A Neural Networks Method to Predict Activity Coefficients for Binary Systems Based on Molecular Functional Group Contribution

by

Harry Rodríguez Vallés

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
INDUSTRIAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2006

Approved by:

_____	_____
Luis Antonio Estévez De Vidts, PhD Member, Graduate Committee	Date
_____	_____
William Hernández Rivera, PhD Member, Graduate Committee	Date
_____	_____
Nazario D. Ramirez Beltrán, PhD President, Graduate Committee	Date
_____	_____
David Suleiman, PhD Representative of Graduate Studies	Date
_____	_____
Agustín Rullán, PhD Chairperson of the Department	Date

ABSTRACT

Artificial neural network (ANN) techniques and functional group contributions were used to develop an algorithm to predict chemical activity coefficients. The ANN algorithm was trained using experimental data for more than 900 binary systems obtained from DECHEMA, a phase-equilibrium database. All experimental data binary systems used in this study are isothermal. The prediction scheme is based on the fact that the atoms in a chemical compound can be grouped in a functional group with its own physical and chemical properties. Thus, almost any chemical compound can be built by combining the right number of functional groups. The functional group interactions among the components in a mixture are estimated and the combination of functional group interactions provides the intermolecular relationship among the components of a mixture and consequently the activity coefficients can be predicted. The intramolecular interactions were not considered in this study. The four-suffix Margules equation was used as the base thermodynamic model to calculate the activity coefficients. The Margules equation is good for modeling enthalpic contributions to the activity coefficient but is not good for modeling entropic contributions to the activity coefficient. The design of functional groups based on quantum mechanics was adopted to develop a method for predicting activity coefficients. ANN techniques are especially useful for modeling a highly nonlinear interaction among the functional groups and the corresponding activity coefficient. One of the major contributions of this research is to propose a method to identify the initial point and the structure of an ANN. The minimum mean squared

prediction error criterion was implemented to determine both a suitable initial point and the structure of the ANN. A random search method was used to determine the optimal initial point and the Levenberg-Marquardt algorithm was used to train the ANN to generate a sample of prediction values and the trim mean based on 20% data elimination was selected as the best representation of the ensemble prediction of the Margules equation parameters. The algorithm was validated with nineteen vapor-liquid equilibrium systems and results show that the ANN provides a relative improvement over the UNIFAC method. The scope of this study is limited to some chemical compound families (i.e. alcohols, phenols, aldehydes, ketones and ethers), it is required to include more experimental data to cover additional chemical compound families such as carboxylic acids, anhydrides, esters, aliphatic hydrocarbons and halogens.

RESUMEN

Se utilizaron redes neuronales artificiales y la contribución de grupos funcionales se utilizaron para desarrollar un algoritmo para predecir coeficientes de actividad. El algoritmo de la red neuronal se adiestró utilizando datos experimentales para más de 900 sistemas obtenidos de DECHEMA, una base de datos de equilibrio líquido-vapor. Todos los sistemas binarios de data experimental utilizados son isotermales. El esquema de predicción se basó en el hecho de que los átomos de un compuesto químico se pueden agrupar formando grupos funcionales con propiedades químicas y físicas únicas. Por lo tanto, cualquier compuesto se puede construir a partir de la combinación exacta de grupos funcionales. Las interacciones de grupos funcionales entre los compuestos de una mezcla son estimadas y la combinación de las interacciones de grupos funcionales provee información sobre las relaciones intermoleculares entre los componentes de la mezcla de modo que los coeficientes de actividad se pueden predecir. Las interacciones intramoleculares no fueron consideradas en este estudio. La ecuación de Margules fue utilizada como el modelo termodinámico base para calcular los coeficientes de actividad. La ecuación de Margules es eficaz para modelar las contribuciones entálpicas al coeficiente de actividad pero no es eficaz para modelar las contribuciones entrópicas. Se adoptó un diseño de grupos funcionales basados en mecánica cuántica para desarrollar el método de predecir coeficientes de actividad. Las técnicas de redes neuronales son útiles para modelar las relaciones no lineales entre los grupos funcionales y su correspondiente coeficiente de actividad. Una de las contribuciones de mayor importancia de esta

investigación es el desarrollo de un método para identificar el punto inicial y la estructura de la red neuronal. El criterio del promedio del cuadrado de los errores se utilizó para determinar tanto el punto inicial óptimo como la estructura óptima de la red neuronal. Un método basado en una búsqueda aleatoria se utilizó para determinar el punto inicial óptimo y el algoritmo de Levenberg-Marquardt se utilizó para entrenar la red neuronal generando una muestra de valores de predicción para la cual el promedio de muestra reducida basado en la eliminación de 20% de los datos se seleccionó como la mejor representación de la predicción de los parámetros de la ecuación de Margules. El algoritmo desarrollado se validó con 19 sistemas de equilibrio líquido-vapor y los resultados muestran que el método de redes neuronales provee una mejora notable sobre el método UNIFAC. Este estudio esta limitado a algunas familias de los compuestos químicos (alcoholes, fenoles, aldehídos, cetonas y éteres), es requerido incluir datos experimentales adicionales para cubrir familias de compuesto químicos adicionales tales como ácidos carboxílicos, anhídridos, esteres, hidrocarburos alifáticos y halógenos.

To God and to my family . . .

ACKNOWLEDGEMENTS

First, I want to thank God for giving me the strength and perseverance to complete this study. Also, I would like to thank my family, for their unconditional support, patience, inspiration, and love during this journey. During the development of my graduate studies at the University of Puerto Rico, several persons collaborated directly and indirectly with my research. Without their support it would have been impossible for me to finish my work. That is why I wish to dedicate this section to recognize their support.

I want to start expressing a sincere acknowledgement to my advisor, Dr. Nazario Ramirez-Beltran for giving me the opportunity to do research under his guidance and supervision. I received motivation, encouragement, guidance, and transmitted knowledge needed for the completion of my work and his support throughout my research. With him, I have learned writing technical papers and sharing my ideas to the public. I also want to thank the support I received from Dr. Antonio Estévez De Vidts and his undergraduate students who help me completing the manual data entry from DECHEMA to an Excel spreadsheet. From these persons, I am completely grateful.

Table of Contents

List of Tables

Tables **Page**

List of Figures

Figures **Page**

1 INTRODUCTION

Several empirical models have been developed to estimate the activity coefficients for a mixture and the most accurate approach is the one that is based on experimental vapor-liquid equilibrium data. Margules and van Laar are two of the empirical models that have been developed to estimate the activity coefficient (Smith et al., 1996). Modern activity coefficient models are based on the local-composition concept, which was introduced by Wilson (1964). Due to molecular size and intermolecular forces, local compositions are assumed to take into account the short-range orders and nonrandom molecular orientations inside a liquid solution. The success of Wilson equation on vapor-liquid equilibrium calculations was supported by the development of alternate local compositions models. Two of the most well known models are the Non-Random-Two-Liquid (NRTL) developed by Renon and Prausnitz (1968) and the Universal QUAsi-Chemical (UNIQUAC) developed by Abrams and Prausnitz (1975). These models are capable of correlating experimental activity coefficients for species *i* in a liquid solution over a wide composition and temperature range. They are also capable of interpolating and/or extrapolating the experimental activity coefficients for a wide range of temperatures and compositions based on a few experimental points.

In the absence of experimental data, group contribution methods have been used to predict activity coefficients. In these methods, atoms in a chemical compound are grouped forming functional groups that are assumed to have their own physical and

chemical identity (Fredenslund et al., 1975). Wilson and Deal (1962) introduced the Analytical Solutions of Groups (ASOG) method. Fredenslund et al. (1975) developed the Universal Functional-group Activity Coefficients (UNIFAC) method to predict activity coefficients based on molecular functional groups contribution. UNIFAC is one of the most prominent methods that uses a combinatorial and a residual part with functional groups parameters such as: group volume, group surface area, and binary group interactions to predict the activity coefficients. The UNIFAC's prediction capabilities have been improved by Weldlich and Gmehling (1987) and also by Larsen et al. (1987). They included a modified empirical combinatorial part, temperature dependant group interactions, and additional main groups (Gmehling, 2003).

The group contribution strategy has also been used to estimate pure compounds physical properties. Constantinou and Gani (1994) proposed a new group contribution design for estimating important physical and thermodynamic properties of pure compounds. Essentially, they proposed a general linear regression model that relates the contribution of the functional groups to several physical and thermodynamic properties such as: critical temperature, critical pressure, critical volume, melting point, normal boiling point, standard Gibbs energy, and standard enthalpy. Skander and Chitour (2002, 2003) proposed a group-contribution method to estimate physical properties of hydrocarbons: boiling point, freezing point, and liquid density. They separated experimental data and created the following compound families: *n*-paraffins, isoparaffins, olefins, alkynes, naphthenes, and aromatics. They used a regression equation to extrapolate physical properties of heavier compounds. More recently, Álvarez and

Valderrama (2004) and, Valderrama and Álvarez (2006) have also proposed group-contribution methods to estimate critical properties.

Almost any chemical compound can be built by combining the right number of functional groups. Thus, in this research we have exploited the idea that functional group interactions among the components in a mixture can be estimated and, by adding these interactions appropriately, the activity coefficients can be predicted.

The aforementioned group-contribution methods use linear and nonlinear regression techniques to represent the relations among the variables of a given system. The relationship between the physical and thermodynamic properties is highly non-linear, and consequently an artificial neural network (ANN) can be a suitable alternative to model the underlying thermodynamic properties. ANN is an especially efficient algorithm to approximate any function with finite number of discontinuities by learning the relationships between input and output vectors (Hagan et al., 1996). Thus, an ANN is an appropriate technique to model the nonlinear behavior of chemical properties. Chow et al. (1995) developed a method for estimating aqueous activity coefficients for aromatic organic compounds using ANN. They demonstrated that when choosing the appropriate network parameters the ANN method provided more accurate predictions of the aqueous activity coefficients than the regression analysis approach.

The purpose of this research is to derive a method to predict activity coefficients. The activity coefficient is a dimensionless parameter; its logarithm is a measure of the deviation of the behavior of a component in a mixture from the ideal-solution behavior. The activity coefficients are crucial in the design of separation processes equipment such as distillation and absorption columns (Smith et al., 1996). Activity coefficients are

predicted to develop feasibility studies and preliminary chemical engineering calculations for separation processes equipment design. Prediction results are usually confirmed with experimental studies on the final design stage of a project. It should be noted that the ANN at the earlier design stage prediction of activity coefficients provides a cost effective decision tool. Our proposed activity coefficients prediction tool based on ANN will provide an additional strategy that may reduce the uncertainties of predicting activity coefficients. Thus, the development of the new method is not intended to fully replace the UNIFAC method but to enhance the forecast and minimize business risk.

Chapter 2 deals with the justification for performing this research. A literature review of publications most relevant to this research is included in Chapter 3. A theoretical background to introduce the fundamental concepts and principles for deriving the activity coefficients and the functionality of the neural networks is presented in Chapter 4. The fifth chapter describes the proposed methodology to estimate activity coefficients. Chapter 6 presents a numerical example to illustrate step by step the application of the suggested methodology. Chapter 7 presents a comparison between the ANN and the UNIFAC methods and Chapter 8 presents some conclusions and future work.

2 JUSTIFICATION

Computer developments lead to use computers as a good design tool in all engineering disciplines. Several computers software had been developed during the last two decades to help engineers in performing efficient designs. Those programs help engineers to deeply study their designs and to predict with accuracy its performance by computer simulations before the construction step. Thus, design process is faster and more efficient since computers development because engineers have more capacity to detect design errors and correct them during the design phase.

