Vapor-Liquid Equilibria for R-32 and R-410A Mixed With a Polyol Ester: Non-Ideality and Local Composition Modeling

C. M. Burton and A. M. Jacobi

ACRC TR-117 May 1997

For additional information:

Air Conditioning and Refrigeration Center
University of Illinois
Mechanical & Industrial Engineering Dept.
1206 West Green Street
Urbana, IL 61801

(217) 333-3115

Prepared as part of ACRC Project 57
Saturation P-v-T Relations for New Refrigerant-Oil Combinations
A. M. Jacobi, Principal Investigator

The Air Conditioning and Refrigeration Center was founded in 1988 with a grant from the estate of Richard W. Kritzer, the founder of Peerless of America Inc. A State of Illinois Technology Challenge Grant helped build the laboratory facilities. The ACRC receives continuing support from the Richard W. Kritzer Endowment and the National Science Foundation. The following organizations have also become sponsors of the Center.

Amana Refrigeration, Inc.
Brazeway, Inc.
Carrier Corporation
Caterpillar, Inc.
Copeland Corporation
Dayton Thermal Products
Delphi Harrison Thermal Systems
Eaton Corporation
Ford Motor Company
Frigidaire Company
General Electric Company
Hydro Aluminum Adrian, Inc.
Indiana Tube Corporation
Lennox International, Inc.
Modine Manufacturing Co.
Peerless of America, Inc.
Redwood Microsystems, Inc.
The Trane Company
Whirlpool Corporation
York International, Inc.

For additional information:

Air Conditioning & Refrigeration Center
Mechanical & Industrial Engineering Dept.
University of Illinois
1206 West Green Street
Urbana IL 61801

217 333 3115

ABSTRACT

Vapor-liquid equilibria (VLE) data were obtained over a wide range of mixture composition and saturation conditions for difluoromethane (R-32) mixed with a polyol ester oil (POE). These data were correlated using the following local composition models from the literature: Wilson, Heil, Wang and Chao, Tsuboka and Katayama, NRTL, and UNIQUAC. The results were used to evaluate the suitability of these models in predicting the saturation behavior of the R-32/POE mixture. The Heil model had the best performance, with a 2-σ error of 4.81% in predicted saturation pressure; UNIQUAC was the worst, with a 2-σ pressure error of more than 12%. Using VLE results from the literature for pentafluoroethane (R-125) mixed with the same oil and model parameters for that mixture, and attempt was undertaken to make *a priori* predictions of the P-T-x behavior of a blend containing R-32, R-125 and the oil (R-410A/POE). Data were obtained for this blend, and the results indicate that the Heil model can make such predictions with a 2-σ pressure error of about 11%.

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

NOMENCLATURE

a, A	Peng-Robinson equation of state parameter
b, B	Peng-Robinson equation of state parameter
f	Fugacity
g	Gibbs free energy
g^E, G_M	Excess Gibbs free energy
h	Height on pressure vessel scale
l	UNIQUAC term
m	Mass
M	Peng-Robinson equation of state parameter
MW	Molecular weight
n	Number of moles
P	Pressure
Pe	Poynting Effect
Q	Objective function
q	UNIQUAC size parameter
R	Universal Gas Constant
R_{mix}	Mixture Gas Constant
r	UNIQUAC structure parameter
T	Temperature
V	Volume
v	Liquid molar specific volume
x	Liquid mass fraction
x_i	Liquid mole fraction of component *i*
x_{ij}	Local mole composition of component *i* around component *j*
y	Vapor mole fraction
z	Compressibility factor
n_C	Coordination number

α	Peng-Robinson equation of state parameter
γ	Activity coefficient
ξ	NRTL parameter
ξ_{ii}	Local volume fractions
Λ_{ij}	Interaction terms
τ_{ij}	Interaction component of Λ_{ij} in Heil & NRTL relations
$\overline{\Phi_i}$	Fugacity coefficient
Φ	UNIQUAC structure parameter ratio
θ	UNIQUAC size parameter ratio
ρ	Density
$\Delta\lambda_i$	Interaction energy

Subscripts

c	at the critical point
i, j, k	component *i*, *j*, or *k*
liq	liquid
mass	on a mass basis
oil	of the lubricating oil
ref	of the refrigerant
s	saturated
t	total
vapor	in the vapor phase

Chapter 1 - INTRODUCTION

1.1 Introduction

Designers of air-conditioning and refrigerating equipment depend on reliable thermophysical property data for refrigerant-oil mixtures. These properties are essential in design because refrigerant-oil solubility affects the desired system performance and reliability.

Prior research in this area has mainly focused on obtaining P-T-x data and correlating those data with simple curve fits (e.g. Antoine-motivated polynomials). This approach is expensive and inefficient in view of the many new refrigerants and refrigerant blends proposed for use in refrigeration and air-conditioning systems. Methods for predicting or correlating mixture VLE that are based on sound physical models may allow accurate modeling with fewer measurements. It is therefore important to investigate such models.

Rather than using physically based models, one could simply try to generalize the form of successful empirical fits to VLE data. This approach was adopted by Thome (1995) in his "comprehensive thermodynamic approach." Thome's method was based on an existing empirical correlation provided by Takaishi and Oguchi (1987) to model R-22 mixed with a synthetic alkyl benzene oil. Takaishi and Oguchi used polynomials to model the effect of oil on bubble temperature; Thome suggested that to model other refrigerant-oil blends, only two constants based on pure refrigerant data need to be changed. In essence, this approach shifts the vapor pressure curve developed for the original R-22 mixture and forces its P-T-x behavior onto other mixtures. Such an approach will only work well when the refrigerant-oil interactions are very similar to R-22 and an alkyl benzene.

Although research seems to have focused on empirical modeling, physically based models have not been ignored. Comprehensive reviews of the archival literature on physically based refrigerant-oil mixture modeling have been provided by Martz (1994) and

Martz et al. (1996a,b). Along with discussions of other empirical models and solubility studies completed at the time, Martz and co-workers provide discussions of local composition models such as the Flory-Huggins polymer theory and the UNIQUAC model.

Martz, et al. (1996a) provided thermophysical property data for R-22, R-134a, R-125, and an R-32/R-125 blend (50%/50%) mixed with a polyol ester lubricant. These fluids were tested at refrigerant mass fractions of 20, 40, 60, 80, and 95%, over a temperature range of -30 °C to 70 °C (-22 °F to 158 °F). The data were interpreted to consider departures from the Lewis-Randall rule, and to relate these departures to molecular size differences, intermolecular forces and other factors. Martz et al. (1996b) modeled refrigerant-oil mixtures using a local composition approach. Such models are based on Gibbs free energy and its relation to the activity coefficient through the Gibbs-Duhem relation. Each model has two free parameters representing the molecular interaction energy associated with the refrigerant and oil molecules. These parameters were estimated by minimizing the error associated with predicting system pressure. The Heil model (Renon and Prausnitz, 1968) performed best for the mixtures studied, predicting mixture pressures to within 3.1% (best-case 2-σ error) and 10.4% (worst-case 2-σ error).

Recently, a study of the solubility of R-32 and a polyol ester lubricant was completed by Takaishi and Oguchi (1995). They tested the complete range of refrigerant mass fractions, for temperatures from -30 °C to 30 °C (-22 °F to 86 °F) and pressures up to 2 MPa (290 psia). The mixture was immiscible over a wide range of temperatures, pressures, and concentrations, and the miscibility limit was determined over the test range. Empirical models for pressure as a function of refrigerant mass fraction were developed for different temperatures.

1.2 Objectives

The research reported now provides a continuation of the work presented by Martz, et al. (1996 a,b). The objectives of this research are to study the P-T-x behavior of R-32 mixed with a POE and R-410A mixed with a POE (including measurements of the vapor

phase composition), and, finally, to extend the use of local composition models to mixtures of 3 components. This extension will allow the modeling of a R-410A/oil mixture from data obtained for the individual components (R-32 and R-125) with oil. The model will be compared to experimental data.

To the author's knowledge, this is the first time such an approach has been explored using local-composition models and refrigerant blends with oil. If successful, this method could significantly reduce the experimental burden associated with predicting P-T-x behavior for refrigerant-oil mixtures. To implement these models, it is necessary to investigate the refrigerant and oil interactions and deviations from ideality of each mixture in terms of fugacity, Poynting effect, and activity coefficient.

Chapter 2 - EXPERIMENTAL APPARATUS AND PROCEDURE

2.1 R-32 Apparatus

An experimental apparatus was designed to allow testing of flammable refrigerants. This apparatus consists of an isochoric circulation loop, an isothermal bath and instrumentation. A schematic of the apparatus is shown in Figure 2.1. Refrigerant mass fractions of 15, 40, 60, 80, and 95% were tested over a temperature range of 5 °C to 60 °C (41 °F to 140 °F), while not exceeding a maximum pressure of 3450 kPa (500 psia).

The isochoric circulation loop consists of a pressure vessel, approximately 2.5 meters (97 in) of 9.525 mm (3/8 in) tubing made from 316 SS, and a variable speed 1/4 horsepower pump with an SCR controller. The total volume of the circulation loop is 0.5757 liters (0.15 gal). A detailed description of the design of the pressure vessel is given in Appendix A.

The vessel itself is contained in an insulated chamber and surrounded by an ethylene-glycol/water mixture. A NESLAB RTE-220 constant temperature bath was used to control the temperature of the ethylene-glycol mixture, thus providing an isothermal environment for the refrigerant-oil system. The pump forces fluid, drawn from the bottom of the vessel, through the densimeter, and back into the top of the vessel, ensuring a well-mixed refrigerant-oil solution.

The system vapor pressure was measured using two four-arm, 350-ohm strain gauge bridges; one transducer had a range of 0-3.45 MPa (0-500 psia), and the other was used for lower pressures of 0-0.345 MPa (0-50 psia). Both transducers were placed outside the thermal reservoir and insulated to prevent heat transfer with the environment. Each transducer had an estimated device uncertainty of 0.5% of the full scale measurement, and the total pressure measurement uncertainty was estimated to be ± 17.3 kPa (2.51 psia).

Two platinum resistance temperature devices (RTDs) were place inside the pressure vessel to measure mixture temperature. The RTDs were mounted in steel sheaths of differ-

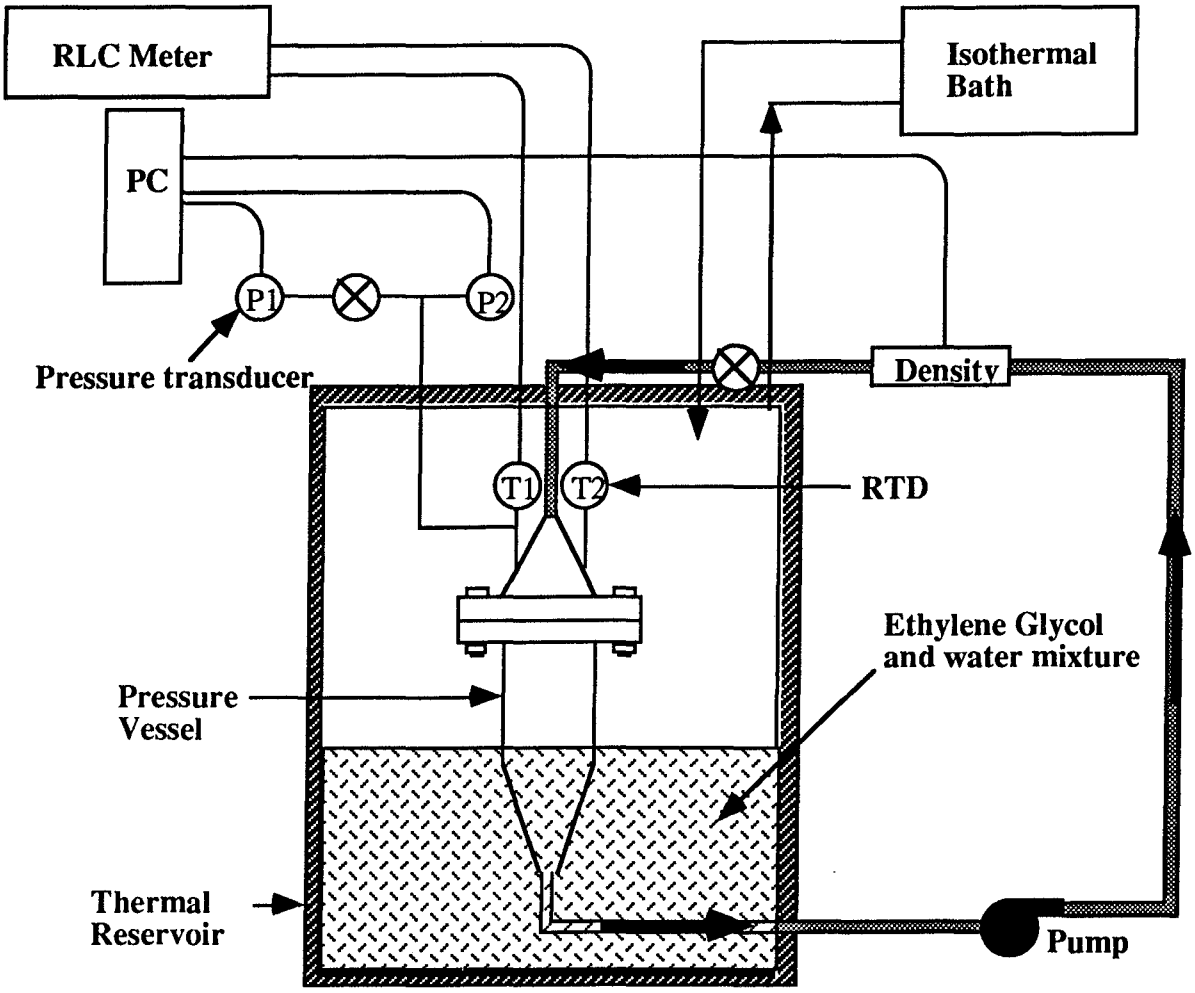

Figure 2.1. Schematic of the R-32 apparatus showing pressure vessel, pump, instrumentation, and isothermal reservoirs.

ing lengths, allowing measurements to be recorded at two different locations within the vessel. Both RTDs were calibrated against NIST-traceable mercury-in-glass thermometers with a resolution of 0.05 °C (0.09°F). The overall uncertainty in temperature measurements was ±0.21 °C (0.38 °F) over the entire temperature range.

Liquid density measurements were made using a temperature-compensated Coriolis densimeter. The Micro Motion D-25 densimeter consists of an isochoric u-shaped tube through which the mixture flows. The open end of the u-tube is fixed, and the closed end vibrates at the tube's resonant frequency. From the resonant frequency, the liquid mass

inside the tube is determined, providing the necessary data to calculate the liquid density. The uncertainty in this measurement was estimated to be ±5 kg/m^3 (0.31 lbm/ft^3).

A computer with two 16-bit A/D boards was used to record measurements obtained with the pressure transducers and densimeter. The resistance of each three-wire RTD was recorded manually to determine temperatures. Measurements of the lead-wire resistance were subtracted from total RTD resistance measurements to determine the change in the device resistance. A 1659 Digibridge RLC meter with a basic accuracy of ±0.2% was used. The measurement range and uncertainties are summarized in Table 2.1.

Table 2.1. Measurement capabilities

Measurement	Instrument	Range	Uncertainty
Temperature	Platinum RTD	-46°C to 121°C- (50°F to 250°F)	±0.21°C (±0.38 °F)
Pressure	Strain gauge transducer	0 to 3.45 MPa (0 to 500 psia) 0 to 0.345 MPa (0 to 50 psia)	±17.3 kPa (±2.51 psia) ±1.73 kPa (±0.251 psia)
Liquid Density	Coriolis meter	400 to 2400 kg/m^3 (25 to 150 lbm/ft^3)	±5 kg/m^3 (±0.31 lbm/ft^3)
Vapor Composition	Gas chromatograph		±4% of each component

System integrity was checked by pressurizing the system with air to 3450 kPa (500 psia) and recording temperature and pressure readings over an extended period of time. Since the system volume was fixed, PV/RT should remain constant during such a leak test. Over a period of at least two hours, PV/RT was always observed to remain constant to within the experimental uncertainty. Therefore, the test apparatus was assumed to be leak free.

2.2 Procedure

The system was carefully flushed and evacuated using an Thermal Engineering Co. Model 1825Z heavy duty high vacuum pump and the procedure detailed by Martz (1994). A known mass of oil (±0.1 gram ($2(10^{-4})$ lbm)) was placed in the test vessel, the temperature was then lowered, and a known mass of refrigerant was added to the system.

Pressure and liquid density data were recorded at a sampling rate of 1 Hz during steady-state conditions for approximately 2 minutes. Equilibrium conditions were determined by observing continuous measurements of system pressure. A period of steady pressure readings (varying less than 1 kPa (0.15 psia) for ten minutes was required before the system was considered to be at equilibrium. This method was verified by Martz (1994). An example of steady-state pressure readings is shown in Figure 2.2, where the variance in mixture pressure data is shown to be well within the device uncertainty. Example steady state liquid density readings are provided in Figure 2.3. The decrease in pressure shown is less than 1 kPa (0.15 kPa) and is due to the temperature drift of the fluid in the isothermal reservoir.

The operation and accuracy of the apparatus were verified by testing R-134a and comparing the experimental results to the data given in ASHRAE (1993). Data obtained in this experimental apparatus and the ASHRAE data for both pressure and liquid density are shown in Figures 2.4 and 2.5, respectively. Using measured temperatures, the experimental data were compared to the ASHRAE data, and the pressure and liquid density measurements were found to be within the estimated experimental uncertainty.

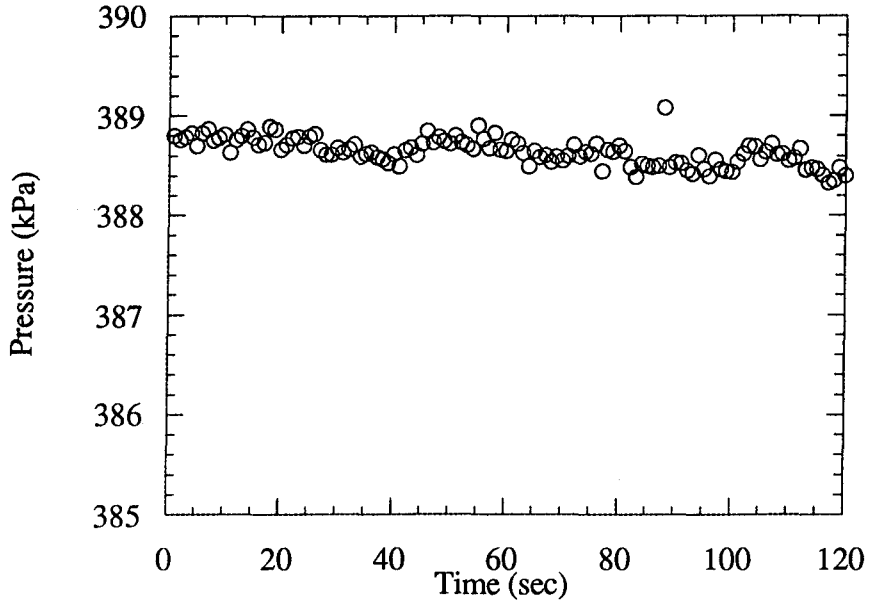

Figure 2.2. Example of sampled mixture pressure data at steady state.

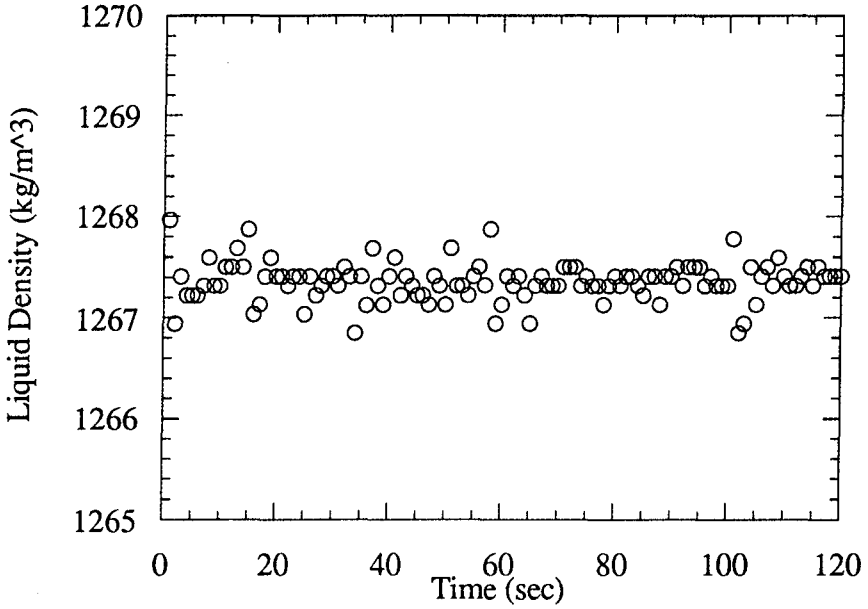

Figure 2.3. Example of sampled liquid density data at steady state.

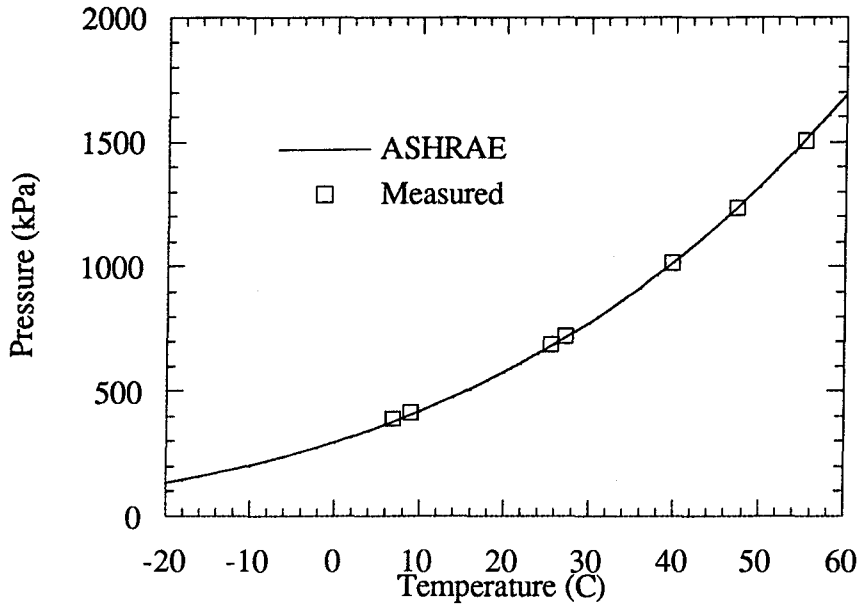

Figure 2.4. Measured pressure and ASHRAE data for pure R-134a.

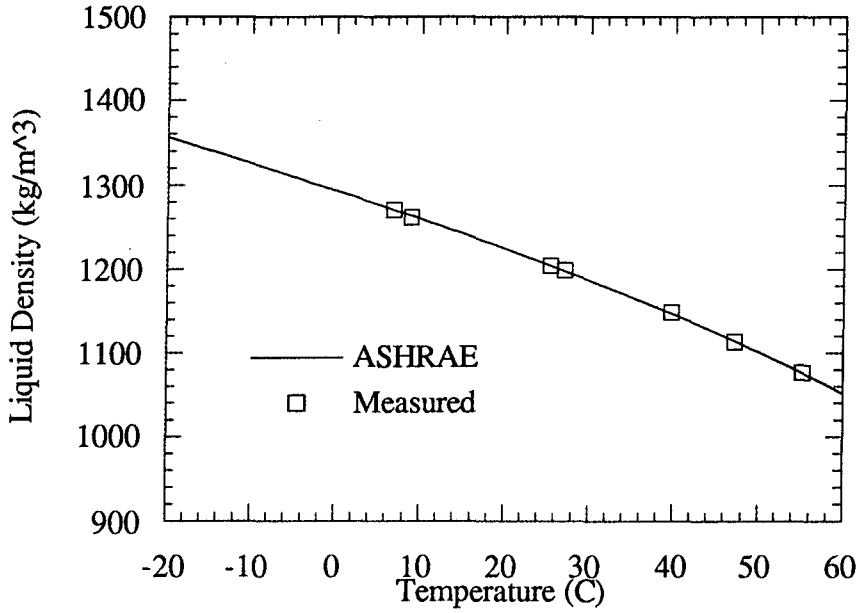

Figure 2.5. Measured liquid density and ASHRAE data for pure R-134a.

2.3 Binary Mixture Apparatus

The experimental apparatus used by Martz, et al. (1996b) was also used in this work to test the R-410A mixtures. A schematic of this apparatus is shown in Fig. 2.6, and a detailed description of its design and construction is given in Martz (1994). The same instrumentation used in the R-32 apparatus was also used in the binary mixture apparatus. Therefore, both experimental apparatus had the same measurement capabilities and uncertainties (Table 2.1). The same charging and flushing procedures were used for both apparatus.

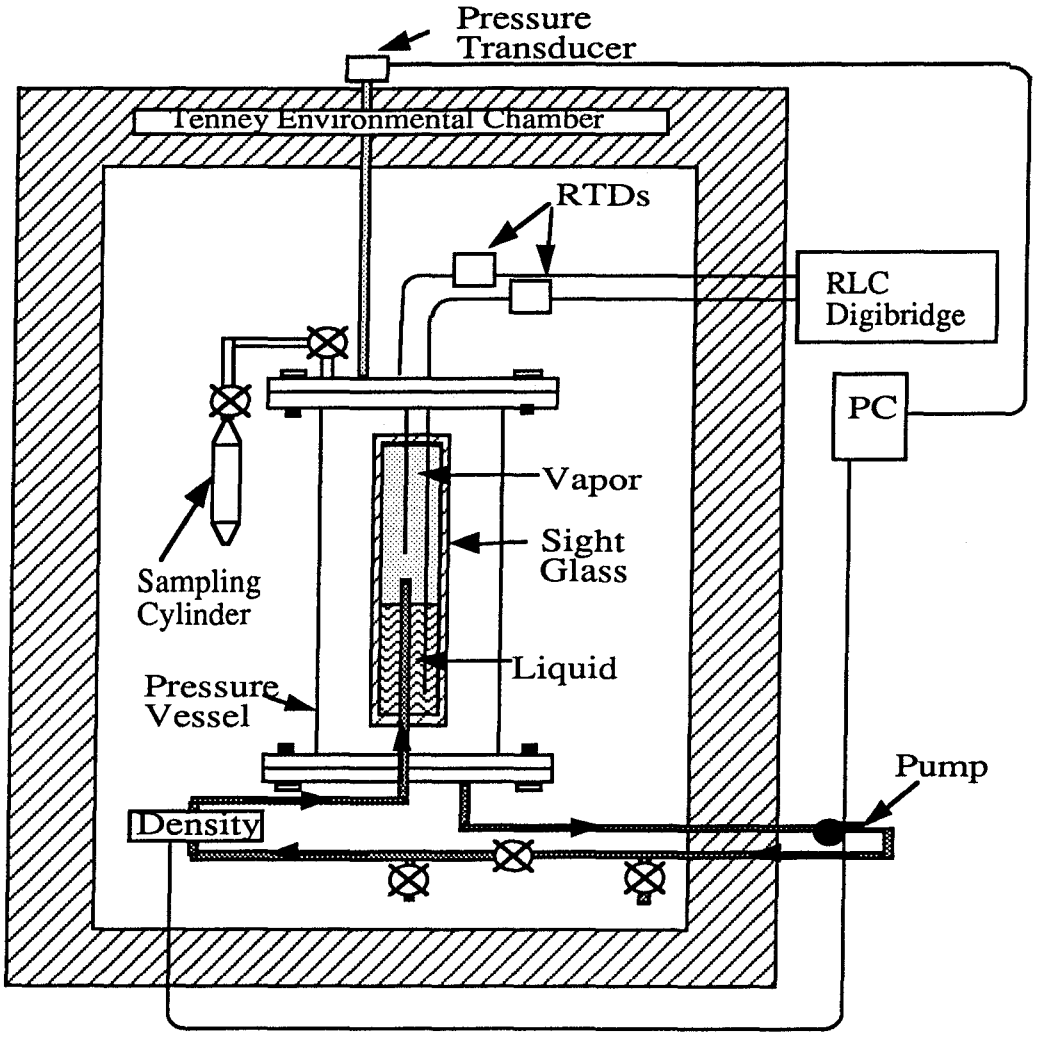

Figure 2.6. Schematic of the binary mixture apparatus showing pressure vessel, sampling cylinder, pump and instrumentation.

In order to determine the composition of the vapor phase, it was necessary to remove a small vapor sample from the mixture. This was done using a 529 ml 316 SS sampling cylinder with a DOT rating of 3E-1800. Vapor from this cylinder was immediately transferred to the gas chromatograph for analysis.

