pc; Tc = ans_set.Tc; kap = ans_set.kap;
    M = ans_set.M; Vc = ans_set.Vc;
    alpha = exp(kap(:,1).*(1-(T./Tc).^kap(:,2)));
    param.ac = R*R*Tc.*Tc./Pc/9/(2^(1/3)-1);
    param.a = param.ac.*(alpha + alpha_bns);
    param.b = (2^(1/3)-1)*R*Tc./Pc/3; param.M = M; param.Vc = Vc;
    param.tau = (exp(5*(1-sqrt(T./Tc)))-1).*(T<Tc); param.T = T;
    ans_set1 = mix_rule(Z,param,13); ans_set2 = mix_rule(Z,param,33);
    a_mix = ans_set1.a_mix; b_mix = ans_set1.b_mix; ac_mix = ans_set2.ac_mix;
    M_mix = ans_set2.M_mix; tau_mix = ans_set2.tau_mix;
    Tc_eff = ac_mix*(2^(1/3)-1)/3/R/b_mix*9*(2^(1/3)-1);
    Pc_eff = R*Tc_eff*(2^(1/3)-1)/3/b_mix; Vc_eff = R*Tc_eff/Pc_eff/3;
    F = N*log(V/(X-b_mix*N))+a_mix*N/R/T/b_mix*log(X/(X+b_mix*N));

    ncm  = [ 0;
             0;
             2*M_mix*R^3*T^3*(-7*T*R*Vc_eff+16*T*R*b_mix*tau_mix+8*T*R*b_mix*tau_mix^2-...
            3*T*R*tau_mix*Vc_eff-16*ac_mix*tau_mix-8*ac_mix*tau_mix^2);
             2*M_mix*R^2*T^2*(32*R^2*T^2*tau_mix^2*b_mix^2+9*T^2*R^2*b_mix*Vc_eff+96*R^2*T^2...
            *tau_mix*b_mix^2+13*T^2*R^2*b_mix*tau_mix*Vc_eff+32*R^2*T^2*b_mix^2-...
            32*R*T*ac_mix*b_mix*tau_mix-16*T*R*ac_mix*tau_mix*Vc_eff+7*T*R*a_mix*Vc_eff-...
            64*R*T*ac_mix*b_mix+3*T*R*a_mix*tau_mix*Vc_eff-16*T*R*ac_mix*Vc_eff-...
            8*a_mix*R*T*b_mix*tau_mix^2-...
            16*a_mix*R*T*b_mix*tau_mix+16*R*T*ac_mix*b_mix*tau_mix^2+32*ac_mix^2+16*a_mix*ac_mix...
            *tau_mix+8*a_mix*ac_mix*tau_mix^2+32*ac_mix^2*tau_mix);
             2*M_mix*R*T*(64*T*R*a_mix*ac_mix*b_mix+16*T*R*a_mix*ac_mix*tau_mix*Vc_eff-...
            32*ac_mix^3+208*R^3*T^3*b_mix^3-8*R^2*T^2*ac_mix*Vc_eff^2-...
            160*R^2*T^2*b_mix^2*ac_mix+32*R^2*T^2*b_mix*ac_mix*tau_mix*Vc_eff+8*R^3*T^3*b_mix*...
            Vc_eff^2+85*R^3*T^3*tau_mix*Vc_eff*b_mix^2+32*R^2*T^2*b_mix^2*ac_mix*tau_mix^2-...
            23*R^2*T^2*b_mix*ac_mix*Vc_eff+32*R*T*ac_mix^2*Vc_eff+208*R^3*T^3*b_mix^3*tau_mix-...
            112*R*T*ac_mix^2*b_mix*tau_mix+16*R^3*T^3*b_mix^3*tau_mix^2-...
            16*R*T*ac_mix^2*b_mix+154*R^3*T^3*Vc_eff*b_mix^2-32*a_mix*ac_mix^2-...
            32*T^2*R^2*a_mix*b_mix^2-19*T^2*R^2*a_mix*b_mix*tau_mix*Vc_eff-...
```

```
16*T^2*R^2*a_mix*tau_mix^2*b_mix^2-23*T^2*R^2*a_mix*b_mix*Vc_eff-...
64*T^2*R^2*a_mix*tau_mix*b_mix^2-...
32*T*R*a_mix*ac_mix*b_mix*tau_mix^2+16*T*R*a_mix*ac_mix*Vc_eff-...
32*a_mix*ac_mix^2*tau_mix);
 -2*M_mix*(80*R^3*T^3*ac_mix*b_mix^3*tau_mix^2-16*a_mix*b_mix^2*ac_mix*tau_mix^2*R^2…
*T^2-32*a_mix*b_mix^2*R^2*T^2*ac_mix+8*a_mix*R^3*T^3*b_mix*Vc_eff^2-...
8*a_mix*ac_mix*Vc_eff^2*R^2*T^2+53*a_mix*R^3*T^3*tau_mix*Vc_eff*b_mix^2+48*a_mix*R^3...
*T^3*b_mix^3*tau_mix-32*a_mix*R^3*T^3*b_mix^3*tau_mix^2+32*a_mix*ac_mix^2*Vc_eff*R*T-...
290*R^4*T^4*b_mix^3*Vc_eff+122*a_mix*R^3*T^3*Vc_eff*b_mix^2-...
16*R^3*T^3*b_mix*ac_mix*Vc_eff^2+15*R^3*T^3*ac_mix*Vc_eff*b_mix^2-...
64*R^3*T^3*ac_mix*b_mix^3*tau_mix-32*R^2*T^2*ac_mix^2*tau_mix*b_mix^2-...
64*R^3*T^3*ac_mix*tau_mix*Vc_eff*b_mix^2-...
160*R*T*ac_mix^3*b_mix+144*a_mix*R^3*T^3*b_mix^3+32*a_mix*b_mix^2*R^2*T^2*ac_mix...
*tau_mix+64*R^3*T^3*ac_mix*b_mix^3+64*R^2*T^2*b_mix^2*ac_mix^2-...
176*a_mix*ac_mix^2*b_mix*tau_mix*R*T-...
80*a_mix*R*T*ac_mix^2*b_mix+9*a_mix*b_mix*R^2*T^2*ac_mix*Vc_eff-...
544*R^4*T^4*b_mix^4+96*R^4*T^4*tau_mix^2*b_mix^4+64*a_mix*b_mix*ac_mix*tau_mix*...
R^2*T^2*Vc_eff-128*R^4*T^4*b_mix^4*tau_mix-32*a_mix*ac_mix^3-...
53*R^4*T^4*b_mix^3*tau_mix*Vc_eff-...
32*R^4*T^4*b_mix^2*Vc_eff^2+112*R^2*T^2*ac_mix^2*b_mix*Vc_eff);
 -2*M_mix*b_mix*(40*R^3*T^3*ac_mix*b_mix^3*tau_mix^2-...
96*a_mix*b_mix^2*ac_mix*tau_mix^2*R^2*T^2+128*a_mix*b_mix^2*R^2*T^2*ac_mix+16*...
a_mix*R^3*T^3*b_mix*Vc_eff^2+32*a_mix*ac_mix*Vc_eff^2*R^2*T^2-...
85*a_mix*R^3*T^3*tau_mix*Vc_eff*b_mix^2-128*a_mix*R^3*T^3*b_mix^3*tau_mix-...
80*a_mix*R^3*T^3*b_mix^3*tau_mix^2-...
176*a_mix*ac_mix^2*Vc_eff*R*T+13*R^4*T^4*b_mix^3*Vc_eff+14*a_mix*R^3*T^3*...
Vc_eff*b_mix^2-32*R^3*T^3*b_mix*ac_mix*Vc_eff^2-22*R^3*T^3*ac_mix*Vc_eff*b_mix^2-...
144*R^3*T^3*ac_mix*b_mix^3*tau_mix-...
304*R^2*T^2*ac_mix^2*tau_mix*b_mix^2+160*R^3*T^3*ac_mix*tau_mix*...
Vc_eff*b_mix^2+264*R*T*ac_mix^3*b_mix+192*a_mix*R^3*T^3*b_mix^3+32...
*a_mix*b_mix^2*R^2*T^2*ac_mix*tau_mix-104*R^3*T^3*ac_mix*b_mix^3-...
40*R^2*T^2*b_mix^2*ac_mix^2+320*a_mix*ac_mix^2*b_mix*tau_mix*R*T+32...
*a_mix*R*T*ac_mix^2*b_mix-a_mix*b_mix*R^2*T^2*ac_mix*Vc_eff-...
696*R^4*T^4*b_mix^4+136*R^4*T^4*tau_mix^2*b_mix^4-...
32*a_mix*b_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff+224*R^4*T^4*b_mix^4*...
tau_mix+224*a_mix*ac_mix^3+255*R^4*T^4*b_mix^3*tau_mix*Vc_eff-...
16*R^4*T^4*b_mix^2*Vc_eff^2-32*R^2*T^2*ac_mix^2*b_mix*Vc_eff);
  2*M_mix*b_mix^2*(160*R^3*T^3*ac_mix*b_mix^3*tau_mix^2-...
136*a_mix*b_mix^2*ac_mix*tau_mix^2*R^2*T^2-...
40*a_mix*b_mix^2*R^2*T^2*ac_mix+32*a_mix*R^3*T^3*b_mix*Vc_eff^2+16...
```

```
*a_mix*ac_mix*Vc_eff^2*R^2*T^2+223*a_mix*R^3*T^3*tau_mix*Vc_eff*...
b_mix^2+224*a_mix*R^3*T^3*b_mix^3*tau_mix-40*a_mix*R^3*T^3*b_mix^3*tau_mix^2-...
320*a_mix*ac_mix^2*Vc_eff*R*T-...
485*R^4*T^4*b_mix^3*Vc_eff+317*a_mix*R^3*T^3*Vc_eff*b_mix^2-...
80*R^3*T^3*b_mix*ac_mix*Vc_eff^2+222*R^3*T^3*ac_mix*Vc_eff*b_mix^2...
+64*R^3*T^3*ac_mix*b_mix^3*tau_mix-320*R^2*T^2*ac_mix^2*tau_mix*b_mix^2-...
80*R^3*T^3*ac_mix*tau_mix*Vc_eff*b_mix^2+48*R*T*ac_mix^3*b_mix+40*a_mix...
*R^3*T^3*b_mix^3-48*a_mix*b_mix^2*R^2*T^2*ac_mix*tau_mix+176*R^3*T^3*ac_mix*b_mix^3-...
32*R^2*T^2*b_mix^2*ac_mix^2+48*a_mix*ac_mix^2*b_mix*tau_mix*R*T-...
72*a_mix*R*T*ac_mix^2*b_mix-...
38*a_mix*b_mix*R^2*T^2*ac_mix*Vc_eff+384*R^4*T^4*b_mix^4+64*R^4*T^4*tau_mix^2*...
b_mix^4+192*a_mix*b_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff-...
448*R^4*T^4*b_mix^4*tau_mix+648*a_mix*ac_mix^3-335*R^4*T^4*b_mix^3*tau_mix*Vc_eff-...
96*R^4*T^4*b_mix^2*Vc_eff^2+304*R^2*T^2*ac_mix^2*b_mix*Vc_eff);
  2*M_mix*b_mix^3*(-64*a_mix*b_mix^2*ac_mix*tau_mix^2*R^2*T^2-...
32*a_mix*b_mix^2*R^2*T^2*ac_mix+80*a_mix*R^3*T^3*b_mix*Vc_eff^2+96*...
a_mix*ac_mix*Vc_eff^2*R^2*T^2-143*a_mix*R^3*T^3*tau_mix*Vc_eff*b_mix^2-...
160*a_mix*R^3*T^3*b_mix^3*tau_mix^2+48*a_mix*ac_mix^2*Vc_eff*R*T-...
320*R^4*T^4*b_mix^3*Vc_eff+155*a_mix*R^3*T^3*Vc_eff*b_mix^2-...
40*R^3*T^3*b_mix*ac_mix*Vc_eff^2+139*R^3*T^3*ac_mix*Vc_eff*b_mix^2-...
256*R^3*T^3*ac_mix*b_mix^3*tau_mix-...
224*R^2*T^2*ac_mix^2*tau_mix*b_mix^2+320*R^3*T^3*ac_mix*tau_mix*Vc_eff...
*b_mix^2+336*R*T*ac_mix^3*b_mix+272*a_mix*R^3*T^3*b_mix^3+128*a_mix*...
b_mix^2*R^2*T^2*ac_mix*tau_mix+272*R^3*T^3*ac_mix*b_mix^3-...
32*R^2*T^2*b_mix^2*ac_mix^2+576*a_mix*ac_mix^2*b_mix*tau_mix*R*T+...
144*a_mix*R*T*ac_mix^2*b_mix-...
162*a_mix*b_mix*R^2*T^2*ac_mix*Vc_eff+224*R^4*T^4*tau_mix^2*b_mix^4-...
272*a_mix*b_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff-224*R^4*T^4*b_mix^4*tau_mix-...
960*a_mix*ac_mix^3+233*R^4*T^4*b_mix^3*tau_mix*Vc_eff-136*R^4*T^4...
*b_mix^2*Vc_eff^2-320*R^2*T^2*ac_mix^2*b_mix*Vc_eff);
  2*M_mix*b_mix^4*(-...
40*b_mix*a_mix*R^3*T^3*Vc_eff^2+245*b_mix*a_mix*R^2*T^2*ac_mix*...
Vc_eff+112*b_mix^3*a_mix*R^3*T^3+224*b_mix^2*a_mix*ac_mix*...
tau_mix^2*R^2*T^2-672*b_mix*a_mix*ac_mix^2*tau_mix*R*T-...
425*b_mix^2*a_mix*R^3*T^3*tau_mix*Vc_eff+128*b_mix^4*R^4*T^4*tau_mix+...
553*b_mix^3*R^4*T^4*tau_mix*Vc_eff+448*b_mix^2*R^2*T^2*ac_mix^2*tau_mix-...
256*b_mix^3*R^3*T^3*ac_mix*tau_mix-...
224*b_mix*R^2*T^2*ac_mix^2*Vc_eff+160*b_mix*R^3*T^3*ac_mix*Vc_eff^2-...
421*b_mix^2*R^3*T^3*ac_mix*Vc_eff+672*a_mix*ac_mix^3+144*b_mix^3*R^4*T^4*Vc_eff+64*...
b_mix^2*R^4*T^4*Vc_eff^2-336*b_mix*R*T*ac_mix^3-...
```