Numerous computers simulation software are used in Chemical Engineering to design industrial processes equipment such as: chemical reactors, distillation columns, cooling towers, gas scrubbers, and boilers. To perform the necessary calculations for designing those equipments it is necessary to know the chemical, physical and thermodynamic properties of the compounds present in a given process. These properties can be obtained from experimental data or can be estimated from empirical models. Experimental data is the best way to obtain the properties of a chemical compound but it is not always possible because of economical and time limitations. Thus, the analytical and/or empirical models are used to obtain the chemical, physical, and thermodynamic properties of compounds.

The Gibbs free energy is a thermodynamic property that is intimated related to vapor-liquid equilibrium calculations, which are needed for separation processes equipment design. Gibbs free energy also leads to calculations of other thermodynamic

properties such as: internal energy, enthalpy, and entropy. If thermodynamic properties of compounds are known, then it is possible to calculate physical properties such as: boiling point, thermal expansivity, and heat capacity. It should be noticed that the Gibbs free energy is obtained by calculating the activity coefficients, which can be estimated using experimental vapor-liquid equilibrium data or using empirical models. The activity coefficient is a dimensioless variable that measures the deviation of a mixture behavior from ideal. The important role of activity coefficients on thermodynamics is that it is possible to describe the real behavior of a given mixture by using a mathematical model derived for the ideal behavior.

There are computer software with specific algorithms that make the necessary calculations to obtain chemical compounds properties and to design industrial process equipments. However, if the approximate methods produce large deviations from the actual values, then design calculations will be incorrect causing either over specification of the equipment or that the designed equipment will not have the capacity to perform the designed task. That situation leads to waste of time and money.

The low accuracy of the existent models to predict activity coefficients causes the design of industrial processes equipment to be not completely reliable. To obtain reliable design of industrial process equipment it is necessary to perform experiments by making pilot plants and then scale up to an industrial process.

The main purpose of the current work was to develop a method to estimate activity coefficients to improve the accuracy of existent methods. It has been shown that a neural network is an efficient tool to model nonlinear relationships among

thermodynamic variables (Chow et al., 1995). Thus, it is expected that neural networks be an appropriate technique to model the nonlinear behavior of activity coefficients. The proposed method uses molecules functional groups interactions contributions and experimental vapor-liquid equilibrium data to develop a model that could accurately predict activity coefficients when experimental data are not available. An advanced optimization method using neural networks was used to model the nonlinear behavior between functional groups interactions and vapor-liquid equilibrium data.

3 LITERATURE REVIEW

The reference more directly related to this study is the publication of UNIFAC method (Fredenslund et al., 1975). It combines a functional-group-contribution concept with a model for activity coefficients based on an extension of the quasi-chemical theory of liquid mixtures, UNIQUAC. The resulting method, UNIFAC, contains two adjustable parameters per pair of functional groups. By using group interaction parameters obtained from experimental data reduction, activity coefficients for binary mixtures may be predicted. Wu and Sandler (1991) improved the UNIFAC method by using quantum mechanics for functional groups. This study provides a strong theoretical background for the identification of functional groups in molecules.

This study is a combination of the use of the functional-groups concept with neural networks to predict activity coefficients. The idea of using neural networks to predict activity coefficients was taken from a Project called Stability Prediction for Drug Products. That project was sponsored by INDUNIV (Industry-University Research Center) during August 1997 to July 1999. The main purpose of the project was to develop a mathematical method to predict expiration dates of pharmaceutical drug products for both liquids and solids formulations. Prediction of the activity coefficients for the drug components were needed to predict the expiration dates. Here comes the idea of using neural networks and functional-groups concept to predict activity coefficients since usually there are not experimental data available for pharmaceutical drugs.

The proposed method consisted on training a neural network using a functional group coefficient that relates the interaction among the functional groups as input and NRTL interaction constants as outputs. Once trained, the neural network was used to predict the NRTL interaction constants and the NRTL model for multicomponent mixtures to calculate the activity coefficients. However, during this investigation it was found that the four-suffix Margules equation offered a better estimation of the activity coefficients for the 921 experimental vapor-liquid equilibrium data systems included in the database. Therefore, the final proposed method used the four-suffix Margules equation as base thermodynamic model to calculate the activity coefficients.

There are several publications regarding the use of ANN to predict physical and thermodynamic properties of chemical compounds. Petersen et al. (1994) developed a predictive tool for activity coefficients using ANN. However, their work was very limited on the data used for training the ANN. Also, a methodology to determine the optimum network structure was not discussed. A comparison with UNIFAC was performed resulting UNIFAC superior to the ANN method. Bilgin (2004) used an ANN approach to estimate activity coefficients for isobaric binary systems. The approach used was based on giving the low boiling component liquid concentration as input to the ANN and the activity coefficients as output. The training of the ANN was performed with half the system experimental data in order to estimate the complete data set. The results ontained were compared between ANN and UNIFAC and it was concluded that the ANN method was superior in estimating the activity coefficients.

Mitchell and Jurs (1998a) developed a model to predict infinite-dilution activity coefficients (γ^{∞}) for organic compounds under the framework of molecular structure information and ANN. Their model demonstrated to be equivalent to the linear solvation energy relationship (LSER) method reported by Sherman et al. (1996). Mitchell and Jurs (1998b), in a second publication, also developed a model to predict aqueous solubility of organic compounds under the framework of molecular structure and ANN obtaining good results. Rani and Dutt (2002) developed a method to predict infinite-dilution activity coefficients for halocarbons in water and organic compounds in hydrofluoroparaffins using ANN. They obtained results equivalent to the method developed by Mitchell and Jurs (1998a).

Dehghani et al. (2006) developed a method to predict the activity coefficient ratio of electrolytes in solutions containing amino acids using ANN. The root-mean-square deviation for the predicted system was less than 0.01. Urata et al. (2002) developed a method for predicting vapor-liquid equilibrium of binary systems containing hydrofluoroeters using ANN. Even though they have good accuracy in the results, it was found that the predicted activity coefficient might not fulfill the restrictions of the Gibbs-Duhem theorem. Taskinen and Yliruusi (2003) performed a literature review on methods for predicting physicochemical properties (i.e. partition coefficient, water solubility, aqueous activity coefficients and boiling point among others) of organic compounds using ANN. From the studied literature they have found that: most physicochemical properties can be predicted from the molecular structure using ANN modeling, it has not been shown that ANN methods are superior to other methods and that much of the

published work can be characterized as exploration of the area rather than development of serious models with properly validated performance.

ANN have been used in several applications to model complex data systems. Design of a pH control system (Kadirkkamanathan and Regunath, 2001), motor fault diagnosis (Gao and Ovaska, 2001), properties of cast irons (Voracek, 2001), fuzzy proportional-integral-derivative controllers (Golob, 2001), image processing (Koppen and Ruiz-del-Solar, 2001), dynamic systems with time delays (Ramirez-Beltran and Montes, 2002) and planning of heating and cooling plants (Inaoka et al., 2001) are cases in which neural networks were applied demonstrating improvement in comparison to the commonly used methods for process data modeling. It should be mentioned that neural networks had been widely applied on different technology disciplines with successful results. The ability to learn the behavior of the data generated by a system gives neural networks its versatility.

4 THEORETICAL BACKGROUND

To properly describe the proposed methodology, a fundamental background related to the theory of activity coefficients and neural network algorithms are provided.

4.1 Activity Coefficients

Given a closed system, the activity coefficient is a function of the temperature, pressure, and mole fraction of the compounds in the mixture. Smith et al. (1996) pointed out that, for the case of non-ideal solutions, the actual thermodynamic properties are obtained by measuring the energy deviation from the ideal solution, which is called the excess property. Thus, if M represents a molar thermodynamic property, i.e., enthalpy (H), entropy (S), internal energy (U), Gibbs free energy (G), and Helmholtz energy (A), the excess property, M^E, is defined as the difference between the actual property, M, and the ideal-solution property, M^{id}, at the same temperature, pressure, and composition. That is,

$$M^E = M - M^{id} \tag{1}$$

Usually, when the activity coefficient is studied, the major property of interest is the excess Gibbs free energy, which can be expressed as follows:

$$G^E = G - G^{id} \tag{2}$$

where, G^E, G, and G^{id} are the excess, the actual, and the ideal-solution Gibbs free energy, respectively. After some mathematical manipulations, the molar partial excess Gibbs free energy for the i^{th} compound can be written as follows:

$$\overline{G}_i^E = RT \ \ln(\gamma_i) \tag{3}$$

and

$$\gamma_i = \frac{\hat{f}_i}{x_i f_i} \ , \tag{4}$$

where, T is the temperature of the mixture, R is the universal gas constant, x_i is the liquid mole fraction for the i^{th} compound, f_i is the fugacity of the i^{th} pure compound, and $\hat{f}_i$ is the actual fugacity of the i^{th} compound in the mixture. Fugacity is a property that measures the degree of freedom of a molecule moving inside of a given compound (Smith et al., 1996). The variable, γ_i, is called the activity coefficient for the i^{th} compound. The activity coefficient is equal to 1 for a pure component. Thus, when the activity coefficient of a compound in a mixture is larger than one, it indicates that its molecules have a larger degree of freedom for moving around comparing with a pure compound. On the other hand, an activity coefficient less than one indicates that the molecules in the mixture are closer to each other due to intermolecular attraction and has a smaller tendency to escape by vaporizing than when comparing to a pure compound (Levine, 1988).

The logarithm of the activity coefficient is a partial property with respect to $\overline{G}_i^E / RT$ according to the following relation (Smith et al., 1996):

$$\ln \gamma_i = \left[\frac{\partial \left(nG^E / RT \right)}{\partial n_i} \right]_{P,T,n_j} \tag{5}$$

where, n is the total number of moles in the mixture, n_i is the number of moles of component i, and n_j is the number of moles of the other compounds in the mixture. Also, the activity coefficient must fulfill the expression:

$$\frac{\overline{G}_i^E}{RT} = \sum_i x_i \ln \gamma_i \qquad \textbf{(6)}$$

and the Gibbs/Duhem equation:

$$\sum_i x_i d \ln \gamma_i = 0 \qquad (const\ T, P) \qquad \textbf{(7)}$$

For a binary system the Gibbs/Duhem equation can be expressed as follows:

$$x_1 \frac{d \ln \gamma_1}{dx_1} + x_2 \frac{d \ln \gamma_2}{dx_1} = 0 \qquad (const\ T, P) \qquad \textbf{(8)}$$

The Gibbs/Duhem equation has the following effects on the activity coefficient behavior with respect to the liquid mole fraction:

- At every composition, the slope of γ_1 curve is opposite in sign to the slope of γ_2 curve.
- When $x_1 \rightarrow 1$, the slope of γ_1 curve is zero and $\gamma_1 \rightarrow 1$ and when $x_2 \rightarrow 1$, the slope of γ_2 curve is zero and $\gamma_2 \rightarrow 1$.

Estimation of the activity coefficient can be obtained using experimental vapor-liquid equilibrium data and selecting a suitable thermodynamic model. While $\overline{G}_i^E / RT$ is a function of T, P and mole fraction, for liquid solutions at low to moderate pressures it is a weak function of P and the pressure dependence of activity coefficients can be neglected (Smith et al., 1996). Therefore, if an empirical mathematical expression is

given for $\overline{G}_i^E / RT$, then equation (5) can be used to derive an empirical model for the activity coefficients and equation (8) can be used to verify the model consistency with the Gibbs/Duhem theorem. Among the most important thermodynamic models to estimate activity coefficients based on experimental data are the following: Margules, van Laar, Wilson, Non-Random Two Liquids (NRTL), and UNIversal QUAsi-Chemical (UNIQUAC).