The GOW-MAC gas chromatograph contained a 0.32 cm (1/8 in) 60/80 CarboPac B (5% Krytox liquid phase) column with a length of 7.3 m (27 ft) which was sensitive to the thermal conductivity of gases. A carrier gas, helium, was used to transport injected vapor samples through the column at an approximate flow rate of 30 ml/min (1.83 in^3/min). The column, detector, and injection port temperatures were set to 50, 55, and 50 °C, respectively (122, 131 °F, and 122 °F), and the detector current was set to 150 mA. An HP Integrator was used to record retention times and peak areas.

To calibrate the GC, known volumes of pure R-32 and pure R-125 were each injected separately into the chromatograph using a syringe. Using the pure-refrigerant data in ASHRAE (1993), the vapor density of each sample was determined, allowing the calculation of the sample mass. The area count given by an HP Integrator was recorded for each sample, and the ratio of the mass to average area count was determined. Typical ratios for R-32 (retention time ~3.4 minutes) and R-125 (retention time ~6.1 minutes) are $2.00(10^{-6})$(kg)/649125(area count) and $3.78(10^{-6})$(kg)/1019557(area count), respectively. The mass of a component in a sample was determined by multiplying the component area count by the appropriate mass to area count ratio. The GC was calibrated at 3 different times for the R-410A/POE data analyzed.

Finally, it was necessary to use the sight glass and millimeter scale on the pressure vessel to determine the liquid and vapor volumes. The vessel was filled with a known volume of water and the liquid level height (h) was read from the millimeter scale. The liquid volume (V_l) calibration data are shown in Fig. 2.7, and the calibration equation is

$$V_l = 7.875h + 984.25 \qquad (2.1)$$

where the liquid volume is given in milliliters and the estimated uncertainty is 18.5 milliliters. The total volume of the apparatus was 4.37 liters (1.15 gal.) (without the sampling cylinder attached).

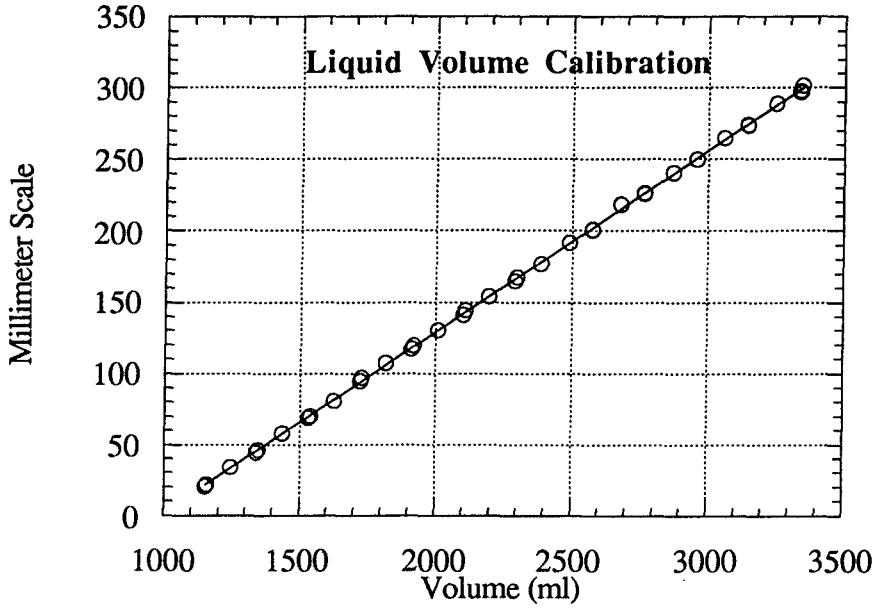

Figure 2.7. Liquid volume calibration of sight glass millimeter scale on binary mixture apparatus.

Chapter 3 - RESULTS AND DISCUSSION

3.1 VLE Data

Pressure, temperature, liquid density, and total component mass data were obtained for mixtures of R-32 with a polyol ester lubricant (RL 68H) using the R-32 apparatus. However, to determine the correct liquid refrigerant concentration, an adjustment must be made for the amount of refrigerant in the vapor phase. Martz (1994) used a technique for determining the liquid refrigerant concentration from a measurement of liquid density, the system charge, and total system volume. This method relies on the following relations:

$$V_{liq} = \frac{m_{ref(liq)} + m_{oil}}{\rho_{liq}} \tag{3.1}$$

$$m_{ref(vapor)} = (V_t - V_{liq})\rho_{vapor} \tag{3.2}$$

$$m_{ref(liq)} = m_{ref(t)} - m_{ref(vapor)} \tag{3.3}$$

In Eqs. 3.1-3.3, the mass of oil (m_{oil}), the total mass of the refrigerant ($m_{ref(t)}$), the total volume (V_t), and the liquid density (ρ_{liq}) are all measured. The Peng-Robinson equation of state (Peng and Robinson, 1976) is used to determine the refrigerant vapor density (ρ_{vapor}), and the set of equations is solved iteratively for the other variables (V_{liq}, $m_{ref(liq)}$, $m_{ref(vap)}$).

For the R-410A/POE mixture, the vapor and liquid volumes are determined using Eq. 2.1, and the vapor composition is determined using a gas chromatograph. These extra measurements, along with the vapor density given by the Peng-Robinson equation, allow the mass composition of both the liquid and vapor to be determined without the need for liquid density measurements.

The lubricant used in this work, an ISO 68 pentaerythritol ester, is the same as used in Martz, et al. (1996a,b); it has a specific weight of 0.997 at 25°C (77°F). Martz, et al.

estimated the molar weight of this oil to be 700, and determined the liquid density as a function of temperature. This relation is:

$$\rho_{oil} = 993.89 - 0.75658T \tag{3.4}$$

The temperature is in degrees Celsius, and the density is in kg/m^3, with an estimated uncertainty of 2.65 kg/m^3 (Martz, 1994).

The pressure and liquid density for the R-32/POE system are shown as functions of temperature and liquid refrigerant mass fraction in Figures 3.1 and 3.2, respectively. A least-squared curve fit is provided for pure refrigerant properties for easy reference. This curve fit, Eq. 3.5a, has a regression factor of 0.99992 (R=1 is a perfect fit) and represents the data for refrigerant mass fractions of 95%, 80%, and 60% well.

$$P = 0.49076T^2 + 21.123T + 846.92 \tag{3.5a}$$

Least-squared curve fits are also provided for the pressure data corresponding to refrigerant mass fractions of 40% and 15%, in Eqs. 3.5b and 3.5c, respectively. Each had a regression factor of 0.99999.

$$P = 0.39718T^2 + 21.78T + 837.57 \tag{3.5b}$$

$$P = 0.17025T^2 + 13.787T + 549.65. \tag{3.5c}$$

In Eqs. 3.5a, b, and c, the pressure has units of kPa, and the temperature, T, is in degrees Celsius.

Because R-125 is present in R-410A, the P-T-x data acquired by Martz, et al. (1996a) for R-125 mixed with a POE are used in this study. The data were acquired with the original version of the binary mixture apparatus used in this work. The system pressure and liquid density are shown as functions of refrigerant mass fraction and temperature in Figures 3.3 and 3.4, respectively. The curves on these figures are provided for readability purposes only, and do not represent modeling efforts.

The binary mixture apparatus was used to acquire P-T-x data for R-410A. Corrections for the vapor shift of the refrigerant were also made using Eqs. 3.1-3.3; however, vapor composition analyses using the GC were necessary to determine the mass

of each component. Pressure and liquid density data for the R-410A system were also acquired by Martz, et al. (1996a), but without the GC measurements. The data from Martz, et al. are shown in Figures 3.5 and 3.6, respectively. The R-32/POE data acquired in this work are listed in Appendix B, and data acquired by Martz, et al. (1996a) can be found in their work.

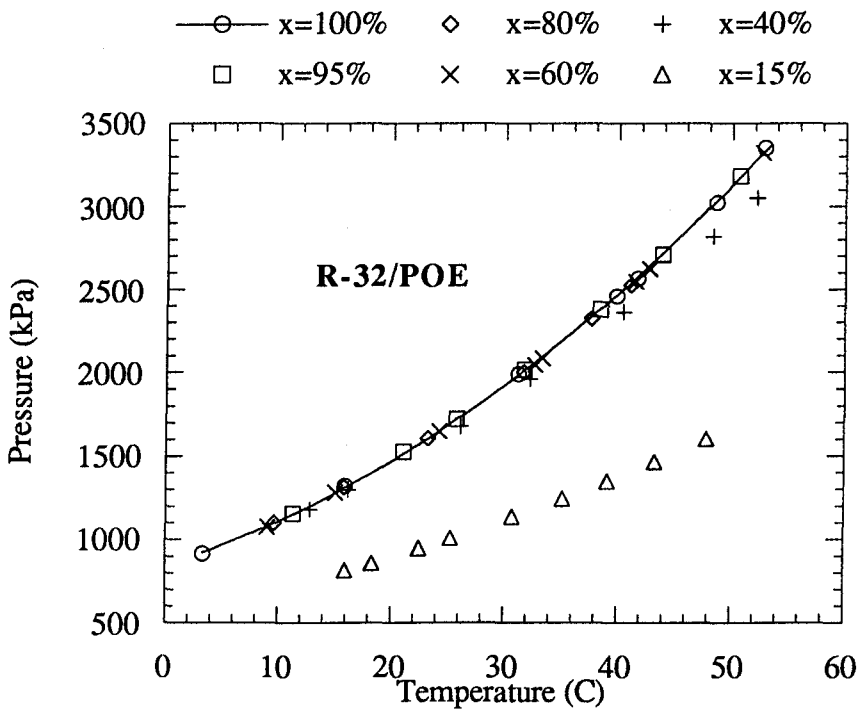

Figure 3.1. R-32 and POE vapor pressure as a function of temperature and refrigerant mass fraction.

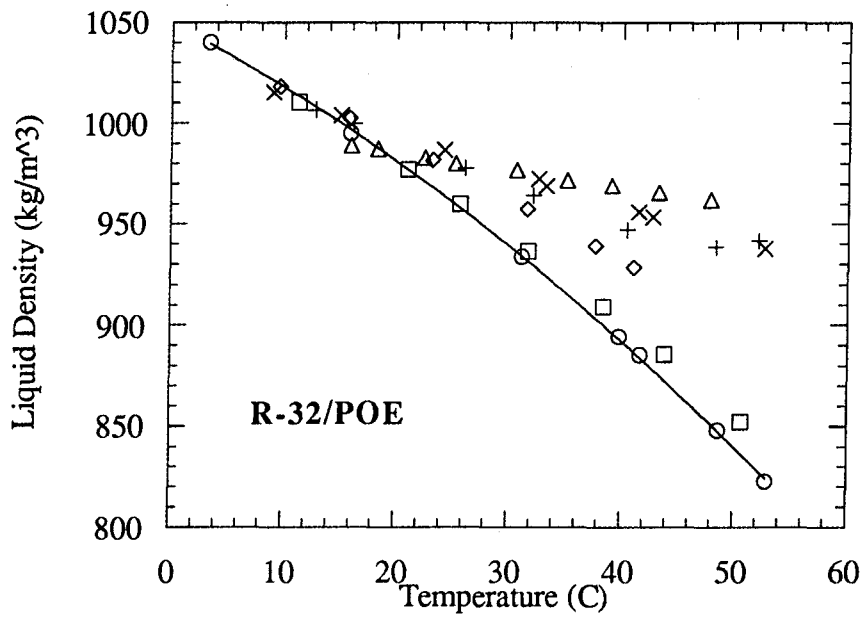

Figure 3.2. R-32 and POE liquid density as a function of temperature and refrigerant mass fraction.

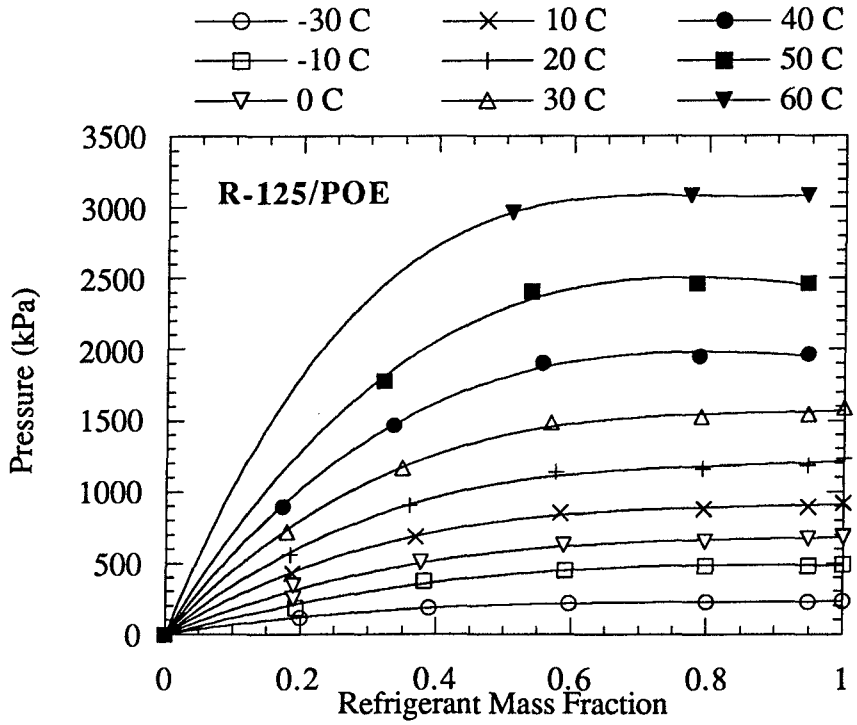

Figure 3.3. R-125 and POE vapor pressure as a function of temperature and refrigerant mass fraction (from Martz, et al. 1996a).

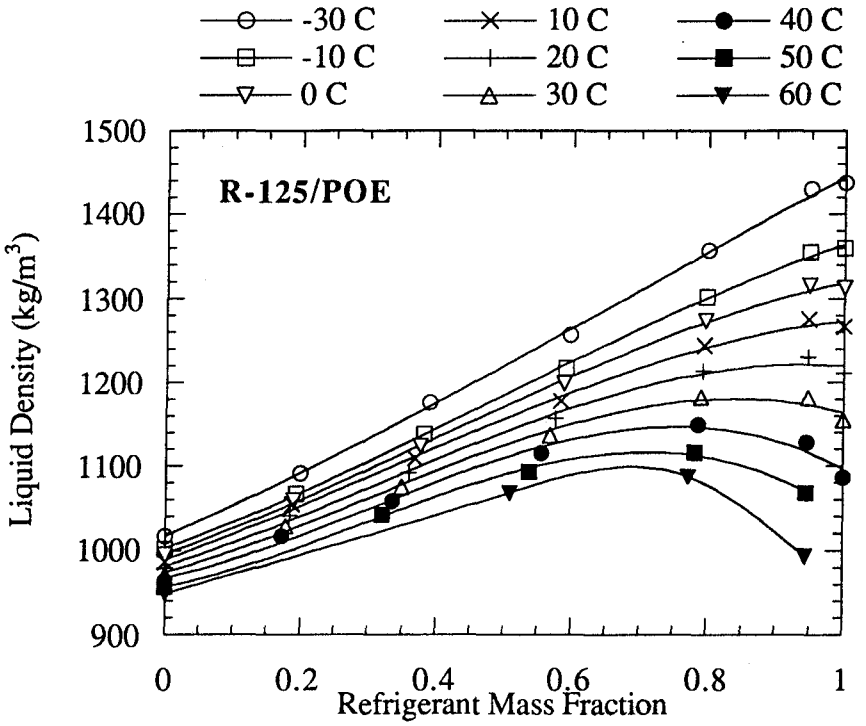

Figure 3.4. R-125 and POE liquid density as a function of temperature and refrigerant mass fraction (from Martz, et al. 1996a).

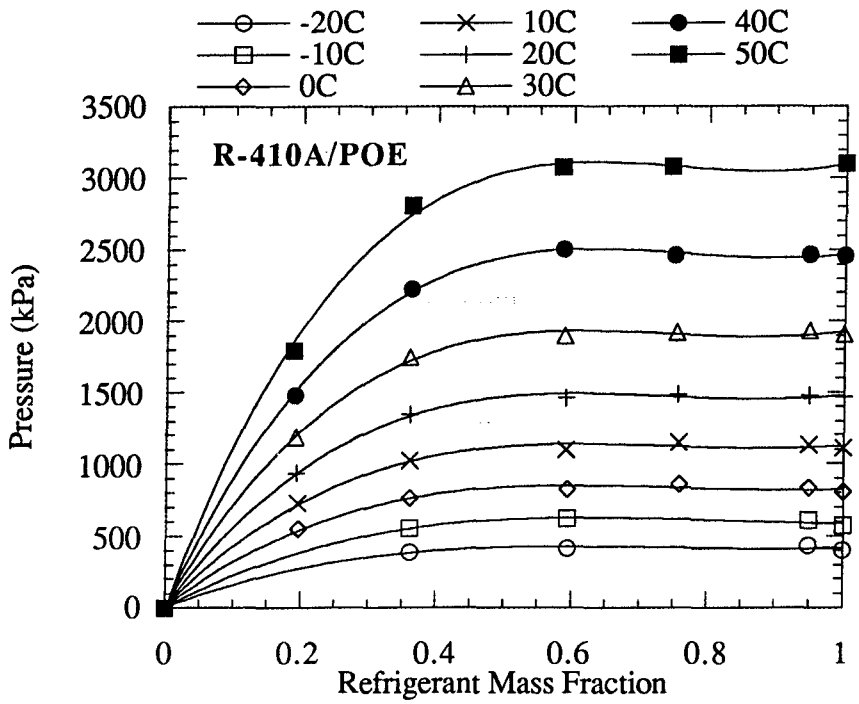

Figure 3.5. R-410A and POE vapor pressure as a function of temperature and refrigerant mass fraction (from Martz, et al. 1996a).

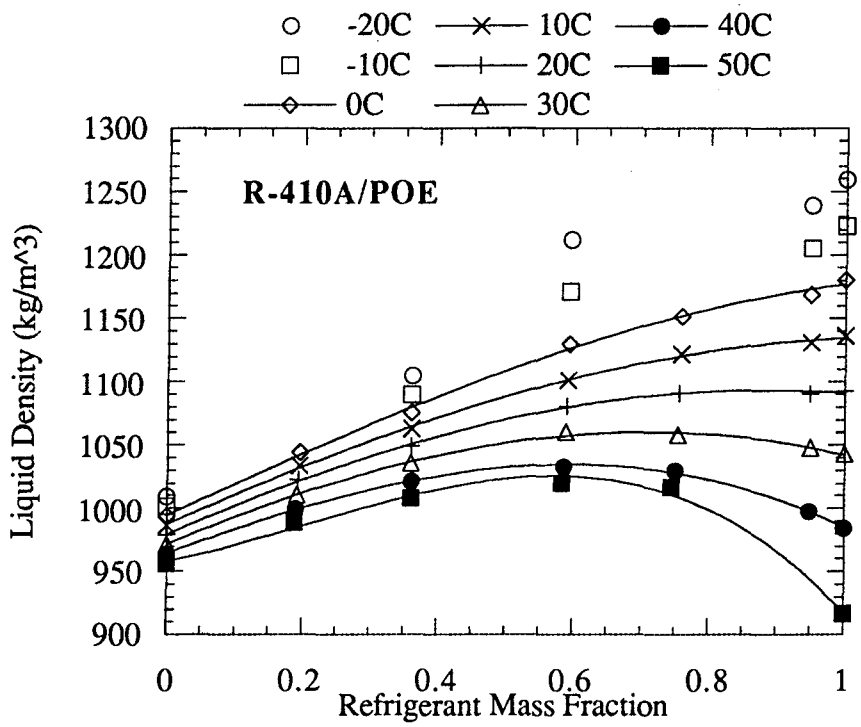

Figure 3.6. R-410A and POE liquid density as a function of temperature and refrigerant mass fraction(from Martz, et al. 1996a).

3.2 Interpretation of the Data

In modeling refrigerant-oil mixtures, departures from ideal behavior are manifested in several ways. Vapor phase non-ideality is interpreted with fugacity, liquid phase non-ideality with fugacity and the Poynting effect, and mixture non-ideality with activity. For the current purpose, these quantities are calculated for the refrigerant components in each mixture.

To determine the fugacity of a mixture component, an equation of state is used with the mixture conditions. For this work, the Peng-Robinson (1976) equation, a cubic equation, was selected. Lin and Daubert (1980) and Moshfeghian et al. (1992) investigated the performance of various equations of state, and determined the Peng-Robinson equation to be well-suited for refrigerant mixtures. Martz, et al. (1996a,b) also used this equation with success. The equation is (Peng and Robinson, 1976):

$$P = \frac{RT}{v-b} - \frac{a\alpha}{v(v+b)+b(v-b)} \tag{3.6a}$$

where

$$a = 0.45724\frac{(RT_c)^2}{P_c} \tag{3.6b}$$

$$b = 0.07780\frac{RT_c}{P_c} \tag{3.6c}$$

$$\alpha = \left(1 + M\left\{1 - \sqrt{T/T_c}\right\}\right)^2 \tag{3.6d}$$

This equation uses two parameters, a and b, which are found using the critical properties of the vapor being modeled. Following Martz, et al. (1996a), a third parameter, M, is determined by minimizing the error associated with predicting saturated refrigerant conditions provided by ASHRAE (1993). This method results in saturation pressure predictions within 1.5% of ASHRAE (Martz, 1996a). The values of M found for R-32 and R-125 are 0.77281 and 0.81485, respectively. Only reduced temperatures below 0.95 were used in this study to eliminate performance problems near the critical temperature.

To determine the fugacity coefficient of a component in a mixture, $\overline{\Phi}_i = f_i/y_iP$, the following equation is used (Peng and Robinson, 1976):

$$\ln\overline{\Phi}_i = \frac{b_i}{b}(z-1) - \ln(z-B) - \frac{A}{2B\sqrt{2}} \times \left(\frac{2\sum_j x_j a_{ji}}{a} - \frac{b_i}{b}\right)\ln\left(\frac{z+2.414B}{z-0.414B}\right) \tag{3.7a}$$

where

$$a = \sum_i \sum_j x_i x_j a_{ij} \tag{3.7b}$$

$$b = \sum_i x_i b_i \tag{3.7c}$$

$$a_{ij} = \left(1-\partial_{ij}\right)a_i^{1/2}a_j^{1/2} \tag{3.7d}$$

and

$$A = \frac{a\alpha P}{(RT)^2} \tag{3.7e}$$

$$B = \frac{bP}{RT}. \tag{3.7f}$$

Eqs. 3.7b-3.7d are mixing rules given by Peng and Robinson. By using Eqs. 3.6b and 3.6c, the parameters, a and b, for each component, i and j, can be determined; thus, Eqs. 3.7b-3.7d provide a, b, and a_{ij}, where the mixing parameter ∂_{ij} is found in the same manner as the parameter M. The value found for R-410A is $\partial_{ij} = \partial_{ji}$=0.0072 ($\partial_{ii} = \partial_{jj} = 1$).

The fugacity coefficient for saturated pure R-32 is shown in Figure 3.7, where it can be seen that $\overline{\Phi}_{i,s}$ ranged from 0.87 to 0.73. Figure 3.8 shows the mixture fugacity coefficient for the R-32/POE system, where both the system temperature and pressure are used. By comparing the results shown in Figs. 3.7 and 3.8, it is found that the lubricant does not significantly affect saturated mixture pressure and temperature until the refrigerant mass fraction is reduced to 40%. A substantial effect is seen for a refrigerant mass fraction of 15%. These results, also evident in Fig. 3.1, would occur for a system that is immiscible at refrigerant mass fractions greater than 40%. Although these results do not

prove immiscibility, Takaishi and Oguchi (1995) observed a large range of immiscibility for the same mixture in a similar study.

The liquid phase non-ideality can also be expressed in terms of fugacity. For thermodynamic equilibrium, the saturated liquid fugacity must equal the saturated vapor fugacity, and therefore, a second term, the Poynting effect (Pe), is used to show the influence of system pressure on the liquid fugacity (Tassios, 1993). The Poynting effect for a component i in a mixture is approximately:

$$Pe_i \approx \exp\left[\frac{v_{i,l}(P - P_{i,s})}{RT}\right]. \tag{3.8}$$

In Eq. 3.8, the influence of system pressure on the liquid fugacity is accounted for by the difference between P, the system pressure, and $P_{i,s}$, the saturation pressure for the pure component. The Poynting effect for R-32 mixed with a POE is shown in Fig. 3.9, where once again the system behaves as if it were immiscible above a refrigerant mass fraction of 40%.

Using the Poynting effect, the liquid-phase fugacity of a component in a mixture may be written as (Tassios, 1993):

$$f_{i,l} = x_i \gamma_i P_{i,s} \overline{\Phi}_{i,s} Pe_i. \tag{3.9}$$

where x_i is the liquid mole fraction of component i, and γ_i is the activity coefficient. Noting that the fugacities of component i in the liquid and vapor must be the same for equilibrium conditions, Eq. 3.9 may be rearranged into an expression for the activity coefficient:

$$\gamma_i = \frac{y_i P \overline{\Phi}_{i,vapor}}{x_i P_{i,s} \overline{\Phi}_{i,s} Pe_i}. \tag{3.10}$$

An in-depth development of the fugacities, Poynting effect, and activity coefficient is found in Martz, et al. (1996a).

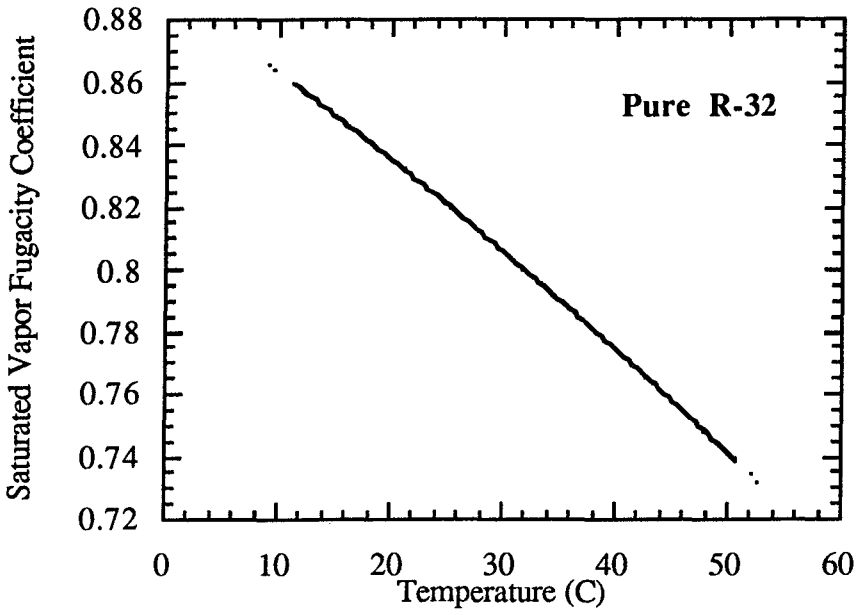

Figure 3.7. Saturated vapor fugacity as a function of temperature for pure R-32.

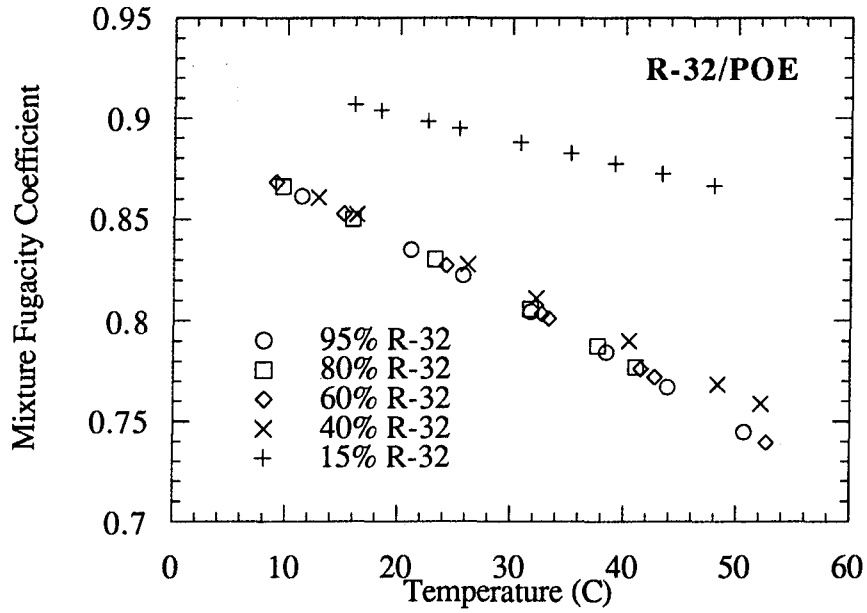

Figure 3.8. Mixture fugacity as a function of temperature and refrigerant mass fraction for R-32 mixed with a POE.