```
160*b_mix^3*R^3*T^3*ac_mix*tau_mix^2+576*a_mix*ac_mix^2*Vc_eff*R*T+64*...
b_mix^4*R^4*T^4*tau_mix^2+160*b_mix^2*R^2*T^2*ac_mix^2-...
136*a_mix*ac_mix*Vc_eff^2*R^2*T^2+112*b_mix^3*R^3*T^3*ac_mix-...
224*b_mix^3*a_mix*R^3*T^3*tau_mix-144*b_mix*a_mix*R*T*ac_mix^2-...
128*b_mix*a_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff-320*b_mix^2*a_mix*R^3*T^3*Vc_eff-...
64*b_mix^2*a_mix*R^2*T^2*ac_mix+192*b_mix^2*a_mix*R^2*T^2*ac_mix*tau_mix);
  -2*M_mix*b_mix^5*R*T*(-111*T^3*R^3*b_mix^3*Vc_eff+136*T^3*R^3*b_mix^4*tau_mix^2-...
224*T^3*R^3*b_mix^2*Vc_eff^2-...
23*T^3*R^3*b_mix^3*tau_mix*Vc_eff+128*T^2*R^2*b_mix^3*a_mix*tau_mix-...
160*T^2*R^2*b_mix^3*a_mix*tau_mix^2+160*T^2*R^2*b_mix*a_mix*Vc_eff^2+112*T^2*...
R^2*b_mix^3*a_mix+224*T^2*R^2*b_mix^3*ac_mix-16*T*R*b_mix^2*ac_mix^2-...
105*T^2*R^2*b_mix^2*a_mix*tau_mix*Vc_eff+316*T^2*R^2*b_mix^2*ac_mix*Vc_eff-...
144*T^2*R^2*b_mix^3*ac_mix*tau_mix+144*T^2*R^2*b_mix^2*a_mix*Vc_eff-...
40*T^2*R^2*b_mix^3*ac_mix*tau_mix^2+320*T^2*R^2*b_mix^2*ac_mix*tau_mix*...
Vc_eff+64*T*R*a_mix*ac_mix*Vc_eff^2+64*T*R*b_mix^2*a_mix*ac_mix*tau_mix^2...
+192*T*R*b_mix^2*a_mix*ac_mix*tau_mix-448*T*R*b_mix*ac_mix^2*Vc_eff-...
448*T*R*b_mix*a_mix*ac_mix*tau_mix*Vc_eff-...
224*T*R*b_mix^2*a_mix*ac_mix+32*T*R*b_mix^2*ac_mix^2*tau_mix-...
315*T*R*b_mix*a_mix*ac_mix*Vc_eff-208*T^3*R^3*b_mix^4*tau_mix-...
144*T^3*R^3*b_mix^4+48*b_mix*a_mix*ac_mix^2+672*a_mix*ac_mix^2*Vc_eff);
  -2*M_mix*b_mix^6*(-480*b_mix*a_mix*ac_mix^2*tau_mix*R*T-...
80*b_mix^2*R^3*T^3*ac_mix*tau_mix*Vc_eff+420*b_mix*a_mix*R^2*T^2*ac_mix*...
Vc_eff+208*b_mix^2*a_mix*R^2*T^2*ac_mix*tau_mix-...
425*b_mix^2*a_mix*R^3*T^3*tau_mix*Vc_eff+136*b_mix^2*a_mix*ac_mix*...
tau_mix^2*R^2*T^2-96*b_mix^4*R^4*T^4*tau_mix+96*b_mix^4*R^4*T^4*tau_mix^2-...
96*b_mix^4*R^4*T^4-48*b_mix^3*a_mix*R^3*T^3*tau_mix-64*b_mix^2*R^4*T^4*Vc_eff^2-...
144*b_mix*R*T*ac_mix^3+336*a_mix*ac_mix^3+49*b_mix^3*R^4*T^4*Vc_eff+...
160*b_mix*R^3*T^3*ac_mix*Vc_eff^2-324*b_mix^2*R^3*T^3*ac_mix*Vc_eff-...
224*a_mix*ac_mix*Vc_eff^2*R^2*T^2-...
192*b_mix*a_mix*R*T*ac_mix^2+32*b_mix^3*a_mix*R^3*T^3+176*b_mix^3*R^3...
*T^3*ac_mix+128*b_mix^2*R^2*T^2*ac_mix^2-...
80*b_mix^3*R^3*T^3*ac_mix*tau_mix^2+128*b_mix*a_mix*ac_mix*tau_mix*...
R^2*T^2*Vc_eff-224*b_mix^3*R^3*T^3*ac_mix*tau_mix-177*b_mix^2*a_mix*R^3...
*T^3*Vc_eff-40*b_mix^3*a_mix*R^3*T^3*tau_mix^2-...
112*b_mix^2*a_mix*R^2*T^2*ac_mix+377*b_mix^3*R^4*T^4*tau_mix*...
Vc_eff+32*b_mix*R^2*T^2*ac_mix^2*Vc_eff+224*b_mix^2*R^2*T^2*ac_mix^2*tau_mix);
2*M_mix*b_mix^7*(-...
288*b_mix*a_mix*ac_mix^2*tau_mix*R*T+160*b_mix^2*R^3*T^3*ac_mix*tau_mix*Vc_eff-...
220*b_mix*a_mix*R^2*T^2*ac_mix*Vc_eff+128*b_mix^2*a_mix*R^2*T^2*ac_mix*tau_mix-...
25*b_mix^2*a_mix*R^3*T^3*tau_mix*Vc_eff+96*b_mix^2*a_mix*ac_mix*tau_mix^2*R^2*...
```

```
T^2+16*b_mix^4*R^4*T^4*tau_mix+16*b_mix^4*R^4*T^4*tau_mix^2-...
16*b_mix^4*R^4*T^4+64*b_mix^3*a_mix*R^3*T^3*tau_mix-136*b_mix^2*R^4*T^4*Vc_eff^2-...
48*b_mix*R*T*ac_mix^3+192*a_mix*ac_mix^3+480*a_mix*ac_mix^2*Vc_eff*R*T+66*...
b_mix^3*R^4*T^4*Vc_eff+40*b_mix*R^3*T^3*ac_mix*Vc_eff^2+239*b_mix^2*...
R^3*T^3*ac_mix*Vc_eff-64*a_mix*ac_mix*Vc_eff^2*R^2*T^2-...
48*b_mix*a_mix*R*T*ac_mix^2+80*b_mix^3*a_mix*R^3*T^3+16*b_mix^3*R^3*T^3*ac_mix-...
16*b_mix^2*R^2*T^2*ac_mix^2-32*b_mix^3*R^3*T^3*ac_mix*tau_mix^2-...
272*b_mix*a_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff-17*b_mix^2*a_mix*R^3*T^3*Vc_eff-...
80*b_mix^3*a_mix*R^3*T^3*tau_mix^2-160*b_mix^2*a_mix*R^2*T^2*ac_mix-...
129*b_mix^3*R^4*T^4*tau_mix*Vc_eff-...
224*b_mix*R^2*T^2*ac_mix^2*Vc_eff+160*b_mix*a_mix*R^3*T^3*Vc_eff^2+80*...
b_mix^2*R^2*T^2*ac_mix^2*tau_mix);
  2*M_mix*b_mix^8*R*T*(-...
16*b_mix*a_mix*ac_mix^2*tau_mix+106*T^3*R^3*b_mix^3*Vc_eff+32*T^3*R^3*b_mix^4*...
tau_mix^2-96*T^3*R^3*b_mix^2*Vc_eff^2-...
16*b_mix*ac_mix^3+95*T^3*R^3*b_mix^3*tau_mix*Vc_eff+16*T^2*R^2*b_mix^3*...
a_mix*tau_mix-...
32*T^2*R^2*b_mix^3*a_mix*tau_mix^2+40*T^2*R^2*b_mix*a_mix*Vc_eff^2+32*T^2...
*R^2*b_mix^3*a_mix+16*T^2*R^2*b_mix^3*ac_mix+32*T*R*b_mix^2*ac_mix^2+80*...
T^2*R^2*b_mix*ac_mix*Vc_eff^2-223*T^2*R^2*b_mix^2*a_mix*tau_mix*Vc_eff-...
105*T^2*R^2*b_mix^2*ac_mix*Vc_eff-64*T^2*R^2*b_mix^3*ac_mix*tau_mix-...
98*T^2*R^2*b_mix^2*a_mix*Vc_eff-16*T^2*R^2*b_mix^3*ac_mix*tau_mix^2-...
64*T^2*R^2*b_mix^2*ac_mix*tau_mix*Vc_eff-...
136*T*R*a_mix*ac_mix*Vc_eff^2+16*T*R*b_mix^2*a_mix*ac_mix*tau_mix^2+96*T*R*...
b_mix^2*a_mix*ac_mix*tau_mix+80*T*R*b_mix*ac_mix^2*Vc_eff+192*T*R*b_mix*...
a_mix*ac_mix*tau_mix*Vc_eff-...
32*T*R*b_mix^2*a_mix*ac_mix+32*T*R*b_mix^2*ac_mix^2*tau_mix+305*T*...
R*b_mix*a_mix*ac_mix*Vc_eff-32*T^3*R^3*b_mix^4-64*b_mix*a_mix*ac_mix^2-...
288*a_mix*ac_mix^2*Vc_eff);
  2*M_mix*b_mix^9*(64*b_mix*a_mix*ac_mix^2*tau_mix*R*T-...
32*b_mix^2*R^3*T^3*ac_mix*tau_mix*Vc_eff+39*b_mix*a_mix*R^2*T^2*ac_mix*Vc_eff-...
32*b_mix^2*a_mix*R^2*T^2*ac_mix*tau_mix-b_mix^2*a_mix*R^3*T^3*tau_mix*Vc_eff-...
32*b_mix^2*a_mix*ac_mix*tau_mix^2*R^2*T^2+8*b_mix^4*R^4*T^4*tau_mix^2-...
8*b_mix^4*R^4*T^4+16*b_mix^2*R^4*T^4*Vc_eff^2+8*b_mix*R*T*ac_mix^3-32*a_mix*ac_mix^3-...
16*a_mix*ac_mix^2*Vc_eff*R*T+5*b_mix^3*R^4*T^4*Vc_eff-32*b_mix*R^3*T^3*ac_mix*...
Vc_eff^2-58*b_mix^2*R^3*T^3*ac_mix*Vc_eff+96*a_mix*ac_mix*...
Vc_eff^2*R^2*T^2+16*b_mix*a_mix*R*T*ac_mix^2-16*b_mix^3*a_mix*R^3*T^3-...
8*b_mix^3*R^3*T^3*ac_mix+8*b_mix^2*R^2*T^2*ac_mix^2+8*b_mix^3*R^3*T^3...
*ac_mix*tau_mix^2+32*b_mix*a_mix*ac_mix*tau_mix*R^2*T^2*Vc_eff-...
16*b_mix^3*R^3*T^3*ac_mix*tau_mix+58*b_mix^2*a_mix*R^3*T^3*Vc_eff+16*b_mix^3*a_mix*...
```

```
R^3*T^3*tau_mix^2+32*b_mix^2*a_mix*R^2*T^2*ac_mix+43*b_mix^3*...
R^4*T^4*tau_mix*Vc_eff+32*b_mix*R^2*T^2*ac_mix^2*Vc_eff-80*b_mix*...
a_mix*R^3*T^3*Vc_eff^2-16*b_mix^2*R^2*T^2*ac_mix^2*tau_mix);
 -2*M_mix*b_mix^10*(-64*a_mix*ac_mix^2*Vc_eff*R*T-...
8*a_mix*ac_mix^3+16*R^3*T^3*b_mix*ac_mix*Vc_eff^2+19*R^4*T^4*b_mix^3*...
Vc_eff-8*a_mix*R*T*ac_mix^2*b_mix-16*R^3*T^3*ac_mix*tau_mix*Vc_eff...
*b_mix^2-43*a_mix*R^3*T^3*Vc_eff*b_mix^2+16*R^2*T^2*ac_mix^2*b_mix*Vc_eff+86...
*a_mix*b_mix*R^2*T^2*ac_mix*Vc_eff-...
8*a_mix*b_mix^2*ac_mix*tau_mix^2*R^2*T^2+16*a_mix*ac_mix^2*b_mix*tau_mix*R*T+...
16*a_mix*b_mix^2*R^2*T^2*ac_mix*tau_mix-14*R^3*T^3*ac_mix*Vc_eff*b_mix^2-...
16*a_mix*ac_mix*Vc_eff^2*R^2*T^2+8*a_mix*b_mix^2*R^2*T^2*ac_mix+64*a_mix*b_mix*...
ac_mix*tau_mix*R^2*T^2*Vc_eff-8*a_mix*R^3*T^3*b_mix^3*tau_mix^2-...
32*R^4*T^4*b_mix^2*Vc_eff^2+5*R^4*T^4*b_mix^3*tau_mix*Vc_eff+8*a_mix*R^3*T^3...
*b_mix^3-53*a_mix*R^3*T^3*tau_mix*Vc_eff*b_mix^2+32*a_mix*R^3*T^3*b_mix*Vc_eff^2);
  2*M_mix*b_mix^11*Vc_eff*R*T*(3*R^3*T^3*b_mix^3*tau_mix+4*R^3*T^3*b_mix^3+8*R^3*...
T^3*b_mix^2*Vc_eff+16*T^2*R^2*Vc_eff*b_mix*a_mix-3*R^2*T^2*ac_mix*b_mix^2-...
5*T^2*R^2*b_mix^2*a_mix*tau_mix-19*b_mix^2*a_mix*R^2*T^2+8*R^2*T^2*ac_mix*...
Vc_eff*b_mix-32*T*R*Vc_eff*a_mix*ac_mix+14*b_mix*a_mix*ac_mix*...
R*T+16*T*R*b_mix*a_mix*ac_mix*tau_mix-16*a_mix*ac_mix^2);
 2*M_mix*b_mix^12*Vc_eff*R^2*T^2*(4*b_mix^3*R^2*T^2+3*T^2*R^2*b_mix^3*tau_mix-...
3*T*R*b_mix^2*ac_mix-3*T*R*b_mix^2*a_mix*tau_mix+8*T*R*Vc_eff*b_mix*a_mix-...
4*b_mix^2*a_mix*R*T+3*b_mix*a_mix*ac_mix+8*Vc_eff*a_mix*ac_mix);
 2*M_mix*b_mix^14*a_mix*R^2*T^2*Vc_eff*(4*T*R*b_mix+3*T*R*b_mix*tau_mix-3*ac_mix)];

dcm  = [10*R^2*T^2*tau_mix+4*R^2*T^2*tau_mix^2+10*R^2*T^2;
 16*tau_mix*b_mix*R^2*T^2+3*R^2*T^2*Vc_eff+20*b_mix*R^2*T^2+4...
*tau_mix*R^2*T^2*Vc_eff-20*ac_mix*R*T-16*ac_mix*tau_mix*R*T;
 -8*ac_mix*R*T*Vc_eff+16*ac_mix^2-2*ac_mix*b_mix*R*T+8*b_mix...
*R^2*T^2*Vc_eff-16*tau_mix^2*b_mix^2*R^2*T^2+6*b_mix^2*R^2*T^2-...
16*tau_mix*b_mix^2*R^2*T^2+24*ac_mix*tau_mix*b_mix*R*T;
 -48*b_mix*ac_mix^2+32*ac_mix*tau_mix*b_mix^2*R*T+12*ac_mix*b_mix*R...
*T*Vc_eff-16*tau_mix*b_mix^2*R^2*T^2*Vc_eff+40*ac_mix*b_mix^2*R*T+...
8*b_mix^3*R^2*T^2-32*tau_mix*b_mix^3*R^2*T^2;
 36*b_mix^2*ac_mix^2+26*b_mix^4*R^2*T^2-...
56*ac_mix*tau_mix*b_mix^3*R*T+4*tau_mix*b_mix^4*R^2*T^2+2*ac_mix*b_mix^3*R*T+24*...
tau_mix^2*b_mix^4*R^2*T^2-16*b_mix^3*R^2*T^2*Vc_eff+16*ac_mix*b_mix^2*R*T*Vc_eff;
 -28*b_mix^3*ac_mix*R*T*Vc_eff-20*ac_mix*b_mix^4*R*T+24*...
tau_mix*b_mix^4*R^2*T^2*Vc_eff-...
10*b_mix^4*R^2*T^2*Vc_eff+4*b_mix^5*R^2*T^2+16*b_mix^3*ac_mix^2+16*tau_mix...
*b_mix^5*R^2*T^2-16*ac_mix*tau_mix*b_mix^4*R*T;
```