For instance, this study used the four-suffix Margules equation as base method for the activity coefficients calculation. The expression for $\overline{G}_i^E / RT$ for the four-suffix Margules equation is written as follows (Reid et al., 1987):

$$\frac{G^E}{RT} = x_1 x_2 \left[A + B(x_1 - x_2) + C(x_1 - x_2)^2 \right] \quad \textbf{(9)}$$

Using equation (5), the expressions for $\ln \gamma_1$ and $\ln \gamma_2$ are:

$$\ln \gamma_1 = (A + 3B + 5C)x_2^2 - 4(B + 4C)x_2^3 + 12Cx_2^4 \quad \textbf{(10)}$$

and

$$\ln \gamma_2 = (A - 3B + 5C)x_1^2 + 4(B - 4C)x_1^3 + 12Cx_1^4 \quad \textbf{(11)}$$

Thermodynamic models such as Margules, Wilson and van Laar can be used to estimate the activity coefficients. Thus, it is required to obtain experimental data for the system of interest to determine the model parameters and generate the analytical expression, based on the chosen model, for calculating the activity coefficients. However, when experimental data are not available, a prediction method is needed to obtain the activity coefficients. The available prediction methods for activity coefficients are based on functional groups contribution and the most well known are the Analytical Solution of

Groups (ASOG) method (Wilson and Deal, 1962) and the Universal Functional-group Activity Coefficients (UNIFAC) method (Fredenslund et al., 1975). These methods express the activity coefficient as a function of the liquid mole fraction in addition of some physical parameters.

UNIFAC is an effective method for predicting the activity coefficients of various chemical mixtures. The criterion for forming functional groups used in the UNIFAC model is based mostly on molecular geometry. In this research, we proposed a thermodynamic model to predict the activity coefficients using a neural networks technique and functional groups based on quantum mechanics calculations introduced by Wu and Sandler (1991). The list of the functional groups is presented in Appendix B.

There are two fundamental conditions for identifying the functional groups in a given mixture. The first condition states that the group should be independent of the molecule in which it appears and should always have the same geometry. The second condition states that each atom in a group and the group as a whole should have approximately the same net charge independently of the molecule that it appears. This is a very important requirement because it is considered that the intermolecular interaction energy between two groups on different molecules come mainly from electrostatic forces (due to charges on the groups), repulsion forces, van der Waals forces, London (dispersion) forces, and hydrogen-bonding forces.

4.2 Neural Networks

Artificial neural network methodology is proposed to model the linear or nonlinear relationships among the functional groups interactions and the activity coefficients. The ANN model technique can be viewed as a general nonlinear modeling approach. Artificial neural networks are a biological inspiration based on various characteristics of the brain functionality. Artificial neurons are simple computational devices that are highly interconnected and the connections between neurons determine the transfer function of the network (Hagan et al., 1996). An artificial neural network determines an empirical relationship between the inputs and outputs of a given system. Where the inputs of the system are the independent variables (*x*'s) and the outputs are the dependent variables (*y*'s). Therefore, it is important for the user to have a good understanding of the science behind the underlying system to provide the appropriate input and, consequently, to support the identified relationship.

Figure 1 shows the selected structure of the implemented neural network for this study. This figure shows that the identified neural network has three neurons in the hidden layer and three neurons in the output layer with a log-sigmoidal and linear transfer functions in the hidden and the output layers, respectively. Hagan et al. (1996) provides a detailed discussion on the structure and functionality of neural networks. The input *P* matrix to train the neural network is formed by physical parameters and functional groups interactions information of a given binary mixture system. The variables *w's* are the weight matrices, *b's* are the bias vectors, *n's* are the net input, and *a* is the output of the

neural network and represents the predicted Margules equation parameters. An efficient procedure to identify the appropriate structure of a neural network is given by Ramirez-Beltran and Montes (2002).

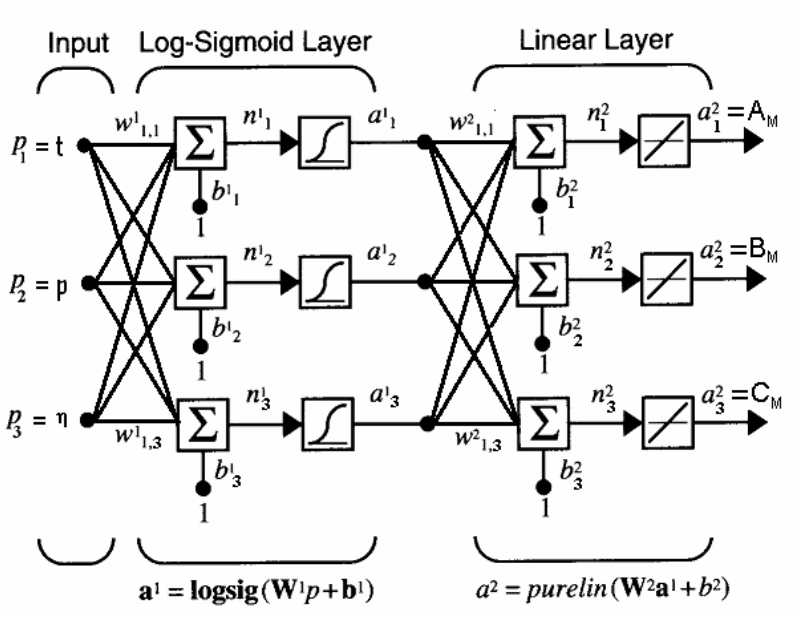


Figure 1. The Neural Network Structure

The input vector (left), the hidden layer (center) and the output layer (right) are the main components of the selected neural network. *t and p* are the temperature and the pressure of the system and η is the interaction proportion, *w's* are the weights, *b's* are the bias *n* is the net input and a's is the output of the network, which are the Margules equation parameters.

Once the structure of the neural network is identified, the next step is to use an optimization technique to estimate the weights and biases in such a way that the output of the network is as close as possible to the target values. In this case the target values are the Margules equation parameters calculated from the experimental data. This optimization strategy is known as the training process and the neural network will learn

the relation between the inputs and the outputs of a dynamic system. Thus, once the neural network is trained the optimal weights are saved with the purpose of predicting the Margules equation parameters.

There are various learning algorithms to train neural networks. Some of the most useful are: Perceptron, Hebbian, Widrow-Hoff, and Backpropagation; the latter is a suitable algorithm for this research because it has the capability of training multilayer neural networks for function approximation (Hagan et al., 1996). However, the original Backpropagation algorithm has several limitations and its major drawback is its low convergence rate for most practical applications. Various modifications of the backpropagation algorithm have been developed and most of them are based on heuristic and numerical optimization strategies.

The heuristic strategies are: the momentum and the variable learning rate. The method of momentum is a heuristic algorithm that improves convergence by using a low-pass filter to smooth out the oscillations of the surface response.

$$y(k) = \gamma y(k-1) + (1-\gamma) w(k) \tag{12}$$

where, $w(k)$ is the input to the filter, $y(k)$ is the output of the filter and the momentum coefficient γ must satisfy: $0 \le \gamma < 1$. The weight and biases are modified according to the following equations that represent the modification to the original backpropagation:

$$\Delta W^m(k) = \gamma \Delta W^m(k-1) - (1-\gamma)\alpha \frac{\partial \hat{F}}{\partial n^m} \left(a^{m-1}\right)^T \quad \text{for weights} \tag{13}$$

and

$$\Delta b^m(k) = \gamma \Delta b^m(k-1) - (1-\gamma)\alpha \frac{\partial \hat{F}}{\partial n^m} \quad \text{for biases} \tag{14}$$

where, k is the iteration number, a^{m-1} is the output of the previous network layer, $\hat{F}$ is the performance index and corresponds to the mean square error of the network, and α is the learning rate.

The key idea of the variable learning rate is to modify the weights and biases update by changing the momentum and learning rate, based on the behavior of the squared errors. There are many variations of the variable learning rate algorithm (Hagan et al., 1996). However, the general rules of the variable learning rate backpropagation algorithm are:

1) If the squared error increases by more than a set percentage ζ after a weight update, then the weight update is discarded, the learning rate is multiplied by a factor $0 < \rho < 1$, and the momentum coefficient γ is set to zero.

2) If the squared error decreases after a weight update, then the weight update is accepted and the learning rate is multiplied by another factor $\eta > 1$. If the momentum coefficient γ has been previously set to zero, it is reset to its original value.

3) If the squared error increases by less than ζ, then the weight update is accepted but the learning rate and the momentum coefficient are unchanged.

The heuristic techniques provide significant improvements over the conventional backpropagation algorithm; however, they also exhibit the disadvantage that a trial an error approach must be implemented to identify the appropriate set of training parameter. The improved techniques based on optimization algorithms are more promising since an automatic learning-rate adjustment can be designed. The most prominent techniques are the conjugate gradient (CG) and the Levenberg-Marquardt (LM) algorithms. The major advantage of the CG algorithm is that it does not require calculation of the second derivatives and yet it still has the quadratic convergence property, i.e., convergence is accomplished as if the surface functions were a quadratic function. The CG algorithm applied to neural networks training consists on the following steps (Hagan et al., 1996):

1) **Interval location:** The interval location step is used to find an initial interval that contains a local minimum. First, the performance index of the network is evaluated at an initial point. Then the performance index is evaluated at a second point, which is at a distance ε from the initial point along the first search direction p_0. The search direction can be obtained from the steepest descent direction obtained from the backpropagation algorithm. The performance index is successively evaluated doubling the distance between the points until the function increases after two consecutive evaluations (see Figure 2). The last three evaluations (points *c*, *d*, and *e*) will bracket the local minimum point defining the initial interval.

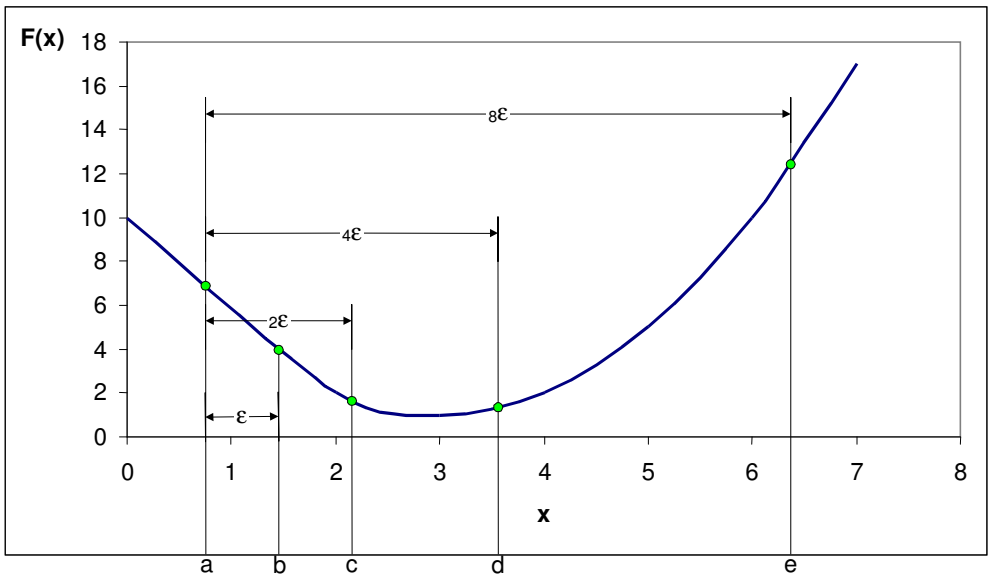


Figure 2. Interval location

2) **Interval reduction:** This step reduces the size of the initial interval until the local minimum is reached to a desired accuracy. The extreme points of the initial interval become points *a* and *b* respectively. The performance index is evaluated at two points (points *c* and *d*) within the initial interval to reduce its size (see Figure 3). A method called the Golden Section Search is used to determine the location of the internal points *c* and *d*. The Golden Section search will reduce the interval until reaching the local minimum point with certain accuracy. The algorithm for the Golden Section Search is as follow.