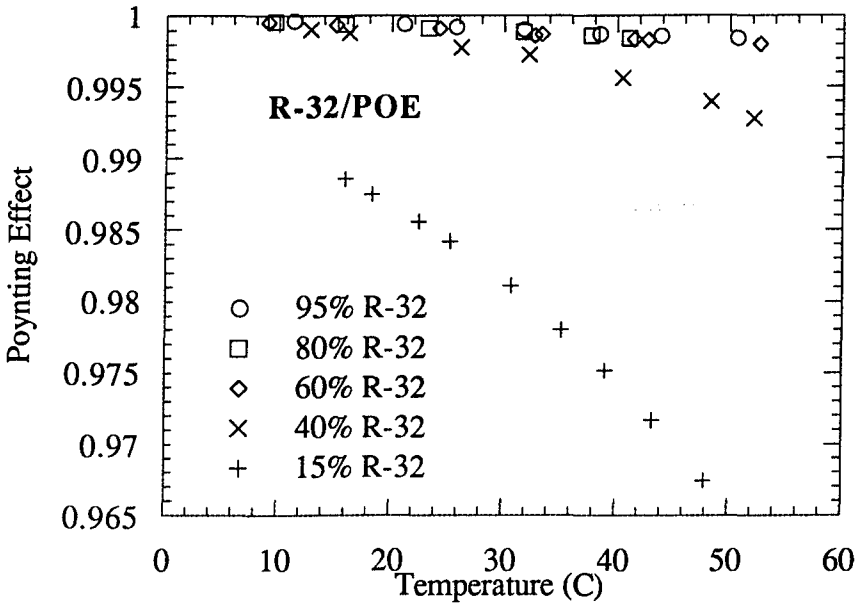

Figure 3.9. Poynting effect as a function of temperature and refrigerant mass fraction for R-32 mixed with a POE.

An interpretation of the activity coefficient is found by considering Raoult's law, which states:

$$P_i = y_i P = x_i P_{i,s}. \tag{3.11}$$

This applies to VLE where the vapor acts ideally. To account for non-ideality in the vapor, the Lewis-Randall rule allows fugacities to be substituted in place of the pressures.

$$f_i = y_i f_{v,i} = x_i f_{l,i} \tag{3.12}$$

These rules apply only to ideal mixtures, and since most mixtures are not ideal, the activity coefficient is used to measure the mixture non-ideality. Activity coefficients greater than one indicate partial pressures higher than the product of the mole fraction and the saturation pressure, and strong attractions between like molecules in the mixture. On the other hand, activity coefficients less than one suggest strong attractions between unlike molecules (Tassios 1993).

The activity coefficient for the R-32/POE system is shown in Fig. 3.10. The values of the activity coefficient for the mixtures with refrigerant mass fractions equal to or greater

than 40% are all greater than one, indicating partial pressures higher than the product of the mole fraction and the saturation pressure and strong attractions between like molecules. This confirms what is readily seen in Fig. 3.1, where the system pressure is shown to remain approximately equal to the saturation pressure even though the refrigerant liquid mass fraction is decreasing. It also helps explain the observed immiscibility of the mixture by Takaishi and Oguchi (1995). At a refrigerant mass fraction of 15%, the activity coefficient is less than one, the system pressure is less than the saturation pressure, and the liquid phase was observed by Takaishi and Oguchi to be miscible.

Martz, et al. (1996a) presented activity coefficients for the R-125/POE system, and those results are provided here (Fig. 3.11) because of their relevance to the R-410A mixture. The system had mixed values for the activity coefficient, and miscibility issues were not a concern.

Martz, et al. acquired data for R-410A mixed with a POE assuming no composition shift of the refrigerant in the vapor and liquid phases. Using these R-410A/POE data, the activity coefficients for both the R-32 and R-125 components were determined, and are shown in Figs. 3.12 and 3.13. For both refrigerants, the activity coefficients found from the ternary data are higher than the values found for the pure refrigerant mixed with a POE (Figs. 3.10 and 3.11). This indicates strong intermolecular attraction between like molecules.

The data presented in Figs. 3.12 and 3.13 follow a general trend: the activity coefficient increases with decreasing liquid mole fraction, except for the lowest liquid mole fraction set, which shows a sudden decrease. This behavior is also seen for pure R-32 mixed with a POE (Fig. 3.10) and indicates a region of immiscibility. This behavior was also observed for the R-410A/POE system studied by Martz et al. (1996a), and a more detailed discussion of the miscibility of their mixture can be found in their work.

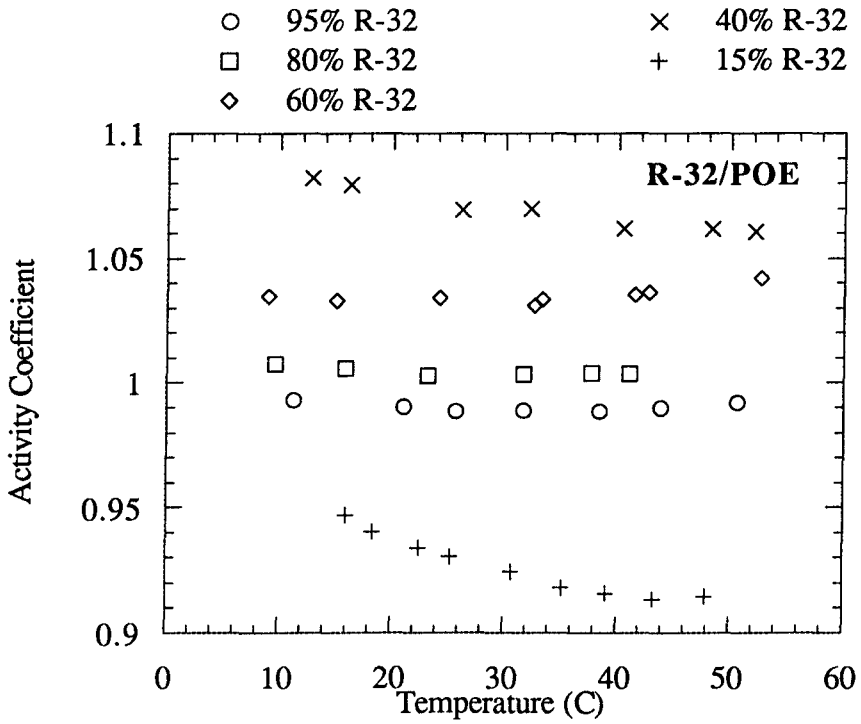

Figure 3.10. Activity coefficient as a function of temperature and refrigerant mass fraction for R-32 mixed with a POE.

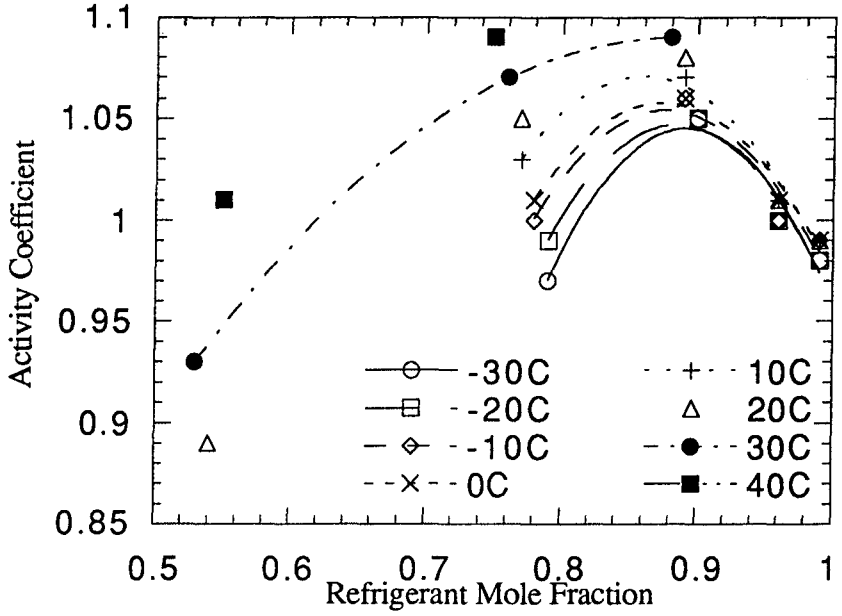

Figure 3.11. Activity coefficient as a function of mole fraction and temperature for R-125 mixed with a POE (from Martz, et al. (1996a)).

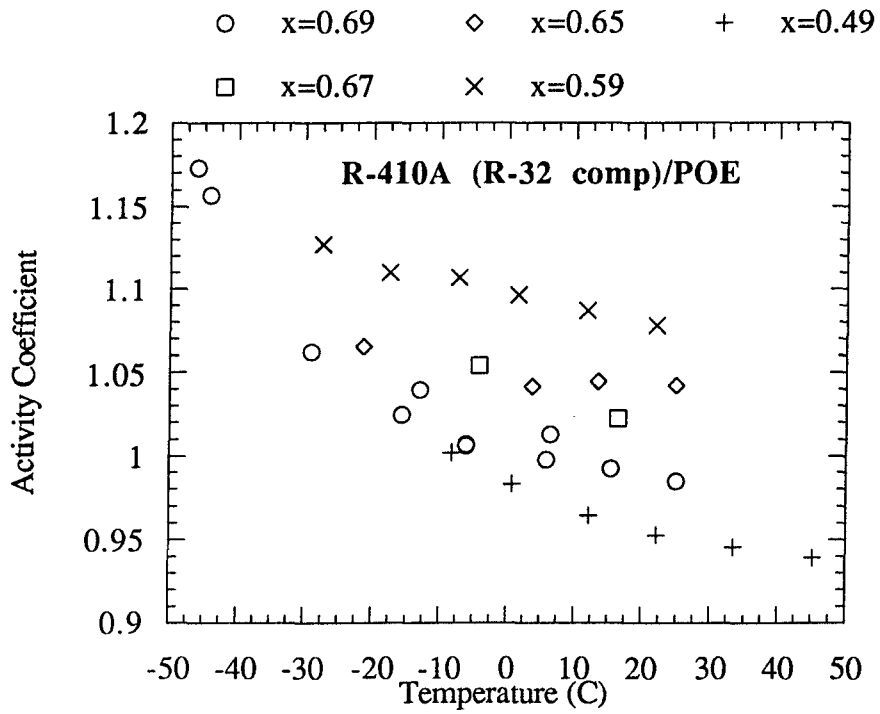

Figure 3.12. Activity coefficient of the R-32 component of R-410A mixed with a POE as a function of liquid mole fraction and temperature.

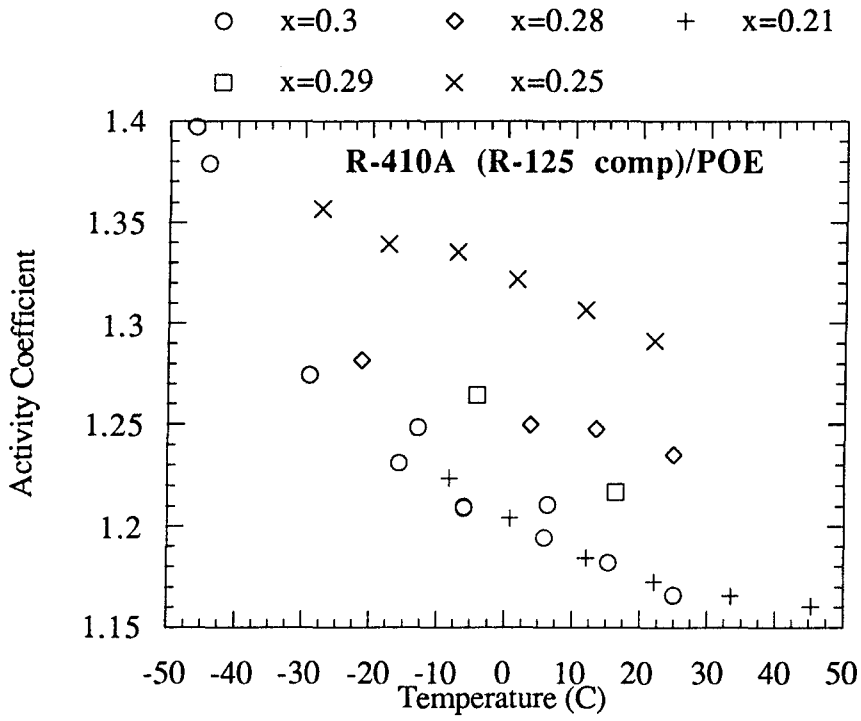

Figure 3.13. Activity coefficient of the R-125 component of R-410A mixed with a POE as a function of liquid mole fraction and temperature.

3.3 Gas Chromatography Results

Preliminary gas chromatograph measurements were made for R-410A/POE at nine different conditions. The measured temperature, pressure, liquid level, and component mass fraction in the vapor phase are given in Table 3.1. The refrigerant vapor used to charge the system was analyzed in the GC to allow the mass of R-32 and R-125 to be determined. The original masses of each refrigerant component are correctly determined in this manner, since the vapor and liquid phases are almost exactly the same composition throughout a wide range of temperatures, pressures, and compositions (Nagel and Bier, 1995). The original masses of R-32 and R-125 were 467.5 g (1.03 lb) and 366.7 g (0.808 lb), respectively. To change the composition, 636.5 g (1.40 lb) of R-32 and 499.3 g (1.10 lb) of R-125 were added for the second half of the data points. The mass of the oil remained constant throughout the experiments at 570.0 g (1.26 lb).

Table 3.1. Temperature, pressure, liquid level, and vapor composition measurements for R-410A/POE.

Temperature (C)	Pressure (kPa)	Liquid level (mm)	Mass fraction R-32 in vapor	Mass fraction R-125 in vapor
-5.1	679.19	15	0.5773	0.4227
9.38	1056.48	12	0.5688	0.4312
23.43	1553.88	3	0.5622	0.4378
37.81	2209.27	0	0.5947	0.4054
-5.17	679.58	133	0.6226	0.3774
9.07	1051.103	137	0.6165	0.3835
23.24	1568.44	137	0.5865	0.4135
37.08	2229.859	135	0.5771	0.4229
51.24	3120.33	127	0.5713	0.4287

Although the variance of the mass fraction (for each set of temperatures) is within the experimental uncertainty, the data suggest that the mass fraction of R-32 decreases slightly with increasing temperature. The only exception to this apparent trend is the data for 37.81°, where there is an increase in R-32 mass fraction which is probably due to poor sampling.

The liquid level measurement was used in Eq. 2.1 to determine the liquid and vapor volumes. Using the Peng-Robinson equation, the vapor density of the mixture was determined, allowing the mass of each component in the vapor to be found from GC data.

With the starting mass of each component determined, the composition in the liquid phase was also determined. The amount of mass of R-32 and R-125 removed in the sampling cylinder was determined in the same manner. Because two components are present in the vapor phase, only one component needs to be specified; however, three components are present in the liquid phase and two components need to be specified. The resulting composition measurements, on both a mass and molar basis, are listed in Table 3.2 (the temperature data are repeated for easy reference).

Table 3.2. Temperature, liquid mass fraction of R-32, liquid mass fraction R-125, vapor mole fraction of R-32, liquid mole fraction of R-32, and liquid mole fraction of R-125 of R-410A/POE.

Temp (C)	Liquid mass fraction R-32	Liquid mass fraction R-125	Vapor mole fraction R-32	Liquid mole fraction R-32	Liquid mole fraction R-125
-5.1	0.3177	0.2510	0.5771	0.6928	0.2372
9.38	0.3061	0.2420	0.5688	0.6885	0.2359
23.43	0.2872	0.2264	0.5620	0.6814	0.2328
37.81	0.2440	0.2096	0.5945	0.6499	0.2419
-5.17	0.4231	0.3362	0.6224	0.7211	0.2483
9.07	0.4195	0.3352	0.6164	0.7195	0.2491
23.24	0.4154	0.3319	0.5864	0.7186	0.2488
37.08	0.4083	0.3272	0.5771	0.7166	0.2488
51.24	0.3950	0.3182	0.5712	0.7127	0.2488

3.4 Model Results

There are two commonly adopted methods available for theoretically modeling refrigerant-oil mixtures: equations of state and activity coefficient modeling. Since the vapor pressure of lubricating oils is typically 12 orders of magnitude smaller than refrigerant vapor pressure, equations of state cannot be accurately used for these mixtures (Spauschus, 1963). However, since activity coefficient modeling does not require extensive property information of the lubricating oil, it is well suited to modeling these mixtures.

Local composition models relate the activity coefficient (Eq. 3.10) of a component in a mixture to temperature, pressure, and mole fraction. The models are based on a hypothesized form of the Gibbs energy in terms of interaction parameters. The accuracy of the model is dependent on the ability of the excess Gibbs energy formulation to predict the

mixture behavior. Each model used in this work requires n(n-1) interaction parameters for a mixture containing n components. These models are limited to low-pressure systems(< 1700 kPa (250 psia)) of non-electrolytes (Martz, 1994).

Six local composition models were used to model the two mixtures studied in this work and the R-125/POE mixture studied by Martz, et al. (1996b). These models were used by Martz and co-workers to study the behavior of 7 different pure refrigerants mixed with an oil. The models used, specifically, are the Wilson relation, Heil equation, Tsuboka and Katayama model, Wang and Chao equation, non-random two liquid theory (NRTL) and universal quasi-chemical theory (UNIQUAC). Each of these models will be discussed here briefly, and a comparison of their results will be provided. Martz, et al. (1996b) provide an in-depth theoretical development and explanation of these models, and successfully used them for two-component systems. Their use will be extended to three component systems in this work.

The most well-known local composition model is the Wilson relation; it can be reduced to Flory-Huggins polymer theory if the binary interaction parameters for each pair of components in a mixture are assumed to be equal. Wilson (1964) developed an expression for the excess Gibbs energy of a mixture, and used the Gibbs-Duhem relation to formulate the activity coefficient as

$$\ln\gamma_i = 1 - \ln\sum_{j=1}^{n}\Lambda_{ji}x_j - \sum_{k=1}^{n}\left[\frac{\Lambda_{ik}x_k}{\sum_{j=1}^{n}\Lambda_{kj}x_j}\right] \tag{3.13a}$$

where

$$\Lambda_{ij} = \frac{v_i}{v_j}\exp\left(-\frac{\Delta\lambda_j}{RT}\right). \tag{3.13b}$$

Once the interaction parameters, $\Delta\lambda_j$, are determined, the model is closed. The activity coefficient, and thus the vapor pressure, for a particular composition and temperature can then be determined. For a two-component system, the Wilson relation becomes

$$\ln\gamma_1 = -\ln(x_1 + x_2\Lambda_{21}) + x_2\left[\frac{\Lambda_{21}}{x_1 + x_2\Lambda_{21}} - \frac{\Lambda_{12}}{x_1\Lambda_{12} + x_2}\right] \tag{3.13c}$$

where

$$\Lambda_{12} = \frac{v_1}{v_2}\exp\left(-\frac{\Delta\lambda_2}{RT}\right) \tag{3.13d}$$

and

$$\Lambda_{21} = \frac{v_2}{v_1}\exp\left(-\frac{\Delta\lambda_1}{RT}\right). \tag{3.13e}$$

Tsuboka and Katayama (1975) modified Wilson's relation to account for excess enthalpy; the resulting equation is:

$$\begin{aligned}\ln\gamma_i = &-\ln\left[\sum_j^n x_j\Lambda_{ji}\right] - \sum_k^n\left(x_k\Lambda_{ik}\Big/\sum_j^n x_j\Lambda_{jk}\right) \\ &+\ln\left[\sum_j^n x_j\rho_{ji}\right] + \sum_k^n\left(x_k\rho_{ik}\Big/\sum_j^n x_j\rho_{jk}\right)\end{aligned} \tag{3.14}$$

The interaction parameters are incorporated as with the Wilson relation using Eq. 3.13b, and $\rho_{ij} = v_i / v_j$.

The Wang and Chao (1983) model (shown here for a two-component system) represents both excess enthalpy and entropy:

$$\begin{aligned}\ln\gamma_i = &\frac{1}{RT}\left(\frac{n_c}{2}\right)\left[x_{21}^2(\Delta\lambda_1) + x_2x_{22}\frac{x_{12}}{x_1}(\Delta\lambda_2)\right] \\ &-\ln(x_1 + x_2\Lambda_{21}) + x_2\left[\frac{\Lambda_{21}}{x_1 + x_2\Lambda_{21}} - \frac{\Lambda_{12}}{x_1\Lambda_{12} + x_2}\right]\end{aligned} \tag{3.15a}$$

where

$$x_{ii} = \frac{x_i}{x_i + x_j\exp(-\Delta\lambda_i/RT)}. \tag{3.15b}$$

$$x_{ij} = \frac{x_{jj}x_i}{x_j}\exp(-\Delta\lambda_j/RT) \tag{3.15c}$$

This model also uses the parameters defined by Wilson in Eq. 3.13b.

Heil and Prausnitz (1966) developed an expression for modeling solutions containing polymers. Their model is based on the following expression for the excess Gibbs energy:

$$\frac{\Delta G_M}{RT} = \sum_{i=1}^{n}\left\{\left[x_i \ln \xi_{ji}\right] + \sum_{\substack{j=1 \\ (j\neq i)}}^{n-1} x_i \xi_{ii}\left(\Delta\lambda_j/RT\right)\right\} \tag{3.16a}$$

where

$$\xi_{ii} = \frac{x_i v_i}{\sum_{k=1}^{n} x_k v_k \exp(-\Delta\lambda_k/RT)} \tag{3.16b}$$

$$\xi_{ji} = \frac{x_j v_j \exp\left(-\Delta\lambda_j/RT\right)}{\sum_{k=1}^{n} x_k v_k \exp(-\Delta\lambda_k/RT)}. \tag{3.16c}$$

Renon and Prausnitz (1968) extended this expression for excess Gibbs energy to the following relation for activity coefficient:

$$\ln\gamma_i = \left(1 - \ln\sum_{j=1}^{n} x_j\Lambda_{ji} - \sum_{j=1}^{n}\frac{x_j\Lambda_{ij}}{\sum_{k=1}^{N}\Lambda_{kj}x_k}\right) + \left[\frac{\sum_{j=1}^{n}\tau_{ji}\Lambda_{ji}x_j}{\sum_{k=1}^{n}\Lambda_{ki}x_k} + \sum_{j=1}^{n}\frac{x_j\Lambda_{ij}}{\sum_{k=1}^{N}\Lambda_{kj}x_k}\left(\tau_{ij} - \frac{\sum_{l=1}^{n}x_l\tau_{lj}\Lambda_{lj}}{\sum_{k=1}^{n}\Lambda_{kj}x_k}\right)\right] \tag{3.16d}$$

where

$$\tau_{ij} = \frac{\Delta\lambda_j}{RT} \tag{3.16e}$$

and the other parameter is defined in the same way as the Wilson model (Eq. 3.13). This model, although developed for use with polysegmented molecules mixed with solvents, is applied here assuming that each molecule has one segment.

Renon and Prausnitz (1968) modified Wilson's relation with a parameter to account for the "non-randomness" of liquid systems. Martz, et al. (1996b) found that a value for this parameter of ξ=0.5 is most effective for refrigerant-oil mixtures. Thus it is used here, also. The NRTL equation is:

$$\ln\gamma_i = \frac{\sum_{j=1}^{n} x_j\tau_{ji}\Lambda_{ji}}{\sum_{k=1}^{n} x_k\Lambda_{ki}} + \sum_{j=1}^{n}\left\{\frac{x_j\Lambda_{ij}}{\sum_{k=1}^{n} x_k\Lambda_{kj}}\left(\tau_{ij} - \frac{\sum_{m=1}^{n} x_m\Lambda_{mj}\tau_{mj}}{\sum_{k=1}^{k} x_k\Lambda_{kj}}\right)\right\} \tag{3.17a}$$

where

$$\tau_{ij} = \Delta\lambda_j/RT \tag{3.17b}$$

and

$$\Lambda_{ij} = \exp\left(-\xi\tau_{ij}\right). \tag{3.17c}$$

The final model considered is the universal quasi-chemical theory (UNIQUAC), which incorporates structural parameters r and q into the local composition theory. These structure parameters are based on the chemical formula of each molecule and these data and their use are provided in Appendix C. The UNIQUAC model is (Tassios, 1993):

$$\ln\gamma_i = \ln\frac{\Phi_i}{x_i} + \frac{n_c}{2}q_i\ln\frac{\theta_i}{\Phi_i} + l_i - \frac{\Phi_i}{x_i}\sum_{j=1}^{n} x_j l_j +$$

$$-q_i\ln\left(\sum_{j=1}^{n}\theta_j\tau_{ji}\right) + q_i - q_i\sum_{j=1}^{n}\left(\frac{\theta_j\tau_{ij}}{\sum_{k=1}^{n}\theta_k\tau_{kj}}\right) \tag{3.18a}$$

where

$$\theta_i = \frac{q_i x_i}{\sum_{j=1}^{n} q_j x_j} \qquad \Phi_i = \frac{r_i x_i}{\sum_{j=1}^{n} r_j x_j} \tag{3.18b,c}$$

$$l_i = \frac{n_c}{2}(r_i - q_i) - (r_i - 1) \tag{3.18d}$$

and

$$\tau_{ij} = \exp(-\Delta\lambda j/RT) \tag{3.18e}$$

The coordination number, n_c, is set equal to 10 for every mixture modeled with this set of equations (Tassios, 1993).

Once the binary interaction parameters are found for each pair of components in a mixture, each model is closed. Though some of the models are defined with three parameters for each pair of components, such an extra parameter is considered a constant in this work.

The choice of the interaction parameters used in each model is based on the following objective function suggested by Silverman and Tassios (1984):

$$Q = \sum_{i=1}^{N} \left(\frac{P_{exp} - P_{calc}}{P_{exp}} \right)^2 \tag{3.19}$$

Once the parameters are determined, a mixture pressure is estimated to allow the calculation of fugacities, and the models are used to predict the activity coefficient of a refrigerant component. The activity coefficient is then used in Eq. 3.10 to calculate a value for the mixture pressure. The calculated and estimated mixture pressures are compared and an iterative scheme continues until the difference between the values is minimized.

Each of the models was used to predict the behavior of the binary combinations in the R-410A/POE blend (R-32/POE, R-125/POE, and R-32/R-125). The performance of the model is based on a 2-σ error (%) in predicting pressure. The results are shown in Tables 3.3-3.5. The binary interaction parameters chosen for the R-32/R-125 blend were obtained using pure R-410A data assuming no composition shifts. These data were acquired by Martz, et al. (1996a).

Table 3.3. Model parameters and 2-σ error associated with predicting pressure for R-32/POE.

Model	$\Delta\lambda_1$	$\Delta\lambda_2$	2-σ Error (%)
Wilson	2650.4	20000.0	6.43
Heil	1204.5	6706.6	4.81
Wang and Chao	2084.5	15744.1	5.96
Tsuboka and Katayama	1143.9	-15057.2	4.92
NRTL	-2852.1	20000.0	5.89
UNIQUAC	-200.0	20000.0	12.7

Table 3.4. Model parameters and 2-σ error associated with predicting pressure for R-125/POE.

Model	$\Delta\lambda_1$	$\Delta\lambda_2$	2-σ Error (%)
Wilson	1441	20000	5.6
Heil	642	5493	5.5
Wang and Chao	-381	4579	3.8
Tsuboka and Katayama	918	-4941	7.4
NRTL	-3363	8481	3.5
UNIQUAC	1715	-570	9.1

Table 3.5. Model parameters and 2-σ error associated with predicting pressure for R-32/R-125.

Model	$\Delta\lambda_1$	$\Delta\lambda_2$	2-σ Error (%)
Wilson	190.7	0.4	1.76
Heil	87.8	11.0	1.76
Wang and Chao	78.7	0.0	1.76
Tsuboka and Katayama	-69.4	-10.1	1.83
NRTL	237.5	-398.1	1.84
UNIQUAC	125.6	0.9	1.75

For the R-32/POE blend, the Heil model performed the best, with a 2-σ error of 4.81%, while the UNIQUAC model performed the worst with an error of 12.7%. The performance results for this mixture are in agreement with the conclusions by Martz, et al. (1996b). They concluded that for mixtures with activity coefficients both greater than and less than one, that the Heil and Tsuboka and Katayama models performed the best, with the Heil model performing the best overall. An in-depth discussion of the R-125/POE results, along with a more robust comparison of model performance is provided by Martz, et al. (1996b).