```
                -16*tau_mix^2*b_mix^6*R^2*T^2+2*ac_mix*b_mix^5*R*T-...
               8*ac_mix*b_mix^4*Vc_eff*R*T+40*ac_mix*tau_mix*b_mix^5*R*T-...
               24*b_mix^4*ac_mix^2-14*b_mix^6*R^2*T^2+8*b_mix^5*Vc_eff*R^2*T^2;
                -16*tau_mix*b_mix^6*R^2*T^2*Vc_eff+20*ac_mix*b_mix^5*R*T*Vc_eff+...
               8*b_mix^6*R^2*T^2*Vc_eff;
                4*b_mix^6*ac_mix^2-2*ac_mix*b_mix^7*R*T-...
               8*ac_mix*tau_mix*b_mix^7*R*T+2*tau_mix*b_mix^8*R^2*T^2+4*b_mix^8*R^2*T^2+4*tau_mix^2...
               *b_mix^8*R^2*T^2;
                -4*ac_mix*b_mix^7*R*T*Vc_eff-b_mix^8*R^2*T^2*Vc_eff+4*tau_mix*b_mix^8*R^2*T^2*Vc_eff];

        dzro = roots(dcm); p = ncm; q = dcm; qp = polyder(q); qpp = polyder(qp);
        A = p(19)/(q(10)*q(10)); F = F + N*A*log(X/N);
        r = deconv(p-A*conv(q,q),[1;0]); p = r; pp = polyder(p);
        B = polyval(p ,dzro)./polyval(qp,dzro).^2; F = F + N*sum(-B./(X/N-dzro));
        C = (polyval(pp,dzro)./polyval(qp,dzro) - B.*(polyval(qpp,dzro)))./polyval(qp,dzro);
        F = F + N*sum(C.*log(X/N-dzro));
end

return;

% ************************************************

function V = FugInt3(X,T,Z,index,data_set)

R = 8.31451e-2; N = sum(Z);

switch(data_set)
    case 1  % RKS Equation
        V = X;
    case 2  % PR Equation
        V = X;
    case 3  % DMT Equation
        ans_set = prop_data(index,5);
        Pc = ans_set.Pc; Tc = ans_set.Tc; M = ans_set.M; Vc = ans_set.Vc;
        param.ac  = R*R*Tc.*Tc./Pc/9/(2^(1/3)-1);
        param.b   = (2^(1/3)-1)*R*Tc./Pc/3;
        param.M   = M;  param.Vc   = Vc;
        param.tau = (exp(5*(1-sqrt(T./Tc)))-1).*(T<Tc);
        ans_set = mix_rule(Z,param,33);
        ac_mix = ans_set.ac_mix; b_mix = ans_set.b_mix;
```

```
        M_mix = ans_set.M_mix; tau_mix = ans_set.tau_mix; Vc_mix = ans_set.Vc_mix;
        Tc_eff = ac_mix*(2^(1/3)-1)/3/R/b_mix*9*(2^(1/3)-1);
        Pc_eff = R*Tc_eff*(2^(1/3)-1)/3/b_mix; Vc_eff = R*Tc_eff/Pc_eff/3;
        lamb = X/N.*X/N./(X/N-b_mix)./(X/N-b_mix) - ...
               (2*ac_mix*X/N+ac_mix*b_mix)/R/T./(X/N+b_mix)./(X/N+b_mix) + tau_mix;
        delt = lamb + (X/N - 3*Vc_eff/4)./(X/N.*lamb + Vc_eff);
        V = X - (Vc_eff - Vc_mix)*N - M_mix*(1./(1 + 2*delt) - 1)*N;
end

return;

% ************************************************

function V = FugInt1(P,T,Z,index,data_set,alpha_bns)

R = 8.31451e-2; N = sum(Z);

switch(data_set)
    case 1  % RKS Equation
        ans_set = prop_data(index,1);
        Pc = ans_set.Pc; Tc = ans_set.Tc; omega = ans_set.omega;
        ac = R*R*Tc.*Tc./Pc/9/(2^(1/3)-1);
        m = (0.480+1.574*omega-0.176*omega.^2);
        alpha = (1+m.*(1-sqrt(T./Tc))).^2 + alpha_bns;
        param.a = ac.*(alpha + alpha_bns);
        param.b = (2^(1/3)-1)*R*Tc./Pc/3; param.T = T;
        ans_set = mix_rule(Z,param,11);
        a_mix = ans_set.a_mix; b_mix = ans_set.b_mix;
        CFS = [-P; R*T; R*T*b_mix-a_mix+P*b_mix^2; a_mix*b_mix];
    case 2  % PR Equation
        ans_set = prop_data(index,1);
        Pc = ans_set.Pc; Tc = ans_set.Tc; omega = ans_set.omega;
        ac = R*R*Tc.*Tc./Pc*0.45724;
        m = (0.37464+1.5422*omega-0.26992*omega.^2);
        alpha = (1+m.*(1-sqrt(T./Tc))).^2 + alpha_bns;
        param.a = ac.*(alpha + alpha_bns);
        param.b= R*Tc./Pc*0.07780; param.T = T;
        ans_set = mix_rule(Z,param,12);
        a_mix = ans_set.a_mix; b_mix = ans_set.b_mix;
        CFS = [-P; R*T-P*b_mix; 2*R*T*b_mix-a_mix+3*P*b_mix^2; ...
```

```
                -R*T*b_mix^2+a_mix*b_mix-P*b_mix^3];
    case 3  % DMT Equation
        ans_set = prop_data(index,5);
        Pc = ans_set.Pc; Tc = ans_set.Tc; kap = ans_set.kap;
        ac = R*R*Tc.*Tc./Pc/9/(2^(1/3)-1);
        alpha = exp(kap(:,1).*(1-(T./Tc).^kap(:,2))) + alpha_bns;
        param.a = ac.*(alpha + alpha_bns);
        param.b = (2^(1/3)-1)*R*Tc./Pc/3; param.T = T;
        ans_set = mix_rule(Z,param,13);
        a_mix = ans_set.a_mix; b_mix = ans_set.b_mix;
        CFS = [-P; R*T; R*T*b_mix-a_mix+P*b_mix^2; a_mix*b_mix];
end

V = roots(CFS)*N; V = V(~imag(V)); V = [min(V),max(V)];

return

% ************************************************

function [phi,gibbsE,molarV] = poly_thermo(P,T,Z,index,phase)

V = pcsaft_vol(P,T,Z,index);
nroots = length(V); ncomps = length(index);
phi = zeros(nroots,ncomps); R = 8.31451e-2;

for j1 = 1:nroots
    for j2 = 1:ncomps
        phi(j1,j2)  = exp(pcsaft_fug(V(j1),T,Z,index,j2));
    end
end

if(phase == 0)
    [gibbsE,phase] = min((R*T*log(phi))*(Z'/sum(Z))); molarV = V(phase);
elseif(phase == 1)
    [molarV,phase] = min(V);
elseif(phase == 2)
    [molarV,phase] = max(V);
end
phi = phi(phase,:);
gibbsE = ((R*T*log(phi))*(Z'/sum(Z)));
```

```
return;

% ************************************************

function [Vroots] = pcsaft_vol(P,T,Z,index)

ans_set = prop_data(index,4);
m = ans_set.m; sig = ans_set.sig; eok = ans_set.eok;
N = sum(Z); Nav = 6.0221415e+23; z = Z/N; Vroots = zeros(1,2);
d  = sig.*(1-0.12*exp(-3*eok/T));

for j1 = 1:2
    eta = 1e-3+(j1-1)*0.6; eta_step = 1e-11*eta;
    Px = 0; etaold = -1; failflag = 0;
    while(abs(P-Px)/P > 1e-4 && abs(eta-etaold)>1e-10 && ~failflag)
        rho = 6*eta/pi/(z*(m.*d.^3)); V = Nav/rho/1e27;
        Px = pcsaft_pres(V,T,Z,index);
        etau = eta+eta_step/2; etad = eta-eta_step/2;
        rhou = 6*etau/pi/(z*(m.*d.^3));   Vu = Nav/1e27/rhou;
        rhod = 6*etad/pi/(z*(m.*d.^3));   Vd = Nav/1e27/rhod;
        Pxu = pcsaft_pres(Vu,T,Z,index); Pxd = pcsaft_pres(Vd,T,Z,index);
        dPdeta = (Pxu-Pxd)/eta_step; Pstepval = (Px-P)/(dPdeta+eps); pressflag = 1;
        Pstepsize = 1.0; etanew = eta-Pstepsize*Pstepval;

        while(pressflag)
            rhonew = 6*etanew/pi/(z*(m.*d.^3)); Vnew = Nav/1e27/rhonew;
            Pxnew = pcsaft_pres(Vnew,T,Z,index);

            if(abs(P-Pxnew) < abs(P-Px))
                eta = etanew; eta_step = 1e-11*eta;
                pressflag = 0; Px = Pxnew; rho = rhonew;
            else
                Pstepsize = Pstepsize/2;  etanew = eta-Pstepsize*Pstepval;
                if(abs(Px-Pxnew)/Px < 1e-5 && abs(P-Px)/P > 1e-4)
                failflag = 1; pressflag = 0; end
            end
        end
    end
    if(~failflag); Vroots(j1) = Nav/rho/1e27; end
```

```
end

if(abs(Vroots(1)-Vroots(2))/(Vroots(1)+Vroots(2)) < 1e-3)
    Vroots(1) = (Vroots(1)+Vroots(2))/2; Vroots(2) = 0; end

Vroots = nonzeros(Vroots);

return;

% ************************************************

function [P] = pcsaft_pres(V,T,Z,index)

ans_set = prop_data(index,4);
m = ans_set.m; sig = ans_set.sig; eok = ans_set.eok;
kbz = 1.3806503e-23; Nav = 6.0221415e+23; N = sum(Z); rho = Nav/1e27/V; z = Z/N;
d  = sig.*(1-0.12*exp(-3*eok/T)); mbar = z*m; squig0 = rho*pi*(z*(m.*d.^0))/6;
squig1 = rho*pi*(z*(m.*d.^1))/6; squig2 = rho*pi*(z*(m.*d.^2))/6;
squig3 = rho*pi*(z*(m.*d.^3))/6; eta    = squig3;
param.eok = eok; param.sig = sig; ans_set = mix_rule(Z,param,44);
eok_mat = ans_set.eok_mat; sig_mat = ans_set.sig_mat;
prefac = [1,(mbar-1)/mbar,(mbar-1)*(mbar-2)/mbar/mbar ];
aval = [  0.9105631445,   -0.3084016918,  -0.0906148351;
          0.6361281449,    0.1860531159,   0.4527842806;
          2.6861347891,   -2.5030047259,   0.5962700728;
        -26.5473624910,   21.4197936290,  -1.7241829131;
         97.7592087840,  -65.2558853300,  -4.1302112531;
       -159.5915408700,   83.3186804810,  13.7766318700;
         91.2977740840,  -33.7469229300,  -8.6728470368];
bval = [  0.7240946941,   -0.5755498075,   0.0976883116;
          2.2382791861,    0.6995095521,  -0.2557574982;
         -4.0025849485,    3.8925673390,  -9.1558561530;
        -21.0035768150,  -17.2154716480,  20.6420759740;
         26.8556413630,  192.6722644700, -38.8044300520;
        206.5513384100, -161.8264616500,  93.6267740770;
       -355.6023561200, -165.2076934600, -29.6669055850];
m2e1s3 = sum(sum(((z)'*(z)).*(m*m').*((eok_mat/T).^1).*sig_mat.^3));
m2e2s3 = sum(sum(((z)'*(z)).*(m*m').*((eok_mat/T).^2).*sig_mat.^3));
aval01 = aval.*((eta.^(0:6)')*ones(1,3));
bval01 = bval.*((eta.^(0:6)')*ones(1,3)); I2 = sum(bval01*prefac');
```