For $\tau = 0.618$,

Set $c_1 = a_1 + (1 - \tau)(b_1 - a_1)$, $F_c = F(c_1)$

$d_1 = b_1 - (1 - \tau)(b_1 - a_1)$, $F_d = F(d_1)$

For $k = 1, 2, \ldots$ repeat

If $F_c < F_d$ then

Set $a_{k+1} = a_k$; $b_{k+1} = d_k$; $d_{k+1} = c_k$

$c_{k+1} = a_{k+1} + (1 - \tau)(b_{k+1} - a_{k+1})$

$F_d = F_c$; $F_c = F(c_{k+1})$

else

Set $a_{k+1} = c_k$; $b_{k+1} = b_k$; $c_{k+1} = d_k$

$d_{k+1} = b_{k+1} - (1 - \tau)(b_{k+1} - a_{k+1})$

$F_c = F_d$; $F_d = F(d_{k+1})$

end

end until $b_{k+1} - a_{k+1} < tol$

where *tol* is the accuracy tolerance set by the user.

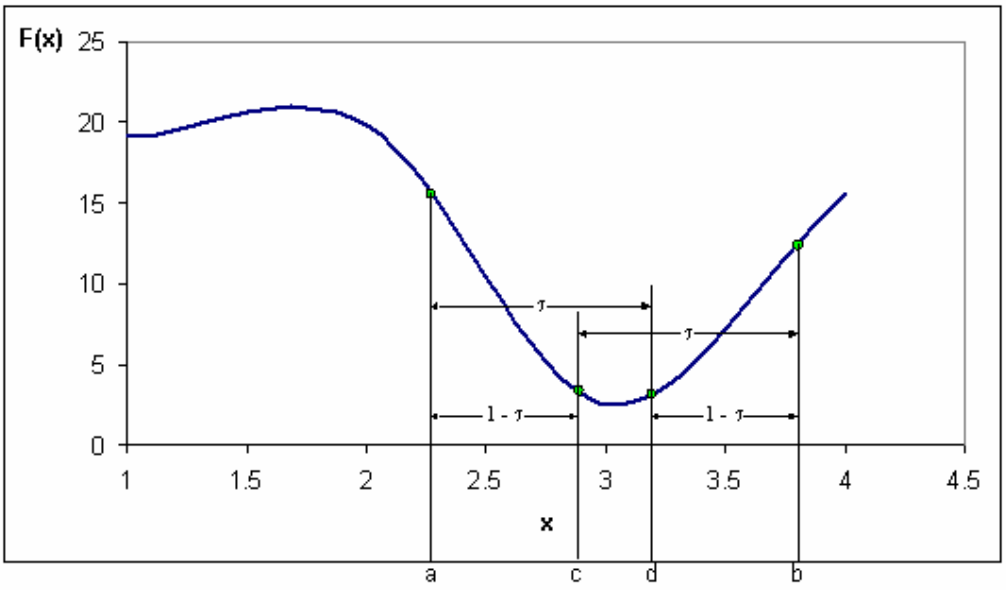


Figure 3. Interval Size Reduction

3) **New search direction**: This step determines a new search direction to continue with the neural network training until reaching the performance index function minimum. The next gradient is calculated with the backpropagation algorithm and the conjugate gradient algorithm is used to update the weights.

4) **Convergence:** The procedure will continue until reaching convergence where the difference between the network response and the targets reaches an acceptable level.

The LM algorithm introduces an automatic calculation of a constant that ensure that the Jacobian matrix will be a positive definite matrix and consequently convergence is accomplished. This method was designed to minimize functions that are sums of squared errors of other nonlinear functions. The LM algorithm steps are as follow (Hagan et al., 1996):

1. Using the inputs and the initial point to the network, calculate the corresponding network outputs. The inputs are the chosen parameters for the system beign analyzed and the initial point consists of the initial values, randomly selected, of the weights and biases. Calculate the errors using the network outputs and the targets values and, calculate the sum of squared errors (*sse*) for all inputs.

$$v^T = \begin{bmatrix} v_1 & v_2 & \cdots & v_N \end{bmatrix} = \begin{bmatrix} e_{1,1} & e_{2,1} & \cdots & e_{S^M,1} & e_{1,2} & \cdots & e_{S^M,Q} \end{bmatrix} \quad \textbf{(15)}$$

$$x^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} w^1_{1,1} & w^1_{1,2} & \cdots & w^1_{S^1,R} & b^1_1 & \cdots & b^1_{S^1} & w^2_{1,1} & \cdots & b^M_{S^M} \end{bmatrix} \quad \textbf{(16)}$$

$$a_q^{(m)} = f^{(m)}\left(w^{(m)} a_q^{(m-1)} + b_q^{(m)}\right) \tag{17}$$

$$e_q = t_q - a_q^{(M)} \tag{18}$$

$$sse(k) = \sum_{i=1}^{S^{(M)}} \sum_{q=1}^{Q} e_{i,q}^2(k) \tag{19}$$

2. Calculate the Jacobian matrix.

$$J(x) = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_{1,1}^1} & \frac{\partial e_{1,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w_{1,1}^1} & \frac{\partial e_{2,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{2,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{2,1}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w_{1,1}^1} & \frac{\partial e_{S^M,1}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{S^M,1}}{\partial w_{S^1,R}^1} & \frac{\partial e_{S^M,1}}{\partial b_1^1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w_{1,1}^1} & \frac{\partial e_{1,2}}{\partial w_{1,2}^1} & \cdots & \frac{\partial e_{1,2}}{\partial w_{S^1,R}^1} & \frac{\partial e_{1,2}}{\partial b_1^1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix} \tag{20}$$

The elements of the Jacobian matrix are calculated using the following equations:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = \tilde{s}_{i,h}^m \times a_{j,q}^{m-1} \tag{21}$$

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{s}_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = \tilde{s}_{i,h}^m \tag{22}$$

$$\tilde{s}_{i,h}^m \equiv \frac{\partial v_h}{\partial n_{i,q}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \tag{23}$$

3. Calculate the new weights and biases.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left[\mathbf{J}^{\mathbf{T}}(x_k)\mathbf{J}(x_k) + \mu_k \mathbf{I}\right]^{-1} \mathbf{J}^{\mathbf{T}}(x_k) v(x_k) \tag{24}$$

4. Re-calculate the sum of squared errors with the values for the weights and biases. If the new *sse* is smaller that those calculated in step 1, then decrease the scalar μ, let the weights and biases become the new values and go back to step 1. Otherwise, increase the scalar μ and go back to step 3.
5. The LM algorithm will converge when the gradient is less than a pre-determined value or if the *sse* are reduced to an error goal.

One of the limitations of this method is the large amount of memory required for performing calculations. However, for the underlying application the LM algorithm becomes a suitable technique since a small data set is used to train the neural network. It should be noted that only the systems that contain interactions of the system of interest are the ones that will be used to predict the activity coefficient and consequently the required memory is reduced. In this research, heuristics and optimizations techniques were tested and it was determined that for the selected training patterns the best performance was obtained by the Levenberg-Marquardt algorithm. The training patterns studied are presented in Table 1.

The system chloroform (1) + n-butyl alcohol (Ohe, 1989) was selected to test the performance of each training pattern shown in Table 1. The activity coefficients were predicted three times using the ANN method and the overall mean absolute error for γ_1 and γ_2 was calculated for each training algorithm. It can be shown that the Levenberg-Marquardt algorithm gave the smaller overall mean absolute error and standard deviation for predicting γ_1. On the other hand, while the Levenberg-Marquardt algorithm did not

give the minimum overall mean absolute error for predicting γ_2 it gave the smaller standard deviation. Thus, the Levenberg-Marquardt algorithm offers the best performance in training the artificial neural network for predicting activity coefficients (see Table 2).

Table 1. ANN Training Algorithms

MatLab Command	**Training Algorithm Description**
traingdm	Batch Gradient Descent with Momentum
traingda	Gradient descent with adaptive learning rate backpropagation
traingdx	Gradient descent with momentum and adaptive learning rate backpropagation
traincgf	Conjugate gradient backpropagation with Fletcher-Reeves updates
traincgb	Conjugate gradient backpropagation with Powell-Beale restarts
traincgp	Conjugate gradient backpropagation with Polak-Ribiere updates
trainscg	Scaled conjugate gradient backpropagation
trainoss	One step secant backpropagation
trainbfg	BFGS quasi-Newton backpropagation
trainrp	Resilient backpropagation
trainlm	Levenberg-Marquardt backpropagation

Table 2. Overall Mean Absolute Error by ANN Training Algorithm

Algorithm	**MAE for γ_1**	**STDEV for γ_1**	**MAE for γ_2**	**STDEV for γ_2**
traingdm	0.1491	0.1067	0.5057	0.4829
traingda	0.1702	0.1643	0.3372	0.5224
traingdx	0.1311	0.0954	0.3046	0.4882
traincgf	0.1541	0.1217	0.2920	0.3735
traincgb	0.1792	0.1357	0.2217	0.3208
traincgp	0.1361	0.1715	0.2042	0.3695
trainscg	0.1595	0.1325	0.2064	0.3529
trainoss	0.0897	0.0826	0.3033	0.3388
trainbfg	0.0491	0.0353	0.2371	0.3143
trainrp	0.1424	0.1225	0.2724	0.3284
trainlm	0.0194	0.0147	0.2815	0.2984

It should be noted that careful attention must be devoted to the application of neural networks, since an automatic application of the neural network may lead to misleading predictions and exaggeratedly large computational time. Thus, a contribution of this research is to improve the method of identifying the appropriate neural network structure. The mean squared prediction error criterion is proposed in this study to ensure that the identification of a network structure provides a reliable prediction scheme. In addition, this method uses the ensemble prediction concept that consists on generating members of ensemble by using different initial conditions and selecting the best representation of the population central tendency as the ensemble prediction.

5 METHODOLOGY

The conventional approach to obtain the activity coefficients in liquid solutions requires vapor-liquid equilibrium experiments. The main purpose of this research is to predict the activity coefficients without experimental data, since in practice experimentation is extremely expensive and time consuming. The proposed approach is possible because there is a large experimental equilibrium data set that can be used to train a neural network to recognize the relation among functional groups interactions and the Margules equation parameters (Sorensen and Arlt, 1979).

5.1 ANN Method Description

The entertained methodology in this research is based on decomposing the chemical compounds of a mixture into its functional groups, identifying the required interactions among functional groups, and training a neural network to predict the activity coefficients (see Figure 4). The four-suffix Margules equation has been used as base method for calculating the activity coefficients. Two MatLab programs were developed to perform the tasks required to apply this methodology. A description of the programs and the MatLab code is included in Appendix C. Figure 4 shows the steps of this methodology.