The R-32/POE interaction parameters for the Wilson and Heil equations were both positive, which agrees with the findings by Martz, et al. (1996b) for mixtures with activity coefficients both greater than and less than one. The NRTL parameters were also in agreement with Martz, et al. The first was negative and the second was positive, which corresponds to both positive and mixed deviations. No clear patterns were found for the Tsuboka and Katayama, Wang and Chao, and UNIQUAC model parameters.

There is a dramatic difference between the 2-σ errors for the R-32/R-125 blend and the single refrigerants blended with oil, as can be seen by comparing Tables 3.3-3.5. This difference can be explained, in part, by considering the UNIQUAC size parameter q_i. In general, the size parameters for oil molecules are an order of magnitude higher than those for refrigerants. For these particular components, the size parameter for the oil is 24.36, while those for R-32 and R-125 are 1.42 and 2.49, respectively. The performance results given in Tables 3.3-3.5 show that the models provide better results for molecules of similar size.

It is also important to note differences between the interaction parameter values for the R-32/R-125 blend and those for the oil blends. The parameters are very large in magnitude for refrigerants blended with oil, while those for R-32/R-125 are relatively small. For example, comparing the R-32/POE Wilson parameters (2650.4, 20000.0) and those for R-32/R-125 (190.7, 0.40)—the differences are apparent. The refrigerant molecules are very close in size, their interaction energies are much closer in value, and they are chemically similar, resulting in much better model performances.

The results for modeling the R-410A/POE data acquired by Martz, et al. (1996a) are given in Table 3.6. Because these data were acquired without vapor composition analysis, the modeling is performed assuming that a 50/50 mass blend of R-32 and R-125 exists in both the vapor and liquid phase. The Heil model performs the best, while the UNIQUAC model has an unacceptable performance (55.1%).

Table 3.6. 2-σ error associated with predicting pressure for R-410A/POE assuming 50/50 mass blend in the liquid and vapor phases.

Model	2-σ Error (%)
Wilson	12.0
Heil	11.6
Tsuboka and Katayama	20.9
NRTL	14.6
UNIQUAC	55.1

The five models used to predict the R-410A/POE data of Martz, et al. (1996a) were also used to model the R-410A data obtained using GC data to determine phase compositions. Since the programs used to implement these models failed on two of the experimental points, and because three of the points had pressures above 1700 kPa, only four points were used with the models. For the points used, the models, except for UNIQUAC, over-predict pressure by approximately 50%. The UNIQUAC model failed for two of the four points, and over-predicted pressure for the remaining two points by approximately 100%. Because the models perform much better with the assumption of a 50/50 mass blend of R-32/R-125, the drastic difference in errors is attributed to poor gas chromatography. It is necessary to acquire more GC data on this blend to determine model performance with accurate vapor and liquid phase compositions.

The programs used in this work are given in Appendix D and are documented with comment statements. The measured and calculated pressures for R-32/POE and R-410A/POE (50/50 mass blend) are also found in Appendix D (Table D.1 and Table D.2).

Martz, et al. (1996a) also investigated the ability to predict the behavior of the refrigerant-oil mixtures with little or no experimental data. One proposed trend was the ability to predict the behavior of the activity coefficient based on the ratio of the molecular weight of the oil to that of the refrigerant. Activity coefficients were greater than one for ratios between 2.7 and 4.1, less than one for ratios greater than 8.1, and mixed for ratios between 5.8 and 6.9. The value of this ratio for the R-32/POE blend is 13.5, but there are mixed deviations from ideal behavior. According to the trend proposed by Martz, et al. this

ratio would suggest only negative deviations. These results do not agree, and further investigation into such generalizations is necessary.

Chapter 4 - CONCLUSIONS

The pressure, temperature, and concentration behavior of a POE blended with R-32 and R-410A has been studied, and thermophysical property data have been provided. The mixtures were tested over a range of refrigerant mass fractions of 15, 20, 40, 60, 80, 95, and 100%, and temperatures varying from -30 °C to 60 °C (-22 °F to 140 °F). The system pressure was restricted to a maximum of 3450 kPa (500 psia). The behavior of R-410A mixed with a POE was predicted using VLE data from R-32/POE, R-125/POE, and R-32/R-125 binary mixtures.

The R-32 and R-410A blends studied in this work were shown to have limited miscibility, which agrees with findings in the literature. Each mixture was investigated in terms of deviation from ideal behavior. The ideality of the vapor phase was interpreted with fugacity, that of the liquid phase with the Poynting effect, and that of the mixture with the activity coefficient.

The activity coefficient for R-32 mixed with a POE had values both greater than and less than one, representing both positive and negative deviations from the Lewis-Randall rule. The effects of miscibility on this system were apparent in the activity coefficient data. The activity coefficient for both refrigerant components in the R-410A mixture was also investigated. The R-32 component had mixed deviations from ideality, but the R-125 component had only a positive deviation from the Lewis-Randall rule. Both R-32/POE and R-125/POE manifested a sudden drop in the activity coefficient when the refrigerant mass fraction entered the miscible region.

Six different local composition models were used to model each binary pair in the R-410A/POE blend (R-32/POE, R-125/POE, and R-32/R-125). Two interaction energy parameters were determined for each model and pair of binary components. For the R-32/POE mixture, the Heil model performed the best, with a 2-σ error in pressure prediction of 4.81%. The worst-case 2-σ error was 12.7% when using the UNIQUAC

relation. These errors were on the order of those found in a comprehensive study by Martz, et al. (1996b). The interaction parameters also behaved according to the trends discussed in Martz, et al.

The errors and interaction energy parameters for the R-32/R-125 binary pair were both significantly smaller than values determined for the other mixtures (2-σ errors less than 1.84%). This difference was discussed in terms of size and structure of the refrigerant and oil molecules. The interaction energy parameters were smaller for molecules closer in size, and the models performed better.

Five of these models were used to predict the behavior of R-410A mixed with a POE, assuming a 50/50 mass blend of R-32 and R-125 in the vapor and liquid phases, and using only the data from each binary combination in the mixture. The models resulted in higher 2-σ errors, with the Heil model performing the best (11.6%) and the UNIQUAC model proving to be unsuitable for this mixture (55.1%). These increases in error were mostly due to combining the errors associated with each binary pair. Another important contributing factor was the error associated with using the Peng-Robinson equation and the corresponding mixing rules. Finally, errors occurred because the liquid phase was treated as a single phase throughout immiscible regions where more than one phase was present, due to the inability to analyze distinct liquid phases.

The vapor composition of R-410A mixed with a POE was determined using gas chromatography. Five of the models were also used to predict the mixture behavior using the measured compositions. The model results were unacceptable, with 2-σ errors of approximately 50%. Since the models had success in predicting the same mixture assuming no composition shifts, the measurements obtained using the GC must be verified. A possible solution to the errors associated with GC sampling is to inject the sample with an automatic sampling valve, instead of the syringe injection technique used in this work.

The ability to predict trends in the activity coefficient for new refrigerant oil mixtures is extremely important, but has not yet been achieved. An initial trend proposed by Martz, et al. (1996b), in which the ratio of the molecular weight of the oil to the refrigerant indicated the type of deviation from ideality, was contradicted by the R-32/POE mixture. Likewise, the inability to predict interaction energy parameters with no data is still a limitation. Investigating a much larger family of mixtures would provide a more robust data set and lead to enhancements in predicting trends and reductions in time-consuming and expensive experiments.

A very important aspect of this work is the extension of local composition models to 3 components. Each binary combination of the mixture must be studied separately, but the actual mixture with all three components does not. This result is very useful to designers, because it reduces the amount of experiments and time needed to determine VLE data. Once the interaction energy parameters for each binary combination have been found, blends containing all three components may be studied without extensive experimental data. The proficiency of these models to predict the behavior of other blends with 3 or more components needs to be investigated.

Further studies into the use of local composition models and empirical models are necessary. Empirical models are quick and simple to implement and easily understood. The local composition models incorporate the physics of mixing, but they are complex and provide little or no advantage in accuracy. The hope of finding a way to predict interaction parameters and activity coefficients used in local composition models is pitted against the ease and accessibility of empirical models. This inherent difference in the modeling strategies is important to the future of research in this area, and must be explicitly investigated.

REFERENCES

ASHRAE, 1993, Fundamentals, American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Atlanta, GA.

Beaton, C. F., and G. F. Hewitt, 1989, *Physical Property Data for the Design Engineer,* Hemisphere Publishing Co., New York.

Bondi, A., 1968, *Physical Properties of Molecular Crystals, Liquids, and Glasses,* Wiley, New York.

Heil, J. F., and J. M. Prausnitz, 1966, "Phase Equilibria in Polymer Solutions," *AIChE J.*, Vol. 12 (4), pp. 678-685.

Lin, C.-T., and T. E. Daubert, 1980, "Estimation of Partial Molar Volume and Fugacity Coefficient of Components in Mixtures from the Soave and Peng-Robinson Equations of State," *Ind. Eng. Chem., Proces Des. Dev.*, Vol. 19, pp. 51-59.

Martz, W. L., 1994, *Refrigerant-Oil Mixtures and Local Composition Modeling*, MS Thesis, University of Illinois, Urbana-Champaign.

Martz, W. L., Burton, C. M., and A. M. Jacobi, 1996a, "Vapor-Liquid Equilibria for R-22, R-134a, R-125, and R-32/125 with a Polyol Ester Lubricant: Measurements and Departure from Ideality," *ASHRAE Trans.*, Vol. 102, Pt. 1, pp. 367-274.

Martz, W. L., Burton, C. M., and A. M. Jacobi, 1996b, "Local Composition Modeling of the Thermodynamic Properties of Refrigerant and Oil Mixtures," *Int. J. Refr.*, Vol. 19, No. 1, pp. 25-33.

Megyesy, Eugene F., *Pressure Vessel Handbook*, 2nd Ed., 1973, Pressure Vessel Handbook Publishing, Inc.; Tulsa, OK, part 1, pp. 8-10.

Moshfeghian, M., Shariat, A., and R. N. Maddox, 1992, "Prediction of Refrigerant Thermodynamic Properties by Equations of State: VLE of Binary Mixtures," *Fluid Phase Equilibria*, Vol. 80, pp. 33-44.

Nagel, M. and K. Bier, 1995, "Vapour-Liquid Equilibrium of Ternary Mixtures of the Refrigerants R32, R125, and R134a," *Int. J. Refr.*, Vol. 18, No. 8, pp. 534-543.

Peng, D. Y., and D. B. Robinson, 1976, "A New Two-Constant Equation of State," *Ind. Eng. Chem., Proces Des. Dev.*, Vol. 15, No. 1, pp. 59-64.

Renon, H., and J. M. Prausnitz, 1968, "Local Compositions in Thermodynamic Excess Functions for Liquid Mixtures," *AIChE J.*, Vol. 14, No. 1, pp. 135-144.

Silverman, N., and D. Tassios, 1984, "Prediction of Multicomponent Vapor-Liquid Equilibrium with the Wilson Equation," *Ind. Eng. Chem., Proces Des. Dev.*, Vol. 23, No. 1, pp. 586-589.

Spauschus, H. O., 1963, " Thermodynamic Properties of Refrigerant-Oil Solutions, Part 1," *ASHRAE Journal*, April, pp. 47-52.

Takaishi, Y., and K. Oguchi, 1987, "Measurements of vapor pressures of R-22/oil solutions," *18th International Congress of Refrigeration Proceedings,* Vol. B, pp. 217-222.

Takaishi, Y., and K. Oguchi, 1995, "Solubility of R-32 and Polyolester Lubricant Mixtures," *19th International Congress of Refrigeration Proceedings,* Vol. IVa, pp. 568-574.

Tassios, D. P., 1993, *Applied Chemical Engineering Thermodynamics,* Springer-Verlag, New York.

Thome, J. R., 1995, "Comprehensive Thermodynamic Approach to Modeling Refrigerant-Lubricating Oil Mixtures," *International Journal of Heating, Ventilating, Air-Conditioning and Refrigerating Research,* Vol. 1, No. 2, pp. 110-126.

Tsuboka, T., and T. Katayama, 1975, "Modified Wilson Equation for Vapor-Liquid and Liquid-Liquid Equilibria," *Journal of Chemical Engineering of Japan,* Vol. 8, No. 3, pp. 181-186.

Wang, W., and K.C. Chao, 1983, "The Complete Local Concentration Models Activity Coefficients," *Chemical Engineering Science,* Vol. 38, No. 9, pp. 1483-1492.

Wilson, G. M., 1964, "A New Expression for the Excess Free Energy of Mixing," *J. Am. Chem., Soc.,* Vol. 86, No. 2, pp. 127-130.

APPENDIX A - DESIGN OF R-32 PRESSURE VESSEL

The pressure vessel used in the R-32 experiments was designed to operate over a pressure range of 0 to 3.45 MPa (500 psia) and a wide temperature range. The vessel was manufactured from 316 Stainless Steel to meet compatibility requirements associated with most refrigerant-oil mixtures. Five pieces were used in the construction of the pressure vessel: a cylindrical body, two end-caps, and a mating set of raised-neck flanges.

The cylindrical body was machined from a seamless, five-inch long piece of Schedule 40 2-inch nominal pipe size 316 SS to the dimensions given in Figure A.1. A factor of safety of n=3 was used in designing the vessel, resulting in a design pressure of 10342 MPa (1500 psia). A minimum wall thickness of 1.42 mm (0.056 in.) was determined using (Megyesy, 1973):

$$t = \frac{PR}{SE - 0.6P} \tag{A1}$$

where,

P = design pressure
R = inside radius
S = material strength
E = joint efficiency (1 for seamless cylinder)
t = wall thickness.

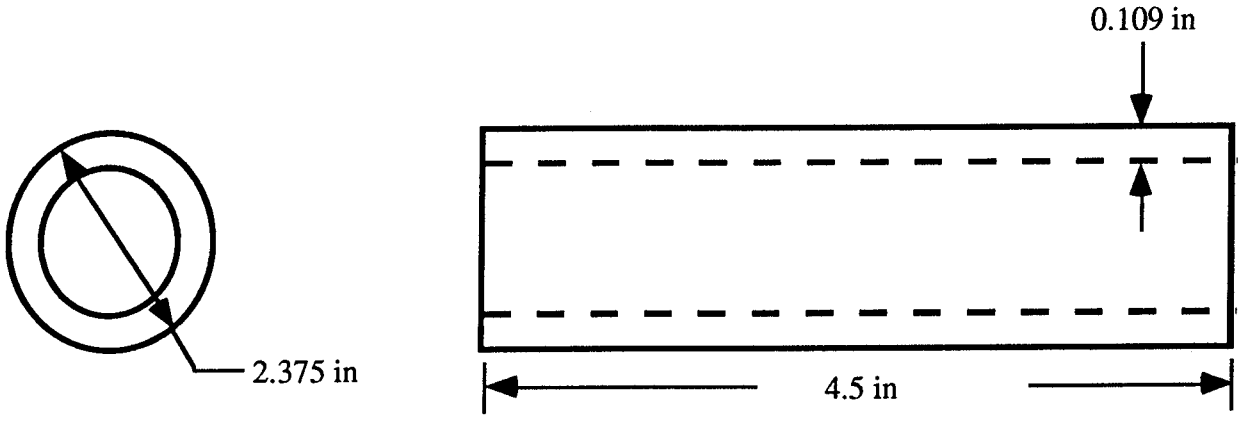

Figure A.1. Dimensioned sketch of body of pressure vessel.

Stainless Steel stock was machined to the dimensions given in Figure A.2 to form the end-caps of the pressure vessel. The minimum wall thickness, 1.48 mm (.058 in), was determined from another analysis provided by Megyesy (1973).

$$t = \frac{PD}{2\cos\alpha(SE - 0.6P)} \tag{A2}$$

where the variables are the same as in Eq. A1, and

D = inside diameter
α = one half of the included (apex) angle.

Each end-cap was tapped for 1/4 NPT.

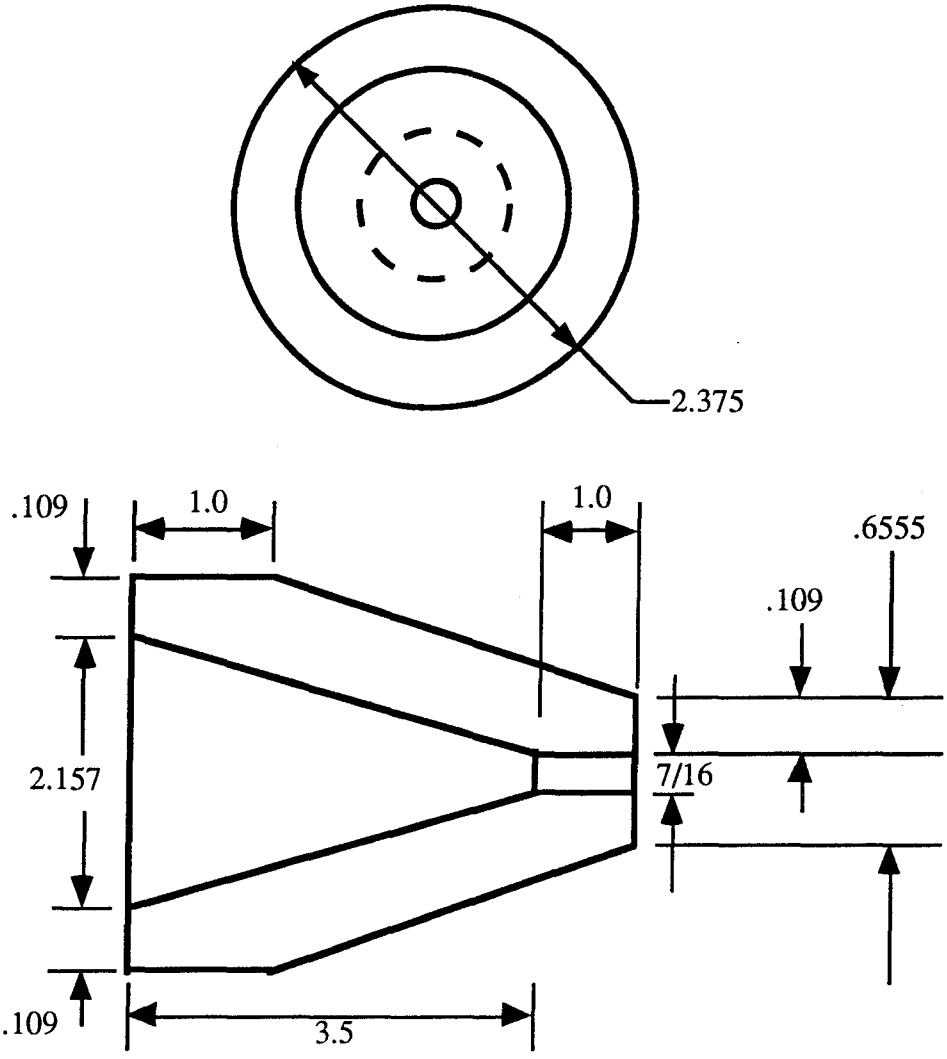

Figure A.2. Dimensioned sketch of end cones of pressure vessel (dimensions in inches).

To assemble the components of the vessel, two 316 SS 300 lb class raised-neck flanges (2-inch size) were used to connect one end, while the opposite end was welded directly together. A Garlock 3400 Gasket was used to create a seal between the flanges. Finally, two pieces of 1/4 in 316 stainless steel tubing were welded into the flanged end-cap to allow placement of instrumentation. The assembled pressure vessel is shown in Fig. A.3.

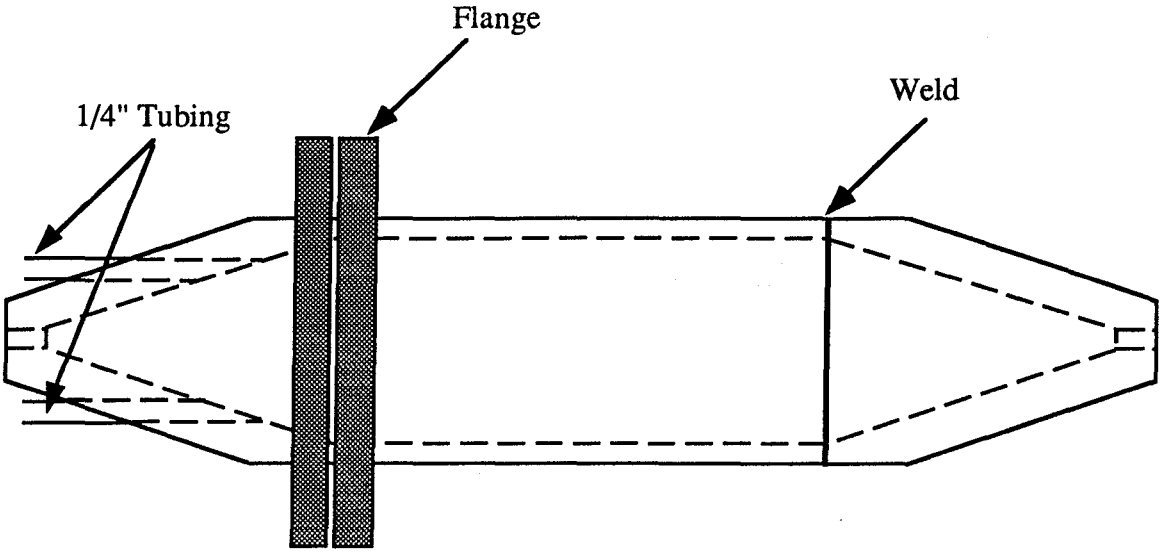

Figure A.3. Sketch of assembled pressure vessel.

APPENDIX B - VLE DATA

Table B.1. R-32/POE thermophysical property data.

T (°C)	P (kPa)	ρ (kg/m^3)	ref. mass fraction
3.36	916.0	1040.20	1.000
15.88	1319.2	995.18	1.000
31.20	1988.8	933.52	1.000
39.80	2458.8	894.21	1.000
41.66	2568.6	885.15	1.000
48.61	3023.4	847.86	1.000
52.89	3351.8	822.60	1.000
11.29	1156.5	1010.40	0.9472
21.11	1523.4	977.00	0.9471
25.74	1721.1	959.74	0.9471
31.76	2014.6	936.52	0.9471
38.46	2378.8	909.32	0.9471
43.89	2714.8	885.47	0.9472
50.66	3183.8	852.21	0.9473
9.65	1097.4	1018.00	0.7721
15.82	1310.6	1002.50	0.7721
23.22	1602.4	981.69	0.7723
31.66	2000.7	957.36	0.7726
37.66	2325.9	938.85	0.7729
41.11	2527.9	928.24	0.7732
9.04	1075.3	1015.10	0.5912
15.08	1279.9	1003.90	0.5893
24.24	1649.0	986.57	0.5859
32.70	2045.6	972.46	0.5818
33.34	2085.1	968.61	0.5815
41.49	2548.6	955.91	0.5762
42.72	2625.9	953.42	0.5752
52.69	3320.0	937.80	0.5655
12.82	1180.7	1006.30	0.3906
16.18	1295.5	999.72	0.3898
26.13	1675.8	977.82	0.3874
32.23	1958.0	963.95	0.3856
40.49	2364.4	946.87	0.3830
48.33	2820.4	938.27	0.3794
52.15	3053.9	941.61	0.3770
25.29	1008.1	980.20	0.1428
30.70	1135.2	976.54	0.1421
35.21	1245.5	971.72	0.1417
39.11	1348.8	969.05	0.1412
43.29	1463.4	965.77	0.1407
47.88	1601.7	962.03	0.1401
15.95	813.3	989.18	0.1436
18.35	858.7	987.56	0.1434
22.50	945.9	982.65	0.1430

Table B.2. R-32/POE non-ideality data.

T (K)	P (kPa)	x_{liq}	$\bar{\Phi}_{i,v}$	$\bar{\Phi}_{i,s}$	Pe	γ_i
284.44	1156.5	0.99588	0.86134	0.85944	0.99965	0.99296
294.26	1523.4	0.99587	0.83510	0.83213	0.99939	0.99029
298.89	1721.1	0.99587	0.82237	0.81865	0.99920	0.98855
304.91	2014.6	0.99587	0.80451	0.80029	0.99903	0.98885
311.61	2378.8	0.99587	0.78434	0.77923	0.99873	0.98838
317.04	2714.8	0.99587	0.76701	0.76165	0.99858	0.98974
323.81	3183.8	0.99588	0.74469	0.73926	0.99843	0.99198
282.80	1097.4	0.97853	0.86613	0.86380	0.99958	1.0074
288.97	1310.6	0.97854	0.85012	0.84710	0.99942	1.0057
296.37	1602.4	0.97855	0.83027	0.82599	0.99911	1.0026
304.81	2000.7	0.97859	0.80565	0.80060	0.99884	1.0032
310.81	2325.9	0.97863	0.78742	0.78178	0.99861	1.0038
314.26	2527.9	0.97866	0.77687	0.77070	0.99842	1.0036
282.19	1075.3	0.95112	0.86798	0.86541	0.99954	1.0346
288.23	1279.9	0.95077	0.85249	0.84915	0.99936	1.0329
297.39	1649.0	0.95009	0.82716	0.82310	0.99914	1.0340
305.85	2045.6	0.94930	0.80337	0.79738	0.99861	1.0310
306.49	2085.1	0.94922	0.80088	0.79539	0.99872	1.0333
314.64	2548.6	0.94817	0.77591	0.76946	0.99834	1.0353
315.87	2625.9	0.94798	0.77196	0.76547	0.99831	1.0359
325.84	3320.0	0.94598	0.73912	0.73234	0.99800	1.0420
298.44	1008.1	0.69141	0.89471	0.81989	0.98417	0.93042
303.85	1135.2	0.69036	0.88772	0.80355	0.98109	0.92437
308.36	1245.5	0.68955	0.88223	0.78945	0.97804	0.91818
312.26	1348.8	0.68873	0.87732	0.77714	0.97517	0.91602
316.44	1463.4	0.68786	0.87228	0.76361	0.97166	0.91344
321.03	1601.7	0.68682	0.86639	0.74848	0.96740	0.91467
289.10	813.30	0.69298	0.90657	0.84675	0.98856	0.94700
291.50	858.67	0.69259	0.90376	0.84002	0.98750	0.94025
295.65	945.90	0.69192	0.89837	0.82809	0.98558	0.9338
285.97	1180.7	0.89609	0.86065	0.85534	0.99901	1.0822
289.33	1295.5	0.89580	0.85237	0.84611	0.99879	1.0796
299.28	1675.8	0.89484	0.82766	0.81739	0.99779	1.0695
305.38	1958.0	0.89411	0.81082	0.79883	0.99724	1.0698
313.64	2364.4	0.89306	0.78988	0.77270	0.99564	1.0620
321.49	2820.4	0.89161	0.76840	0.74695	0.99399	1.0617
325.30	3053.9	0.89062	0.75862	0.73416	0.99280	1.0607

Table B.3. R-410A/POE component activity coefficient data (assuming 50/50 blend—thermophysical property data for this blend were obtained by Martz et al., 1996a).

Temp (°C)	$x_{liq,R-32}$	γ_{R-32}	$x_{liq,R-125}$	γ_{R-125}
-44.26	0.69386	1.15661	0.30071	1.37890
-29.00	0.69386	1.06222	0.30071	1.27446
-15.66	0.69385	1.02403	0.30071	1.23117
-6.03	0.69384	1.00669	0.30070	1.20968
-6.01	0.69384	1.00581	0.30070	1.20870
15.36	0.69383	0.99203	0.30070	1.18209
24.98	0.69383	0.98469	0.30070	1.16606
-46.08	0.69380	1.17295	0.30069	1.39694
-12.96	0.69379	1.03942	0.30068	1.24843
5.87	0.69378	0.99748	0.30068	1.19436
6.33	0.69377	1.01267	0.30068	1.21065
-4.24	0.67554	1.05395	0.29277	1.26429
16.38	0.67516	1.02215	0.29261	1.21672
-21.29	0.65179	1.06512	0.28248	1.28178
3.69	0.65141	1.04112	0.28231	1.24988
13.38	0.65116	1.04453	0.28221	1.24773
24.94	0.65079	1.04195	0.28205	1.23525
-27.51	0.58980	1.12709	0.25561	1.35695
-17.59	0.58971	1.10984	0.25557	1.33906
-7.31	0.58961	1.10654	0.25553	1.33515
1.56	0.58952	1.09628	0.25549	1.32158
11.71	0.58943	1.08667	0.25546	1.30670
21.91	0.58937	1.07802	0.25543	1.29139
-8.26	0.49075	1.00172	0.21269	1.22350
0.77	0.48985	0.98298	0.21230	1.20402
12.14	0.48847	0.96407	0.21170	1.18425
22.11	0.48709	0.95284	0.21110	1.17275
33.49	0.48527	0.94556	0.21031	1.16576
45.25	0.48317	0.93902	0.20940	1.16028

APPENDIX C - UNIQUAC SIZE AND STRUCTURE PARAMETERS

The UNIQUAC model requires the use of structural data for each constituent in the mixture. These structural data are represented by a size parameter, r, and a surface area parameter, q (Tassios, 1993). These parameters, listed in Table C.1, have been determined for different chemical groups by Bondi (1968) and Beaton and Hewitt (1989).