```
daval01 = aval01.*(((1:7)')*ones(1,3)); dI1 = sum(daval01*prefac');
dbval01 = bval01.*(((1:7)')*ones(1,3)); dI2 = sum(dbval01*prefac');
Zhs = squig3/(1-squig3) + 3*squig1*squig2/squig0/(1-squig3)^2 + ...
      (3*squig2^3-squig3*squig2^3)/squig0/(1-squig3)^3;
ghs = [d.^0,d/2,(d/2).^2]*...
      [1/(1-squig3);3*squig2/(1-squig3)^2;2*squig2^2/(1-squig3)^3];
dgh = [d.^0,d/2,(d/2).^2]*...
      [squig3/(1-squig3)^2;3*squig2/(1-squig3)^2 + 6*squig2*squig3/(1-squig3)^3;
       4*squig2^2/(1-squig3)^3+6*squig2^2*squig3/(1-squig3)^4];
Zhc = mbar*Zhs - z*((m-1)./ghs.*dgh);
C1 = 1/(1 + mbar*(8*eta-2*eta^2)/(1-eta)^4 + ...
     (1-mbar)*(20*eta-27*eta^2+12*eta^3-2*eta^4)/(1-eta)^2/(2-eta)^2);
C2 = -(C1^2)*(mbar*(-4*eta^2+20*eta+8)/(1-eta)^5 + ...
             (1-mbar)*(2*eta^3+12*eta^2-48*eta+40)/(1-eta)^3/(2-eta)^3);
Zdisp = -2*pi*rho*dI1*m2e1s3 - pi*rho*mbar*(dI2*C1+C2*eta*I2)*m2e2s3;
Ztot = 1 + Zhc + Zdisp; P = Ztot*kbz*T*rho*1e25;

return

% ************************************************

function [logphi] = pcsaft_fug(V,T,Z,index,comp)

ans_set = prop_data(index,4);
m = ans_set.m; sig = ans_set.sig; eok = ans_set.eok;
Nav = 6.0221415e+23; N = sum(Z); rho = Nav/1e27/V; z = Z/N;
d  = sig.*(1-0.12*exp(-3*eok/T)); mbar = z*m; squig0 = rho*pi*(z*(m.*d.^0))/6;
squig1 = rho*pi*(z*(m.*d.^1))/6; squig2 = rho*pi*(z*(m.*d.^2))/6;
squig3 = rho*pi*(z*(m.*d.^3))/6; eta    = squig3;
param.eok = eok; param.sig = sig; ans_set = mix_rule(Z,param,44);
eok_mat = ans_set.eok_mat; sig_mat = ans_set.sig_mat;
prefac = [1,(mbar-1)/mbar,(mbar-1)*(mbar-2)/mbar/mbar ];
aval = [  0.9105631445,   -0.3084016918,  -0.0906148351;
          0.6361281449,    0.1860531159,   0.4527842806;
          2.6861347891,   -2.5030047259,   0.5962700728;
        -26.5473624910,   21.4197936290,  -1.7241829131;
         97.7592087840,  -65.2558853300,  -4.1302112531;
       -159.5915408700,   83.3186804810,  13.7766318700;
         91.2977740840,  -33.7469229300,  -8.6728470368];
bval = [  0.7240946941,   -0.5755498075,   0.0976883116;
```

```
          2.2382791861,    0.6995095521,  -0.2557574982;
         -4.0025849485,    3.8925673390,  -9.1558561530;
        -21.0035768150,  -17.2154716480,  20.6420759740;
         26.8556413630,  192.6722644700, -38.8044300520;
        206.5513384100, -161.8264616500,  93.6267740770;
       -355.6023561200, -165.2076934600, -29.6669055850];
aval01 = aval.*((eta.^(0:6)')*ones(1,3)); I1 = sum(aval01*prefac');
bval01 = bval.*((eta.^(0:6)')*ones(1,3)); I2 = sum(bval01*prefac');
daval01 = aval01.*(((1:7)')*ones(1,3)); dI1 = sum(daval01*prefac');
dbval01 = bval01.*(((1:7)')*ones(1,3)); dI2 = sum(dbval01*prefac');
Zhs = squig3/(1-squig3) + 3*squig1*squig2/squig0/(1-squig3)^2 + ...
      (3*squig2^3-squig3*squig2^3)/squig0/(1-squig3)^3;
ghs = [d.^0,d/2,(d/2).^2]*...
         [1/(1-squig3);3*squig2/(1-squig3)^2;2*squig2^2/(1-squig3)^3];
dgh = [d.^0,d/2,(d/2).^2]*...
      [squig3/(1-squig3)^2;3*squig2/(1-squig3)^2 + 6*squig2*squig3/(1-squig3)^3;
       4*squig2^2/(1-squig3)^3+6*squig2^2*squig3/(1-squig3)^4];
Zhc = mbar*Zhs - z*((m-1)./ghs.*dgh);
C1 = 1/(1 + mbar*(8*eta-2*eta^2)/(1-eta)^4 + ...
     (1-mbar)*(20*eta-27*eta^2+12*eta^3-2*eta^4)/(1-eta)^2/(2-eta)^2);
C2 = -(C1^2)*(mbar*(-4*eta^2+20*eta+8)/(1-eta)^5 + ...
              (1-mbar)*(2*eta^3+12*eta^2-48*eta+40)/(1-eta)^3/(2-eta)^3);
m2e1s3 = sum(sum(((z)'*(z)).*(m*m').*((eok_mat/T).^1).*sig_mat.^3));
m2e2s3 = sum(sum(((z)'*(z)).*(m*m').*((eok_mat/T).^2).*sig_mat.^3));
Zdisp = -2*pi*rho*dI1*m2e1s3 - pi*rho*mbar*(dI2*C1+C2*eta*I2)*m2e2s3;
Ztot = 1 + Zhc + Zdisp; adisp = -2*pi*rho*I1*m2e1s3 - pi*rho*mbar*C1*I2*m2e2s3;
ahs = (3*squig1*squig2/(1-squig3) + squig2^3/squig3/(1-squig3)^2 + ...
       (squig2^3/squig3^2-squig0)*log(1-squig3))/squig0;
ahc = mbar*ahs - z*((m-1).*log(ghs)); ares = ahc + adisp;

squig0xk = pi*rho.*m.*(d.^0)/6; squig1xk = pi*rho.*m.*(d.^1)/6;
squig2xk = pi*rho.*m.*(d.^2)/6; squig3xk = pi*rho.*m.*(d.^3)/6;
dahsdx = -squig0xk/squig0*ahs+(3*(squig1xk*squig2+squig1*squig2xk)/(1-squig3)+...
          3*squig1*squig2*squig3xk/(1-squig3)^2+3*squig2^2*squig2xk/squig3/(1-squig3)^2+...
          squig2^3*squig3xk*(3*squig3-1)/squig3^2/(1-squig3)^3+...
          ((3*squig2^2*squig2xk*squig3-2*squig2^3*squig3xk)/squig3^3-squig0xk)...
          *log(1-squig3)+(squig0-squig2^3/squig3^2)*squig3xk/(1-squig3))/squig0;
dghxk = [d.^0,d/2,(d/2).^2]*[(squig3xk'/(1-squig3)^2);(3*squig2xk'/(1-squig3)^2+6*squig2*...
squig3xk'/(1-squig3)^3);(4*squig2*squig2xk'/(1-squig3)^3+6*squig2^2*squig3xk'/(1-squig3)^4)];
dahcdx = m.*ahs+mbar*dahsdx-((z.*(m-1)'./ghs')*dghxk)'-(m-1).*log(ghs);
```

```
C1xk = C2*squig3xk-C1^2*(m*(8*eta-2*eta^2)/(1-eta)^4-...
       m*(20*eta-27*eta^2+12*eta^3-2*eta^4)/(1-eta)^2/(2-eta)^2);
m2e1s3xk = 2*m.*(((eok_mat/T).^1.*sig_mat.^3)*(z'.*m));
m2e2s3xk = 2*m.*(((eok_mat/T).^2.*sig_mat.^3)*(z'.*m));

prefacn = [0,1/mbar/mbar,(3-4/mbar)/mbar/mbar];
naval01 = aval.*((eta.^(-1:5)')*ones(1,3)).*((0:6)'*ones(1,3)); nI1p1 = squig3xk*sum(naval01*prefac');
nbval01 = bval.*((eta.^(-1:5)')*ones(1,3)).*((0:6)'*ones(1,3)); nI2p1 = squig3xk*sum(nbval01*prefac');
naval02 = aval.*((eta.^(0:6)')*ones(1,3)); nI1p2 = m*sum(naval02*prefacn'); I1xk = nI1p1+nI1p2;
nbval02 = bval.*((eta.^(0:6)')*ones(1,3)); nI2p2 = m*sum(nbval02*prefacn'); I2xk = nI2p1+nI2p2;
dadispdx = -2*pi*rho*(I1xk*m2e1s3+I1*m2e1s3xk) - ...
           pi*rho*((m*C1*I2+mbar*C1xk*I2+mbar*C1*I2xk)*m2e2s3+mbar*C1*I2*m2e2s3xk);
daresdx = dahcdx + dadispdx;
logphi = ares + (Ztot-1) - log(Ztot) -(z)*daresdx + daresdx(comp);

return;

% ************************************************
```

Appendix A.9 - Alpha Function Regression

This script "alpha set" is used to determine the parameters of an alpha function needed to accurately represent pure component vapor pressure. The output is a vector of temperatures, required alpha values, and sensitivities that are to be used when fitting the required alpha values to a model. If the argument *optim_flag* is set to zero, then only the current vapor pressure error will be calculated and not the required alpha values.

```
function [T,Tc,alpha_base,alpha,sat_err,sensitiv] = alpha_set(index,optim_flag)

ans_set = prop_data(index,2); kap = ans_set.kap;
Tt = ans_set.Tt; Tc = ans_set.Tc; T = linspace(Tt+1,Tc-1,50)';
alpha = zeros(50,1); sat_err = zeros(50,1); sensitiv = zeros(50,1);
alpha_base = exp(kap(1)*(1-(T/Tc).^kap(2)));

for j1 = 1:50
    disp(j1);
    P = sat_data(T(j1),index);phi_rat = 1; Pwork = P;
    while(abs(phi_rat)>1e-4)
        [phi_L]  = thermo(Pwork-1e-3*Pwork,T(j1),1,index,1);
        [phi_V]  = thermo(Pwork-1e-3*Pwork,T(j1),1,index,2);
        [phi_L2] = thermo(Pwork+1e-3*Pwork,T(j1),1,index,1);
        [phi_V2] = thermo(Pwork+1e-3*Pwork,T(j1),1,index,2);
        phi_rat = log(phi_L/phi_V); phi_rat2 = log(phi_L2/phi_V2);
        phi_prim = (phi_rat2-phi_rat)/2e-3/Pwork;
        if(phi_rat/phi_prim < Pwork)
            Pwork = Pwork - phi_rat/phi_prim;
        else
            Pwork = Pwork/2;
        end
    end
    sat_err(j1) = 100*(P-Pwork)/P;

    if(optim_flag)
        alpha_bns = 0; phi_rat = 1;
        while(abs(phi_rat) > 1e-5)
```

```
            [phi_L]  = thermo(P,T(j1),1,index,1,alpha_bns-1e-6);
            [phi_V]  = thermo(P,T(j1),1,index,2,alpha_bns-1e-6);
            [phi_L2] = thermo(P,T(j1),1,index,1,alpha_bns+1e-6);
            [phi_V2] = thermo(P,T(j1),1,index,2,alpha_bns+1e-6);
            phi_rat = log(phi_L/phi_V); phi_rat2 = log(phi_L2/phi_V2);
            phi_prim = (phi_rat2-phi_rat)/2e-6;
            alpha_bns = alpha_bns - phi_rat/phi_prim;
        end

        alpha(j1) = alpha_bns;

        if(~isnan(alpha_bns))
            [phi_L1] = thermo(P-1e-5*P,T(j1),1,index,1,alpha_bns);
            [phi_V1] = thermo(P-1e-5*P,T(j1),1,index,2,alpha_bns);
            [phi_L2] = thermo(P+1e-5*P,T(j1),1,index,1,alpha_bns);
            [phi_V2] = thermo(P+1e-5*P,T(j1),1,index,2,alpha_bns);
            phi_rat1 = log(phi_L1/phi_V1); phi_rat2 = log(phi_L2/phi_V2);
            phi_prim1 = (phi_rat2-phi_rat1)/2e-5/P;

            [phi_L3] = thermo(P       ,T(j1),1,index,1,alpha_bns-1e-6);
            [phi_V3] = thermo(P       ,T(j1),1,index,2,alpha_bns-1e-6);
            [phi_L4] = thermo(P       ,T(j1),1,index,1,alpha_bns+1e-6);
            [phi_V4] = thermo(P       ,T(j1),1,index,2,alpha_bns+1e-6);
            phi_rat3 = log(phi_L3/phi_V3); phi_rat4 = log(phi_L4/phi_V4);
            phi_prim2 = (phi_rat4-phi_rat3)/2e-6;
            sensitiv(j1) = phi_prim2/phi_prim1/P;
        else
            alpha(j1) = 0; sensitiv(j1) = 0;
        end
    end
end

% ************************************************
```

Appendix A.10 - Binary Phase Envelope Calculations

This script calculates the compositions and either pressure or temperature as indicated by the Txy and Pxy flags in line 3. When evaluating a Pxy cross-section, the least volatile component needs to be listed first in the *index* argument.

```
function [X1,Y1,P1,X2,Y2,T2,VVAL] = binary_mix(T,index)

iters = 100; LLflag = 0; Pxyflag = 1; Txyflag = 0;
X1 = zeros(iters,2); Y1 = zeros(iters,2); P1 = zeros(iters,1);
X2 = zeros(iters,2); Y2 = zeros(iters,2); T2 = zeros(iters,1);
cvals = [0.05;0.05;0.05;0.05;0.1;0.1;0.1]; failflag = 0;

if(LLflag)
    ans_set = prop_data(index,1); Pc = ans_set.Pc; Tc = ans_set.Tc;
    Psat(1) = sat_data(T,index(1)); Psat(2) = sat_data(T,index(2));
    P = ((T<Tc(1))*Psat(1)+(T>Tc(1))*Pc(1)+(T<Tc(2))*Psat(2)+(T>Tc(2))*Pc(2))/2;

    X = [1 0]; W = [0 1]; phiX = thermo(P,T,X,index,1); F = 1; cur_iter = 0;
    phiW = thermo(P,T,W,index,1); K = phiX./phiW; VAL = [X';W';log(K)'];

    while(sum(abs(F))>1e-5 && ~failflag)
        G = obj_grad3(VAL,P,T,index); F = obj_fun3(VAL,P,T,index);
        newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun3(nextVAL,P,T,index); cur_iter = cur_iter + 1;

        while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
            newt_step = newt_step/2;
            nextVAL = VAL - newt_step*step_val;
            nextF = obj_fun3(nextVAL,P,T,index);
        end

        F = nextF; VAL = nextVAL;
        if(cur_iter > 50)
            failflag = 1; end
    end
    X = [VAL(1),VAL(2)]; W = [VAL(3),VAL(4)];
```

```
    phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
    Y(1) = X(1)*phiX(1)/(X(1)*phiX(1)+X(2)*phiX(2)); Y(2) = 1-Y(1);
    phiY = thermo(P,T,Y,index,2); K1 = phiX./phiW; K2 = phiX./phiY;
    VAL = [X';W';Y';log(K1)';log(K2)';log(P)]; F = 1; cur_iter = 0;

    while(sum(abs(F))>1e-5 && ~failflag)
        G = obj_grad4(VAL,T,index); F = obj_fun4(VAL,T,index);
        newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun4(nextVAL,T,index); cur_iter = cur_iter + 1;

        while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
            newt_step = newt_step/2;
            nextVAL = VAL - newt_step*step_val;
            nextF = obj_fun4(nextVAL,T,index);
        end
        F = nextF; VAL = nextVAL;
        if(cur_iter > 50)
            failflag = 1; end
    end
    X = [VAL(1),VAL(2)]; W = [VAL(3),VAL(4)]; Y = [VAL(5),VAL(6)]; disp([X;Y;W]);
end


if(Pxyflag)
    P = sat_data(T,index(1)); X = [1 0]; Y = [1 0];
    phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,Y,index,2); K = phiX./phiY;
    VAL = [X';Y';log(K)';log(P)]; breakflag = false;
    next_val_dex = 7; step_dir = zeros(7,1); step_size = 0.0;
    Pbreak = 400; critflagx = 0; critflagy = 0;

    for j1 = 1:iters
        F = 1; init_set = VAL; sensit = zeros(7,1); sensit(7) = 1; step_con = 0;