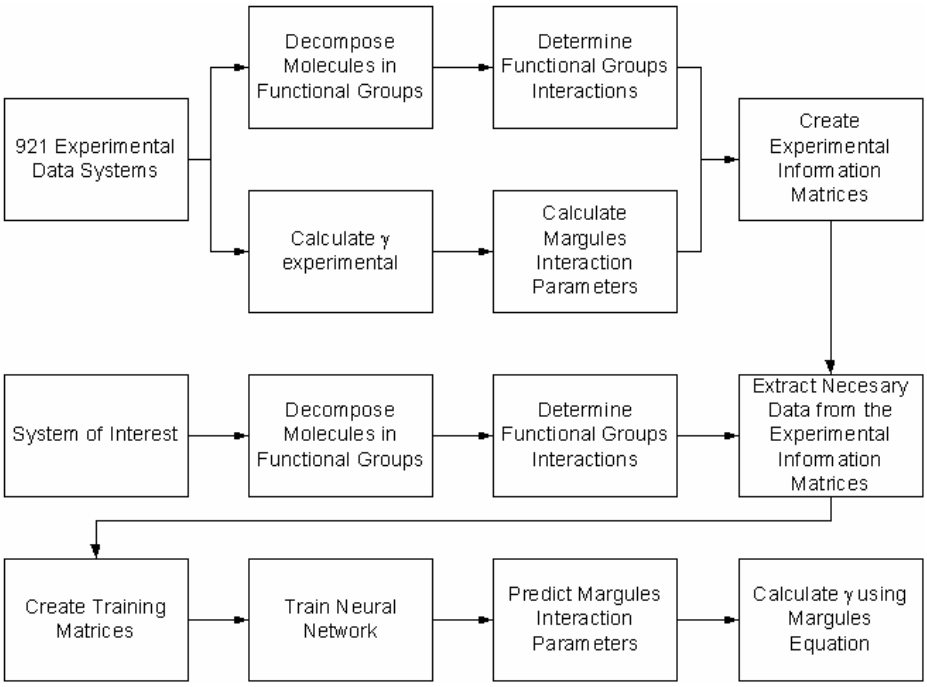


Figure 4. Methodology for Activity Coefficients Prediction with ANN

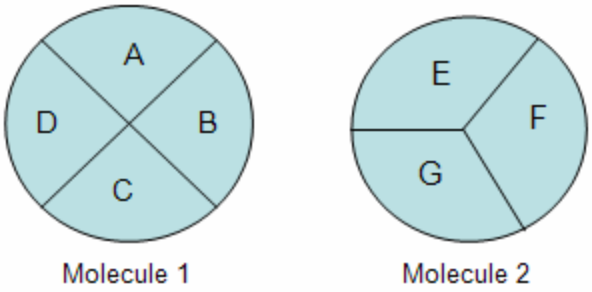
Step 1: Experimental Data

The vapor-liquid equilibrium data used in this study are limited to the systems of binary mixtures containing several equilibrium data points. Nine hundred and twenty one (921) binary systems were used to create the information matrix and train a neural network. The sample size (921) was limited because of the availability of experimental data systems and because of required computational time. The experimental data were entered into a spreadsheet to be used by the MatLab program.

Step 2: Decompose Molecules in Functional Groups and Determine Functional Groups Interactions

The functional groups used in this work were introduced by Wu and Sandler (1991) and are based on quantum mechanics calculations. The list of the functional

groups is presented in Appendix B. Figure 5 shows a symbolic representation of two molecules. It is assumed that the first and second molecules contain 4 and 3 functional groups, respectively. Figure 5 also shows all possible interactions ($2 \times 4 \times 3=24$) among the functional groups.



Interactions 1-2		Interactions 2-1	
AE	CE	EA	FC
AF	CF	EB	FD
AG	CG	EC	GA
BE	DE	ED	GB
BF	DF	FA	GC
BG	DG	FB	GD

Figure 5. Functional groups decomposition and interactions

Molecule 1 is composed of 4 functional groups and molecule 2 is composed of 3 functional groups. The total amount of possible functional groups interactions is 12 for each molecule.

Step 3: Calculate the Experimental Activity Coefficients and the Margules Constants

The experimental activity coefficients are calculated from the vapor-liquid experimental data using the modified Raoult's law and the Antoine equation (Smith et al., 1996). That is, for the i^{th} component in the solution:

$$y_i P = x_i \gamma_i P_i^{sat} \quad \textbf{and} \quad \log P_i^{sat} = A' - \frac{B'}{t - C'} \qquad \textbf{(25)}$$

where, y_i is the vapor-phase mole fraction, P is the pressure of the system, x_i is the liquid-phase mole fraction, γ_i is the activity coefficient, P_i^{sat} is the vapor pressure in mm Hg, *A', B'* and *C'* are the Antoine constants and *t* is the temperature of the system in degree Celsius.

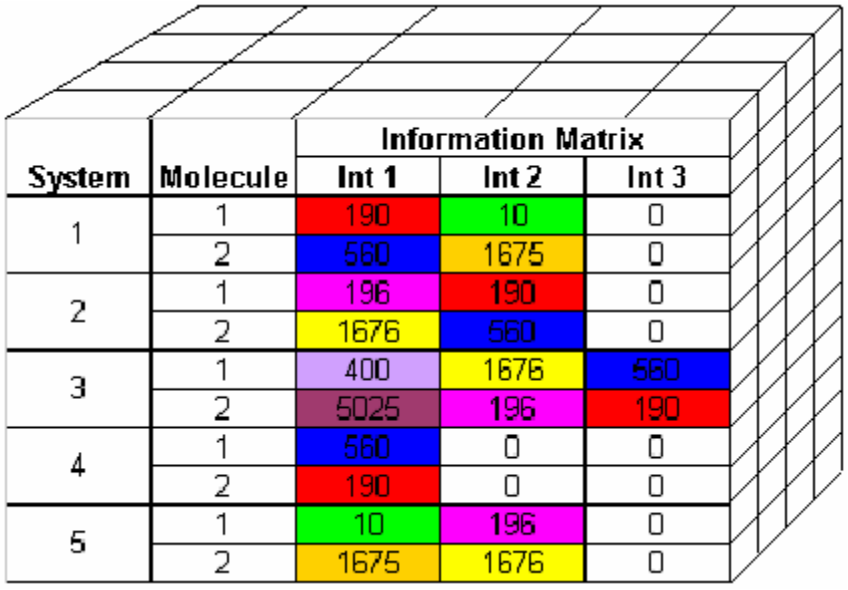
The Margules constants were estimated for each of the experimental data systems using a quasi-Newton method. The objective function consisted on minimizing the sum of squared errors (SSE). The problem formulation was to find the Margules parameters such that:

$$\min SSE(\hat{\gamma}_i) = \sum_{i=1}^{m} \left(\hat{\gamma}_i - \gamma_i\right)^2 \qquad \textbf{(26)}$$

where γ_i is the experimental value and $\hat{\gamma}_i$ is the predicted value calculated using the Margules equations (10) and (11).

Step 4: Create the Experimental Data Information Matrix

The vapor-liquid equilibrium experimental data for each system was organized in columns and rows within a three dimensional matrix called the experimental data information matrix. The data in each row of the experimental data information matrix belong to one of the compounds of a binary system. Thus, every two rows of data belong to the experimental data of one binary system. The columns contain the information that belongs to the functional groups interactions. The columns are expanded tri-dimensionally into eight columns of data each containing the information of a functional group interaction (See Figure 6).



System	Molecule	Information Matrix		
		Int 1	Int 2	Int 3
1	1	190	10	0
	2	560	1675	0
2	1	196	190	0
	2	1676	560	0
3	1	400	1676	560
	2	5025	196	190
4	1	560	0	0
	2	190	0	0
5	1	10	196	0
	2	1675	1676	0

Figure 6. Experimental Data Information Matrix

Each interaction column represents eight columns with experimental data information. Note that the interactions are not sequentially ordered since the program position the information as it appears in the experimental data.

The first column of a group of eight columns within the experimental data information matrix contains the chemical interaction number between the h and k functional groups. The interaction number ($N_{h,k}$) is computed by using the following equation:

$$N_{h,k} = g(h-1)+k \tag{27}$$

where, g is the total number of selected functional groups, h and k are the numbers associated to the first and the second functional groups, respectively. For instance, this study includes a total of 186 functional groups and the interaction number between the second and fourth functional groups can be computed as follows:

$$N_{2,4} = 186(2-1) + 4 = 190$$

In a similar fashion, the interaction number between the fourth and second functional group would be:

$$N_{4,2} = 186(4-1) + 2 = 560$$

The remaining seven columns from a columns group are used to save the following parameters: system temperature, system pressure, functional groups quantities, functional groups molecular weights, and functional groups mole fractions. Thus, any interaction between a functional group and itself is not included in the neural network training because it is expected to follow an ideal behavior and thus would have no contribution to the activity coefficient.

Step 5: System of Interest

The data from the system of interest is entered into a spreadsheet, similar to that used for the experimental data, to be used by the MatLab program.

Step 6: Determine the Functional Groups and Functional Groups Interactions for the System of Interest

The molecules from the system of interest are decomposed into their respective functional groups and the functional group interactions are determined as described in step 2. The data from the system of interest are arranged into an information matrix the same way as the experimental data.

Step 5: Extract Necessary Data from the Experimental Data Matrix

The next step is to extract the required data from the experimental data information matrix. Searching techniques are used to identify the columns of the

experimental data information matrix that contains any of the functional group interactions of interest to create the training patterns. The selected information is called the reduced experimental data information matrix and is used to train a neural network and finally the optimal weights are used to predict the Margules equation parameters. Then, the four-suffix Margules equation is used to calculate the activity coefficients.

It should be noted that the proposed approach develops a model using only the required interactions with the purpose of reducing the computational effort and increase prediction capability.

Step 6: Create Training Matrices

The training patterns of the neural networks are formed by the input and output vectors. In this study, the input vector, ***P***, includes the following variables: system temperature, system pressure, and the proportions of the functional groups interactions. The target, ***T***, of this prediction scheme contains the observed Margules equation parameters that are computed using experimental data. Thus the training patterns can be expressed as follows:

$$P = \left[t, p, \eta^{i}_{1,1} \cdots \eta^{i}_{h,k} \cdots \eta^{i}_{H,K}\right] \quad \textbf{and} \quad T = \left[A_M, B_M, C_M\right] \qquad i = 1,2,...,m \qquad \textbf{(28)}$$

where, t and p are the temperature and pressure of the system, respectively; m is the total number of rows of the reduced experimental data information matrix, the upper and lower supercripts corresponds to the i^{th} row, and the subcripts corresponds to the h^{th} functional group, and the k^{th} functional group, respectively. The functional group h belongs to the compound in the i^{th} row and the functional group k belongs to other compound in that mixture. H and K are the total number of functional groups, respectively in the first and

second compounds of the mixture. The $\eta_{h,k}^{i}$ is the amount of interactions between a functional group *h* and *k* divided by the total amount of possible interactions between the functional groups present in the mixture.

$$\eta_{h,k}^{i} = \frac{q_h^i \times q_k^j}{\sum_{h=1}^{H}\sum_{k=1}^{K} q_h^i \times q_k^j + \sum_{k=1}^{K}\sum_{h=1}^{H} q_k^j \times q_h^i} \quad \textbf{(29)}$$

where, q_h^i is the amount of functional groups in the i[th] molecule and q_k^j is the amount of functional groups in the j[th] molecule.

Step 7: Train the Neural Network

A systematic method is introduced to identify the neural network structure for modeling the non-linear behavior between the functional groups interactions and the Margules equation parameters. The neural networks with more than two layers were discarded since those networks over-parameterize the learning process causing reduction of skill prediction capability. On the other hand, a neural network composed with a single layer was not used because it does not have the capability of modeling nonlinear relationships. Therefore, the selected general structure included two layers, the hidden and the output layer (Figure 1).

A searching procedure was implemented to identify the minimum number of neurons in the hidden layer and the type of transfer function that minimizes the mean squared prediction error. The suggested identification procedure consists on selecting a transfer function and then varying the number of neurons in the hidden layer while measuring prediction accuracy by the mean squared prediction errors (*mspe*). This

process was repeated for three different transfer functions and varying the number of neuron from 1 to 5 in the hidden layer. Finally, the structure of the neural network is selected such that the *mspe* is minimized. It should be noted that the prediction was done with data that were not used for training. To perform this identification procedure, the data were split in two equal parts. The first part of the data was used to perform training and the second part was used to predict the Margules parameters. The mean squared prediction errors were calculated between the known experimental Margules parameters and the predicted values with the neural network. The mean squared prediction errors were computed as follows:

$$mspe = \frac{1}{9}\sum_{i=1}^{m}\left[\left(\hat{A}_{M\,i} - A_{M\,i}\right) + \left(\hat{B}_{M\,i} - B_{M\,i}\right) + \left(\hat{C}_{M\,i} - C_{M\,i}\right)\right]^2 \qquad \textbf{(30)}$$

where, $\hat{A}_{M\,i}$, $\hat{B}_{M\,i}$ *and* $\hat{C}_{M\,i}$ are the predicted Margules parameters, $A_{M\,i}$, $B_{M\,i}$ *and* $C_{M\,i}$ are the observed Margules parameters and were obtained from the experimental data, and *m* is the number of data rows of the reduced experimental data information matrix used to predict the Margules parameters.