Table C.1. Size and surface area parameters for common chemical groups.

Chemical Group	size (r_i)	surface area (q_i)
C	0.2195	0
CH	0.4469	0.228
CH_2	0.6744	0.540
CH_3	0.9011	0.848
F	0.3771	0.440
CF_3	1.4061	1.380
CH_2COO	1.6764	1.420
Cl	0.7660	0.720
CHCl	1.2380	0.952
CCl_2	1.8016	1.448

Each constituent used in this work, R-32, R-125, and POE, has a molecular formula that is made up of the chemical groups listed in Table C.1. To find the parameters for each constituent, the following relations are used:

$$r = \sum_i n_i r_i \qquad \text{(C1)}$$

$$q = \sum_i n_i q_i \qquad \text{(C2)}$$

In these equations, r and q are the constituent parameters, r_i and q_i are the parameters for the different chemical groups, and n_i is the number of each chemical group in the constituent.

For example, the molecular formula for R-32 is CH_2F_2, and it is made up of the chemical groups of CH_2 and F. Using Eq. C1, the size parameter, r, for R-32 is 1.43. The values of both parameters for each constituent are given in Table C2. The molecular formula and parameter values for the POE were estimated by Martz (1994).

Table C.2. Size and surface area parameter values for constituents studied in this work.

Constituent	Molecular Formula	r	q
R-32	CH_2F_2	1.43	1.42
R-125	C_2HF_5	2.61	2.49
POE	estimated	29.40	24.36

APPENDIX D - COMPUTER PROGRAMS AND OUTPUTS

D.1 Computer Programs

Program "kfind2" is used to find the parameter M in the Peng-Robinson equation of state. The parameter M is used in Eq. 6d. The original version of this program was written by Martz (1994).

```
            Program kfind2
**************************************
*       program to calculate the parameter in the
Peng-Robinson EoS
**************************************
      double precision Tc, Pc, MW, R, kmin,
Tcel(100),Pdata(100)
          double precision k, error, errort, avgt,
avgtmin,tdata
          double precision pcalc
          integer n,i,j
**************************************
*         set the input/output files
**************************************
          open(2,file='r32data', status='old')
          open(3,file='r32check2',status='unknown'
)
**************************************
*         Declare the constants for the particular
refrigerant
**************************************
          Tc=78.41+273.15d0
          Pc=5857.90d0
          MW=52.02d0
          n=0
**************************************
*         Initialize variables
**************************************
          R=8.31441/MW
          kmin=9999.0d0
          avgtmin=9999.0d0
**************************************
*         Read in ASHRAE data for T (Celsius) and P
(kPa)
**************************************
          do 5 i=1, 100
          read(2,*,end=5)Tcel(i),Pdata(i)
          n=n+1
 5        continue
**************************************
*         Use this outer "do - loop to vary the Peng-
Robinson parameter "k"
**************************************
          do 10 k=0.772805d0, 0.772805d0,.001
          write(*,*)"k=",k
          write(3,*)"k=",k
          error=0.0d0
          errort=0.0d0
**************************************
*It is necessary to nest another do-loop to calculate
the saturation pressure
*         at each temperature for the given "k." The
saturation pressure will be
*         calculated using the subroutine Psat
**************************************
          do 15 j=1,n
          Tdata=0.0d0
          Tdata=Tcel(j)+273.15d0
          Pcalc=0.0d0
          call Psatc(Tdata,Pcalc,Tc,Pc,k)
           write(3,*)Tdata,Pdata(j),Pcalc
          error=Pdata(j)-Pcalc
          errort=error**2 + errort
 15       continue
**************************************
*         To minimize the squared error, the
parameter "errort"
*          should be averaged by the
*         number of data points "n" and compared to
previous values.
**************************************
          avgt=errort/float(n)
          write(3,*)"avgt=",avgt
       if (avgt.lt.avgtmin)kmin=k
          if  (avgt.lt.avgtmin)avgtmin=avgt
 10       continue
          write(*,*)"kmin=",kmin
          write(*,*)"avgtmin=",avgtmin
          write(3,*)"kmin=",kmin
          write(3,*)"avgtmin=",avgtmin
          stop
          end

**************************************
**
*   subroutine to calculate saturation pressure
**************************************
**
          Subroutine Psatc(Tk, Psat1,Tc,Pc,rk)
          implicit double precision (a-h,l, o-z)
```

```
          Trr = Tk/Tc
          P = Pc*0.75*Trr
           diff = 0.0d0
           dp = 1.0d0
           itermmax = 25000
****************************************
**
*          interaction loop to find Psat
****************************************
**
         do 200 iter=1,itermmax
****************************************
*
*  determine the constants of the PR EOS
****************************************
*
          alpha=(1.0+rk*(1.0-Trr**0.5))**2.0
          Prr = P/Pc
          A = 0.45724*alpha*Prr/(Trr*Trr)
          B = 0.07780*Prr/Trr
*write(*,*)"A,B=",A,B
          a1 = -1.0*(1.0-B)
          a2 = A-3.0*B*B-2.0*B
          a3 = -1.0*(A*B-B*B-B*B*B)
****************************************
*  determine the roots of the PR EOS
****************************************
          Q = (a1*a1-3.0*a2)/9.0
          RR =(2.0*a1*a1*a1-
9.0*a1*a2+27.0*a3)/54.0
*write(*,*)"q,RR=",Q,RR
          if (q.lt.0)then
          P = 0.5*P
          go to 200
          endif
          if(abs(RR/((Q*Q*Q)**.5)).gt.1.0d0)then
          phi=0.0d0
          else
          phi = dacos(RR/((Q*Q*Q)**.5))
          endif
          x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
          x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
          x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0
************************************
*  determine if vapor fugacity equals the liquid
fugacity
************************************
     Zv = max(x1,x2,x3)
     Zl = min(x1,x2,x3)
************************************
*      These next few lines are used to eliminate
system errors, where
*      the computer cannot evaluate the natural
logarithm of negative
*     numbers.
************************************
     ch1=Zv-B
     ch2=Zl-B
      if((Zv-B).lt.0.0d0)then
          P=0.5*P
*write(*,*)"p is halved because vapor fugacity "
          go to 200
      endif
       if((Zl-B).lt.0.0d0)then
          P=0.5*P
 *write(*,*)"p is halved because liquid fugacity"
          go to 200
      endif
************************************

     lnfv1= Zv-1.0-dlog(Zv-B)
     lnfv=lnfv1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zv+2.414*B)/(Zv-
0.414*B))
     lnfl1=Zl-1.0-dlog(Zl-B)
     lnfl=lnfl1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zl+2.414*B)/(Zl-
0.414*B))
    diffold = diff
    diff = lnfv-lnfl
************************************
*  determine the next guess
************************************
     if ((diffold*diff).lt.0) dp = 0.1*dp
    if (diff.lt.0) P = P+dp
    if (diff.gt.0) P = P-dp
************************************
*  check for convergence
************************************
    adiff = abs(diff)
     if (adiff.lt.abs(0.0001*lnfv)) goto 28
200  continue
28   psat1 = p
   return
   end
```

Program "activity" is used to calculate the activity coefficient for the mixtures studied in this work.

```
          Program activity
****************************************
************
*          This program calculates the activity
coefficients given the
*          Temperature, Pressure, Liquid Density, and
MOLE Fraction.
****************************************
************
****************************************
************
          implicit none
          double precision
Vliq,Vtot,Vvap,rhov,rhol(100),T(100)
          double precision
P(100),mrefl,mrefv,moil,MWoil,MWref
          double precision
Zvap,Tcrit,Pcrit,Runiv,Rref,mreft
```

```
      double precision xrmass,
 xref(100),k,dval,ter
      double precision
 Pe(100),fcm(100),fcr(100),act(100)
      double precision
 kval,props(100,4),pld,ps,actcf(100)
      integer  j,np,i,ndp,ii,jj,m,dcol,kk
*************************************
**********
*       Vliq     liquid volume
*       Vtot     Total volume of apparatus
*       Vvap     vapor volume
*       rhov     vapor density
*       rhol     liquid density (measured)
*       T        temperature (measured)
*       P        pressure (measured)
*       mrefl    mass of refrigerant in the liquid
phase
*       mrefv    mass of refrigerant in the vapor
phase
*       moil     mass of oil
*       mreft    mass of refrigerant (in both
phases)
*       MWoil    molecular weight of the oil
*       MWref    molecular weight of the
refrigerant
*       np       number of points
*       Tcrit    critical temperature of refirgerant
*       Pcrit    critical pressure of refrigerant
*       Runiv    Universal gas constant
*       Rref     specific gas constant for
refrigerant
*       Zvap     compressibility factor for
refrigerant
*       k        Peng-Robinson parameter
*       xrmass   corrected mass fraction of
refrigerant in liquid phase
*       xref     corrected mole fraction of
refrigerant in liquid phase
*       act      activity
*       actcf    activity coefficient
*       fcr      fugacity coefficient of the
saturated refrigerant
*       fcm      fugacity coefficient of the
mixture
*       Pe       Poynting effect
*************************************
**********
*************************************
**********
        Tcrit=78.41d0+273.15d0
        Pcrit=5857.9d0
        Runiv=8.31441d0
        MWoil=700
        MWref=52.02d0
        Rref=Runiv/MWref
        k=0.772805d0
*************************************
*********
*************************************
*********
*       This section reads in the temperature (C),
pressure (kPa),
*       liquid density (kg/m^3), and liquid mole
fraction of ref.
*        for each concentration tested
*************************************
*******
        open (11, file="r32rl68h", status="old")
        np = 0
        do 5 j=1,1000
     read(11,*,end=5)T(j),P(j),rhol(j),xref(j)
    T(j) = T(j) + 273.15d0
    np = np + 1
5        continue
         close(11)
*************************************
*******
*************************************
*******
*       This section reads in the pure ref data: T, P,
liqdens, vap dens
*************************************
*******
        open (22, file="r32data", status="old")
        ndp= 0
        do 10 jj=1,1000
     read(22,*,end=10) (props(jj,i),i=1,4)
     props(jj,1) = props(jj,1) + 273.15
    ndp = ndp + 1
10  continue
   close(22)
*************************************
******
*************************************
*******
*       This next section calculates the saturation
pressure,
*       saturation fugacity coefficient, mixture
fugacity, activity
*       coefficient, activity, and Poynting effect.
*       Several Subroutines are called to perform
these calculations.
*************************************
******

        do 20 i=1,np
*************************************
*****
*       This next section linearly interpolates the
pure refrigerant
*       liquid density at mixture temperature from
the pure refrigerant
*       data file.
*************************************
****
    kval=T(i)
        m=1
   dcol = 3
    do 716 ii=1,ndp
      if(props(ii,m).lt.kval)then
        goto 716
    else
         ter = (kval-props(ii-1,m))/(props(ii,m)-
props(ii-1,m))
        goto 70
    endif
```

```
 716  continue
 70   dval = ter*(props(ii,dcol)-props(ii-
1,dcol))+props(ii-1,dcol)
     pld=dval
      write(*,*)"the iteration for liquid ref. density
gives pld=",pld
*****************************************
**************
*          The following section calculates the
saturation Pressure
*          based on the mixture temperature using
subroutine Psat
*****************************************
**************
          call Psatc(T(i), Ps, Tcrit, Pcrit, k)
*****************************************
**************
*****************************************
**************
*          The fugacity coefficient for both the
saturation pressure
*          at mixture temperature (fcr) and the actual
mixture pressure
*          (fcm) is calculated now.
*****************************************
**************
          call fuga(T(i),Ps,fcr(i),Tcrit,Pcrit,k)

          call fuga(T(i),P(i),fcm(i),Tcrit,Pcrit,k)
*****************************************
**************
*****************************************
**************
*          This section calculates the Poynting effect
using the
*          above parameters
*****************************************
************
          Pe(i)=exp((P(i)-Ps)/(pld*Rref*T(i)))

*****************************************
************
*****************************************
************
*          Using all of the above parameters, the
activity
*          coefficient can be found
*****************************************
***********
          actcf(i)=P(i)*fcm(i)/(xref(i)*Ps*fcr(i)*Pe(i
))
          act(i)=actcf(i)*xref(i)
*****************************************
***********
*          The following statements are used to
format and provide
*          the output of the calculated parameters
*****************************************
***********

 15     format(6(1xe11.5))

          write(*,15)T(i),fcm(i),fcr(i),Pe(i),actcf(i),
act(i)

 20       continue
*****************************************
*********
*          this section just repeats the output in a
formatted manner
*****************************************
*********
          do 25 kk=1,np

write(*,15)T(kk),fcm(kk),fcr(kk),Pe(kk),actcf(kk),a
ct(kk)
 25       continue
          stop
          end

*****************************************
************
*****************************************
************
*          This subroutine calculates the fugacity
based on temperature
*          and pressure using the Peng-Robinson
EoS.
*****************************************
***********
     Subroutine Fuga(T, Pm, FC,Tc,Pc,rk)
     implicit double precision (a-h, o-z)
    Prr = Pm/Pc
    Trr = T/Tc
     alpha =(1+rk*(1-Trr**0.5))**2
     A = 0.45724*alpha*Prr/(Trr*Trr)
     B = 0.07780*Prr/Trr
    a1 = -1*(1-B)
    a2 = A-3*B*B-2*B
    a3 = -1*(A*B-B*B-B*B*B)
    QQ = (a1*a1-3*a2)/9
     RR =(2*a1*a1*a1-9*a1*a2+27*a3)/54
    if ((RR*RR).gt.(QQ*QQ*QQ)) then
      if(RR.gt.0) sign = 1.0
      if(RR.lt.0) sign = -1.0
      AA = -1*sign*(abs(RR)+(RR*RR-
QQ*QQ*QQ)**(1/3))
     BB = QQ/AA
     if(aa.eq.0) bb = 0
     Zv = (AA+BB)-a1/3
     goto 6
    endif
    phi = acos(RR/((QQ*QQ*QQ)**.5))
     x1 = -2*(QQ**0.5)*cos(phi/3)-a1/3
     x2 = -2*(QQ**0.5)*cos((phi+2*3.1415926)/3)-
a1/3
     x3 = -2*(QQ**0.5)*cos((phi-2*3.1415926)/3)-
a1/3
    Zv = max(x1,x2,x3)
    ppp = Zv-b
    Zl = min(x1,x2,x3)
 6   rlnfv=Zv-1.0-log(ppp)
    rlnfv=rlnfv-
(A/(2.*(2.**0.5)*B))*log((Zv+2.414*B)/(Zv-
0.414*B))
    FC = EXP(rlnfv)
```

```
      return
      end
*************************************
*****************
*************************************
**
*   subroutine to calculate saturation pressure
      Subroutine Psatc(T, Psat,Tc,Pc,rk)
      implicit double precision (a-h,l, o-z)
*
*   interaction loop to find Psat
     Trr = T/Tc
     P = Pc*0.75*Trr
     diff = 0
     dp = 1
      itenmax = 25000
       do 200 itern=1,itenmax
*
*   determine the constants of the PR EOS
       alpha=(1.0+rk*(1.0-Trr**0.5))**2.0
     Prr = P/Pc
      A = 0.45724*alpha*Prr/(Trr*Trr)
      B = 0.07780*Prr/Trr
      a1 = -1.0*(1.0-B)
      a2 = A-3.0*B*B-2.0*B
      a3 = -1.0*(A*B-B*B-B*B*B)
*
*   determine the roots of the PR EOS
      Q = (a1*a1-3.0*a2)/9.0
       RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
     if (q.lt.0)then
      P = 0.5*P
      go to 200
     endif
     phi = dacos(RR/((Q*Q*Q)**.5))
       x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
     x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
      x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0
*   determine if vapor fugacity equals the liquid
fugacity
     Zv = max(x1,x2,x3)
     Zl = min(x1,x2,x3)
     lnfv1= Zv-1.0-dlog(Zv-B)
     lnfv=lnfv1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zv+2.414*B)/(Zv-
0.414*B))
     lnfl1=Zl-1.0-dlog(Zl-B)
     lnfl=lnfl1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zl+2.414*B)/(Zl-
0.414*B))
    diffold = diff
    diff = lnfv-lnfl
*
*   determine the next guess
     if ((diffold*diff).lt.0) dp = 0.1*dp
    if (diff.lt.0) P = P+dp
    if (diff.gt.0) P = P-dp
*
*   check for convergence
    adiff = abs(diff)
     if (adiff.lt.abs(0.0001*lnfv)) goto 28
 200  continue
 28   psat = p
       write(*,*)"psat=",P,"subroutine Psatc complete
in",itern
     return
     end
```

Program "model" finds the interaction parameters, $\Delta\lambda_1$ and $\Delta\lambda_2$, as shown in Eqs 12e and 12d. This program was written by Martz (1994), and is used with one refrigerant mixed with and oil.

```
      PROGRAM GENERAL
          implicit double precision (a-h, o-z)
      include 'variable.com'
*****************************************
****
*     VARIABLES
*     Tc = critical temperature
*     Pc = critical pressure
*     rmw = refrigerant molar weight
*     rmwo = refrigerant molar weight
*     rk = Peng-Robinson EOS parameter
*     la1 = interaction parameter
*     la2 = interaction parameter
*     x1, xref(i) = refrigerant mole fraction
*     T, Tem(i) = mixture temperature
*     Pres, Pr(i) = mixture pressure
*     Psat, ps(i) = saturation pressure at T
*     ar, art = activity coefficients: measured,
calculated
*     p1, pld(i) = pure refrigerant liquid density at T
*     p2 = oil density at T
*     oyi, oys = coefficients of oil density relation
*     props(i,j) = pure refrigerant properties from
ASHRAE
*     r1, r2, q1, q2 = UNIQUAC structure and size
parameters
*     std, stda = percent error for pressure, activity
*     fcr, fcm = fugacities of pure refrigerant, mixture
*     model variables as defined in chapter 6
*     Peng-Robinson EOS variables as defined in
chapter 5

*****************************************
****
*     define model constants
     rp = 0.0
     q = 1.0
```

```
      rNRTL =0.0
       alpha12 = 1.0
      Tak = 0.0
      Wac = 0.0
      UNI = 0.0
*****************************************
****
*      define initial la1 values
      start = 20000
      stop = -20000
      delta = -5000
*      initial step value
      delti = 10
      rmn =9
*****************************************
****
*     R22/RL68H   1
*     R125/RL68H  2
*     R134a/PAG   3
*     R134a/Ester 4
*     R134a/RL68H 5
*     R12/nap     6
*     R12/par     7
*          R22/alk     8
*          R32/RL68H   9
*     input mixture number
*****************************************
***
      if (rmn.eq.1) then
         print*, 'r22rl68h'
         open (22, file='RL68h', status='old')
         open (44, file='r22RL68h.out', status = 'old')
         open (66, file='r22data', status = 'old')
         open (88, file='r22crit', status='old')
      endif
      if (rmn.eq.2) then
         print*, 'r125rl68h'
         open (22, file='RL68h', status='old')
         open (44, file='r125RL68h.out', status =
'old')
         open (66, file='r125data', status = 'old')
         open (88, file='r125crit', status='old')
      endif
      if (rmn.eq.3) then
         print*, 'r134apag'
         open (22, file='PAG', status='old')
         open (44, file='r134aPAG.out', status = 'old')
         open (66, file='r134adata', status = 'old')
         open (88, file='r134acrit', status='old')
      endif
      if (rmn.eq.4) then
         print*, 'r134aester'
         open (22, file='ester', status='old')
         open (44, file='r134aester.out', status = 'old')
         open (66, file='r134adata', status = 'old')
         open (88, file='r134acrit', status='old')
      endif
      if (rmn.eq.5) then
         print*, 'r134aRL68h'
         open (22, file='RL68h', status='old')
         open (44, file='r134aRL68h.out', status =
'old')
         open (66, file='r134adata', status = 'old')
         open (88, file='r134acrit', status='old')
      endif
      if (rmn.eq.6) then
         print*, 'r12nap3'
         open (22, file='nap', status='old')
         open (44, file='r12nap3.out', status = 'old')
         open (66, file='r12data', status = 'old')
         open (88, file='r12crit', status='old')
      endif
      if (rmn.eq.7) then
         print*, 'r12par'
         open (22, file='par', status='old')
         open (44, file='r12par.out', status = 'old')
         open (66, file='r12data', status = 'old')
         open (88, file='r12crit', status='old')
      endif
         if (rmn.eq.8) then
            print*, 'r22alk'
            open (22, file='alk',status='old')
            open (44, file='glovadata',status='old')
            open (66, file='r22data',status='old')
            open (88, file='r22crit',status='old')
         endif
         if (rmn.eq.9) then
            print*, 'r32rl68h'
            open (22, file='rl68h', status='old')
            open (44, file='r32rl68h.out', status =
'old')
            open (66, file='r32data',status='old')
            open (88, file='r32crit',status='old')
         endif
*****************************************
**
*     open mixture data file
     ndp = 0
     do 43 j=1, 1000
        read(44,*,end=43) xref(j),aref(j),Tem(j),Pr(j)
       ndp = ndp + 1
           Tem(j)=Tem(j)+273.15
 43    continue
     close (44)
*****************************************
**
*     open ASHRAE data file
     np = 0
     do 11 j=1, 1000
        read(66,*,end=11) (props(j,i),i=1,4)
        props(j,1) = props(j,1) +273.15
       np = np + 1
 11    continue
     close (66)
*****************************************
**
*      open refrigerant data file
     read(88,*) Tc, Pc, rmw, rk, r1, q1
     close(88)
     Tc = Tc + 273.15
*****************************************
***
*      open oil data file
     read(22,*) rmwo, oyi, oys, r2, q2
     close (22)
*****************************************
**
```

```
*     linearly interpolate to find ref.liq.dens. and
pressure
*     at measured temperature

      do 28 ii = 1, ndp
      Tval = Tem(ii)
           iPcol = 2
      idcol = 3
      do 716 i=1, np
        if (props(i,1) .lt. Tval) then
          goto 716
        else
            ter = (Tval-props(i-1,1))/(props(i,1)-
     &              props(i-1,1))
          goto 70
        endif
 716    continue
 70     dval = ter*(props(i,idcol)-props(i-
1,idcol))+props(i-1,idcol)
           Pval = ter*(props(i,iPcol)-props(i-
1,iPcol))+props(i-1,iPcol)
       pld(ii) = dval
       ps(ii) = Pval
            print*,pld(ii)
 28    continue
*****************************************
*******
*       define other constants
      il = 0
      stdmin = 99999
      R=8.31441
      do 57 rla1init = start, stop, delta
      rla1 = rla1init
      rla2 = 0
      if (il.eq.1) then
        rla1 = rla1f
        rla2 = rla2f
      endif
      rla2min = rla2
      rla1min = rla1
      rlamax = 20000
      errortmin = 999999999
      iter = 100
      delt = delti
      change = 20*delt

*****************************************
**
*     Determine rla2
*****************************************
**
 33     do 9 ipp=1,iter,1
       trial = (rla2min) - change
       htrial = (rla2min) + change
         do 14 rkk = trial, htrial, delt
           errort = 0.0
           errora = 0.0
           do 92 ib = 1, ndp
             ar = aref(ib)
             x1 = xref(ib)
             Pres = pr(ib)
             psat = ps(ib)
            T = Tem(ib)
            rla2 = rkk
             p1 = pld(ib)
          Call Model(pres,p1,p2,rla1, rla2,T,
     &       x1, ar, zero,zeroa,psat,delt,pcalc,art)
         errort = errort+ zero*zero/(pres*pres)
        errora = errora+ (zeroa)*(zeroa)/(ar*ar)
 92        continue
          if((errort).lt.(errortmin)) then
           rla2min = rla2
           errortmin = errort
         endif
 14     continue
       rla2 = rla2min
        if (rla1min.ge.rlamax) then
          rla1min = rlamax
          rla1min = rlamax
          goto 88
       endif
        if (rla1min.le.(-1*rlamax)) then
          rla1min = -1*rlamax
          rla1min = -1*rlamax
          goto 88
       endif
*****************************************
**
*      Determine la1
*****************************************
**
       trial = (rla1min)-change
       htrial = (rla1min)+change
       do 930 rkk = trial, htrial, delt
          errort = 0.0
          errora = 0.0
          do 94 ib = 1, ndp
            ar = aref(ib)
            x1 = xref(ib)
           T = Tem(ib)
            pres = pr(ib)
            psat = ps(ib)
            p1 = pld(ib)
            rla1 = rkk
        Call Model(pres,p1,p2, rla1,rla2,T,
     &     x1, ar, zero,zeroa,psat,delt,pcalc,art)
       errort = errort+ zero*zero/(pres*pres)
       errora = errora+ zeroa*zeroa/(ar*ar)
 94          continue
            if(abs(errort).lt.abs(errortmin)) then
             rla1min = rla1
          endif
 930       continue
        rla1 = rla1min
*****************************************
***********
*       check for convergence
          if(rla1mino.eq.rla1min.and.ipp.gt.2) goto
88
        rla1mino = rla1min
 9     continue
*****************************************
***********
*     change delt for convergence
 88          if (delt.lt.(1)) goto 115
           delt = delt/10
           change = 20*delt
           goto 33
```

```
*****************************************
***********
 115     rla1 = rla1min
         rla2 = rla2min
*****************************************
***********
        std = 2*(errort/float(ndp-1))**0.5
        stda = 2*(errora/float(ndp-1))**0.5
*       print statistics
        print*,  'la1init =', rla1init
        print*, 'la1 =', rla1, 'la2 =', rla2
        print*, 'error P = ', std, 'error A =', stda
        if (std.lt.stdmin) then
         stdmin  = std
         stdamin = stda
          rla2initf = rla2init
         rla1f = rla1
         rla2f = rla2
       endif
*****************************************
***********
 57     continue
         rla1 = rla1f
         rla2 = rla2f
       do 41 ib=1,ndp
              ar = aref(ib)
              x1 = xref(ib)
             T = Tem(ib)
              pres = pr(ib)
              psat = ps(ib)
              p1 = pld(ib)
         Call Model(pres,p1,p2,rla1,rla2,T,
     &   x1, ar, zero,zeroa,psat,delt,pcalc,art)
 513
format(1x,f6.2,3x,f7.2,3x,f7.2,3x,f7.2,3x,f7.5,3x,
f7.5,3x,f7.5)
         write(*,513) t,Pcalc, pres, art, ar
 41     continue
        print*,  'la1init =', rla1initf
        print*, 'la1 =', rla1, 'la2 =', rla2
        print*, 'error P = ', stdmin, 'error A =', stdamin
 7       format(1x, 7(f5.2, 3x))
       write(*,7) rp, q, rNRTL,alpha12, TaK, WaC,
UNI
       print*, 'number of data points = ', ndp , 'mixture
# =',rmn
      end

*****************************************
********************
*     Subroutine to calculate activity coefficients
       Subroutine Model(pres,p1,p2,rla1,rla2,T,
     & x1, ar, zero,zeroa,psat,delt,pcalc,art)
       implicit double precision (a-h, o-z)
      include 'variable.com'
*****************************************
***********
*       calculate variables
            x2 = 1.0 -x1
             p2 = oyi+oys*(T-273.15)
            v1 = (1/p1)*(rmw)
            v2 = (1/p2)*(rmwo)
            if (rNRTL.eq.1) then
               rho12 = 1.0
               rho21 = 1.0
            else
               rho12 = v1/v2
               rho21 = v2/v1
            endif
             Tao12 = rla1/(R*T)
             Tao21 = rla2/(R*T)
              G12 = (rho12)*exp(-alpha12*Tao12)
              G21 = (rho21)*exp(-alpha12*Tao21)
*************************************
***********
*       Calculate Wang and Chao coefficients

            if (WaC.eq.1) then
               x11 = x1/(x1+x2*exp(-Tao21))
               x22 = x2/(x2+x1*exp(-Tao12))
               x12 = x22*(x1/x2)*exp(-Tao12)
               x21 = x11*(x2/x1)*exp(-Tao21)
              z = 6.0
            endif
*****************************************
***********
* calculate Wilson, Heil, NRTL, T&K, and WaC
log(act coef)

            if((x1+x2*g12).gt.0.01.or.q.eq.0) then
         rlogar=q*(x2*G21/(x1+x2*G21) -
x2*G12/(x2+x1*G12)
     &       - log(x1+x2*G21))
     &       +
rp*x2*x2*(Tao12*G12/((x2+x1*G12)**2.0)
     &         + Tao21*G21*G21/((x1+x2*G21)**2.0))
     &      + Tak*(-x2*rho21/(x1+x2*rho21) +
x2*rho12/(x2+x1*rho12)
     &      +  log(x1+x2*rho21))
     &   +
WaC*(z/(R*T*2.0))*(x21*x21*rla2+x2*x22*x12
*rla1/x1)
*****************************************
***********
* calculate UNIQUAC log(act coef)

        if (UNI.eq.1) then
         thet1 = q1*x1/(q1*x1+q2*x2)
         thet2 = q2*x2/(q2*x2+q1*x1)
         ph1 = r1*x1/(r1*x1+r2*x2)
         ph2 = r2*x2/(r1*x1+r2*x2)
         rL1 = (10./2.)*(r1-q1)-(r1-1.)
         rL2 = (10./2.)*(r2-q2)-(r2-1.)
         tao12 = exp(-1*tao12)
         tao21 = exp(-1*tao21)
         test = thet1+thet2*tao21
         if(test.le.0) then
            zero = 1000
           zeroa = 3
            goto 89
        endif

rlogar=log(ph1/x1)+(10./2.)*q1*log(thet1/ph1)
     &       +ph2*(rL1-(r1/r2)*rL2)
-q1*log(thet1+thet2*tao21)
     &        +thet2*q1*((Tao21/(thet1+thet2*Tao21))
     &        -(Tao12/(thet2+thet1*tao12)))
```

```
      endif
*****************************************
***********
*     calculate activity coefficient
      art = exp(rlogar)
*****************************************
***********
*     iterate to calculate pressure
      call Fuga(T, psat, Fcr)
      do 115 ihh=1,3
        call fuga(T, Pres, Fcm)
        Pe = exp((1/p1)*(Pm-psat)/(R*T))
        Pcalc = (art*x1*fcr*Psat*Pe)/fcm
115   continue
     endif
*****************************************
***********
*     calculate deviations
      zeroa = art - ar
      zero = pcalc - pres
89    Return
     End
*****************************************
******************
*      Routine to calculate fugacity
*****************************************
******************
      Subroutine fuga(T, Pm, FC)
      implicit double precision (a-h, o-z)
      include 'variable.com'
      Prr = Pm/Pc
      Trr = T/Tc
       alpha =(1+rk*(1-Trr**0.5))**2
       A = 0.45724*alpha*Prr/(Trr*Trr)
       B = 0.07780*Prr/Trr
      a1 = -1*(1-B)
      a2 = A-3*B*B-2*B
      a3 = -1*(A*B-B*B-B*B*B)
      QQ = (a1*a1-3*a2)/9
       RR =(2*a1*a1*a1-9*a1*a2+27*a3)/54
      if ((RR*RR).gt.(QQ*QQ*QQ)) then
        if(Rr.gt.0) sign = 1.0
        if(RR.lt.0) sign = -1.0
        AA= -1*sign*(abs(rR)+(RR*RR-
QQ*QQ*QQ)**(1/3))
       BB= QQ/AA
        if(aa.eq.0) bb = 0
       Zv = (AA+BB) - a1/3
        goto 3
      endif
      phi = acos(RR/((QQ*QQ*QQ)**.5))
       x1 = -2*(QQ**0.5)*COS(phi/3) - a1/3
      x2 =
-2*(QQ**0.5)*COS((phi+2*3.1415926)/3)  -  a1/3
       x3 = -2*(QQ**0.5)*COS((phi-
2*3.1415926)/3)  -   a1/3
      Zv = max(x1,x2,x3)
      ppp = Zv-b
       Zl = min(x1,x2,x3)
3     rlnfv=Zv -1.0  -log(ppp)
      rlnfv=rlnfv-
(A/(2.*(2.**0.5)*B))*log((Zv+2.414*B)/(Zv-
0.414*B))
     FC = EXP(rlnfv)
     Return
     End
*****************************************
***********
```