        G = obj_grad1(VAL,T,index,next_val_dex); pre_sign = G\sensit;
        [jnk,next_val_dex] = max(abs(pre_sign)); failflag = 0;
        G = obj_grad1(VAL,T,index,next_val_dex); pre_sign = G\sensit;
        targ_val = init_set(next_val_dex)+step_dir(next_val_dex)*step_size*cvals(next_val_dex);

        if(VAL(7)>log(Pbreak))
            next_val_dex = 7; targ_val = log(Pbreak); breakflag = true; end
```

```
while(sum(abs(F))>1e-5 && ~failflag)
    if(step_con > 5 && step_size > 1e-3)
        step_size = step_size/2; VAL = init_set; step_con = 0;
        targ_val = init_set(next_val_dex) + ...
                   step_dir(next_val_dex)*step_size*cvals(next_val_dex);
    end
    G = obj_grad1(VAL,T,index,next_val_dex); newt_step = 1;
    F = obj_fun1(VAL,T,index,targ_val,next_val_dex); step_val = G\F;

    nextVAL = VAL - newt_step*step_val;
    nextF = obj_fun1(nextVAL,T,index,targ_val,next_val_dex);

    while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-8);
        newt_step = newt_step/2;
        nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun1(nextVAL,T,index,targ_val,next_val_dex);
    end
    if(newt_step < 1e-8 && step_size < 1e-3)
        failflag = 1;
    else
        F = nextF; VAL = nextVAL; step_con = step_con + 1;
    end
end

if(j1<5);
    critflagx = sign(VAL(5)); critflagy = sign(VAL(6)); end

step_dir = sign(VAL-init_set+eps);  step_size = 1;
X1(j1,:) = VAL(1:2); Y1(j1,:) = VAL(3:4); P1(j1,1) = exp(VAL(7));
step_dir = step_dir*(j1>1) + sign(pre_sign)*(j1==1);
step_dir = step_dir - 2*sign(pre_sign)*(j1==1)*(step_dir(1)>0);

disp([init_set,VAL,pre_sign,step_dir]);

[phi,Gex,VVAL(j1,1)] = thermo(exp(VAL(7)),T,VAL(1:2)',index,1);
[phi,Gex,VVAL(j1,2)] = thermo(exp(VAL(7)),T,VAL(3:4)',index,2);

if(sign(VAL(5))~=critflagx || sign(VAL(6))~=critflagy)
    failflag = true; end
```

```
        if(breakflag || failflag)
            break; end
    end
    X1 = X1(1:j1,:); Y1 = Y1(1:j1,:); P1 = P1(1:j1,:);
end


if(Txyflag && ~failflag)
    if(Pxyflag)
        P = Pbreak; X = VAL(1:2)'; Y = VAL(3:4)';
    else
        P = sat_data(T,index(1)); X = [1 0]; Y = [1 0];
    end

    phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,Y,index,2); K = phiX./phiY;
    VAL = [X';Y';log(K)';log(T)]; breakflag = false;

    next_val_dex = 7; step_dir = zeros(7,1); step_size = 0.0;
    Tbreak = 523.15; critflagx = 0; critflagy = 0;

    for j1 = 1:iters
        F = 1; init_set = VAL; sensit = zeros(7,1); sensit(7) = 1; step_con = 0;

        G = obj_grad2(VAL,P,index,next_val_dex); pre_sign = G\sensit;
        [jnk,next_val_dex] = max(abs(pre_sign)); failflag = 0;
        G = obj_grad2(VAL,P,index,next_val_dex); pre_sign = G\sensit;
        targ_val = init_set(next_val_dex)+step_dir(next_val_dex)*step_size*cvals(next_val_dex);

        if(VAL(7)>log(Tbreak))
            next_val_dex = 7; targ_val = log(Tbreak); breakflag = true; end

        while(sum(abs(F))>1e-5 && ~failflag)
            if(step_con > 5 && step_size > 1e-3)
                step_size = step_size/2; VAL = init_set; step_con = 0;
                targ_val = init_set(next_val_dex) + ...
                           step_dir(next_val_dex)*step_size*cvals(next_val_dex);
            end
            G = obj_grad2(VAL,P,index,next_val_dex); newt_step = 1;
            F = obj_fun2(VAL,P,index,targ_val,next_val_dex); step_val = G\F;
```

```
            nextVAL = VAL - newt_step*step_val;
            nextF = obj_fun2(nextVAL,P,index,targ_val,next_val_dex);

            while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-8);
                newt_step = newt_step/2;
                nextVAL = VAL - newt_step*step_val;
                nextF = obj_fun2(nextVAL,P,index,targ_val,next_val_dex);
            end
            if(newt_step < 1e-8 && step_size < 1e-3)
                failflag = 1;
            else
                F = nextF; VAL = nextVAL; step_con = step_con + 1;
            end
        end

        if(j1<5);
            critflagx = sign(VAL(5)); critflagy = sign(VAL(6)); end

        step_dir = sign(VAL-init_set+eps);  step_size = 1;
        X2(j1,:) = VAL(1:2); Y2(j1,:) = VAL(3:4); T2(j1,1) = exp(VAL(7));
        step_dir = step_dir*(j1>1) + sign(pre_sign)*(j1==1);
        step_dir = step_dir - 2*sign(pre_sign)*(j1==1)*(step_dir(1)>0);

        disp([init_set,VAL,pre_sign,step_dir]);

        if(sign(VAL(5))~=critflagx || sign(VAL(6))~=critflagy)
            failflag = true; end

        if(breakflag || failflag)
            break; end
    end
    X2 = X2(1:j1,:); Y2 = Y2(1:j1,:); T2 = T2(1:j1,:);
end

return

% ************************************************

function F = obj_fun1(VAL,T,index,next_val,next_val_dex)
```

```
    F = zeros(7,1);
    X = VAL(1:2)'; Y = VAL(3:4)'; logK = VAL(5:6); logP = VAL(7);
    phiX = thermo(exp(logP),T,X,index,1); phiY = thermo(exp(logP),T,Y,index,2);
    F(1,1) = logK(1)+log(phiY(1))-log(phiX(1));
    F(2,1) = logK(2)+log(phiY(2))-log(phiX(2));
    F(3,1) = Y(1)-exp(logK(1))*X(1); F(4,1) = Y(2)-exp(logK(2))*X(2);
    F(5,1) = X(1)+X(2)-1; F(6,1) = Y(1)+Y(2)-1; F(7,1) = VAL(next_val_dex)-next_val;

return

% ************************************************

function G = obj_grad1(VAL,T,index,next_val_dex)

    G = zeros(7,7);
    X = VAL(1:2)'; Y = VAL(3:4)'; logK = VAL(5:6); logP = VAL(7);
    phiX = thermo(exp(logP),T,X,index,1); phiY = thermo(exp(logP),T,Y,index,2);
    phiXpu = thermo(exp(logP)+exp(logP)*1e-3,T,X,index,1);
    phiXpd = thermo(exp(logP)-exp(logP)*1e-3,T,X,index,1);
    phiX1u = thermo(exp(logP),T,[X(1)+1e-3,X(2)],index,1);
    phiX1d = thermo(exp(logP),T,[X(1)-1e-3,X(2)],index,1);
    phiX2u = thermo(exp(logP),T,[X(1),X(2)+1e-3],index,1);
    phiX2d = thermo(exp(logP),T,[X(1),X(2)-1e-3],index,1);
    phiYpu = thermo(exp(logP)+exp(logP)*1e-3,T,Y,index,2);
    phiYpd = thermo(exp(logP)-exp(logP)*1e-3,T,Y,index,2);
    phiY1u = thermo(exp(logP),T,[Y(1)+1e-3,Y(2)],index,2);
    phiY1d = thermo(exp(logP),T,[Y(1)-1e-3,Y(2)],index,2);
    phiY2u = thermo(exp(logP),T,[Y(1),Y(2)+1e-3],index,2);
    phiY2d = thermo(exp(logP),T,[Y(1),Y(2)-1e-3],index,2);
    phiXP = (phiXpu-phiXpd)/exp(logP)/2e-3; phiYP = (phiYpu-phiYpd)/exp(logP)/2e-3;
    phiX1 = (phiX1u-phiX1d)/2e-3;   phiY1 = (phiY1u-phiY1d)/2e-3;
    phiX2 = (phiX2u-phiX2d)/2e-3;   phiY2 = (phiY2u-phiY2d)/2e-3;

    G(1,1) = -phiX1(1)/phiX(1); G(1,2) = -phiX2(1)/phiX(1);
    G(1,3) = phiY1(1)/phiY(1); G(1,4) = phiY2(1)/phiY(1);
    G(1,5) = 1; G(1,7) = (phiYP(1)/phiY(1) - phiXP(1)/phiX(1))*exp(logP);
    G(2,1) = -phiX1(2)/phiX(2); G(2,2) = -phiX2(2)/phiX(2);
    G(2,3) = phiY1(2)/phiY(2); G(2,4) = phiY2(2)/phiY(2);
    G(2,6) = 1; G(2,7) = (phiYP(2)/phiY(2) - phiXP(2)/phiX(2))*exp(logP);
```

```
    G(3,1) = -exp(logK(1)); G(3,3) = 1; G(3,5) = -X(1)*exp(logK(1));
    G(4,2) = -exp(logK(2)); G(4,4) = 1; G(4,6) = -X(2)*exp(logK(2));
    G(5,1:2) = 1; G(6,3:4) = 1; G(7,next_val_dex) = 1;

return

% ************************************************

function F = obj_fun2(VAL,P,index,next_val,next_val_dex)

    F = zeros(7,1);
    X = VAL(1:2)'; Y = VAL(3:4)'; logK = VAL(5:6); logT = VAL(7);
    phiX = thermo(P,exp(logT),X,index,1); phiY = thermo(P,exp(logT),Y,index,2);
    F(1,1) = logK(1)+log(phiY(1))-log(phiX(1));
    F(2,1) = logK(2)+log(phiY(2))-log(phiX(2));
    F(3,1) = Y(1)-exp(logK(1))*X(1); F(4,1) = Y(2)-exp(logK(2))*X(2);
    F(5,1) = X(1)+X(2)-1; F(6,1) = Y(1)+Y(2)-1; F(7,1) = VAL(next_val_dex)-next_val;

return

% ************************************************

function G = obj_grad2(VAL,P,index,next_val_dex)

    G = zeros(7,7);
    X = VAL(1:2)'; Y = VAL(3:4)'; logK = VAL(5:6); logT = VAL(7);
    phiX = thermo(P,exp(logT),X,index,1); phiY = thermo(P,exp(logT),Y,index,2);
    phiXpu = thermo(P,exp(logT)+exp(logT)*1e-3,X,index,1);
    phiXpd = thermo(P,exp(logT)-exp(logT)*1e-3,X,index,1);
    phiX1u = thermo(P,exp(logT),[X(1)+1e-3,X(2)],index,1);
    phiX1d = thermo(P,exp(logT),[X(1)-1e-3,X(2)],index,1);
    phiX2u = thermo(P,exp(logT),[X(1),X(2)+1e-3],index,1);
    phiX2d = thermo(P,exp(logT),[X(1),X(2)-1e-3],index,1);
    phiYpu = thermo(P,exp(logT)+exp(logT)*1e-3,Y,index,2);
    phiYpd = thermo(P,exp(logT)-exp(logT)*1e-3,Y,index,2);
    phiY1u = thermo(P,exp(logT),[Y(1)+1e-3,Y(2)],index,2);
    phiY1d = thermo(P,exp(logT),[Y(1)-1e-3,Y(2)],index,2);
    phiY2u = thermo(P,exp(logT),[Y(1),Y(2)+1e-3],index,2);
    phiY2d = thermo(P,exp(logT),[Y(1),Y(2)-1e-3],index,2);
    phiXP = (phiXpu-phiXpd)/exp(logT)/2e-3; phiYP = (phiYpu-phiYpd)/exp(logT)/2e-3;
```

```
    phiX1 = (phiX1u-phiX1d)/2e-3;   phiY1 = (phiY1u-phiY1d)/2e-3;
    phiX2 = (phiX2u-phiX2d)/2e-3;   phiY2 = (phiY2u-phiY2d)/2e-3;

    G(1,1) = -phiX1(1)/phiX(1); G(1,2) = -phiX2(1)/phiX(1);
    G(1,3) = phiY1(1)/phiY(1); G(1,4) = phiY2(1)/phiY(1);
    G(1,5) = 1; G(1,7) = (phiYP(1)/phiY(1) - phiXP(1)/phiX(1))*exp(logT);
    G(2,1) = -phiX1(2)/phiX(2); G(2,2) = -phiX2(2)/phiX(2);
    G(2,3) = phiY1(2)/phiY(2); G(2,4) = phiY2(2)/phiY(2);
    G(2,6) = 1; G(2,7) = (phiYP(2)/phiY(2) - phiXP(2)/phiX(2))*exp(logT);
    G(3,1) = -exp(logK(1)); G(3,3) = 1; G(3,5) = -X(1)*exp(logK(1));
    G(4,2) = -exp(logK(2)); G(4,4) = 1; G(4,6) = -X(2)*exp(logK(2));
    G(5,1:2) = 1; G(6,3:4) = 1; G(7,next_val_dex) = 1;

return

% ************************************************

function F = obj_fun3(VAL,P,T,index)

    F = zeros(6,1); X = VAL(1:2)'; W = VAL(3:4)'; logK = VAL(5:6)';
    phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
    F(1,1) = logK(1)+log(phiW(1))-log(phiX(1));
    F(2,1) = logK(2)+log(phiW(2))-log(phiX(2));
    F(3,1) = W(1)-exp(logK(1))*X(1); F(4,1) = W(2)-exp(logK(2))*X(2);
    F(5,1) = X(1)+X(2)-1; F(6,1) = W(1)+W(2)-1;

return

% ************************************************

function G = obj_grad3(VAL,P,T,index)

    X = VAL(1:2)'; W = VAL(3:4)'; logK = VAL(5:6)'; G = zeros(6,6);
    phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
    phiX1u = thermo(P,T,[X(1)+1e-5,X(2)],index,1); phiX1d = thermo(P,T,[X(1)-1e-5,X(2)],index,1);
    phiX2u = thermo(P,T,[X(1),X(2)+1e-5],index,1); phiX2d = thermo(P,T,[X(1),X(2)-1e-5],index,1);
    phiW1u = thermo(P,T,[W(1)+1e-5,W(2)],index,1); phiW1d = thermo(P,T,[W(1)-1e-5,W(2)],index,1);
    phiW2u = thermo(P,T,[W(1),W(2)+1e-5],index,1); phiW2d = thermo(P,T,[W(1),W(2)-1e-5],index,1);
    phiX1 = (phiX1u-phiX1d)/2e-5;   phiW1 = (phiW1u-phiW1d)/2e-5;
    phiX2 = (phiX2u-phiX2d)/2e-5;   phiW2 = (phiW2u-phiW2d)/2e-5;
```