Three different transfer functions (TF) combinations for the neural network structure were explored in this research:

- purelin-purelin combination: a linear TF used in both, the hidden and output layers.
- logsig-purelin combination: a log-sigmoid TF in the hidden layer and a linear TF in the output layer.

- tansig-purelin combination: a hyperbolic tangent TF in the hidden layer and a linear TF in the output layer.

The mathematical representation of the three different TF is the following:

$$\textbf{Linear function:} \quad f_1(n) = n = wp + b \tag{31}$$

$$\textbf{Log-sigmoidal function:} \quad f_2(n) = \frac{1}{1+e^{-n}} \tag{32}$$

$$\textbf{Hyperbolic tangent function:} \quad f_3(n) = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{33}$$

where, n represents the net input into the network, w is the weight matrix, and b is the bias vector.

Thus, the structures of the neural network change depending on the natural relationship of the available data. The minimum *mspe* criterion was used to identify the appropriate transfer functions and the number of neurons in the hidden layer. The best performance was accomplished with a log-sigmoidal TF in the hidden layer, and a linear TF in the output layer. The number of neurons in the hidden layer varies for every case, depending on the *mspe* values. Once the optimal neural network is determined the network is trained with the LM algorithm using the training patterns from the reduced experimental data information matrix.

Step 8: Predict the Margules Equation Parameters

The input vector for prediction is constructed with the system of interest data following equation (28). Then, the next step is to predict the Margules parameters with the trained ANN and the input vector constructed with the system of interest data. It was observed that, even though the TF and the optimum number of neurons were determined, the

outputs of the ANN provide different prediction values. This is due to the fact that the selected initial point to train the ANN is different every time the training is performed and consequently it may converge to a different local minimum. Thus, the ensemble prediction strategy was used to derive consistent prediction values. A searching process was designed to select the initial point that minimizes the *mspe* and perform prediction. The neural network was trained nine times with a different initial point and each time a new member of ensemble prediction was obtained. Finally, the Margules equation parameters were obtained by computing the trim mean based on 20% data elimination from the ensemble members. In this study just nine members of the ensemble population were generated because of computational effort limitations.

Step 9: Calculate γ using the Margules Equation

Once the Margules parameters are obtained, the activity coefficients are calculated using the four-suffix Margules equations as shown in equations (10) and (11).

In this study, a feedforward ANN was trained to predict the Margules parameters to calculate the activity coefficients. It was shown that it is possible to combine the interactions of the functional groups from a mixture and reasonable prediction of activity coefficients can be obtained under the framework of the artificial neural networks. The accuracy of the prediction tool is illustrated by comparing the neural network approach with the conventional functional groups contribution procedure UNIFAC.

5.2 Limitation of the Proposed Methodology

The proposed methodology is limited to estimate activity coefficients for binary liquid mixtures. A list of the DECHEMA database books is presented in Table 3. The

selected 921 systems used to build the experimental database are limited to alcohols, phenols, aldehydes, ketones and ethers. In the case of a system of interest is out of the mentioned compound families it is possible that all the interactions of the system may not be available and consequently the ANN method will provide either no results or a poor estimation of the activity coefficients. Thus, an expansion of the experimental database built during this work is recommended to include a wider range of compound families and functional group interactions.

Table 3. DECHEMA Chemistry Database

Part	Title	ISBN	Pages	Published	Number of Systems
1	Aqueous-Organic Systems	3-926959-30-4	750	1991	7
1a	Supplement 1	3-926959-90-8	750	1998	0
1b	Supplement 2	3-921567-91-2	658	1988	0
2a	Alcohols	3-921567-62-9	750	1986	189
2b	Alcohols and Phenols	3-926959-18-5	620	1990	147
2c	Alcohols Supplement 1	3-921567-29-7	730	2001	197
2d	Alcohols and Phenols Supplement 2	3-921567-43-2	830	1982	0
2e	Alcohols and Phenols Supplement 3	3-921567-92-0	650	1988	96
2f	Alcohols and Phenols Supplement 4	3-926959-16-9	716	1990	76
3/4	Aldehydes, Ketones, Ethers	3-921567-14-9	650	1979	113
3a	Aldehydes, Supplement 1	3-921567-93-9	280	1993	0
3b	Ketones, Supplement 1	3-921567-44-4	750	1993	0
4a	Ethers, Supplement 1	3-921567-86-6	534	1996	0
4b	Ethers, Supplement 2	3-926959-98-3	476	1999	0
5	Carboxylic Acids, Anhydrides, Esters	3-921567-20-3	750	2001	0
5a	Carboxylic Acids, Anhydrides, Supplement 1	3-89746-040-8	500	2002	0
5b	Esters, Supplement 2	3-89746-041-6	600	2002	0
6a	Aliphatic Hydrocarbons C_4-C_6	3-926959-42-8	716	1997	0
6b	Aliphatic Hydrocarbons C_7- C_{18}	3-926959-87-8	540	1997	0
6c	Aliphatic Hydrocarbons Supplement 1	3-921567-51-3	689	1983	0
6d+e	Aliphatic Hydrocarbons C_4- C_{30}	3-89746-007-6	1011	1999	0
7	Aromatic Hydrocarbons	3-926959-88-6	600	1997	77

Part	Title	ISBN	Pages	Published	Number of Systems
7a/7b	Supplement 1	3-89746-012-2	789	2000	19
8	Halogen, Nitrogen, Sulfur and other Compounds	3-921567-24-6	600	1984	0
8a	Halogen, Nitrogen, Sulfur and other Compounds, Supplement 1	3-89746-028-9	600	2001	0

Because the activity coefficient is derived from the molar partial Gibbs free energy it has two thermodynamic contributions, the enthalpic and the entropic, as defined by the Gibbs energy,

$$G_i = H_i - TS_i \quad \textbf{(34)}$$

where, G_i is the Gibbs energy, H_i is the enthalpy and S_i is the entropy for the i^{th} compound in a mixture. The enthalpic contribution to the activity coefficient is associated to the molecular interactions while the entropic contribution is associated to the molecules size and geometry. It is known that the Margules equation takes into consideration only the enthalpic contribution to the activity coefficient while the UNIFAC is based on the UNIQUAC equation, which takes into account both the enthalpic and the entropic contributions to the activity coefficient. It is expected that the UNIFAC method will better predict the activity coefficient for a system where the entropic contribution domains over the enthalpic contribution due to large molecular size. The ANN method should be able to use other thermodynamic models such as NRTL and UNIQUAC, to increase its capability of predicting the activity coefficient for systems with large molecular size.

6 NUMERICAL EXAMPLE

The purpose of this example is to illustrate a step-by-step calculation of predicting the activity coefficients for a binary system. An arbitrary binary system was selected and corresponds to a mixture of 2,4-dimethylpentane and benzene (Ohe, 1989). Table 4 shows the vapor-liquid equilibrium data for the system. Table 5 shows the Antoine constants of the selected system and eight steps were designed to predict the activity coefficients.

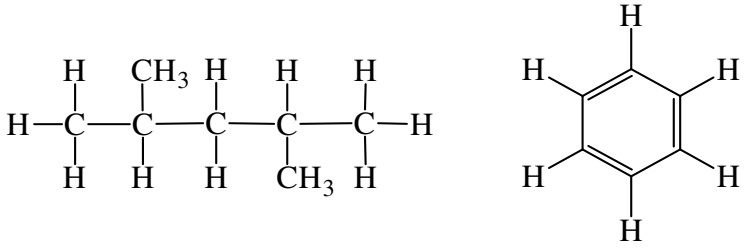
Table 4. VLE Data for the system 2,4-dimethylpentane (1) + benzene (2)

x_1	y_1	**Temperature (ºC)**	**Pressure (mmHg)**
0.078	0.125	77.3	757
0.106	0.164	77.0	757
0.140	0.195	76.5	757
0.152	0.219	76.3	757
0.192	0.242	76.1	757
0.224	0.268	75.9	757
0.251	0.292	75.7	757
0.281	0.318	75.5	757
0.335	0.357	75.4	757
0.378	0.392	75.4	757
0.432	0.432	75.2	757
0.480	0.469	75.4	757
0.525	0.504	75.3	757
0.572	0.541	75.5	757
0.616	0.576	75.5	757
0.662	0.614	75.9	757
0.700	0.652	76.1	757
0.74	0.686	76.3	757
0.785	0.731	76.6	757
0.839	0.785	77.2	757

Table 5. Antoine Constants

Component	A	B	C
1	6.82621	1192.041	221.634
2	6.90565	1211.033	220.790

Step 1. Structural formula. The structural formula for each compound of the mixture is written to facilitate the identification of the functional groups. Figure 7 shows the structural formula for each chemical compound.



2, 4-Dimethylpentane **Benzene**

Figure 7. Chemical Structures

Step 2. Identifying chemical structures functional groups. The number of functional groups associated to each compound was identified. Table 6 sumarizes the functional groups associated to each compound, which are:

- 2,4-Dimethylpentane has two units of the functional group number 2, (CH); one unit of the functional group number 3, (CH_2); and four units of the functional group number 4, (CH_3). Appendix B shows the enumeration of the functional groups.
- Benzene has six (6) units of the functional group number 135, (aCH).

Table 6. Compounds Functional Groups

Compound	Functional Groups and Quantities		
2,4-Dimethylpentane	2 × G2	1 × G3	4 × G4
Benzene	6 × G135		

Step 3. Identify functional groups interactions. The functional group interaction numbers are calculated and arranged into the system of interest information matrix. Calculations are obtained by using equation (27) and Table 7 shows the functional group interactions for the system 2, 4-dimethylpentane (1) + benzene (2).

Table 7. System Interactions

Interactions 1-2	Interaction Number	Interactions 2-1	Interaction Number
G2 G135	321	G135 G2	24926
G3 G135	507	G135 G3	24927
G4 G135	693	G135 G4	24928

The functional groups numbers were assigned using Appendix B.

Step 4. Develop the information matrix. The system of interest information matrix is similar to the experimental data information matrix. The data is arranged in a tri-dimensional matrix as presented in Figure 8.

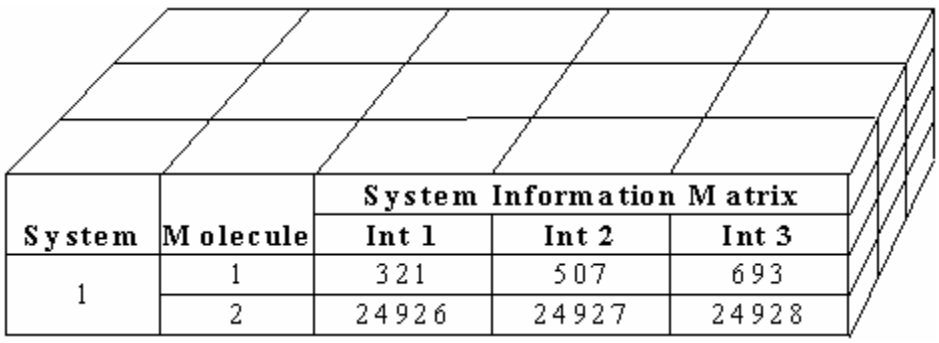


Figure 8. System Information Matrix

This system information matrix was generated by the MatLab program NNvsUNIFAC.m and is used to create the input matrix required to predict the Magurles parameters with the trained neural network.