The program "biparams1.f" is used to find $\Delta\lambda_1$ and $\Delta\lambda_2$ for the binary mixture of R-32 and R-125. It differs from the above program in that it must use the Peng-Robinson equation of state for two components.

```
    PROGRAM GENERAL
          implicit double precision (a-h, o-z)
          double precision
aref(100),tem(100),props(100,10),props1(100,10)
          double precision
pr(100),ps(100),pld(100),pld2(100)
*****************************************
****
*      define model constants
     rp = 0.0
     q = 0.0
     rNRTL =0.0
      alpha12 = 0.0
     Tak = 0.0
     Wac = 0.0
     UNI = 1.0
*****************************************
****
*      define initial la1 values
     start = 20000
     stop = -20000
     delta = -5000
*      initial step value
     delti = 10
          open(22,file="r32data",status="old")
          open(44,file="r125data",status="old")
          open(66,file="r410aact",status="old")

*     open mixture data file
    ndp = 0
    do 43 j=1, 1000
        read(66,*,end=43) Tem(j),pr(j),aref(j)
       ndp = ndp + 1
43    continue
    close (66)
*****************************************
**
*     open ASHRAE data file
    np = 0
         np1=0
    do 11 j=1, 1000
```

```
          read(22,*,end=11) (props(j,i),i=1,4)
          props(j,1) = props(j,1) +273.15
         np = np + 1
 11    continue
          do 12 j=1, 1000
          read(44,*,end=12) (props1(j,i),i=1,4)
          props1(j,1) = props1(j,1) +273.15
         npl = npl + 1
 12    continue
      close (22)
          close (44)
*****************************************
**
*****************************************
**
*     linearly interpolate to find ref.liq.dens. and
pressure
*     at measured temperature

      do 28 ii = 1, ndp
      Tval = Tem(ii)
           iPcol = 2
      idcol = 3
      do 716 i=1, np
        if (props(i,1) .lt. Tval) then
          goto 716
       else
            ter = (Tval-props(i-1,1))/(props(i,1)-
   &              props(i-1,1))
          goto 70
       endif
 716     continue
 70      dval = ter*(props(i,idcol)-props(i-
1,idcol))+props(i-1,idcol)
           Pval = ter*(props(i,iPcol)-props(i-
1,iPcol))+props(i-1,iPcol)
      pld(ii) = dval
      ps(ii) = Pval
          do 816 i=1, npl
        if (props1(i,1) .lt. Tval) then
          goto 816
       else
            ter = (Tval-props1(i-1,1))/(props1(i,1)-
   &              props1(i-1,1))
          goto 80
       endif
 816     continue
 80      dval = ter*(props1(i,idcol)-props1(i-
1,idcol))+props1(i-1,idcol)
      pld2(ii) = dval
 28    continue
*****************************************
*******
*       define other constants
     il = 0
     stdmin = 99999
     R=8.31441
     do 57 rla1init = start, stop, delta
     rla1 = rla1init
     rla2 = 0
     if (il.eq.1) then
      rla1 = rla1f
      rla2 = rla2f
     endif
      rla2min = rla2
      rla1min = rla1
      rlamax = 20000
      errortmin = 999999999
      iter = 100
      delt = delti
      change = 20*delt

*****************************************
**
*     Determine rla2
*****************************************
**
 33     do 9 ipp=1,iter,1
      trial = (rla2min) - change
      htrial = (rla2min) + change
        do 14 rkk = trial, htrial, delt
          errort = 0.0
          errora = 0.0
          do 92 ib = 1, ndp
           ar = aref(ib)
            x1 = 0.698d0
           Pres = pr(ib)
           psat = ps(ib)
          T = Tem(ib)
          rla2 = rkk
          p1 = pld(ib)
                    p2= pld2(ib)
       Call Model(pres,p1,p2,rla1, rla2,T,
   &      x1, ar, zero,zeroa,psat,delt,pcalc,art,
   + rp,q,rnrtl,alpha12,tak,wac,uni)
        errort = errort+ zero*zero/(pres*pres)
       errora = errora+ (zeroa)*(zeroa)/(ar*ar)
 92       continue
         if((errort).lt.(errortmin)) then
          rla2min = rla2
          errortmin = errort
        endif
 14    continue
     rla2 = rla2min
      if (rla1min.ge.rlamax) then
       rla1min = rlamax
       rla1min = rlamax
       goto 88
     endif
      if (rla1min.le.(-1*rlamax)) then
       rla1min = -1*rlamax
       rla1min = -1*rlamax
       goto 88
     endif
*****************************************
**
*       Determine la1
*****************************************
**
      trial = (rla1min)-change
      htrial = (rla1min)+change
      do 930 rkk = trial, htrial, delt
          errort = 0.0
          errora = 0.0
          do 94 ib = 1, ndp
            ar = aref(ib)
             x1 = 0.698d0
            T = Tem(ib)
```

```
            pres = pr(ib)
            psat = ps(ib)
            p1 = pld(ib)
                  p2 = pld2(ib)
            rla1 = rkk
          Call Model(pres,p1,p2, rla1,rla2,T,
     &     x1, ar, zero,zeroa,psat,delt,pcalc,art,
     + rp,q,rnrtl,alpha12,tak,wac,uni)
        errort = errort+ zero*zero/(pres*pres)
        errora = errora+ zeroa*zeroa/(ar*ar)
 94         continue
            if(abs(errort).lt.abs(errortmin)) then
             rla1min = rla1
           endif
 930       continue
        rla1 = rla1min
*****************************************
***********
*       check for convergence
          if(rla1mino.eq.rla1min.and.ipp.gt.2) goto
88
        rla1mino = rla1min
 9     continue
*****************************************
***********
*     change delt for convergence
 88          if (delt.lt.(1)) goto 115
          delt = delt/10
          change = 20*delt
          goto 33
*****************************************
***********
 115     rla1 = rla1min
       rla2 = rla2min
*****************************************
***********
       std = 2*(errort/float(ndp-1))**0.5
       stda = 2*(errora/float(ndp-1))**0.5
*      print statistics
       print*,  'la1init =', rla1init
       print*, 'la1 =', rla1, 'la2 =', rla2
      print*, 'error P = ', std, 'error A =', stda
      if (std.lt.stdmin) then
       stdmin  = std
       stdamin = stda
       rla2initf = rla2init
       rla1f = rla1
       rla2f = rla2
     endif
*****************************************
***********
 57    continue
       rla1 = rla1f
       rla2 = rla2f
     do 41 ib=1,ndp
           ar = aref(ib)
           x1 = 0.698d0
          T = Tem(ib)
           pres = pr(ib)
           psat = ps(ib)
           p1 = pld(ib)
                  p2 = pld2(ib)
       Call Model(pres,p1,p2,rla1,rla2,T,
   &   x1, ar, zero,zeroa,psat,delt,pcalc,art,
     + rp,q,rnrtl,alpha12,tak,wac,uni)
 513
format(1x,f6.2,3x,f7.2,3x,f7.2,3x,f7.5,3x,
f7.5,3x,f7.5)
        write(*,513) t,Pcalc, pres, art, ar
 41    continue
       print*,  'la1init =', rla1initf
       print*, 'la1 =', rla1, 'la2 =', rla2
      print*, 'error P = ', stdmin, 'error A =', stdamin
 7     format(1x, 7(f5.2, 3x))
      write(*,7) rp, q, rNRTL,alpha12, TaK, WaC,
UNI
      print*, 'number of data points = ', ndp , 'mixture
# =',rmn
     end

*****************************************
*******************
*     Subroutine to calculate activity coefficients
      Subroutine Model(pres,p1,p2,rla1,rla2,T,
     & x1, ar, zero,zeroa,psat,delt,pcalc,art,
     + rp,q,rnrtl,alpha12,tak,wac,uni)
      implicit double precision (a-h, o-z)
          rmwi=52.02
          rmwj=120.03
          R=8.31441d0
*****************************************
***********
*     calculate variables
           x2 = 1.0 -x1
           v1 = (1/p1)*(rmwi)
           v2 = (1/p2)*(rmwj)
           if (rNRTL.eq.1) then
             rho12 = 1.0
             rho21 = 1.0
           else
             rho12 = v1/v2
             rho21 = v2/v1
           endif
           Tao12 = rla1/(R*T)
           Tao21 = rla2/(R*T)
            G12 = (rho12)*exp(-alpha12*Tao12)
            G21 = (rho21)*exp(-alpha12*Tao21)
*****************************************
***********
*     Calculate Wang and Chao coefficients

           if (WaC.eq.1) then
             x11 = x1/(x1+x2*exp(-Tao21))
             x22 = x2/(x2+x1*exp(-Tao12))
             x12 = x22*(x1/x2)*exp(-Tao12)
             x21 = x11*(x2/x1)*exp(-Tao21)
            z = 6.0
           endif
*****************************************
***********
* calculate Wilson, Heil, NRTL, T&K, and WaC
log(act coef)

           if((x1+x2*g12).gt.0.01.or.q.eq.0) then
      rlogar=q*(x2*G21/(x1+x2*G21) -
x2*G12/(x2+x1*G12)
   &       - log(x1+x2*G21))
```

```
     &      +
 rp*x2*x2*(Tao12*G12/((x2+x1*G12)**2.0)
     &          + Tao21*G21*G21/((x1+x2*G21)**2.0))
     &      + Tak*(-x2*rho21/(x1+x2*rho21) +
 x2*rho12/(x2+x1*rho12)
     &      +  log(x1+x2*rho21))
     &     +
 WaC*(z/(R*T*2.0))*(x21*x21*rla2+x2*x22*x12
 *rla1/x1)
 ****************************************
 ***********
 *  calculate UNIQUAC log(act coef)

           r1=1.43
           q1=1.42
           r2=2.61
           q2=2.49

        if (UNI.eq.1) then
          thet1 = q1*x1/(q1*x1+q2*x2)
          thet2 = q2*x2/(q2*x2+q1*x1)
          ph1 = r1*x1/(r1*x1+r2*x2)
          ph2 = r2*x2/(r1*x1+r2*x2)
          rL1 = (10./2.)*(r1-q1)-(r1-1.)
          rL2 = (10./2.)*(r2-q2)-(r2-1.)
          tao12 = exp(-1*tao12)
          tao21 = exp(-1*tao21)
          test = thet1+thet2*tao21
          if(test.le.0) then
              zero = 1000
              zeroa = 3
              goto 89
          endif

 rlogar=log(ph1/x1)+(10./2.)*q1*log(thet1/ph1)
     &       +ph2*(rL1-(r1/r2)*rL2)
 -q1*log(thet1+thet2*tao21)
     &       +thet2*q1*((Tao21/(thet1+thet2*Tao21))
     &       -(Tao12/(thet2+thet1*tao12)))

        endif
 ****************************************
 ***********
 *      calculate activity coefficient
        art = exp(rlogar)
 ****************************************
 ***********
 *      iterate to calculate pressure
        call Fuga(T, psat, Fcr)
        do 115 ihh=1,3
          call fugabin(T, Pres, Fcm)
          Pe = exp((1/p1)*(Pm-psat)/(R*T))
          Pcalc = (art*fcr*Psat*Pe)/fcm
 115    continue
      endif
 ****************************************
 ***********
 *      calculate deviations
      zeroa = art - ar
      zero = pcalc - pres
 89   Return
      End
 ****************************************
 ******************
 *       Routine to calculate fugacity
 ****************************************
 ******************
       Subroutine fuga(T, Pm, FC)
       implicit double precision (a-h, o-z)
           Tc=78.41d0+273.15d0
       Pc=5857.90d0
       rmw=52.02d0
       rk=0.772805d0
       R=8.31441d0
       Prr = Pm/Pc
       Trr = T/Tc
         alpha =(1+rk*(1-Trr**0.5))**2
         A = 0.45724*alpha*Prr/(Trr*Trr)
         B = 0.07780*Prr/Trr
         a1 = -1*(1-B)
         a2 = A-3*B*B-2*B
         a3 = -1*(A*B-B*B-B*B*B)
         QQ = (a1*a1-3*a2)/9
          RR =(2*a1*a1*a1-9*a1*a2+27*a3)/54
         if ((RR*RR).gt.(QQ*QQ*QQ)) then
           if(Rr.gt.0) sign = 1.0
           if(RR.lt.0) sign = -1.0
          AA= -1*sign*(abs(rR)+(RR*RR-
 QQ*QQ*QQ)**(1/3))
          BB= QQ/AA
           if(aa.eq.0) bb = 0
          Zv = (AA+BB) - a1/3
          goto 3
         endif
         phi = acos(RR/((QQ*QQ*QQ)**.5))
          x1 = -2*(QQ**0.5)*COS(phi/3) - a1/3
         x2 =
 -2*(QQ**0.5)*COS((phi+2*3.1415926)/3)  -  a1/3
          x3 = -2*(QQ**0.5)*COS((phi-
 2*3.1415926)/3)  -   a1/3
          Zv = max(x1,x2,x3)
          Zl = min(x1,x2,x3)
 3       rlnfv=Zv -1.0  -log(Zv-b)
         rlnfv=rlnfv-
 (A/(2.*(2.**0.5)*B))*log((Zv+2.414*B)/(Zv-
 0.414*B))
        FC = EXP(rlnfv)
        Return
        End
 ****************************************
 ***********
           Subroutine fugabin(Tk, P, Fcm)
         implicit double precision (a-h, o-z)
        double precision
 lnfv1i,lnfl1i,lnfvi,lnfli,lnfv1j
         double precision lnfvj,lnfl1j,lnflj
         Tci=78.41+273.15d0
         Tcj=66.04+273.15d0
         Pci=5857.90d0
         Pcj=3595.0d0
        Rmwi=52.02d0
        Rmwj=120.03d0
        rki=0.772805d0
        rkj=0.81485d0
        xi=0.698d0
        xj=0.302d0
        R=8.31441d0
          dij=0.0072d0
```

```
      Tri = Tk/Tci
      Trj = Tk/Tcj
           alphai=(1.0d0+rki*(1.0d0-
 Tri**0.5d0))**2.0d0
       alphaj=(1.0d0+rkj*(1.0d0-
 Trj**0.5d0))**2.0d0
        ai=(0.45724d0)*alphai*(R**2*Tci**2)/Pci
        aj=(0.45724d0)*alphaj*(R**2*Tcj**2)/Pcj
       bi=(0.07780d0)*R*Tci/Pci
       bj=(0.07780d0)*R*Tcj/Pcj
       bm=xi*bi+xj*bj
        aij=(1.0d0-dij)*ai**0.5d0*aj**0.5d0
        am=xi**2*ai + 2.0d0*xj*xi*aij + xj**2*aj
*****************************************
*   determine the constants of the PR EOS
*****************************************
*
      A = am*P/(R**2*Tk**2)
      B = bm*P/(R*Tk)
      a1 = -1.0*(1.0-B)
      a2 = A-3.0*B*B-2.0*B
      a3 = -1.0*(A*B-B*B-B*B*B)
*****************************************
*   determine the roots of the PR EOS
*****************************************
      Q = (a1*a1-3.0*a2)/9.0
       RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
       if(abs(RR/((Q*Q*Q)**.5)).gt.1.0d0)then
      phi=0.0d0
      else
      phi = dacos(RR/((Q*Q*Q)**.5))
      endif
        x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
      x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
       x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0
*************************************
*   determine if vapor fugacity equals the liquid
fugacity
*************************************
     Zv = max(x1,x2,x3)
     Zl = min(x1,x2,x3)
*************************************
*       calculate the fugacity coefficient for the vapor
phase of component i
*************************************
          lnfv1i=  bi/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xj*aij+2.0*xi*ai)/am - bi/bm
     lnfvi=lnfv1i-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zv+2.414*B)/(Zv
-0.414*B))
*************************************
*       calculate the fugacity coefficient for the vapor
phase of component j
*************************************
     lnfv1j= bj/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xi*aij+2.0*xj*aj)/am - bj/bm
     lnfvj=lnfv1j-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zv+2.414*B)/(Zv
-0.414*B))
*************************************
*************************************
```

```
      fcm=exp(lnfvi)
     return
     end
```

The program "dcalc.f" is used to calculate the mixing parameter ∂_{ij}, which is found in Eq. 7d.

```
            Program dcalc
*************************************
*           program to calculate the binary
*           interaction  parameter in the Peng-
Robinson EoS
*************************************
           implicit double precision (a-h, o-z)
           dimension Tcel(100),Pdata(100)
           integer  i,j,n
       include 'critprps.com'
*************************************
*           The common block "critprps" (critical
properties) contains:
*
           Tci,Tcj,Pci,Pcj,rmwi,rmwj,rki,rkj,xi,xj,R
,dij
*************************************
*           set the input/output files
*************************************
           open(2,file='r404adata', status='old')
           open(3,file='r404adcheck',status='unknow
n')
*************************************
*           Declare the constants for the particular
refrigerant
*           i=R-32
*           j=R-125
*           A 50/50 mass mixture translates to mole
fractions of
*           yi=.698,  yj=.302
*************************************
           Tci=78.41+273.15d0
           Tcj=66.04+273.15d0
           Pci=5857.90d0
           Pcj=3595.0d0
           Rmwi=52.02d0
           Rmwj=120.03d0
           rki=0.772805d0
           rkj=0.81485d0
           xi=0.698d0
           xj=0.302d0
           n=0
*************************************
*           Initialize variables
*************************************
```

```
      R=8.31441d0
      dmin=9999.0d0
      avgtmin=9999.0d0
      stdmin=9999.0d0
**********************************
*     Read in ASHRAE data for T (Celsius) and P
(kPa)
**********************************
      do 5 i=1, 100
      read(2,*,end=5)Tcel(i),Pdata(i)
      n=n+1
5     continue
**********************************
*     Use this outer "do - loop to vary the Peng-
Robinson
*     interaction parameter "d"
**********************************
      do 10 dij=0.007d0, 0.008d0, 0.0001d0
      write(*,*)"d=",dij
      write(3,*)"d=",dij
      error=0.0d0
      errort=0.0d0
**********************************
*It is necessary to nest another do-loop to calculate
the saturation pressure
*     at each temperature for the given "d." The
saturation pressure will be
*     calculated using the subroutine Psat
**********************************
      do 15 j=1,n
      Tdata=0.0d0
      Tdata=Tcel(j)+273.15d0
      Pcalc=0.0d0
      call Psatcmx(Tdata,Pcalc)
       write(3,*)Tdata,Pdata(j),Pcalc
      error=Pdata(j)-Pcalc
      errort=(error**2)/(Pdata(j)**2) + errort
15    continue
**********************************
*     To minimize the squared error, the
parameter "errort"
*     should be averaged by the
*     number of data points "n" and compared to
previous values.
**********************************
      avgt=errort/float(n)
      std=2*sqrt(errort/float(n-1))
      write(3,*)"avgt=",avgt,"std=",std
    if (avgt.lt.avgtmin)dmin=dij
      if (avgt.lt.avgtmin)stdmin=std
      if (avgt.lt.avgtmin)avgtmin=avgt
10    continue
      write(*,*)"dmin=",dmin
      write(*,*)"avgtmin=",avgtmin,"stdmin="
,stdmin
      write(3,*)"dmin=",dmin
      write(3,*)"avgtmin=",avgtmin,"std=",std
      stop
      end

**************************************
**
*  subroutine to calculate saturation pressure for
binary mixtures
*  "j" calculations verified against kcalc2.f and OK
(5/16/96)
*  "i"    "    "     "     "   " (5/20/96)
**************************************
**
      Subroutine Psatcmx(Tk, Psat1)
      implicit double precision (a-h, o-z)
      double precision
lnfv1i,lnfl1i,lnfvi,lnfli,lnfv1j
    double precision lnfvj,lnfl1j,lnflj
      include 'critprps.com'
      Tri = Tk/Tci
      Trj = Tk/Tcj
      alphai=(1.0d0+rki*(1.0d0-
Tri**0.5d0))**2.0d0
    alphaj=(1.0d0+rkj*(1.0d0-
Trj**0.5d0))**2.0d0
      ai=(0.45724d0)*alphai*(R**2*Tci**2)/P
ci
      aj=(0.45724d0)*alphaj*(R**2*Tcj**2)/P
cj
      bi=(0.07780d0)*R*Tci/Pci
      bj=(0.07780d0)*R*Tcj/Pcj
      bm=xi*bi+xj*bj
      aij=(1.0d0-dij)*ai**0.5d0*aj**0.5d0
      am=xi**2*ai + 2.0d0*xj*xi*aij +
xj**2*aj
**************************************
*     The constants to this point have been
checked and are correct 5/1/96
**************************************
      P
=(Pci+Pcj)/2.0d0*0.75d0*(Tri+Trj)/2.0d0
*   diffi = 0.0d0
*      diffj=0.0d0
      diff=0.0d0
     dp = 10.0d0
      itermmax = 25000
**************************************
**
*     interaction loop to find Psat
**************************************
**
      do 200 iterm=1,itermmax
**************************************
*
*  determine the constants of the PR EOS
**************************************
*
      A = am*P/(R**2*Tk**2)
      B = bm*P/(R*Tk)
      a1 = -1.0*(1.0-B)
      a2 = A-3.0*B*B-2.0*B
      a3 = -1.0*(A*B-B*B-B*B*B)
**************************************
*  determine the roots of the PR EOS
**************************************
      Q = (a1*a1-3.0*a2)/9.0
      RR =(2.0*a1*a1*a1-
9.0*a1*a2+27.0*a3)/54.0
      if (q.lt.0)then
      P = 0.5*P
      go to 200
      endif
```

```
            if(abs(RR/((Q*Q*Q)**.5)).gt.1.0d0)then
            phi=0.0d0
            else
            phi = dacos(RR/((Q*Q*Q)**.5))
            endif
            x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
            x2 =
 -2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
 a1/3.0
            x3 = -2.0*(Q**0.5)*cos((phi-
 2.0*3.1415926)/3.0)-a1/3.0
 ************************************
 *  determine if vapor fugacity equals the liquid
 fugacity
 ************************************
      Zv = max(x1,x2,x3)
      Zl = min(x1,x2,x3)
 ************************************
 *          These next few lines are used to eliminate
 system errors, where
 *          the computer cannot evaluate the natural
 logarithm of negative
 *          numbers.
 ************************************
            ch1=Zv-B
            ch2=Zl-B
            if((Zv-B).lt.0.0d0)then
                        P=0.5*P
 *write(*,*)"p is halved because vapor fugacity "
                        go to 200
            endif
            if((Zl-B).lt.0.0d0)then
                        P=0.5*P
 *write(*,*)"p is halved because liquid fugacity"
                        go to 200
            endif
 ************************************
 *          calculate the fugacity coefficient for the
 vapor phase of component i
 ************************************
      lnfv1i= bi/bm*(Zv-1.0)-dlog(Zv-B)
      u=(2.0*xj*aij+2.0*xi*ai)/am - bi/bm
     lnfvi=lnfv1i-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zv+2.414*B)/(Zv
-0.414*B))
 ************************************
 *          calculate the fugacity coefficient for the
 liquid phase of component j
 ************************************
      lnfl1i= bi/bm*(Zl-1.0)-dlog(Zl-B)
      u=(2.0*xj*aij+2.0*xi*ai)/am - bi/bm
     lnfli=lnfl1i-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zl+2.414*B)/(Zl-
0.414*B))
 ************************************
 *          calculate the fugacity coefficient for the
 vapor phase of component j
 ************************************
      lnfv1j= bj/bm*(Zv-1.0)-dlog(Zv-B)
      u=(2.0*xi*aij+2.0*xj*aj)/am - bj/bm
     lnfvj=lnfv1j-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zv+2.414*B)/(Zv
-0.414*B))
 ************************************
 *          calculate the fugacity coefficient for the
 liquid phase of component j
 ************************************
      lnfl1j= bj/bm*(Zl-1.0)-dlog(Zl-B)
      u=(2.0*xi*aij+2.0*xj*aj)/am - bj/bm
     lnflj=lnfl1j-
(A/(2.0*(2.0**0.5)*B))*u*dlog((Zl+2.414*B)/(Zl-
0.414*B))
 ************************************
 *          calculate the differences between liquid and
 vapor phase fugacities for
 *          each component
 ************************************
 *      diffoldi = diffi
 *       diffi =abs(lnfvi-lnfli)
 *      diffoldj = diffj
 *       diffj=abs(lnfvj-lnflj)
            diffold=diff
            diff=(lnfvi-lnfli)+(lnfvj-lnflj)
 *write(*,*)lnfli,lnfvi,lnflj,lnfvj
 ************************************
 *  determine the next guess
 ************************************
     if ((diffold*diff).lt.0) dp = 0.1*dp
     if (diff.lt.0.0d0)then
            P = P+dp
           endif
     if (diff.gt.0.0d0)then
            P = P-dp
           endif
 *write(*,*)"Pnew=",P
 *write(*,*)"diff",diff
 ************************************
 *  check for convergence
 ************************************
 *      adiffi = abs(diffi)
 *      adiffj = abs(diffj)
 *       cci=abs(0.0001*lnfvi)
 *       ccj=abs(0.0001*lnfvj)
 *       if (adiffi.lt.cci.and.adiffj.lt.ccj) goto 28
            adiff=abs(diff)
            if(adiff.lt.(0.0001*abs(lnfvi)))goto  28
 200  continue
 28   psat1 = p
    return
    end
```