```
    G(1,1) = -phiX1(1)/phiX(1); G(1,2) = -phiX2(1)/phiX(1);
    G(1,3) = phiW1(1)/phiW(1); G(1,4) = phiW2(1)/phiW(1); G(1,5) = 1;
    G(2,1) = -phiX1(2)/phiX(2); G(2,2) = -phiX2(2)/phiX(2);
    G(2,3) = phiW1(2)/phiW(2); G(2,4) = phiW2(2)/phiW(2); G(2,6) = 1;
    G(3,1) = -exp(logK(1)); G(3,3) = 1; G(3,5) = -X(1)*exp(logK(1));
    G(4,2) = -exp(logK(2)); G(4,4) = 1; G(4,6) = -X(2)*exp(logK(2));
    G(5,1:2) = 1; G(6,3:4) = 1;

return

% ************************************************

function F = obj_fun4(VAL,T,index)

    X = VAL(1:2)'; W = VAL(3:4)'; Y = VAL(5:6)';
    logK1 = VAL(7:8)'; logK2 = VAL(9:10)'; logP = VAL(11);
    phiX = thermo(exp(logP),T,X,index,1); phiW = thermo(exp(logP),T,W,index,1);
    phiY = thermo(exp(logP),T,Y,index,2); F = zeros(11,1);
    F(1,1) = logK1(1)+log(phiW(1))-log(phiX(1));
    F(2,1) = logK1(2)+log(phiW(2))-log(phiX(2));
    F(3,1) = logK2(1)+log(phiY(1))-log(phiX(1));
    F(4,1) = logK2(2)+log(phiY(2))-log(phiX(2));
    F(5,1) = W(1)-exp(logK1(1))*X(1); F(6,1) = W(2)-exp(logK1(2))*X(2);
    F(7,1) = Y(1)-exp(logK2(1))*X(1); F(8,1) = Y(2)-exp(logK2(2))*X(2);
    F(9,1) = X(1)+X(2)-1; F(10,1) = W(1)+W(2)-1;  F(11,1) = Y(1)+Y(2)-1;

return

% ************************************************

function G = obj_grad4(VAL,T,index)

    X = VAL(1:2)'; W = VAL(3:4)'; Y = VAL(5:6)';
    logK1 = VAL(7:8)'; logK2 = VAL(9:10)'; logP = VAL(11);
    phiX = thermo(exp(logP),T,X,index,1); phiW = thermo(exp(logP),T,W,index,1);
    phiY = thermo(exp(logP),T,Y,index,2); G = zeros(11,11);

    phiX1u = thermo(exp(logP),T,[X(1)+1e-5,X(2)],index,1);
    phiX1d = thermo(exp(logP),T,[X(1)-1e-5,X(2)],index,1);
```

```
phiX2u = thermo(exp(logP),T,[X(1),X(2)+1e-5],index,1);
phiX2d = thermo(exp(logP),T,[X(1),X(2)-1e-5],index,1);
phiW1u = thermo(exp(logP),T,[W(1)+1e-5,W(2)],index,1);
phiW1d = thermo(exp(logP),T,[W(1)-1e-5,W(2)],index,1);
phiW2u = thermo(exp(logP),T,[W(1),W(2)+1e-5],index,1);
phiW2d = thermo(exp(logP),T,[W(1),W(2)-1e-5],index,1);
phiY1u = thermo(exp(logP),T,[Y(1)+1e-5,Y(2)],index,2);
phiY1d = thermo(exp(logP),T,[Y(1)-1e-5,Y(2)],index,2);
phiY2u = thermo(exp(logP),T,[Y(1),Y(2)+1e-5],index,2);
phiY2d = thermo(exp(logP),T,[Y(1),Y(2)-1e-5],index,2);
phiXpu = thermo(exp(logP)+exp(logP)*1e-5,T,X,index,1);
phiXpd = thermo(exp(logP)-exp(logP)*1e-5,T,X,index,1);
phiWpu = thermo(exp(logP)+exp(logP)*1e-5,T,W,index,1);
phiWpd = thermo(exp(logP)-exp(logP)*1e-5,T,W,index,1);
phiYpu = thermo(exp(logP)+exp(logP)*1e-5,T,Y,index,2);
phiYpd = thermo(exp(logP)-exp(logP)*1e-5,T,Y,index,2);
phiX1 = (phiX1u-phiX1d)/2e-5; phiW1 = (phiW1u-phiW1d)/2e-5; phiY1 = (phiY1u-phiY1d)/2e-5;
phiX2 = (phiX2u-phiX2d)/2e-5; phiW2 = (phiW2u-phiW2d)/2e-5; phiY2 = (phiY2u-phiY2d)/2e-5;
phiXP = (phiXpu-phiXpd)/exp(logP)/2e-5; phiYP = (phiYpu-phiYpd)/exp(logP)/2e-5;
phiWP = (phiWpu-phiWpd)/exp(logP)/2e-5;

G(1,1)  = -phiX1(1)/phiX(1);
G(1,2)  = -phiX2(1)/phiX(1);
G(1,3)  = phiW1(1)/phiW(1);
G(1,4)  = phiW2(1)/phiW(1);
G(1,7)  = 1;
G(1,11) = (phiWP(1)/phiW(1) - phiXP(1)/phiX(1))*exp(logP);

G(2,1)  = -phiX1(2)/phiX(2);
G(2,2)  = -phiX2(2)/phiX(2);
G(2,3)  = phiW1(2)/phiW(2);
G(2,4)  = phiW2(2)/phiW(2);
G(2,8)  = 1;
G(2,11) = (phiWP(2)/phiW(2) - phiXP(2)/phiX(2))*exp(logP);

G(3,1)  = -phiX1(1)/phiX(1);
G(3,2)  = -phiX2(1)/phiX(1);
G(3,5)  = phiY1(1)/phiY(1);
G(3,6)  = phiY2(1)/phiY(1);
G(3,9)  = 1;
```

```
    G(3,11) = (phiYP(1)/phiY(1) - phiXP(1)/phiX(1))*exp(logP);

    G(4,1)  = -phiX1(2)/phiX(2);
    G(4,2)  = -phiX2(2)/phiX(2);
    G(4,5)  = phiY1(2)/phiY(2);
    G(4,6)  = phiY2(2)/phiY(2);
    G(4,10) = 1;
    G(4,11) = (phiYP(2)/phiY(2) - phiXP(2)/phiX(2))*exp(logP);

    G(5,1) = -exp(logK1(1)); G(5,3) = 1; G(5,7)  = -X(1)*exp(logK1(1));
    G(6,2) = -exp(logK1(2)); G(6,4) = 1; G(6,8)  = -X(2)*exp(logK1(2));
    G(7,1) = -exp(logK2(1)); G(7,5) = 1; G(7,9)  = -X(1)*exp(logK2(1));
    G(8,2) = -exp(logK2(2)); G(8,6) = 1; G(8,10) = -X(2)*exp(logK2(2));
    G(9,1:2) = 1; G(10,3:4) = 1; G(11,5:6) = 1;

return

% ************************************************
```

Appendix A.11 - Ternary Phase Envelope Calculations

This script is used to calculate the phase envelope cross-section of a ternary mixture. The pressure is specified as an argument, while the temperature is fixed in line 22.

```
function [X1,Y1,W1,P1,T1] = ternary_mix(P,index)

iters = 100; failflag = 0; limdex = index(1:2);
X1 = zeros(iters,3); W1 = zeros(iters,3); Y1 = zeros(iters,3);
P1 = zeros(iters,1); T1 = zeros(iters,1);

ans_set = prop_data(index,1);
Tc = ans_set.Tc; Pc = ans_set.Pc; Tsat = zeros(3,1);

for j1 = 1:3
    infun = @(T) sat_data(T,index(j1))-P;
    if(P>Pc(j1))
        Tsat(j1) = Tc(j1);
    else
        Tsat(j1) = fzero(infun,[100 Tc(j1)]);
    end
end

T = max(Tsat); X = [1 0 0]; Y = [1 0 0]; F = 1; cur_iter = 0;
phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,X,index,2);
K = phiX./phiY; VAL = [X';Y';log(K)']; c_frac_dex = 2; c_frac = 0;

T = 400

while(sum(abs(F))>1e-5 && ~failflag)
    G = obj_grad6(VAL,P,T,index,c_frac_dex,0); F = obj_fun6(VAL,P,T,index,c_frac,c_frac_dex,0);
    newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
    nextF = obj_fun6(nextVAL,P,T,index,c_frac,c_frac_dex,0); cur_iter = cur_iter + 1;

    while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
        newt_step = newt_step/2;
```

```
        nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun6(nextVAL,P,T,index,c_frac,c_frac_dex,0);
    end

    F = nextF; VAL = nextVAL;
    if(cur_iter > 50)
        failflag = 1; end
end

c_frac_vec = linspace(VAL(6),0,25);

for j1 = 1:25
    X = [VAL(1) VAL(2) VAL(3)]; Y = [VAL(4) VAL(5) VAL(6)];
    F = 1; cur_iter = 0; phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,X,index,2);
    K = phiX./phiY; VAL = [X';Y';log(K)']; c_frac_dex = 3; c_frac = c_frac_vec(j1);

    while(sum(abs(F))>1e-5 && ~failflag)
        G = obj_grad6(VAL,P,T,index,c_frac_dex,0); F = obj_fun6(VAL,P,T,index,c_frac,c_frac_dex,0);
        newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun6(nextVAL,P,T,index,c_frac,c_frac_dex,0); cur_iter = cur_iter + 1;

        while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
            newt_step = newt_step/2;
            nextVAL = VAL - newt_step*step_val;
            nextF = obj_fun6(nextVAL,P,T,index,c_frac,c_frac_dex,0);
        end

        F = nextF; VAL = nextVAL;
        if(cur_iter > 50)
            failflag = 1; end
    end
    VAL'
    X1(j1,:) = [VAL(1),VAL(2),VAL(3)]; Y1(j1,:) = [VAL(4),VAL(5),VAL(6)];
end

X1 = X1(1:j1,:); Y1 = Y1(1:j1,:);

return
```

```
T = min(Tsat);
X = [1 0]; W = [0 1]; phiX = thermo(P,T,X,limdex,1); F = 1; cur_iter = 0;
phiW = thermo(P,T,W,limdex,1); K = phiX./phiW; VAL = [X';W';log(K)'];

while(sum(abs(F))>1e-5 && ~failflag)
    G = obj_grad3(VAL,P,T,limdex); F = obj_fun3(VAL,P,T,limdex);
    newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
    nextF = obj_fun3(nextVAL,P,T,limdex); cur_iter = cur_iter + 1;

    while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
        newt_step = newt_step/2;
        nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun3(nextVAL,P,T,limdex);
    end

    F = nextF; VAL = nextVAL;
    if(cur_iter > 50)
        failflag = 1; end
end

X = [VAL(1),VAL(2)]; W = [VAL(3),VAL(4)];
phiX = thermo(P,T,X,limdex,1); phiW = thermo(P,T,W,limdex,1);
Y(1) = X(1)*phiX(1)/(X(1)*phiX(1)+X(2)*phiX(2)); Y(2) = 1-Y(1);
phiY = thermo(P,T,Y,limdex,2); K1 = phiX./phiW; K2 = phiX./phiY;
VAL = [X';W';Y';log(K1)';log(K2)';log(T)]; F = 1; cur_iter = 0;

while(sum(abs(F))>1e-5 && ~failflag)
    G = obj_grad4(VAL,P,limdex); F = obj_fun4(VAL,P,limdex);
    newt_step = 1; step_val = G\F; nextVAL = VAL - newt_step*step_val;
    nextF = obj_fun4(nextVAL,P,limdex); cur_iter = cur_iter + 1;

    while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
        newt_step = newt_step/2;
        nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun4(nextVAL,P,limdex);
    end
    F = nextF; VAL = nextVAL;
    if(cur_iter > 50)
        failflag = 1; end
```

```
end

X = [VAL(1),VAL(2),0]; W = [VAL(3),VAL(4),0]; Y = [VAL(5),VAL(6),0];
T = exp(VAL(11)); phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
phiY = thermo(P,T,Y,index,2); K1 = phiX./phiW; K2 = phiX./phiY;
VAL = [X';W';Y';log(K1)';log(K2)';log(T)]; cval = 0.1; step_size = 0.0;
next_val_dex = 16; step_dir = zeros(16,1); breakflag = false;

for j1 = 1:iters
    F = 1; init_set = VAL; sensit = zeros(16,1); sensit(16) = 1;
    G = obj_grad5(VAL,P,index,next_val_dex); pre_sign = G\sensit;
    [jnk,next_val_dex] = max(abs(pre_sign)); failflag = 0; step_con = 0;
    G = obj_grad5(VAL,P,index,next_val_dex); pre_sign = G\sensit;
    targ_val = init_set(next_val_dex)+cval*step_dir(next_val_dex)*step_size;

    if(VAL(5)<0)
        next_val_dex = 5; targ_val = 0; breakflag = true; j1 = j1 -1; end

    while(sum(abs(F))>1e-5 && ~failflag)
        if(step_con > 5 && step_size > 1e-3)
            step_size = step_size/2; VAL = init_set; step_con = 0;
            targ_val = init_set(next_val_dex) + ...
                       step_dir(next_val_dex)*step_size*cval;
        end
        G = obj_grad5(VAL,P,index,next_val_dex); newt_step = 1;
        F = obj_fun5(VAL,P,index,targ_val,next_val_dex); step_val = G\F;

        nextVAL = VAL - newt_step*step_val;
        nextF = obj_fun5(nextVAL,P,index,targ_val,next_val_dex);

        while(sum(abs(nextF)) > sum(abs(F)) && newt_step > 1e-3);
            newt_step = newt_step/2;
            nextVAL = VAL - newt_step*step_val;
            nextF = obj_fun5(nextVAL,P,index,targ_val,next_val_dex);
        end
        F = nextF; VAL = nextVAL; step_con = step_con + 1;
        if(newt_step < 1e-8 && step_size < 1e-3)
            failflag = 1; end
    end
```