The data contained for each functional group interaction along the 3rd dimension of the system information matrix are presented in Table 8.

Table 8. System Functional Group Interactions Data

Int#1	t	p	GF1	GF2	wGF1	wGF2	Ones
321	76.005	757	2	6	26	78	1
24926	76.005	757	6	2	78	26	1
Int#2	t	p	GF1	GF2	wGF1	wGF2	Ones
507	76.005	757	1	6	14	78	1
24927	76.005	757	6	1	78	14	1
Int#3	t	p	GF1	GF2	wGF1	wGF2	Ones
693	76.005	757	4	6	60	78	1
24928	76.005	757	6	4	78	60	1

The system information matrix is used to search within the experimental data information matrix for all systems that contain at least one of the functional group interactions of interest and build a reduce experimental data information matrix. The reduced experimental data information matrix consists of all systems within the experimental data, whose functional group interactions match at least one of the interactions of interest. The reduced experimental data information matrix was used to train the neural network.

Step 5. Generate the training patterns. The training patterns are obtained from the reduced experimental data information matrix and are organized to perform the neural network-training task following equation (28). All available training patterns for the system of interest were arranged using the reduced experimental data information matrix with 165 rows and 208 columns, i.e., there were 165 systems out of the 921 systems within the experimental data information matrix that contain at least one of the interactions of interest.

Step 6. Train the neural network. The neural network is trained with the reduced experimental data information matrix. The inputs patterns include the variables described by equation (28). The target includes the Margules parameters as shown by equation 28. The structure of the network includes a log-sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. The ANN was trained with nine different initial points and the optimum initial point is the one that minimizes the *mspe*. Figure 9 shows the training process of the MatLab software. The horizontal axis shows the number of epochs, which represents the number of time the training patterns are presented to the ANN and the vertical axis represents the *mspe*.

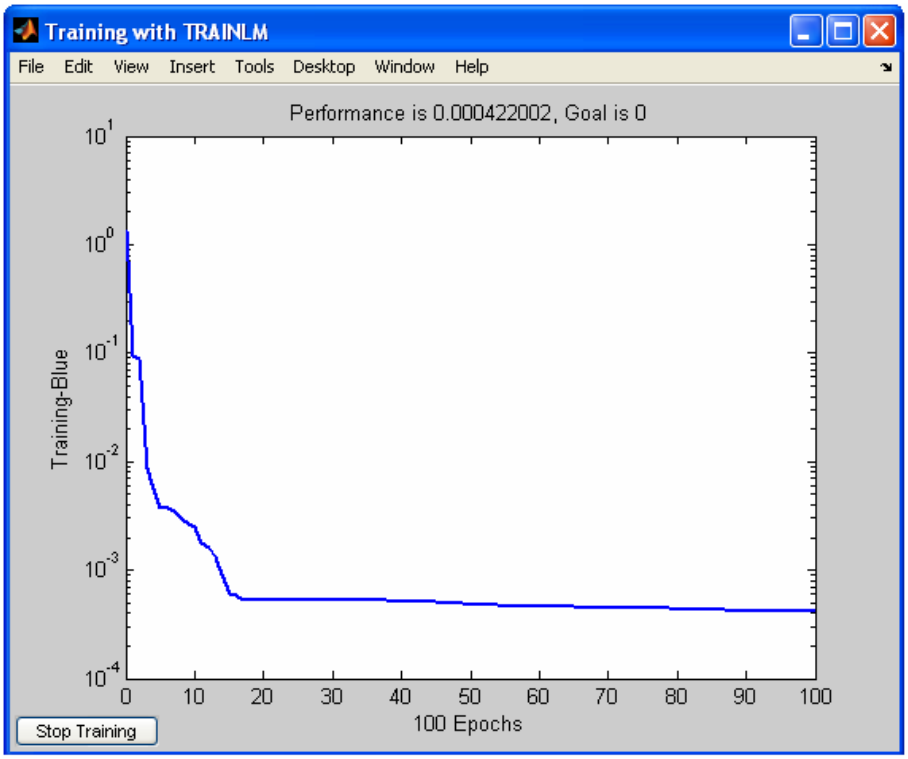


Figure 9. Neural Network Training

Step 7. Predicting the Margules parameters. The ensemble prediction technique was use to generate the nine ensemble members and the trim mean with 20% data elimination was used to calculate the predicted value for the Margules parameters. Table 9 shows the members of the ensemble prediction. Note that within the nine predicted values for the Margules parameters, there are outliers that deviate considerably from the other values. The ensemble prediction method is capable of eliminating those outliers from the calculation of the predicted Margules parameters giving more consistent results (see Table 10).

Table 9. Predicted Margules parameters values

Predicted Value	**A_M**	**B_M**	**C_M**
1	0.5958	0.1204	-0.0262
2	0.5301	0.0993	0.0394
3	0.9227	-0.0641	0.0164
4	0.9226	-0.0641	0.0163
5	0.5253	-0.1985	-0.1219
6	-1.0525	-1.3918	-1.7782
7	0.5188	0.0518	0.1006
8	0.9453	-0.2725	-0.0657
9	0.5453	0.0926	-0.0054

Table 10. Predicted Margules parameters values with 20% data elimination

	A_M	**B_M**	**C_M**
	0.5253	-0.1985	-0.0657
	0.5301	-0.0641	-0.0262
	0.5453	-0.0641	-0.0054
	0.5958	0.0518	0.0163
	0.9226	0.0926	0.0164
Average	**0.6238**	**-0.0365**	**-0.0129**

Step 8. Predicting the activity coefficients using Margules equation. The predicted Margules parameters are used to calculate the activity coefficients for both

components of the mixture. Figure 10 and Table 11 shows the results of the predicted activity coefficients using the ANN method. This table also shows the activity coefficients predicted by the UNIFAC method and the observed activity coefficients obtained by the modified Raoult's Law and the Antoine equation.

Table 11. Activity Coefficients Estimation

Mole fraction		Activity Coefficient Estimated from Experimental Data		Activity Coefficient Estimated from Neural Networks (using functional groups)		Activity Coefficient Estimated from UNIFAC (using functional groups)	
x_1	x_2	γ_1	γ_2	γ_1	γ_2	γ_1	γ_2
0.078	0.922	1.7593	1.0313	1.7276	1.0041	1.5491	1.0054
0.106	0.894	1.7142	1.0258	1.6693	1.0076	1.4863	1.0097
0.140	0.860	1.5672	1.0431	1.6034	1.0133	1.4203	1.0162
0.152	0.848	1.6312	1.0328	1.5814	1.0157	1.3992	1.0188
0.192	0.808	1.4359	1.0587	1.5122	1.0252	1.3359	1.0287
0.224	0.776	1.3714	1.0713	1.4614	1.0344	1.2925	1.0377
0.251	0.749	1.3418	1.0803	1.4215	1.0434	1.2601	1.0460
0.281	0.719	1.3134	1.0910	1.3801	1.0546	1.2280	1.0559
0.335	0.665	1.2407	1.1157	1.3130	1.0783	1.1789	1.0753
0.378	0.622	1.2073	1.1279	1.2658	1.1004	1.1467	1.0919
0.432	0.568	1.1715	1.1612	1.2136	1.1323	1.1133	1.1142
0.480	0.520	1.1375	1.1783	1.1734	1.1648	1.0889	1.1349
0.525	0.475	1.1211	1.2087	1.1404	1.1988	1.0700	1.1552
0.572	0.428	1.0977	1.2335	1.1105	1.2381	1.0536	1.1769
0.616	0.384	1.0852	1.2700	1.0865	1.2784	1.0409	1.1979
0.662	0.338	1.0632	1.2970	1.0650	1.3243	1.0300	1.2200
0.700	0.300	1.0611	1.3091	1.0500	1.3652	1.0227	1.2387
0.740	0.260	1.0495	1.3543	1.0366	1.4110	1.0163	1.2585
0.785	0.215	1.0446	1.3898	1.0243	1.4661	1.0106	1.2810
0.839	0.161	1.0303	1.4557	1.0131	1.5369	1.0056	1.3079

This table presents the final results for the activity coefficients predicted by the ANN method in addition to the results from UNIFAC and the activity coefficients calculated from the experimental vapor-liquid equilibrium data for the system: 2, 4 – dimethylpentane (1) + benzene (2).

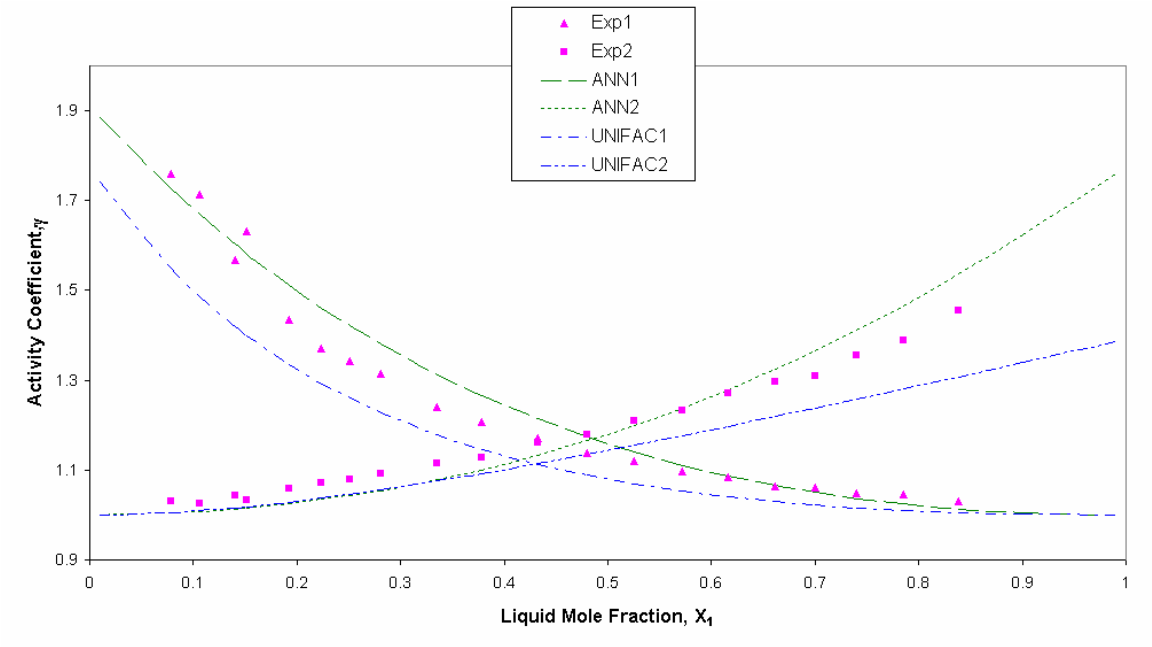


Figure 10. Comparison of activity coefficients for system:
2,4-dimethylpentane (1) + benzene (2)

This figure shows a graphical comparison for the activity coefficients γ_1 and γ_2 between the experimental values, predicted by ANN method and predicted by UNIFAC method. The prediction performance for both ANN and UNIFAC methods is about the same.