The program "pvtbin1.f" uses local composition models to model the behavior of R-410a mixed with a polyol ester, assuming a 50/50 mass fraction of

R-410a in both the vapor and liquid phases.

```
*****************************************
******
      PROGRAM pvtbin1
      implicit double precision (a-h, o-z)
            double precision
 props(100,10),tk(100),pk(100),xki(100)
            double precision
 pdata(100),pcalc(100),xkj(100)
            double precision props1(100,10)
*****************************************
******
*           This program models a binary refrigerant
 blend (R-410a)
*           mixed with a polyol ester (RL68H)
*           This is a preliminary modelling scheme in
 that it assumes that
*           the 50/50 mass concentration of R-410a
 remains constant throughout
*           all experimental conditions.
*****************************************
******
*****************************************
******
*           The "subscripts" i, j, and k correspond to
 R-32, R-125,
*           and RL68H, respectively.
*****************************************
******
*   VARIABLES
*    Tci,Tcj = critical temperature
*    Pci,Pcj = critical pressure
*    rmwi,rmwj, = refrigerant molar weight
*    rmwo = oil molar weight
*    rki,rkj = Peng-Robinson EOS Parameter
*     x1, xref,xi,xj = refrigerant mole fraction
*    xk = oil mole fraction
*    T, Tem = mixture temperature
*    Pres, Pr = mixture pressure
*     Psati,Psatj, psi,psj = saturation pressure at T
*     arti,artj = activity coefficient, calculated
*     p1i,plj, pldi,pldj = pure refrigerant liquid density
 at T or Pres
*    p2,pk = oil density at T
*     oyi, oys = coefficients of oil density relation
*     props(i,j) = pure refigerant properties from
ASHRAE
*    fcri,fcrj,fcmi,fcmj = fugacities of pure
refrigerant, mixture
*****************************************
******
          open (66, file="r32data", status="old")
          open (22, file="r125data", status="old")
          open (44, file="r410arl68h",
status="old")
          open (11, file="output",status="new")
     Rewind (unit=66)
          rewind(unit=44)
          rewind(unit=22)
*****************************************
*******
*    Read in the experimental data for R410a:
*          Temp (C), Pressure (kPa), liquid mole
fraction R-32,
*          liquid mole fraction R-125
*****************************************
*******
          nkp=0
          do 14 l=1,1000
          read(44,*,end=14)Tk(l),Pk(l),xki(l),xkj(l)
          tk(l)=tk(l)+273.15
          nkp=nkp+1
 14       continue
*****************************************
*******
*   Read in pure refrigerant (R-32 or R-125) from
ASHRAE file
*****************************************
*******
     np = 0
          np1 = 0
     do 11 j=1,1000
          read(66,*,end=11) (props(j,i),i=1,4)
          props(j,1) = props(j,1) + 273.15
          np = np + 1
 11   continue
          do 12 j=1,1000
       read(22,*,end=12) (props1(j,i),i=1,4)
       props1(j,1) = props1(j,1) + 273.15
      np1 = np1 + 1
 12   continue
     close(66)
          close(22)
          close(44)
*****************************************
*******
*****************************************
*******
*          Declare critical properties for R-32,R-125,
and POE
*****************************************
*******
          Tci=78.41d0+273.15d0
          Tcj=66.04d0+273.15d0
          Pci=5857.90d0
          Pcj=3595.0d0
          rmwi=52.02d0
          rmwj=120.03d0
          rki=0.772805d0
          rkj=0.81485d0
          dij=0.0072d0
          R=8.31441d0
          yi=0.698d0
          yj=0.302d0
     rmwo=700.d0
      oyi=993.89
      oys=-0.7566
*****************************************
*******
*    linearly interpolate to find ref.liq.dens
*    at measured temperature or pressure
*
```

```
          perrort=0.
          sum=0.
          tpct=0.
          do 121 kk=1,nkp
          if(tk(kk).gt.0.95*Tci.or.pk(kk).gt.1700)t
 hen
          go to 121
          endif
          tem=tk(kk)
          x1=xki(kk)
          x2=xkj(kk)
          kval=Tem
     dcol = 3
     do 716 i=1,np
          if(props(i,1).lt.kval)then
                    goto 716
          else
                    ter = (kval-props(i-
 1,1))/(props(i,1)-props(i-1,1))
                    goto 70
          endif
  716  continue
  70   dval = ter*(props(i,dcol)-props(i-
 1,dcol))+props(i-1,dcol)
     pldi=dval
          do 816 i=1,npl
        if(props1(i,1).lt.kval)then
          goto 816
     else
           ter = (kval-props1(i-1,1))/(props1(i,1)-
 props1(i-1,1))
          goto 80
     endif
  816  continue
  80   dval = ter*(props(i,dcol)-props(i-
 1,dcol))+props(i-1,dcol)
     pldj=dval
*****************************************
******
*   choose a model
     write(*,*)"Choose a model"
     write(*,*)"Enter a 1 for the Wilson Model"
     write(*,*)"2=Heil Model"
     write(*,*)"3=NRTL Model"
     write(*,*)"4=Tsuboka and Katayama Model"
     write(*,*)"5=Wang and Chao Model is not
 available"
     write(*,*)"6=UNIQUAC Model"
*          SEt model #
          m=3
*****************************************
******
          call Pressure(Tem,pr, x1,
 m,pldi,pldj,oyi,oys,
   +
 rmwi,rmwo,R,Tci,Pci,rki,rmwj,Tcj,Pcj,rkj,dij,x2)
          perror=((pr-pk(kk))/pk(kk))**2.
          perrort=perrort+perror
          pdata(kk)=pk(kk)
          pcalc(kk)=pr
          pct=100.*(pk(kk)-pr)/pk(kk)
          tpct=tpct+pct
          sum=sum+1.
  121     continue
          pstd=2*(perrort/float(nkp-1))**0.5
          avpct=tpct/nkp
          do 131 jj=1,nkp
          write(11,*)pdata(jj),pcalc(jj)
          write(*,*)pdata(jj),pcalc(jj)
  131     continue
          write(11,*)"the pstd=",pstd
          write(*,*)"the pstd=",pstd
     END
*****************************************
******
*   subroutine to calculate pressure from temperature
 and concentration
     subroutine Pressure(Tem, Pr, x1,
 m,pldi,pldj,oyi,oys,
   +
 rmwi,rmwo,R,Tci,Pci,rki,rmwj,Tcj,Pcj,rkj,dij,x2)
     implicit double precision (a-h, o-z)
*****************************************
******
          yi=0.698d0
*   call model based on users choice
     if(m.eq.1)then
*   find activity coefficient from Wilson
      call Wilson(Tem,x1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
          elseif(m.eq.2)then
      call Heil(Tem,x1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.3)then
      call NRTL(Tem,X1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.4)then
      call TandK(Tem,x1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.5)then
      call WandC(Tem,X1,x2,art,rmn,
 pld,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.6)then
     call
 Uniquac(Tem,X1,x2,art,rmn,pldi,pldj,oyi,oys,rmwi
 ,rmwj,rmwo,R)
     endif
*****************************************
******
*   find saturation pressure
          call Psatc(Tem, Psat1,Tci,Pci,rki)
*****************************************
******
*   estimate mixture Pressure using act coeff
     Pnew = (art*x1*Psat1)
*****************************************
******
*   get fugacity coefficients
          call Fuga(Tem, Psat1, Fcr,Tci,Pci,rki)
     do 215 inh=1,1000
          call Fugabin(Tem, Pnew,
 Fcm,Tci,Pci,rki,Tcj,Pcj,rkj,dij)
          Pe = exp((1/pldi)*(Pnew-Psat1)/(R*Tem))
          Pcalc = (art*x1*Psat1*fcr*Pe)/(yi*fcm)
*   test for convergence
          delt = abs(Pcalc-Pnew)
          conv = 0.0005*Pcalc
     if (delt.lt.conv) then
```

```
            goto 614
        else
            Pnew=(Pcalc+Pnew)/2
        endif
  215  continue
  614  Pr = Pcalc
            write(*,*)"The predicted
 pressure=",Pcalc,inh
        return
        end
*****************************************
******
*   subroutine to calculate activity coefficients
      Subroutine
 Wilson(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj
 ,rmwo,R)
       implicit double precision (a-h, o-z)
*****************************************
******
*   calculate variables
            rla12=0.4
            rla21=190.7
            rla13=20000.
            rla31=2650.4
            rla23=20000.
            rla32=1441.
            p3 = oyi + oys*(T-273.15)
            x3=1.-x1-x2
            v1 = (1/p1)*(rmwi)
            v2 = (1/p2)*(rmwj)
            v3 = (1/p3)*(rmwo)
            rho12 = v1/v2
            rho21 = v2/v1
            rho13 = v1/v3
            rho31 = v3/v1
            rho23 = v2/v3
            rho32 = v3/v2
            Tao12 = rla12/(R*T)
            Tao21 = rla21/(R*T)
            Tao13 = rla13/(R*T)
            Tao31 = rla31/(R*T)
            Tao23 = rla23/(R*T)
            Tao32 = rla32/(R*T)
            G12 = (rho12*(exp(-Tao12)))
            G21= (rho21*(exp(-Tao21)))
            G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
            G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
*   calculate Wilson log(act coeff)
           rlogar=1.-log(x1+x2*G21+x3*G31)-
x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
x3*G13/(x1*G13+x2*G23+x3)
*   calculate activity coefficient
           art = exp(rlogar)
     return
     end
*****************************************
*****
*          Heil model subroutine
     subroutine
heil(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,rm
wo,R)
       implicit double precision (a-h,o-z)
*****************************************
******
*   calculate variables
            rla12=11.0
       rla21=87.8
       rla13=6706.6
       rla31=1204.5
       rla23=5493.
       rla32=642.
       p3 = oyi + oys*(T-273.15)
       x3=1.-x1-x2
       v1 = (1/p1)*(rmwi)
       v2 = (1/p2)*(rmwj)
       v3 = (1/p3)*(rmwo)
       rho12 = v1/v2
       rho21 = v2/v1
       rho13 = v1/v3
       rho31 = v3/v1
       rho23 = v2/v3
       rho32 = v3/v2
       Tao12 = rla12/(R*T)
       Tao21 = rla21/(R*T)
       Tao13 = rla13/(R*T)
       Tao31 = rla31/(R*T)
       Tao23 = rla23/(R*T)
       Tao32 = rla32/(R*T)
       G12 = (rho12*(exp(-Tao12)))
       G21= (rho21*(exp(-Tao21)))
       G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
       G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
            z1=(tao21*G21*x2+Tao31*G31*x3)/(x1
+G21*x2+G31*x3)
            z2=x1/(x1+G21*x2+G31*x3)*(0.0d0-
(x2*Tao21*G21+x3*Tao31*G31)/
     +  (x1+G21*x2+G31*x3))
            z3=x2*G12/(G12*x1+x2+G32*x3)*(Tao
12-
    +
(x1*Tao12*G12+x3*Tao32*G32)/(G12*x1+x2+G
32*x3))
            z4=x3*G13/(G13*x1+G23*x2+x3)*(Tao
13-
    +
(x1*Tao13*G13+x2*Tao23*G23)/(G13*x1+G23*
x2+x3))
*   calculate Heil log(act coeff)
           rlogar=1.-log(x1+x2*G21+x3*G31)-
x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
x3*G13/(x1*G13+x2*G23+x3)
    + +z1+z2+z3+z4
*   calculate activity coefficient
      art = exp(rlogar)
     return
     end
*****************************************
**********
*
*          Tsuboka and Katayama Subroutine
```

```
          Subroutine
 TandK(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,r
 mwo,R)
      implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla12=-10.1
       rla21=-69.4
       rla13=-15057.2
       rla31=1143.9
       rla23=-4941.
       rla32=918.0
       p3 = oyi + oys*(T-273.15)
       x3=1.-x1-x2
       v1 = (1/p1)*(rmwi)
       v2 = (1/p2)*(rmwj)
       v3 = (1/p3)*(rmwo)
       rho12 = v1/v2
       rho21 = v2/v1
       rho13 = v1/v3
       rho31 = v3/v1
       rho23 = v2/v3
       rho32 = v3/v2
       Tao12 = rla12/(R*T)
       Tao21 = rla21/(R*T)
       Tao13 = rla13/(R*T)
       Tao31 = rla31/(R*T)
       Tao23 = rla23/(R*T)
       Tao32 = rla32/(R*T)
       G12 = (rho12*(exp(-Tao12)))
       G21= (rho21*(exp(-Tao21)))
       G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
       G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
 *   calculate TandK  log(act coeff)
        rlogar=0.0d0-log(x1+x2*G21+x3*G31)-
 x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
 x3*G13/(x1*G13+x2*G23+x3)
    +
 +log(x1+x2*rho21+x3*rho31)+x1/(x1+x2*rho21
 +x3*rho31)
    +
 +x2*rho12/(x1*rho12+x2+x3*rho32)+x3*rho13/(
 x1*rho13+x2*rho23+x3)
 *   calculate Tsuboka and Katayama log(act coeff)
           art = exp(rlogar)
     return
     end

 ****************************************
 **********
 *
 * Wang and Chao Model subroutine
           subroutine
 WandC(T,x1,x2,art,rmn,p1,oyi,oys,rmwi,rmwj,rm
 wo,R)
      implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
       if(rmn.eq.1)then
            rla1=2084.5
            rla2=15744.1
                    rmw=rmwi
       endif
        if(rmn.eq.2)then
            rla2=4579.
            rla1=-381.0
                    rmw=rmwj
       endif
        p2 = oyi + oys*(T-273.15)
       v1 = (1/p1)*(rmw)
       v2 = (1/p2)*(rmwo)
       rho12 = v1/v2
       rho21 = v2/v1
       Tao12 = rla2/(R*T)
       Tao21 = rla1/(R*T)
       G12 = (rho12*(exp(-Tao12)))
       G21= (rho21*(exp(-Tao21)))
            x11=x1/(x1+x2*exp(-Tao21))
            x22=x2/(x2+x1*exp(-Tao12))
            x12=x22*x1*exp(-Tao12)/x2
            x21=x11*x2*exp(-Tao21)/x1
 *   calculate Wand and Chao log(act coeff)
      rlogar =
 -log(x1+x2*G21)+x2*((G21/(x1+G21*x2))
     & -
 (G12/(x2+G12*x1)))+(6./(R*T*2.))*x21*x21*rla1
      &    +(6./(R*T*2.))*x2*x22*x12*rla2/x1
 *   calculate activity coefficient
       art = exp(rlogar)
     return
     end
 ****************************************
 **********
 *
 *          NRTL model subroutine
 **
           subroutine
 NRTL(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,r
 mwo,R)
      implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla21=-398.1
           rla12=237.5
       rla31=20000.
       rla13=-2852.1
       rla32=8481.
       rla23=-3363.0
            x3=1.0-x2-x1
      Tao12 = rla12/(R*T)
      Tao21 = rla21/(R*T)
      Tao13 = rla13/(R*T)
      Tao31 = rla31/(R*T)
      Tao23 = rla23/(R*T)
      Tao32 = rla32/(R*T)
      G12 = ((exp(-Tao12/2.)))
      G21= ((exp(-Tao21/2.)))
      G13 = ((exp(-Tao13/2.)))
      G31= ((exp(-Tao31/2.)))
      G32 = ((exp(-Tao32/2.)))
      G23= ((exp(-Tao23/2.)))
 *  calculate NRTL log(act coeff)
```

```
          rlogar=(x2*tao21*G21+x3*Tao31*G31)/
 (x1+x2*G21+x3*G31) +
     +x1/(x1+x2*G21+x3*G31)*(-
 (x2*G21*tao21+x3*G31*tao31)/
     +(x1+x2*G21+x3*G31)) +
 x2*G12/(x1*G12+x2+x3*G32)*
     +(tao12-
 (x1*G12*tao12+x3*G32*tao32)/(x1*G12+x2+x3
 *G32))+
     +x3*G13/(x1*G13+x2*G23+x3)*(Tao13-
 (x1*G13*tao13+x2*G23*tao23)/
     +(x1*G13+x2*G23+x3))
 *   calculate activity coefficient
      art = exp(rlogar)
     return
     end
 ****************************************
 **********
 *  Uniquac Model subroutine
     subroutine
 UNIQUAC(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rm
 wj,rmwo,R)
     implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla12=0.9
           rla21=125.6
           rla23=-570.
           rla32=1715.
           rla31=-200.0
           rla13=20000.
                 r1=1.43
                 q1=1.42
                 r3=29.4
                 q3=24.36
                 r2=2.61
                 q2=2.49
      p3 = oyi + oys*(T-273.15)
          x3=1.0-x2-x1
      Tao12 = exp(-rla12/(R*T))
      Tao21 = exp(-rla21/(R*T))
      Tao13 = exp(-rla13/(R*T))
      Tao31 = exp(-rla31/(R*T))
      Tao23 = exp(-rla23/(R*T))
      Tao32 = exp(-rla32/(R*T))
           t1=q1*x1/(q1*x1+q2*x2+q3*x3)
           t2=q2*x2/(q1*x1+q2*x2+q3*x3)
           t3=q3*x3/(q1*x1+q2*x2+q3*x3)
           s1=r1*x1/(r1*x1+r2*x2+r3*x3)
           s2=r2*x2/(r1*x1+r2*x2+r3*x3)
           s3=r3*x3/(r1*x1+r2*x2+r3*x3)
           u1=10./2.*(r1-q1)-(r1-1.)
           u2=10./2.*(r2-q2)-(r2-1.)
           u3=10./2.*(r3-q3)-(r3-1.)
           rlnc=0.0
           rlnr=0.0
 *   calculate UNIQUAC log(act coeff)
           rlnc=log(s1/x1)+10./2.*q1*log(t1/s1)+u1
           rlnc=rlnc-s1/x1*(x1*u1+x2*u2+x3*u3)
           rlnr=q1-q1*log(t1+t2*tao21+t3*tao31)
           rlnr=rlnr-
 q1*((t1/(t1+t2*tao21+t3*tao31))+
 +t2*tao12/(t1*tao12+t2+t3*tao32)+t3*tao13/(t1*
 tao13+t2*tao23+t3))
           rlogar=rlnc+rlnr
 *   calculate activity coefficient
      art = exp(rlogar)
     return
           end
 ****************************************
 ********
 *   subroutine to calculate fugacity
      Subroutine Fuga(T, Pm, FC,Tc,Pc,rk)
      implicit double precision (a-h, o-z)
     Prr = Pm/Pc
     Trr = T/Tc
      alpha =(1+rk*(1-Trr**0.5))**2
      A = 0.45724*alpha*Prr/(Trr*Trr)
      B = 0.07780*Prr/Trr
      a1 = -1*(1-B)
      a2 = A-3*B*B-2*B
      a3 = -1*(A*B-B*B-B*B*B)
      QQ = (a1*a1-3*a2)/9
       RR =(2*a1*a1*a1-9*a1*a2+27*a3)/54
     if ((RR*RR).gt.(QQ*QQ*QQ)) then
           if(RR.gt.0) sign = 1.0
           if(RR.lt.0) sign = -1.0
           AA = -1*sign*(abs(RR)+(RR*RR-
 QQ*QQ*QQ)**(1/3))
           BB = QQ/AA
           if(aa.eq.0) bb = 0
           Zv = (AA+BB)-a1/3
           goto 6
     endif
     phi = acos(RR/((QQ*QQ*QQ)**.5))
      x1 = -2*(QQ**0.5)*cos(phi/3)-a1/3
      x2 = -2*(QQ**0.5)*cos((phi+2*3.1415926)/3)-
 a1/3
      x3 = -2*(QQ**0.5)*cos((phi-2*3.1415926)/3)-
 a1/3
     Zv = max(x1,x2,x3)
     ppp = Zv-b
     Zl = min(x1,x2,x3)
 6    rlnfv=Zv-1.0-log(ppp)
     rlnfv=rlnfv-
 (A/(2.*(2.**0.5)*B))*log((Zv+2.414*B)/(Zv-
 0.414*B))
    FC = EXP(rlnfv)
    return
    end
 ****************************************
 **
 ****************************************
 **
 *         subroutine to calculate component
 fugacity
           Subroutine fugabin(Tk, P,
 Fcm,Tci,Pci,rki,Tcj,Pcj,rkj,dij)
           implicit double precision (a-h, o-z)
      double precision
 lnfv1i,lnfl1i,lnfvi,lnfli,lnfv1j
       double precision lnfvj,lnfl1j,lnflj
      Tci=78.41+273.15d0
      Tcj=66.04+273.15d0
      Pci=5857.90d0
```

```
      Pcj=3595.0d0
      Rmwi=52.02d0
      Rmwj=120.03d0
      rki=0.772805d0
      rkj=0.81485d0
      xi=0.698d0
      xj=0.302d0
      R=8.31441d0
     Tri = Tk/Tci
     Trj = Tk/Tcj
      alphai=(1.0d0+rki*(1.0d0-
Tri**0.5d0))**2.0d0
      alphaj=(1.0d0+rkj*(1.0d0-
Trj**0.5d0))**2.0d0
       ai=(0.45724d0)*alphai*(R**2*Tci**2)/Pci
       aj=(0.45724d0)*alphaj*(R**2*Tcj**2)/Pcj
      bi=(0.07780d0)*R*Tci/Pci
      bj=(0.07780d0)*R*Tcj/Pcj
      bm=xi*bi+xj*bj
       aij=(1.0d0-dij)*ai**0.5d0*aj**0.5d0
      am=xi**2*ai + 2.0d0*xj*xi*aij + xj**2*aj
*****************************************
*  determine the constants of the PR EOS
*****************************************
*
     A = am*P/(R**2*Tk**2)
     B = bm*P/(R*Tk)
     a1 = -1.0*(1.0-B)
     a2 = A-3.0*B*B-2.0*B
     a3 = -1.0*(A*B-B*B-B*B*B)
*****************************************
*  determine the roots of the PR EOS
*****************************************
      Q = (a1*a1-3.0*a2)/9.0
       RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
       if(abs(RR/((Q*Q*Q)**.5)).gt.1.0d0)then
     phi=0.0d0
     else
      phi = dacos(RR/((Q*Q*Q)**.5))
     endif
       x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
     x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
      x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0
*****************************************
*  determine if vapor fugacity equals the liquid
fugacity
*****************************************
    Zv = max(x1,x2,x3)
    Zl = min(x1,x2,x3)
*****************************************
*      calculate the fugacity coefficient for the vapor
phase of component i
*****************************************
     lnfv1i= bi/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xj*aij+2.0*xi*ai)/am - bi/bm
      lnfvi=lnfv1i-(A/(2.0*(2.0**0.5)*B))*
    +u*dlog((Zv+2.414*B)/(Zv-0.414*B))
*****************************************
*      calculate the fugacity coefficient for the vapor
phase of component j
*****************************************
     lnfv1j= bj/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xi*aij+2.0*xj*aj)/am - bj/bm
      lnfvj=lnfv1j-(A/(2.0*(2.0**0.5)*B))*
    +u*dlog((Zv+2.414*B)/(Zv-0.414*B))
*****************************************
*****************************************
      if(rmn.eq.1)then
      fcm=exp(lnfvi)
     else
      fcm=exp(lnfvj)
     endif
    return
    end



*****************************************
*  subroutine to calculate saturation pressure
     Subroutine Psatc(T, Psat,Tc,Pc,rk)
      implicit double precision (a-h,l, o-z)
*
*  interaction loop to find Psat
    Trr = T/Tc
     P = Pc*0.75*Trr
    diff = 0
    dp = 1
     iternmax = 25000
          do 200 itern=1,iternmax
*
*  determine the constants of the PR EOS
      alpha=(1.0+rk*(1.0-Trr**0.5))**2.0
    Prr = P/Pc
     A = 0.45724*alpha*Prr/(Trr*Trr)
     B = 0.07780*Prr/Trr
     a1 = -1.0*(1.0-B)
     a2 = A-3.0*B*B-2.0*B
     a3 = -1.0*(A*B-B*B-B*B*B)
*
*  determine the roots of the PR EOS
     Q = (a1*a1-3.0*a2)/9.0
      RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
     if (q.lt.0)then
          P = 0.5*P
          go to 200
    endif
     phi = dacos(RR/((Q*Q*Q)**.5))
      x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
    x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
     x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0

*  determine if vapor fugacity equals the liquid
fugacity
    Zv = max(x1,x2,x3)
    Zl = min(x1,x2,x3)
    lnfv1= Zv-1.0-dlog(Zv-B)
    lnfv=lnfv1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zv+2.414*B)/(Zv-
0.414*B))
    lnfl1=Zl-1.0-dlog(Zl-B)
```

```
      lnfl=lnfl1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zl+2.414*B)/(Zl-
0.414*B))
      diffold = diff
      diff = lnfv-lnfl
*
*   determine the next guess
      if ((diffold*diff).lt.0) dp = 0.1*dp
      if (diff.lt.0) P = P+dp
      if (diff.gt.0) P = P-dp
*
*   check for convergence
      adiff = abs(diff)
       if (adiff.lt.abs(0.0001*lnfv)) goto 28
 200  continue
 28   psat = p
           write(*,*)"psat=",P,"subroutine  Psatc
complete  in",itern
     return
     end
**************************************
**
```