```
    X1(j1,:) = VAL(1:3)'; W1(j1,:) = VAL(4:6)'; Y1(j1,:) = VAL(7:9)';
    P1(j1) = P; T1(j1) = exp(VAL(16)); step_size = 1;
    step_dir = step_dir*(j1>1) + sign(pre_sign)*(j1==1);
    if(((init_set(5)-VAL(5))<0));step_dir = -step_dir;end
    disp([init_set,VAL,pre_sign,step_dir]);

    if(failflag || breakflag)
        break; end
end

X1 = X1(1:j1,:); W1 = W1(1:j1,:); Y1 = Y1(1:j1,:);
P1 = P1(1:j1,:); T1 = T1(1:j1,:);

% ************************************************

function F = obj_fun3(VAL,P,T,index)

    F = zeros(6,1); X = VAL(1:2)'; W = VAL(3:4)'; logK = VAL(5:6)';
    phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
    F(1,1) = logK(1)+log(phiW(1))-log(phiX(1));
    F(2,1) = logK(2)+log(phiW(2))-log(phiX(2));
    F(3,1) = W(1)-exp(logK(1))*X(1); F(4,1) = W(2)-exp(logK(2))*X(2);
    F(5,1) = X(1)+X(2)-1; F(6,1) = W(1)+W(2)-1;

return

% ************************************************

function G = obj_grad3(VAL,P,T,index)

    X = VAL(1:2)'; W = VAL(3:4)'; logK = VAL(5:6)'; G = zeros(6,6);
    phiX = thermo(P,T,X,index,1); phiW = thermo(P,T,W,index,1);
    phiX1u = thermo(P,T,[X(1)+1e-5,X(2)],index,1); phiX1d = thermo(P,T,[X(1)-1e-5,X(2)],index,1);
    phiX2u = thermo(P,T,[X(1),X(2)+1e-5],index,1); phiX2d = thermo(P,T,[X(1),X(2)-1e-5],index,1);
    phiW1u = thermo(P,T,[W(1)+1e-5,W(2)],index,1); phiW1d = thermo(P,T,[W(1)-1e-5,W(2)],index,1);
    phiW2u = thermo(P,T,[W(1),W(2)+1e-5],index,1); phiW2d = thermo(P,T,[W(1),W(2)-1e-5],index,1);
    phiX1 = (phiX1u-phiX1d)/2e-5;   phiW1 = (phiW1u-phiW1d)/2e-5;
    phiX2 = (phiX2u-phiX2d)/2e-5;   phiW2 = (phiW2u-phiW2d)/2e-5;

    G(1,1) = -phiX1(1)/phiX(1); G(1,2) = -phiX2(1)/phiX(1);
```

```
    G(1,3) = phiW1(1)/phiW(1); G(1,4) = phiW2(1)/phiW(1); G(1,5) = 1;
    G(2,1) = -phiX1(2)/phiX(2); G(2,2) = -phiX2(2)/phiX(2);
    G(2,3) = phiW1(2)/phiW(2); G(2,4) = phiW2(2)/phiW(2); G(2,6) = 1;
    G(3,1) = -exp(logK(1)); G(3,3) = 1; G(3,5) = -X(1)*exp(logK(1));
    G(4,2) = -exp(logK(2)); G(4,4) = 1; G(4,6) = -X(2)*exp(logK(2));
    G(5,1:2) = 1; G(6,3:4) = 1;

return

% ************************************************

function F = obj_fun4(VAL,P,index)

    X = VAL(1:2)'; W = VAL(3:4)'; Y = VAL(5:6)';
    logK1 = VAL(7:8)'; logK2 = VAL(9:10)'; logT = VAL(11);
    phiX = thermo(P,exp(logT),X,index,1); phiW = thermo(P,exp(logT),W,index,1);
    phiY = thermo(P,exp(logT),Y,index,2); F = zeros(11,1);
    F(1,1) = logK1(1)+log(phiW(1))-log(phiX(1));
    F(2,1) = logK1(2)+log(phiW(2))-log(phiX(2));
    F(3,1) = logK2(1)+log(phiY(1))-log(phiX(1));
    F(4,1) = logK2(2)+log(phiY(2))-log(phiX(2));
    F(5,1) = W(1)-exp(logK1(1))*X(1); F(6,1) = W(2)-exp(logK1(2))*X(2);
    F(7,1) = Y(1)-exp(logK2(1))*X(1); F(8,1) = Y(2)-exp(logK2(2))*X(2);
    F(9,1) = X(1)+X(2)-1; F(10,1) = W(1)+W(2)-1;  F(11,1) = Y(1)+Y(2)-1;

return

% ************************************************

function G = obj_grad4(VAL,P,index)

    X = VAL(1:2)'; W = VAL(3:4)'; Y = VAL(5:6)';
    logK1 = VAL(7:8)'; logK2 = VAL(9:10)'; logT = VAL(11);
    phiX = thermo(P,exp(logT),X,index,1); phiW = thermo(P,exp(logT),W,index,1);
    phiY = thermo(P,exp(logT),Y,index,2); G = zeros(11,11);

    phiX1u = thermo(P,exp(logT),[X(1)+1e-5,X(2)],index,1);
    phiX1d = thermo(P,exp(logT),[X(1)-1e-5,X(2)],index,1);
    phiX2u = thermo(P,exp(logT),[X(1),X(2)+1e-5],index,1);
    phiX2d = thermo(P,exp(logT),[X(1),X(2)-1e-5],index,1);
```

```
phiW1u = thermo(P,exp(logT),[W(1)+1e-5,W(2)],index,1);
phiW1d = thermo(P,exp(logT),[W(1)-1e-5,W(2)],index,1);
phiW2u = thermo(P,exp(logT),[W(1),W(2)+1e-5],index,1);
phiW2d = thermo(P,exp(logT),[W(1),W(2)-1e-5],index,1);
phiY1u = thermo(P,exp(logT),[Y(1)+1e-5,Y(2)],index,2);
phiY1d = thermo(P,exp(logT),[Y(1)-1e-5,Y(2)],index,2);
phiY2u = thermo(P,exp(logT),[Y(1),Y(2)+1e-5],index,2);
phiY2d = thermo(P,exp(logT),[Y(1),Y(2)-1e-5],index,2);
phiXtu = thermo(P,exp(logT)+exp(logT)*1e-5,X,index,1);
phiXtd = thermo(P,exp(logT)-exp(logT)*1e-5,X,index,1);
phiWtu = thermo(P,exp(logT)+exp(logT)*1e-5,W,index,1);
phiWtd = thermo(P,exp(logT)-exp(logT)*1e-5,W,index,1);
phiYtu = thermo(P,exp(logT)+exp(logT)*1e-5,Y,index,2);
phiYtd = thermo(P,exp(logT)-exp(logT)*1e-5,Y,index,2);
phiX1 = (phiX1u-phiX1d)/2e-5; phiW1 = (phiW1u-phiW1d)/2e-5; phiY1 = (phiY1u-phiY1d)/2e-5;
phiX2 = (phiX2u-phiX2d)/2e-5; phiW2 = (phiW2u-phiW2d)/2e-5; phiY2 = (phiY2u-phiY2d)/2e-5;
phiXT = (phiXtu-phiXtd)/exp(logT)/2e-5; phiYT = (phiYtu-phiYtd)/exp(logT)/2e-5;
phiWT = (phiWtu-phiWtd)/exp(logT)/2e-5;

G(1,1)  = -phiX1(1)/phiX(1);
G(1,2)  = -phiX2(1)/phiX(1);
G(1,3)  = phiW1(1)/phiW(1);
G(1,4)  = phiW2(1)/phiW(1);
G(1,7)  = 1;
G(1,11) = (phiWT(1)/phiW(1) - phiXT(1)/phiX(1))*exp(logT);

G(2,1)  = -phiX1(2)/phiX(2);
G(2,2)  = -phiX2(2)/phiX(2);
G(2,3)  = phiW1(2)/phiW(2);
G(2,4)  = phiW2(2)/phiW(2);
G(2,8)  = 1;
G(2,11) = (phiWT(2)/phiW(2) - phiXT(2)/phiX(2))*exp(logT);

G(3,1)  = -phiX1(1)/phiX(1);
G(3,2)  = -phiX2(1)/phiX(1);
G(3,5)  = phiY1(1)/phiY(1);
G(3,6)  = phiY2(1)/phiY(1);
G(3,9)  = 1;
G(3,11) = (phiYT(1)/phiY(1) - phiXT(1)/phiX(1))*exp(logT);
```

```
    G(4,1)  = -phiX1(2)/phiX(2);
    G(4,2)  = -phiX2(2)/phiX(2);
    G(4,5)  = phiY1(2)/phiY(2);
    G(4,6)  = phiY2(2)/phiY(2);
    G(4,10) = 1;
    G(4,11) = (phiYT(2)/phiY(2) - phiXT(2)/phiX(2))*exp(logT);

    G(5,1) = -exp(logK1(1)); G(5,3) = 1; G(5,7)  = -X(1)*exp(logK1(1));
    G(6,2) = -exp(logK1(2)); G(6,4) = 1; G(6,8)  = -X(2)*exp(logK1(2));
    G(7,1) = -exp(logK2(1)); G(7,5) = 1; G(7,9)  = -X(1)*exp(logK2(1));
    G(8,2) = -exp(logK2(2)); G(8,6) = 1; G(8,10) = -X(2)*exp(logK2(2));
    G(9,1:2) = 1; G(10,3:4) = 1; G(11,5:6) = 1;

return

% ************************************************

function F = obj_fun5(VAL,P,index,next_val,next_val_dex)

    X = VAL(1:3)'; W = VAL(4:6)'; Y = VAL(7:9)';
    logK1 = VAL(10:12)'; logK2 = VAL(13:15)'; logT = VAL(16);
    phiX = thermo(P,exp(logT),X,index,1); phiW = thermo(P,exp(logT),W,index,1);
    phiY = thermo(P,exp(logT),Y,index,2); F = zeros(16,1);

    F(1,1) = logK1(1)+log(phiW(1))-log(phiX(1));
    F(2,1) = logK1(2)+log(phiW(2))-log(phiX(2));
    F(3,1) = logK1(3)+log(phiW(3))-log(phiX(3));

    F(4,1) = logK2(1)+log(phiY(1))-log(phiX(1));
    F(5,1) = logK2(2)+log(phiY(2))-log(phiX(2));
    F(6,1) = logK2(3)+log(phiY(3))-log(phiX(3));

    F(7,1) = W(1)-exp(logK1(1))*X(1);
    F(8,1) = W(2)-exp(logK1(2))*X(2);
    F(9,1) = W(3)-exp(logK1(3))*X(3);

    F(10,1) = Y(1)-exp(logK2(1))*X(1);
    F(11,1) = Y(2)-exp(logK2(2))*X(2);
    F(12,1) = Y(3)-exp(logK2(3))*X(3);
```

```
    F(13,1) = X(1)+X(2)+X(3)-1;
    F(14,1) = W(1)+W(2)+W(3)-1;
    F(15,1) = Y(1)+Y(2)+Y(3)-1;

    F(16,1) = VAL(next_val_dex)-next_val;

return

% ************************************************

function G = obj_grad5(VAL,P,index,next_val_dex)

    X = VAL(1:3)'; W = VAL(4:6)'; Y = VAL(7:9)';
    logK1 = VAL(10:12)'; logK2 = VAL(13:15)'; logT = VAL(16);
    phiX = thermo(P,exp(logT),X,index,1); phiW = thermo(P,exp(logT),W,index,1);
    phiY = thermo(P,exp(logT),Y,index,2); G = zeros(16,16);

    phiX1u = thermo(P,exp(logT),[X(1)+1e-5,X(2),X(3)],index,1);
    phiX1d = thermo(P,exp(logT),[X(1)-1e-5,X(2),X(3)],index,1);
    phiX2u = thermo(P,exp(logT),[X(1),X(2)+1e-5,X(3)],index,1);
    phiX2d = thermo(P,exp(logT),[X(1),X(2)-1e-5,X(3)],index,1);
    phiX3u = thermo(P,exp(logT),[X(1),X(2),X(3)+1e-5],index,1);
    phiX3d = thermo(P,exp(logT),[X(1),X(2),X(3)-1e-5],index,1);
    phiW1u = thermo(P,exp(logT),[W(1)+1e-5,W(2),W(3)],index,1);
    phiW1d = thermo(P,exp(logT),[W(1)-1e-5,W(2),W(3)],index,1);
    phiW2u = thermo(P,exp(logT),[W(1),W(2)+1e-5,W(3)],index,1);
    phiW2d = thermo(P,exp(logT),[W(1),W(2)-1e-5,W(3)],index,1);
    phiW3u = thermo(P,exp(logT),[W(1),W(2),W(3)+1e-5],index,1);
    phiW3d = thermo(P,exp(logT),[W(1),W(2),W(3)-1e-5],index,1);
    phiY1u = thermo(P,exp(logT),[Y(1)+1e-5,Y(2),Y(3)],index,2);
    phiY1d = thermo(P,exp(logT),[Y(1)-1e-5,Y(2),Y(3)],index,2);
    phiY2u = thermo(P,exp(logT),[Y(1),Y(2)+1e-5,Y(3)],index,2);
    phiY2d = thermo(P,exp(logT),[Y(1),Y(2)-1e-5,Y(3)],index,2);
    phiY3u = thermo(P,exp(logT),[Y(1),Y(2),Y(3)+1e-5],index,2);
    phiY3d = thermo(P,exp(logT),[Y(1),Y(2),Y(3)-1e-5],index,2);
    phiXtu = thermo(P,exp(logT)+exp(logT)*1e-5,X,index,1);
    phiXtd = thermo(P,exp(logT)-exp(logT)*1e-5,X,index,1);
    phiWtu = thermo(P,exp(logT)+exp(logT)*1e-5,W,index,1);
    phiWtd = thermo(P,exp(logT)-exp(logT)*1e-5,W,index,1);
    phiYtu = thermo(P,exp(logT)+exp(logT)*1e-5,Y,index,2);
```

```
phiYtd = thermo(P,exp(logT)-exp(logT)*1e-5,Y,index,2);
phiX1 = (phiX1u-phiX1d)/2e-5; phiW1 = (phiW1u-phiW1d)/2e-5; phiY1 = (phiY1u-phiY1d)/2e-5;
phiX2 = (phiX2u-phiX2d)/2e-5; phiW2 = (phiW2u-phiW2d)/2e-5; phiY2 = (phiY2u-phiY2d)/2e-5;
phiX3 = (phiX3u-phiX3d)/2e-5; phiW3 = (phiW3u-phiW3d)/2e-5; phiY3 = (phiY3u-phiY3d)/2e-5;
phiXT = (phiXtu-phiXtd)/exp(logT)/2e-5; phiYT = (phiYtu-phiYtd)/exp(logT)/2e-5;
phiWT = (phiWtu-phiWtd)/exp(logT)/2e-5;