The program "pvtbin2.f" is a modified version of "pvtbin1.f," allowing the mole fraction of each component in both the vapor and liquid phase to be specified.

```
**************************************
******
     PROGRAM pvtbin2
      implicit double precision (a-h, o-z)
           double precision
props(100,10),tk(100),pk(100),xki(100)
           double precision
pdata(100),pcalc(100),xkj(100)
           double precision
props1(100,10),yki(100)
**************************************
******
*          This program models a binary refrigerant
blend (R-410a)
*          mixed with a polyol ester (RL68H)
*          This is a  modelling scheme which allows
the user to input
*          the known mole fractions of the vapor and
liquid phases
**************************************
******
**************************************
******
*          The "subscripts" i, j, and k correspond to
R-32,  R-125,
*          and RL68H, respectively.
**************************************
******
*   VARIABLES
*    Tci,Tcj = critical temperature
*    Pci,Pcj = critical pressure
*    rmwi,rmwj, = refrigerant molar weight
*    rmwo = oil molar weight
*    rki,rkj = Peng-Robinson EOS Parameter
*     x1, xref,xki,xkj = refrigerant mole
fraction(liquid)
*    xkk = oil mole fraction(liquid)
*    yki,ykj = vapor phase mole fractions of comp i
and j
*    T, Tem = mixture temperature
*    Pres, Pr = mixture pressure
*     Psati,Psatj, psi,psj = saturation pressure at T
*     arti,artj = activity coefficient, calculated
*     p1i,plj, pldi,pldj = pure refrigerant liquid density
at T or Pres
*    p2,pk = oil density at T
*     oyi, oys = coefficients of oil density relation
*     props(i,j) = pure refigerant properties from
ASHRAE
*    fcri,fcrj,fcmi,fcmj = fugacities of pure
refrigerant,  mixture
**************************************
******
           open (66, file="r32data", status="old")
           open (22, file="r125data", status="old")
           open (44, file="r410arl68h2",
status="old")
           open (11, file="output",status="new")
     Rewind (unit=66)
           rewind(unit=44)
           rewind(unit=22)
**************************************
*******
*    Read in the experimental data for R410a:
*          Temp (C), Pressure (kPa), liquid mole
fraction  R-32,
*          liquid mole fraction R-125
**************************************
*******
           nkp=0
           do 14 l=1,1000
           read(44,*,end=14)Tk(l),Pk(l),yki(l),xki(l)
,xkj(l)
           tk(l)=tk(l)+273.15
           nkp=nkp+1
 14        continue
**************************************
*******
*    Read in pure refrigerant (R-32 or R-125) from
ASHRAE file
**************************************
*******
    np = 0
           np1 = 0
    do 11 j=1,1000
          read(66,*,end=11)  (props(j,i),i=1,4)
          props(j,1) = props(j,1) + 273.15
          np = np + 1
 11   continue
          do 12 j=1,1000
```

```
      read(22,*,end=12) (props1(j,i),i=1,4)
      props1(j,1) = props1(j,1) + 273.15
     npl = npl + 1
 12  continue
    close(66)
         close(22)
         close(44)
*************************************
*******
*************************************
*******
*       Declare critical properties for R-32,R-125,
and POE
*************************************
*******
         Tci=78.41d0+273.15d0
         Tcj=66.04d0+273.15d0
         Pci=5857.90d0
         Pcj=3595.0d0
         rmwi=52.02d0
         rmwj=120.03d0
         rki=0.772805d0
         rkj=0.81485d0
         dij=0.0072d0
         R=8.31441d0
    rmwo=700.d0
     oyi=993.89
     oys=-0.7566
*************************************
*******
*   linearly interpolate to find ref.liq.dens
*   at measured temperature or pressure
*
         perrort=0.
         sum=0.
         tpct=0.
         do 121 kk=1,nkp
         if(tk(kk).gt.0.95*Tci.or.pk(kk).gt.1700)t
hen
         go to 121
         endif
         tem=tk(kk)
         x1=xki(kk)
         x2=xkj(kk)
         y1=yki(kk)
         kval=Tem
   dcol = 3
   do 716 i=1,np
         if(props(i,1).lt.kval)then
                  goto 716
         else
                  ter = (kval-props(i-
1,1))/(props(i,1)-props(i-1,1))
                  goto 70
         endif
 716  continue
 70   dval = ter*(props(i,dcol)-props(i-
1,dcol))+props(i-1,dcol)
   pldi=dval
         do 816 i=1,npl
      if(props1(i,1).lt.kval)then
         goto 816
    else
            ter = (kval-props1(i-1,1))/(props1(i,1)-
 props1(i-1,1))
         goto 80
      endif
 816  continue
 80   dval = ter*(props(i,dcol)-props(i-
1,dcol))+props(i-1,dcol)
    pldj=dval
*************************************
******
*   choose a model
     write(*,*)"Choose a model"
     write(*,*)"Enter a 1 for the Wilson Model"
     write(*,*)"2=Heil Model"
     write(*,*)"3=NRTL Model"
     write(*,*)"4=Tsuboka and Katayama Model"
     write(*,*)"5=Wang and Chao Model is not
available"
    write(*,*)"6=UNIQUAC Model"
*         SEt model #
         m=6
*************************************
******
         call Pressure(Tem,pr,y1, x1,
m,pldi,pldj,oyi,oys,
   +
rmwi,rmwo,R,Tci,Pci,rki,rmwj,Tcj,Pcj,rkj,dij,x2)
         perror=((pr-pk(kk))/pk(kk))**2.
         perrort=perrort+perror
         pdata(kk)=pk(kk)
         pcalc(kk)=pr
         pct=100.*(pk(kk)-pr)/pk(kk)
         tpct=tpct+pct
         sum=sum+1.
 121     continue
         pstd=2*(perrort/float(nkp-1))**0.5
         avpct=tpct/nkp
         do 131 jj=1,nkp
         write(11,*)pdata(jj),pcalc(jj)
         write(*,*)pdata(jj),pcalc(jj)
 131     continue
         write(11,*)"the pstd=",pstd
         write(*,*)"the pstd=",pstd
   END
*************************************
******
*  subroutine to calculate pressure from temperature
and concentration
     subroutine Pressure(Tem, Pr,y1, x1,
m,pldi,pldj,oyi,oys,
   +
rmwi,rmwo,R,Tci,Pci,rki,rmwj,Tcj,Pcj,rkj,dij,x2)
     implicit double precision (a-h, o-z)
*************************************
******
*  call model based on users choice
     if(m.eq.1)then
*   find activity coefficient from Wilson
     call Wilson(Tem,
    +x1,x2,art,rmn,
pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
          elseif(m.eq.2)then
    call Heil(Tem,
```

```
     +x1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.3)then
     call NRTL(Tem,
     +X1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.4)then
     call TandK(Tem,
     +x1,x2,art,rmn,
 pldi,pldj,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.5)then
     call WandC(Tem,
     +X1,x2,art,rmn,
 pld,oyi,oys,rmwi,rmwj,rmwo,R)
       elseif(m.eq.6)then
     call
 Uniquac(Tem,x1,x2,art,rmn,pldi,pldj,oyi,oys,rmwi
 ,rmwj,rmwo,R)
     endif
 ****************************************
 ******
 *   find saturation pressure
           call Psatc(Tem, Psat1,Tci,Pci,rki)
 ****************************************
 ******
 *   estimate mixture Pressure using act coeff
     Pnew = (art*x1*Psat1)
 ****************************************
 ******
 *   get fugacity coefficients
           call Fuga(Tem, Psat1, Fcr,Tci,Pci,rki)
     do 215 inh=1,1000
           call Fugabin(Tem, Pnew,
 Fcm,y1,Tci,Pci,rki,Tcj,Pcj,rkj,dij)
           Pe = exp((1/pldi)*(Pnew-Psat1)/(R*Tem))
           Pcalc = (art*x1*Psat1*fcr*Pe)/(y1*fcm)
 *   test for convergence
           delt = abs(Pcalc-Pnew)
           conv = 0.0005*Pcalc
     if (delt.lt.conv) then
           goto 614
    else
           Pnew=(Pcalc+Pnew)/2
    endif
  215 continue
  614 Pr = Pcalc
           write(*,*)"The predicted
 pressure=",Pcalc,inh
    return
    end
 ****************************************
 ******
 *   subroutine to calculate activity coefficients
    Subroutine
 Wilson(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj
 ,rmwo,R)
     implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla12=0.4
           rla21=190.7
           rla13=20000.
           rla31=2650.4
           rla23=20000.
           rla32=1441.
           p3 = oyi + oys*(T-273.15)
           x3=1.-x1-x2
           v1 = (1/p1)*(rmwi)
           v2 = (1/p2)*(rmwj)
           v3 = (1/p3)*(rmwo)
           rho12 = v1/v2
           rho21 = v2/v1
           rho13 = v1/v3
           rho31 = v3/v1
           rho23 = v2/v3
           rho32 = v3/v2
           Tao12 = rla12/(R*T)
           Tao21 = rla21/(R*T)
           Tao13 = rla13/(R*T)
           Tao31 = rla31/(R*T)
           Tao23 = rla23/(R*T)
           Tao32 = rla32/(R*T)
           G12 = (rho12*(exp(-Tao12)))
           G21= (rho21*(exp(-Tao21)))
           G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
           G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
 *   calculate Wilson log(act coeff)
           rlogar=1.-log(x1+x2*G21+x3*G31)-
 x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
 x3*G13/(x1*G13+x2*G23+x3)
 *   calculate activity coefficient
           art = exp(rlogar)
    return
    end
 ****************************************
 *****
 *          Heil model subroutine
    subroutine
 heil(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,rm
 wo,R)
          implicit double precision (a-h,o-z)
 ****************************************
 ******
 *   calculate variables
           rla12=11.0
      rla21=87.8
      rla13=6706.6
      rla31=1204.5
      rla23=5493.
      rla32=642.
      p3 = oyi + oys*(T-273.15)
      x3=1.-x1-x2
      v1 = (1/p1)*(rmwi)
      v2 = (1/p2)*(rmwj)
      v3 = (1/p3)*(rmwo)
      rho12 = v1/v2
      rho21 = v2/v1
      rho13 = v1/v3
      rho31 = v3/v1
      rho23 = v2/v3
      rho32 = v3/v2
      Tao12 = rla12/(R*T)
      Tao21 = rla21/(R*T)
      Tao13 = rla13/(R*T)
```

```
      Tao31 = rla31/(R*T)
      Tao23 = rla23/(R*T)
      Tao32 = rla32/(R*T)
       G12 = (rho12*(exp(-Tao12)))
       G21= (rho21*(exp(-Tao21)))
       G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
       G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
            z1=(tao21*G21*x2+Tao31*G31*x3)/(x1
+G21*x2+G31*x3)
            z2=x1/(x1+G21*x2+G31*x3)*(0.0d0-
(x2*Tao21*G21+x3*Tao31*G31)/
     + (x1+G21*x2+G31*x3))
            z3=x2*G12/(G12*x1+x2+G32*x3)*(Tao
12-
     +
(x1*Tao12*G12+x3*Tao32*G32)/(G12*x1+x2+G
32*x3))
            z4=x3*G13/(G13*x1+G23*x2+x3)*(Tao
13-
     +
(x1*Tao13*G13+x2*Tao23*G23)/(G13*x1+G23*
x2+x3))
*    calculate Heil log(act coeff)
            rlogar=1.-log(x1+x2*G21+x3*G31)-
x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
x3*G13/(x1*G13+x2*G23+x3)
     + +z1+z2+z3+z4
*    calculate activity coefficient
       art = exp(rlogar)
     return
     end
****************************************
**********
*
*           Tsuboka and Katayama Subroutine
           Subroutine
TandK(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,r
mwo,R)
      implicit double precision (a-h, o-z)
****************************************
******
*    calculate variables
           rla12=-10.1
       rla21=-69.4
       rla13=-15057.2
       rla31=1143.9
       rla23=-4941.
       rla32=918.0
       p3 = oyi + oys*(T-273.15)
       x3=1.-x1-x2
      v1 = (1/p1)*(rmwi)
      v2 = (1/p2)*(rmwj)
      v3 = (1/p3)*(rmwo)
      rho12 = v1/v2
      rho21 = v2/v1
      rho13 = v1/v3
      rho31 = v3/v1
      rho23 = v2/v3
      rho32 = v3/v2
     Tao12 = rla12/(R*T)
     Tao21 = rla21/(R*T)
      Tao13 = rla13/(R*T)
      Tao31 = rla31/(R*T)
      Tao23 = rla23/(R*T)
      Tao32 = rla32/(R*T)
       G12 = (rho12*(exp(-Tao12)))
       G21= (rho21*(exp(-Tao21)))
       G13 = (rho13*(exp(-Tao13)))
       G31= (rho31*(exp(-Tao31)))
       G32 = (rho32*(exp(-Tao32)))
       G23= (rho23*(exp(-Tao23)))
*    calculate TandK  log(act coeff)
       rlogar=0.0d0-log(x1+x2*G21+x3*G31)-
x1/(x1+x2*G21+x3*G31)
     + -x2*G12/(x1*G12+x2+x3*G32) -
x3*G13/(x1*G13+x2*G23+x3)
   +
+log(x1+x2*rho21+x3*rho31)+x1/(x1+x2*rho21
+x3*rho31)
   +
+x2*rho12/(x1*rho12+x2+x3*rho32)+x3*rho13/(
x1*rho13+x2*rho23+x3)
*    calculate Tsuboka and Katayama log(act coeff)
           art = exp(rlogar)
     return
     end

****************************************
**********
*
* Wang and Chao Model subroutine
           subroutine
WandC(T,x1,x2,art,rmn,p1,oyi,oys,rmwi,rmwj,rm
wo,R)
      implicit double precision (a-h, o-z)
****************************************
******
*    calculate variables
       if(rmn.eq.1)then
            rla1=2084.5
            rla2=15744.1
                  rmw=rmwi
      endif
       if(rmn.eq.2)then
            rla2=4579.
            rla1=-381.0
                  rmw=rmwj
      endif
       p2 = oyi + oys*(T-273.15)
      v1 = (1/p1)*(rmw)
      v2 = (1/p2)*(rmwo)
      rho12 = v1/v2
      rho21 = v2/v1
     Tao12 = rla2/(R*T)
     Tao21 = rla1/(R*T)
      G12 = (rho12*(exp(-Tao12)))
      G21= (rho21*(exp(-Tao21)))
           x11=x1/(x1+x2*exp(-Tao21))
           x22=x2/(x2+x1*exp(-Tao12))
           x12=x22*x1*exp(-Tao12)/x2
           x21=x11*x2*exp(-Tao21)/x1
*    calculate Wand and Chao log(act coeff)
     rlogar =
-log(x1+x2*G21)+x2*((G21/(x1+G21*x2))
```

```
     & -
 (G12/(x2+G12*x1)))+(6./(R*T*2.))*x21*x21*rla1
      &   +(6./(R*T*2.))*x2*x22*x12*rla2/x1
 *   calculate activity coefficient
       art = exp(rlogar)
     return
     end
 ****************************************
 **********
 *
 *          NRTL model subroutine
 **
          subroutine
 NRTL(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rmwj,r
 mwo,R)
     implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla21=-398.1
           rla12=237.5
       rla31=20000.
       rla13=-2852.1
       rla32=8481.
       rla23=-3363.0
           x3=1.0-x2-x1
       Tao12 = rla12/(R*T)
       Tao21 = rla21/(R*T)
       Tao13 = rla13/(R*T)
       Tao31 = rla31/(R*T)
       Tao23 = rla23/(R*T)
       Tao32 = rla32/(R*T)
       G12 = ((exp(-Tao12/2.)))
       G21= ((exp(-Tao21/2.)))
       G13 = ((exp(-Tao13/2.)))
       G31= ((exp(-Tao31/2.)))
       G32 = ((exp(-Tao32/2.)))
       G23= ((exp(-Tao23/2.)))
 *   calculate NRTL log(act coeff)
           rlogar=(x2*tao21*G21+x3*Tao31*G31)/
 (x1+x2*G21+x3*G31) +
      +x1/(x1+x2*G21+x3*G31)*(-
 (x2*G21*tao21+x3*G31*tao31)/
     +(x1+x2*G21+x3*G31)) +
 x2*G12/(x1*G12+x2+x3*G32)*
    +(tao12-
 (x1*G12*tao12+x3*G32*tao32)/(x1*G12+x2+x3
 *G32))+
      +x3*G13/(x1*G13+x2*G23+x3)*(Tao13-
 (x1*G13*tao13+x2*G23*tao23)/
     +(x1*G13+x2*G23+x3))
 *   calculate activity coefficient
       art = exp(rlogar)
     return
     end
 ****************************************
 **********
 * Uniquac Model subroutine
     subroutine
 UNIQUAC(T,x1,x2,art,rmn,p1,p2,oyi,oys,rmwi,rm
 wj,rmwo,R)
     implicit double precision (a-h, o-z)
 ****************************************
 ******
 *   calculate variables
           rla12=0.9
           rla21=125.6
           rla23=-570.
           rla32=1715.
           rla31=-200.0
           rla13=20000.
                 r1=1.43
                 q1=1.42
                 r3=29.4
                 q3=24.36
                 r2=2.61
                 q2=2.49
       p3 = oyi + oys*(T-273.15)
           x3=1.0-x2-x1
       Tao12 = exp(-rla12/(R*T))
       Tao21 = exp(-rla21/(R*T))
       Tao13 = exp(-rla13/(R*T))
       Tao31 = exp(-rla31/(R*T))
       Tao23 = exp(-rla23/(R*T))
       Tao32 = exp(-rla32/(R*T))
           t1=q1*x1/(q1*x1+q2*x2+q3*x3)
           t2=q2*x2/(q1*x1+q2*x2+q3*x3)
           t3=q3*x3/(q1*x1+q2*x2+q3*x3)
           s1=r1*x1/(r1*x1+r2*x2+r3*x3)
           s2=r2*x2/(r1*x1+r2*x2+r3*x3)
           s3=r3*x3/(r1*x1+r2*x2+r3*x3)
           u1=10./2.*(r1-q1)-(r1-1.)
           u2=10./2.*(r2-q2)-(r2-1.)
           u3=10./2.*(r3-q3)-(r3-1.)
           rlnc=0.0
           rlnr=0.0
 *   calculate UNIQUAC log(act coeff)
           rlnc=log(s1/x1)+10./2.*q1*log(t1/s1)+u1
           rlnc=rlnc-s1/x1*(x1*u1+x2*u2+x3*u3)
           rlnr=q1-q1*log(t1+t2*tao21+t3*tao31)
           rlnr=rlnr-
 q1*((t1/(t1+t2*tao21+t3*tao31))+

 +t2*tao12/(t1*tao12+t2+t3*tao32)+t3*tao13/(t1*
 tao13+t2*tao23+t3))
           rlogar=rlnc+rlnr
 *   calculate activity coefficient
       art = exp(rlogar)
     return
           end
 ****************************************
 ********
 *   subroutine to calculate fugacity
      Subroutine Fuga(T, Pm, FC,Tc,Pc,rk)
      implicit double precision (a-h, o-z)
     Prr = Pm/Pc
     Trr = T/Tc
      alpha =(1+rk*(1-Trr**0.5))**2
     A = 0.45724*alpha*Prr/(Trr*Trr)
     B = 0.07780*Prr/Trr
     a1 = -1*(1-B)
     a2 = A-3*B*B-2*B
     a3 = -1*(A*B-B*B-B*B*B)
     QQ = (a1*a1-3*a2)/9
      RR =(2*a1*a1*a1-9*a1*a2+27*a3)/54
     if ((RR*RR).gt.(QQ*QQ*QQ)) then
           if(RR.gt.0) sign = 1.0
           if(RR.lt.0) sign = -1.0
```

```
          AA = -1*sign*(abs(RR)+(RR*RR-
 QQ*QQ*QQ)**(1/3))
          BB = QQ/AA
          if(aa.eq.0) bb = 0
          Zv = (AA+BB)-a1/3
          goto 6
     endif
     phi = acos(RR/((QQ*QQ*QQ)**.5))
      x1 = -2*(QQ**0.5)*cos(phi/3)-a1/3
      x2 = -2*(QQ**0.5)*cos((phi+2*3.1415926)/3)-
 a1/3
      x3 = -2*(QQ**0.5)*cos((phi-2*3.1415926)/3)-
 a1/3
     Zv = max(x1,x2,x3)
     ppp = Zv-b
     Zl = min(x1,x2,x3)
 6    rlnfv=Zv-1.0-log(ppp)
     rlnfv=rlnfv-
 (A/(2.*(2.**0.5)*B))*log((Zv+2.414*B)/(Zv-
 0.414*B))
     FC = EXP(rlnfv)
     return
     end
**************************************
**
**************************************
**
*          subroutine to calculate component
 fugacity
          Subroutine fugabin(Tk, P,
 Fcm,y1,Tci,Pci,rki,Tcj,Pcj,rkj,dij)
          implicit double precision (a-h, o-z)
      double precision
 lnfv1i,lnfl1i,lnfvi,lnfli,lnfv1j
       double precision lnfvj,lnfl1j,lnflj
      Tci=78.41+273.15d0
      Tcj=66.04+273.15d0
      Pci=5857.90d0
      Pcj=3595.0d0
      Rmwi=52.02d0
      Rmwj=120.03d0
      rki=0.772805d0
      rkj=0.81485d0
     xi=y1
     xj=1.0d0-xi
     R=8.31441d0
    Tri = Tk/Tci
    Trj = Tk/Tcj
      alphai=(1.0d0+rki*(1.0d0-
 Tri**0.5d0))**2.0d0
      alphaj=(1.0d0+rkj*(1.0d0-
 Trj**0.5d0))**2.0d0
       ai=(0.45724d0)*alphai*(R**2*Tci**2)/Pci
       aj=(0.45724d0)*alphaj*(R**2*Tcj**2)/Pcj
      bi=(0.07780d0)*R*Tci/Pci
      bj=(0.07780d0)*R*Tcj/Pcj
      bm=xi*bi+xj*bj
       aij=(1.0d0-dij)*ai**0.5d0*aj**0.5d0
      am=xi**2*ai + 2.0d0*xj*xi*aij + xj**2*aj
**************************************
*  determine the constants of the PR EOS
**************************************
*
     A = am*P/(R**2*Tk**2)
      B = bm*P/(R*Tk)
       a1 = -1.0*(1.0-B)
       a2 = A-3.0*B*B-2.0*B
       a3 = -1.0*(A*B-B*B-B*B*B)
**************************************
*  determine the roots of the PR EOS
**************************************
       Q = (a1*a1-3.0*a2)/9.0
        RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
       if(abs(RR/((Q*Q*Q)**.5)).gt.1.0d0)then
      phi=0.0d0
      else
       phi = dacos(RR/((Q*Q*Q)**.5))
      endif
        x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
      x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
       x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0
**************************************
*  determine if vapor fugacity equals the liquid
fugacity
**************************************
     Zv = max(x1,x2,x3)
     Zl = min(x1,x2,x3)
**************************************
*      calculate the fugacity coefficient for the vapor
phase of component i
**************************************
     lnfv1i= bi/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xj*aij+2.0*xi*ai)/am - bi/bm
      lnfvi=lnfv1i-(A/(2.0*(2.0**0.5)*B))*
    +u*dlog((Zv+2.414*B)/(Zv-0.414*B))
**************************************
*      calculate the fugacity coefficient for the vapor
phase of component j
**************************************
     lnfv1j= bj/bm*(Zv-1.0)-dlog(Zv-B)
     u=(2.0*xi*aij+2.0*xj*aj)/am - bj/bm
      lnfvj=lnfv1j-(A/(2.0*(2.0**0.5)*B))*
    +u*dlog((Zv+2.414*B)/(Zv-0.414*B))
**************************************
**************************************
      if(rmn.eq.1)then
      fcm=exp(lnfvi)
     else
      fcm=exp(lnfvj)
     endif
    return
    end



**************************************
*  subroutine to calculate saturation pressure
     Subroutine Psatc(T, Psat,Tc,Pc,rk)
     implicit double precision (a-h,l, o-z)
*
*  interaction loop to find Psat
    Trr = T/Tc
    P = Pc*0.75*Trr
    diff = 0
    dp = 1
```

```
      itermmax = 25000
            do 200 itern=1,iternmax
*
*   determine the constants of the PR EOS
        alpha=(1.0+rk*(1.0-Trr**0.5))**2.0
      Prr = P/Pc
       A = 0.45724*alpha*Prr/(Trr*Trr)
       B = 0.07780*Prr/Trr
       a1 = -1.0*(1.0-B)
       a2 = A-3.0*B*B-2.0*B
       a3 = -1.0*(A*B-B*B-B*B*B)
*
*   determine the roots of the PR EOS
       Q = (a1*a1-3.0*a2)/9.0
        RR =(2.0*a1*a1*a1-9.0*a1*a2+27.0*a3)/54.0
       if (q.lt.0)then
            P = 0.5*P
            go to 200
      endif
       phi = dacos(RR/((Q*Q*Q)**.5))
        x1 = -2.0*(Q**0.5)*cos(phi/3.0)-a1/3.0
      x2 =
-2.0*(Q**0.5)*cos((phi+2.0*3.1415926)/3.0)-
a1/3.0
       x3 = -2.0*(Q**0.5)*cos((phi-
2.0*3.1415926)/3.0)-a1/3.0

*   determine if vapor fugacity equals the liquid
fugacity
      Zv = max(x1,x2,x3)
      Zl = min(x1,x2,x3)
      lnfv1= Zv-1.0-dlog(Zv-B)
      lnfv=lnfv1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zv+2.414*B)/(Zv-
0.414*B))
      lnfl1=Zl-1.0-dlog(Zl-B)
      lnfl=lnfl1-
(A/(2.0*(2.0**0.5)*B))*dlog((Zl+2.414*B)/(Zl-
0.414*B))
      diffold = diff
      diff = lnfv-lnfl
*
*   determine the next guess
      if ((diffold*diff).lt.0) dp = 0.1*dp
      if (diff.lt.0) P = P+dp
      if (diff.gt.0) P = P-dp
*
*   check for convergence
      adiff = abs(diff)
       if (adiff.lt.abs(0.0001*lnfv)) goto 28
 200  continue
 28   psat = p
           write(*,*)"psat=",P,"subroutine  Psatc
complete  in",itern
     return
     end
**************************************
**
```

D.2 Model Outputs

Tables D.1 and D.2 give the measured and calculated (modeled) pressures (kPa) for the models used. The symbol "NaNQ" signifies that the program failed for these points.

Table D.1. Measured and predicted pressures for the R-32/POE mixture.

Measured Pressure (kPa)	Wilson model (kPa)	Heil model (kPa)	W & C model (kPa)	T & K model (kPa)	NRTL model (kPa)	UNI-QUAC (kPa)
1156.5	1150.08	1151.25	1167.19	1146.35	1152.39	1145.55
1523.4	1519.15	1520.32	1543.52	1514.14	1521.24	1513.17
1721.1	1720.62	1721.74	1748.96	1714.9	1722.51	1713.85
2014.6	2013.3	2014.28	2047.36	2006.53	2014.82	2005.39
2378.8	2380.95	2381.68	2422.05	2372.86	2381.93	2371.61
2714.8	2716.13	2716.55	2763.47	2706.8	2716.54	2705.48
3183.8	3184.02	3183.86	3239.62	3172.95	3183.52	3171.57
1097.4	1090.4	1099.27	1112.38	1086.04	1118.36	1071.6
1310.6	1304.02	1314.42	1333.62	1297.97	1337.38	1281.61
1602.4	1600.25	1612.67	1641.67	1591.52	1640.64	1572.82
2000.7	1998.02	2012.98	2057.22	1985.14	2046.95	1963.82
2325.9	2323.19	2340.09	2398.34	2306.46	2378.39	2283.41
2527.9	2527.07	2545.11	2612.77	2507.71	2585.87	2483.76
2896.2	2757.32	2776.67	2854.19	2735.16	2820.25	2710.05
3334.8	3057.26	3078.24	3169.22	3031.24	3125.22	3004.83
1075.3	1053.64	1070.68	1068.61	1066.67	1091.19	1022.57
1279.9	1255.97	1275.89	1277.46	1269.47	1304.39	1219.11
1649	1617.14	1642.09	1652.48	1630.26	1685.62	1569.68
2045.6	2014.01	2044.36	2067.65	2024.85	2105.07	1954.38
2085.1	2047.98	2078.79	2103.27	2058.57	2140.97	1987.27
2548.6	2503.61	2540.56	2583.64	2508.98	2622.76	2427.89
2625.9	2578.88	2616.84	2663.32	2583.12	2702.34	2500.55
3320	3254.38	3301.56	3382.39	3244.65	3416.45	3150.57
1008.1	1009.24	1005.31	997.98	1015.97	986.54	1081.95
1135.2	1147.31	1142.74	1141.43	1147.07	1135.25	1230.34
1245.5	1272.13	1267.13	1272.3	1263.64	1270.2	1363.21
1348.8	1388.04	1382.73	1394.82	1369.82	1395.34	1485.33
1463.4	1520.31	1514.84	1535.78	1488.56	1537.7	1622.85
1601.7	1676.67	1671.24	1703.93	1625.35	1704.69	1782.68
813.3	798.67	795.92	781.97	811.59	761.82	853.66
858.67	849.39	846.33	833.69	861.24	815.6	908.8
945.9	942.68	939.11	929.33	951.91	915.15	1009.98
1180.7	1108.86	1134.4	1107.2	1161.24	1142.26	1073.62
1295.5	1219.2	1246.74	1219.75	1274.44	1260.61	1180.95
1675.8	1592.52	1626.63	1603.23	1654.61	1663.12	1543.6
1958	1862.06	1900.8	1882.44	1926.45	1954.91	1804.74
2364.4	2273.19	2318.85	2312.23	2336.09	2400.93	2201.36
2820.4	2724.86	2778.14	2789.15	2779.07	2890.33	2634.09
3053.9	2963.32	3020.68	3043.03	3009.34	3147.94	2860.87
3260.1	2985.5	3044.43	3081.59	2981.42	3150.65	2856.63

Table D.2. Measured and predicted pressures for R-410a/POE assuming a 50/50 mass blend in the vapor and liquid phases.

Measured Pressure (kPa)	Wilson model (kPa)	Heil model (kPa)	T & K model (kPa)	NRTL model (kPa)	UNIQUAC model (kPa)
170.5	NaNQ	NaNQ	NaNQ	NaNQ	NaNQ
306.3	NaNQ	NaNQ	NaNQ	NaNQ	NaNQ
493.7	488.00	489.13	499.90	497.35	532.48
679	692.06	693.52	710.58	707.12	759.84
678.8	692.56	694.02	711.09	707.63	760.39
1295.3	1402.70	1405.05	1451.35	1444.70	1576.70
1673	1891.20	1894.02	1970.31	1961.05	2182.90
158.5	761.69	765.04	782.65	775.41	856.09
553.2	539.50	540.74	553.00	550.18	590.12
984.6	1035.18	1037.15	1066.77	1061.69	1149.47
1016.3	1050.83	1052.83	1083.10	1077.98	1167.58
736.6	721.96	731.31	748.24	726.84	957.57
1337.6	1410.24	1429.50	1469.62	1429.11	2093.82
390.1	374.28	382.00	392.12	370.29	508.76
899.3	904.26	922.09	949.49	902.88	1339.63
1206.3	1227.19	1251.77	1290.12	1229.27	1962.03
1655.5	1735.53	1772.54	1828.82	1744.99	"-NanQ"
291.7	260.02	265.83	270.16	252.68	331.11
420.7	380.21	387.96	394.87	372.41	497.00
602.8	546.25	556.53	567.04	539.18	735.74
796.4	730.75	743.79	757.52	725.51	1013.74
1070.8	997.98	1015.06	1032.31	996.83	1444.23
1409.3	1340.64	1363.43	1379.55	1344.44	2068.99
429.2	400.98	399.08	355.14	390.42	462.96
560.9	532.09	528.86	468.92	523.59	627.04
764.3	739.60	734.23	644.63	735.12	891.63
982	964.89	957.14	830.40	965.65	1185.23
1280	1277.82	1266.91	1078.59	1284.68	1599.98
1638.8	1466.31	1454.80	1208.34	1468.09	1832.57