G(1,1)  = -phiX1(1)/phiX(1);
G(1,2)  = -phiX2(1)/phiX(1);
G(1,3)  = -phiX3(1)/phiX(1);
G(1,4)  = phiW1(1)/phiW(1);
G(1,5)  = phiW2(1)/phiW(1);
G(1,6)  = phiW3(1)/phiW(1);
G(1,10)  = 1;
G(1,16) = (phiWT(1)/phiW(1) - phiXT(1)/phiX(1))*exp(logT);

G(2,1)  = -phiX1(2)/phiX(2);
G(2,2)  = -phiX2(2)/phiX(2);
G(2,3)  = -phiX3(2)/phiX(2);
G(2,4)  = phiW1(2)/phiW(2);
G(2,5)  = phiW2(2)/phiW(2);
G(2,6)  = phiW3(2)/phiW(2);
G(2,11)  = 1;
G(2,16) = (phiWT(2)/phiW(2) - phiXT(2)/phiX(2))*exp(logT);

G(3,1)  = -phiX1(3)/phiX(3);
G(3,2)  = -phiX2(3)/phiX(3);
G(3,3)  = -phiX3(3)/phiX(3);
G(3,4)  = phiW1(3)/phiW(3);
G(3,5)  = phiW2(3)/phiW(3);
G(3,6)  = phiW3(3)/phiW(3);
G(3,12)  = 1;
G(3,16) = (phiWT(3)/phiW(3) - phiXT(3)/phiX(3))*exp(logT);

G(4,1)  = -phiX1(1)/phiX(1);
G(4,2)  = -phiX2(1)/phiX(1);
G(4,3)  = -phiX3(1)/phiX(1);
G(4,7)  = phiY1(1)/phiY(1);
G(4,8)  = phiY2(1)/phiY(1);
G(4,9)  = phiY3(1)/phiY(1);
```

```
    G(4,13)  = 1;
    G(4,16) = (phiYT(1)/phiY(1) - phiXT(1)/phiX(1))*exp(logT);

    G(5,1)  = -phiX1(2)/phiX(2);
    G(5,2)  = -phiX2(2)/phiX(2);
    G(5,3)  = -phiX3(2)/phiX(2);
    G(5,7)  = phiY1(2)/phiY(2);
    G(5,8)  = phiY2(2)/phiY(2);
    G(5,9)  = phiY3(2)/phiY(2);
    G(5,14)  = 1;
    G(5,16) = (phiYT(2)/phiY(2) - phiXT(2)/phiX(2))*exp(logT);

    G(6,1)  = -phiX1(3)/phiX(3);
    G(6,2)  = -phiX2(3)/phiX(3);
    G(6,3)  = -phiX3(3)/phiX(3);
    G(6,7)  = phiY1(3)/phiY(3);
    G(6,8)  = phiY2(3)/phiY(3);
    G(6,9)  = phiY3(3)/phiY(3);
    G(6,15)  = 1;
    G(6,16) = (phiYT(3)/phiY(3) - phiXT(3)/phiX(3))*exp(logT);

    G(7,1) = -exp(logK1(1)); G(7,4) = 1; G(7,10)  = -X(1)*exp(logK1(1));
    G(8,2) = -exp(logK1(2)); G(8,5) = 1; G(8,11)  = -X(2)*exp(logK1(2));
    G(9,3) = -exp(logK1(3)); G(9,6) = 1; G(9,12)  = -X(3)*exp(logK1(3));

    G(10,1) = -exp(logK2(1)); G(10,7) = 1; G(10,13)  = -X(1)*exp(logK2(1));
    G(11,2) = -exp(logK2(2)); G(11,8) = 1; G(11,14)  = -X(2)*exp(logK2(2));
    G(12,3) = -exp(logK2(3)); G(12,9) = 1; G(12,15)  = -X(3)*exp(logK2(3));

    G(13,1:3) = 1; G(14,4:6) = 1; G(15,7:9) = 1;

    G(16,next_val_dex) = 1;

return

% ************************************************

function F = obj_fun6(VAL,P,T,index,c_frac,c_frac_dex,LLleg)

    X = VAL(1:3)'; Y = VAL(4:6)'; logK = VAL(7:9)';
```

```
    phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,Y,index,2-LLleg);

    F = zeros(9,1);
    F(1,1) = logK(1)+log(phiY(1))-log(phiX(1));
    F(2,1) = logK(2)+log(phiY(2))-log(phiX(2));
    F(3,1) = logK(3)+log(phiY(3))-log(phiX(3));
    F(4,1) = Y(1)-exp(logK(1))*X(1);
    F(5,1) = Y(2)-exp(logK(2))*X(2);
    F(6,1) = Y(3)-exp(logK(3))*X(3);
    F(7,1) = X(1)+X(2)+X(3)-1;
    F(8,1) = Y(1)+Y(2)+Y(3)-1;
    F(9,1) = Y(c_frac_dex)-c_frac;

return

% ************************************************

function G = obj_grad6(VAL,P,T,index,c_frac_dex,LLleg)

    X = VAL(1:3)'; Y = VAL(4:6)'; logK = VAL(7:9)';
    phiX = thermo(P,T,X,index,1); phiY = thermo(P,T,Y,index,2-LLleg); G = zeros(9,9);
    phiX1u = thermo(P,T,[X(1)+1e-5,X(2),X(3)],index,1);
    phiX1d = thermo(P,T,[X(1)-1e-5,X(2),X(3)],index,1);
    phiX2u = thermo(P,T,[X(1),X(2)+1e-5,X(3)],index,1);
    phiX2d = thermo(P,T,[X(1),X(2)-1e-5,X(3)],index,1);
    phiX3u = thermo(P,T,[X(1),X(2),X(3)+1e-5],index,1);
    phiX3d = thermo(P,T,[X(1),X(2),X(3)-1e-5],index,1);
    phiY1u = thermo(P,T,[Y(1)+1e-5,Y(2),Y(3)],index,2-LLleg);
    phiY1d = thermo(P,T,[Y(1)-1e-5,Y(2),Y(3)],index,2-LLleg);
    phiY2u = thermo(P,T,[Y(1),Y(2)+1e-5,Y(3)],index,2-LLleg);
    phiY2d = thermo(P,T,[Y(1),Y(2)-1e-5,Y(3)],index,2-LLleg);
    phiY3u = thermo(P,T,[Y(1),Y(2),Y(3)+1e-5],index,2-LLleg);
    phiY3d = thermo(P,T,[Y(1),Y(2),Y(3)-1e-5],index,2-LLleg);
    phiX1 = (phiX1u-phiX1d)/2e-5; phiY1 = (phiY1u-phiY1d)/2e-5;
    phiX2 = (phiX2u-phiX2d)/2e-5; phiY2 = (phiY2u-phiY2d)/2e-5;
    phiX3 = (phiX3u-phiX3d)/2e-5; phiY3 = (phiY3u-phiY3d)/2e-5;

    G(1,1) = -phiX1(1)/phiX(1); G(2,1) = -phiX1(2)/phiX(2); G(3,1) = -phiX1(3)/phiX(3);
    G(1,2) = -phiX2(1)/phiX(1); G(2,2) = -phiX2(2)/phiX(2); G(3,2) = -phiX2(3)/phiX(3);
    G(1,3) = -phiX3(1)/phiX(1); G(2,3) = -phiX3(2)/phiX(2); G(3,3) = -phiX3(3)/phiX(3);
```

```
    G(1,4) = phiY1(1)/phiY(1);  G(2,4) = phiY1(2)/phiY(2);  G(3,4) = phiY1(3)/phiY(3);
    G(1,5) = phiY2(1)/phiY(1);  G(2,5) = phiY2(2)/phiY(2);  G(3,5) = phiY2(3)/phiY(3);
    G(1,6) = phiY3(1)/phiY(1);  G(2,6) = phiY3(2)/phiY(2);  G(3,6) = phiY3(3)/phiY(3);
    G(1,7) = 1;                 G(2,8) = 1;                 G(3,9) = 1;

    G(4,1) = -exp(logK(1)); G(4,4) = 1; G(4,7)  = -X(1)*exp(logK(1));
    G(5,2) = -exp(logK(2)); G(5,5) = 1; G(5,8)  = -X(2)*exp(logK(2));
    G(6,3) = -exp(logK(3)); G(6,6) = 1; G(6,9)  = -X(3)*exp(logK(3));
    G(7,1:3) = 1; G(8,4:6) = 1; G(9,3+c_frac_dex) = 1;

return

% ************************************************
```

Appendix A.12 - Mixing Rule Specifications

This script implements the mixing rules for determining the values of the mixture parameters. The mixing rules implemented here are the Mathias, Klotz, and Prausnitz rules from section 5.2. Other mixing rules can be implemented in this script and the changes will propagate as appropriate to the "thermo" script.

```
function [ans_set] = mix_rule(Z,param,mixflag)

ncomps = length(Z); N = sum(Z); z = Z/N;

switch(mixflag)
    case {11,12,13} % Generic 2 Parameter Mixing Rule
        kij = zeros(ncomps); lij = zeros(ncomps);

%         kij(5,2) = kij(1,2) + 0.12; kij(2,5) = kij(2,1) + 0.12;
%         kij(5,3) = kij(2,3) + 0.14; kij(3,5) = kij(3,2) + 0.14;
%         kij(5,4) = kij(2,3) + 0.16; kij(4,5) = kij(3,2) + 0.16;

%         kij(1,2) = kij(1,2) + 0.14; kij(2,1) = kij(2,1) + 0.14;

        a_mat = sqrt(param.a*param.a');
        ans_set.a_mix = z*((ones(ncomps)-kij).*a_mat)*z' + ...
                        ((z*(nthroot((lij.*a_mat),3))).^3)*z';

        b1 = ones(ncomps); b2 = diag(param.b,0);
        ans_set.b_mix = z*((b1*b2+b2*b1)/2)*z';

    case {33} % DMT Volume Translation Mixing Rule
        ac_mat = sqrt(param.ac*param.ac');
        ans_set.ac_mix = z*ac_mat*z';

        b1 = ones(ncomps); b2 = diag(param.b,0);
        ans_set.b_mix = z*((b1*b2+b2*b1)/2)*z';

        M1 = ones(ncomps); M2 = diag(param.M,0);
        ans_set.M_mix = z*((M1*M2+M2*M1)/2)*z';
```

```
        tau1 = ones(ncomps); tau2 = diag(param.tau,0);
        ans_set.tau_mix = z*((tau1*tau2+tau2*tau1)/2)*z';

        Vc1 = ones(ncomps); Vc2 = diag(param.Vc,0);
        ans_set.Vc_mix = z*((Vc1*Vc2+Vc2*Vc1)/2)*z';

    case {44} % PC-SAFT Mixing Rule
        kij = zeros(ncomps);
        kij(3,2) = kij(3,2) + 0.0312; kij(2,3) = kij(2,3) + 0.0312;
        kij(1,3) = kij(1,3) + 0.0969; kij(3,1) = kij(3,1) + 0.0969;

        ans_set.eok_mat = sqrt(param.eok*param.eok').*(ones(ncomps)-kij);

        sig1 = ones(ncomps); sig2 = diag(param.sig,0);
        ans_set.sig_mat = (sig1*sig2+sig2*sig1)/2;

    case {70} % Rackett Molar Volumes
        Tc_mat = sqrt(param.Tc*param.Tc');
        ans_set.Tc_mix = z*(Tc_mat.*(param.Vc*param.Vc'))*z';
end

return
```

Appendix A.13 - Flash Algorithm

This flash algorithm is an implementation of the procedure described in section 2.4.

```
function [betas,fracs,phis,chk] = multiflash(P,T,Z,index)

ncomps = length(Z); N = sum(Z); ans_set = prop_data(index,1); chk = 0;
Pc = ans_set.Pc; Tc = ans_set.Tc; nphase = 2; betas = ones(nphase,1);
betas = betas/sum(betas); logK = zeros(ncomps,nphase);
logK(:,2) = log(Pc/P) + 5.373*(1-Tc/T);
phis = exp(logK); betas_old = zeros(nphase,1);

% Two Phase Flash
while(sum(abs(betas-betas_old)) > 1e-5)
    betas_old = betas;  betas = multi_beta(betas,Z,phis);
    fracs = diag((Z'/N)./((1./phis)*betas),0)*ones(ncomps,nphase)./phis;
    for j1 = 1:nphase
        phis(:,j1) = thermo(P,T,fracs(:,j1)',index,1+(j1==1))';
    end
end

% % Pure Phase Check
% purphi = zeros(1,ncomps);
% for j1 = 1:ncomps
%     purfrac = zeros(1,ncomps); purfrac(j1) = purfrac(j1) + 1;
%     tempval = thermo(P,T,purfrac,index,1); purphi(j1) = tempval(j1);
% end
%
% chk = 1 - Z'/N.*(1./purphi')./((1./phis)*betas);
%
% betas = betas';
return

% ************************************************
```

```
function betas = multi_beta(betas,Z,phis)

doitagain = 1; lostphase = 0; N = sum(Z);
ncomps = length(Z); nphase = length(betas);
while(doitagain || lostphase)
    Q = sum(betas)-Z/N*log((1./phis)*betas);
    G = 1-((((Z'/N)./((1./phis)*betas))')*(1./phis))'; Gbak = G;
    H = (ones(nphase,ncomps)*diag((((Z'/N)./...
         ((1./phis)*betas).^2)'),0).*((1./phis)'))*(1./phis);

    alpha_step = 1.0; activec = (betas<=1e-6); G(activec) = 0;
    H(activec,:) = 0; H(:,activec) = 0; H = H+(H==0).*eye(nphase);
    betas_step = -H\G;  betas_new = betas + alpha_step*betas_step;
    [minbval,minbdex] = min(betas_new); lostphase = 0;

    if(minbval < -1e-5)
        alpha_step = -betas(minbdex)/betas_step(minbdex);
        betas_new = betas + alpha_step*betas_step;
        Q_new = sum(betas_new)-Z/N*log((1./phis)*betas_new);

        if(Q_new < Q)
            Q = Q_new; betas = betas_new; lostphase = 1;
        else
            alpha_step = alpha_step/2;
            betas_new = betas + alpha_step*betas_step;
            Q_new = sum(betas_new)-Z/N*log((1./phis)*betas_new);
        end
    else
        Q_new = sum(betas_new)-Z/N*log((1./phis)*betas_new);
    end

    while(Q < Q_new && alpha_step > 1e-5 && ~lostphase)
        alpha_step = alpha_step/2;
        betas_new = betas + alpha_step*betas_step;
        Q_new = sum(betas_new)-Z/N*log((1./phis)*betas_new);
    end
    betas = betas_new;
    doitagain = (sum(abs(betas_step))>1e-5);
    if(~doitagain)
        maxgval = max(Gbak(activec));
```

```
        if(maxgval<-1e-7)
            [mingval,mingdex] = min(Gbak);
            betas(mingdex) = 1e-4; doitagain = 1;
            betas = betas/sum(betas);
        end
    end
end

return

% ************************************************
```