One of the well-known measurements of prediction accuracy is the mean absolute error (MAE), which is defined as follows:

$$MAE_i = \frac{1}{n}\sum_{j=1}^{n}\left|\gamma_{i,j} - \hat{\gamma}_{i,j}\right| \qquad \textbf{(35)}$$

where, n is the number of concentration levels. MAE_i is the mean absolute error for the i^{th} compound, $\gamma_{i,j}$ and $\hat{\gamma}_{i,j}$ are the activity coefficients obtained from experimental data and from the neural network methodology, respectively. Subscript j refers to the applied concentration and the subscript i represents the chemical compound. The *MAE* for the activity coefficients predicted by the ANN and UNIFAC methods for the system

presented above are shown in Table 12. The ANN method shows an improvement of the activity coefficients prediction over the UNIFAC method, as also shown in Table 12.

Table 12. Mean Absolute Error for the System: 2, 4-dimethylpentane (1) + benzene (2)

Liquid Mole fraction		**Absolute Prediction Errors for Neural Networks**		**Absolute Prediction Errors for UNIFAC**	
x_1	x_2	γ_1 **Error**	γ_2 **Error**	γ_1 **Error**	γ_2 **Error**
0.078	0.922	0.0317	0.0271	0.2102	0.0259
0.106	0.894	0.0449	0.0181	0.2279	0.0161
0.140	0.860	0.0362	0.0297	0.1469	0.0269
0.152	0.848	0.0498	0.0170	0.2320	0.0140
0.192	0.808	0.0763	0.0333	0.1000	0.0300
0.224	0.776	0.0900	0.0367	0.0789	0.0336
0.251	0.749	0.0797	0.0368	0.0817	0.0343
0.281	0.719	0.0667	0.0362	0.0854	0.0351
0.335	0.665	0.0723	0.0372	0.0618	0.0404
0.378	0.622	0.0585	0.0273	0.0606	0.0360
0.432	0.568	0.0421	0.0287	0.0582	0.0470
0.480	0.520	0.0359	0.0133	0.0486	0.0434
0.525	0.475	0.0193	0.0097	0.0511	0.0535
0.572	0.428	0.0128	0.0048	0.0441	0.0566
0.616	0.384	0.0013	0.0086	0.0443	0.0721
0.662	0.338	0.0018	0.0275	0.0332	0.0770
0.700	0.300	0.0111	0.0563	0.0384	0.0704
0.740	0.260	0.0129	0.0569	0.0332	0.0958
0.785	0.215	0.0203	0.0764	0.0340	0.1088
0.839	0.161	0.0172	0.0814	0.0247	0.1478
Mean Absolute Error		0.0390	0.0331	0.0848	0.0531

This table presents the absolute prediction errors for the activity coefficients predicted by the ANN and UNIFAC methods. The mean absolute error is presented at the end of the table.

7 COMPARISON BETWEEN ANN AND UNIFAC

Initially, we tried to find a data set from a source other than the DECHEMA database to perform a fair comparison between the ANN and UNIFAC methods; however, the identified number of systems was very small. Nineteen systems were selected to perform a comparison between the ANN method and the UNIFAC method. Nine out of nineteen systems were obtained from Ohe (1989) and the other ten systems were taken directly from DECHEMA database (Sorensen and Arlt, 1979). Although the information comes from different sources it was noticed that all of the nineteen systems are available in the complete DECHEMA database and the UNIFAC method was developed using all of the 8,600 systems available in the complete DECHEMA database (Gmehling, 2003). The ANN method was developed using only 921 systems from DECHEMA. The activity coefficients, γ_1 and γ_2 were predicted for the nineteen systems using the ANN and the UNIFAC methods (see Appendix A). Appendix A also shows graphically the performance comparison between the ANN and UNIFAC methods for all nineteen systems. Appendix A figures show that the prediction capabilities of the considered methods are about the same.

The mean absolute errors (MAE) from the nineteen systems were used to measure the accuracy of the prediction methods. The *MAE's* do not follow a normal distribution due to the outliers' results as shown in Figure 11.

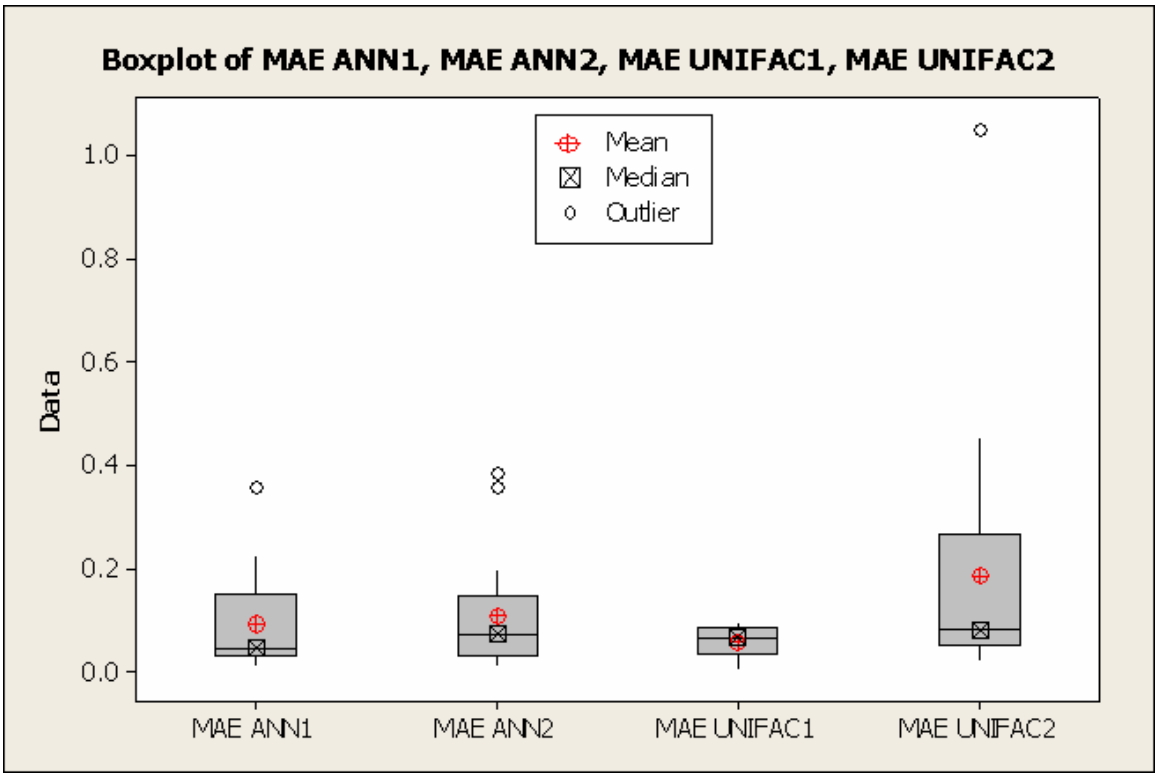


Figure 11. MAE Populations Boxplot

It is well known that the median is a better representation of the central tendency of the population when outliers are present. Thus, a non-parametric test was used to prove whether or not the median of the *MAE* from the ANN method is smaller than the median of the *MAE* obtained from UNIFAC. The selected non-parametric test was the Mann-Whitney (Hayter, 1996) test. When predicting the activity coefficient γ_1, the medians of the *MAE* from the ANN method and from UNIFAC show no significant difference at 95% confidence since the p-value was 0.6614. The ANN method performance is slightly better, with a median of the *MAE* equal to 0.0483 while the median of the *MAE* for the UNIFAC method is equal to 0.0669.

Mann-Whitney Test and CI: MAE ANN1_M, MAE U1

```
             N   Median
MAE ANN1_M  19  0.04831
MAE U1      19  0.06691
Point estimate for ETA1-ETA2 is 0.00716
95.3 Percent CI for ETA1-ETA2 is (-0.02561,0.05846)
W = 386.0
Test of ETA1 = ETA2 vs. ETA1 not = ETA2 is significant at 0.6614
```

When predicting γ_2 the medians of the *MAE* from the ANN method and from UNIFAC show no significant difference at 95% confidence since the p-value was 0.2933. The ANN method performance is slightly better, with a median of the *MAE* equal to 0.0764 while the median of the *MAE* for the UNIFAC method is equal to 0.0842.

Mann-Whitney Test and CI: MAE ANN2_M, MAE U2

```
             N  Median
MAE ANN2_M  19  0.0764
MAE U2      19  0.0842
Point estimate for ETA1-ETA2 is -0.0193
95.3 Percent CI for ETA1-ETA2 is (-0.0785,0.0308)
W = 334.0
Test of ETA1 = ETA2 vs. ETA1 not = ETA2 is significant at 0.2933
```

Even though, the ANN method performs a slightly better than the UNIFAC method, the improvement is not statistically significant. In summary, based on the nineteen systems studied, the performance of the ANN method is slightly better than the UNIFAC method. Table 13 summarizes the results for the mean absolute errors.

Table 13. Mean Absolute Errors for the Activity Coefficients

System	Molecule 1	Molecule 2	Mean Absolute Prediction Errors for Neural Networks		Mean Absolute Prediction Errors for UNIFAC		Source
			γ_1	γ_2	γ_1	γ_2	
1	acetone	methanol	0.0155	0.0268	0.0112	0.0227	1
2	methanol	3-methylbutanol	0.0227	0.0732	0.0066	0.0957	1
3	acetone	ethanol	0.0536	0.0438	0.0875	0.0405	1
4	tetrachloromethane	2-propanol	0.1254	0.0919	0.0276	0.4035	1
5	methanol	1-propanol	0.0311	0.0598	0.0825	0.0504	1
6	methanol	1-butanol	0.0398	0.0764	0.0669	0.0642	1
7	methyl acetate	methanol	0.0138	0.0214	0.0333	0.0329	1
8	acetone	benzene	0.0273	0.0138	0.0412	0.0291	1
9	acetone	hexane	0.0320	0.0794	0.0725	0.0677	1
10	acetone	toluene	0.0454	0.0390	0.0421	0.0804	1
11	2,4-dimethyl pentane	benzene	0.0390	0.0331	0.0848	0.0531	2
12	cyclohexane	isopropyl alcohol	0.2253	0.1311	0.1001	0.2922	2
13	ethylcyclohexane	isopropyl alcohol	0.3572	0.1501	0.0999	1.0492	2
14	benzene	isopropyl alcohol	0.1546	0.3573	0.0701	0.0842	2
15	toluene	isopropyl alcohol	0.0849	0.1542	0.0596	0.2226	2
16	ethyl ether	acetone	0.1240	0.1998	0.0971	0.2669	2
17	acetone	isopropyl alcohol	0.1585	0.3837	0.0938	0.4554	2
18	isopropyl alcohol	ethyl acetate	0.0483	0.0200	0.0355	0.0953	2
19	chloroform	n-butyl alcohol	0.2012	0.1355	0.0397	0.1548	2
		Average	**0.0947**	**0.1100**	**0.0606**	**0.1874**	
		Median	**0.0483**	**0.0764**	**0.0669**	**0.0842**	

Sources: (1) Sorensen and Arlt (1979) (*DECHEMA*); (2) Ohe (1989)

This table presents the mean absolute errors for the activity coefficients for each of the nineteen systems studied. The mean absolute error is used to standardize the data and compare the prediction results populations.

The relative comparison between two prediction methods can be accomplished by using a skill score (SS) that relates the mean absolute prediction of the forecast methods. The skill score is widely used in different fields (Wilks, 2006) and is defined as follows:

$$SS = \frac{A - A_{ref}}{A_{perf} - A_{ref}} \times 100 \qquad \textbf{(36)}$$

where, A is a particular measure of accuracy of the method to be compared such as *MAE*, *MSE* or *RMSE*. A_{ref} is the measure of accuracy of the reference method, A_{perf} is the value of the accuracy measure that would be achieved by a perfect forecast and is equal to 0 when using *MSE* or *MAE* as accuracy measure.

The skill score is a positive value when the method of interest is better than the reference method and negative otherwise. The skill score based on the median of the *MAE* for the ANN method was 28% when predicting γ_1 and 9% when predicting γ_2.

8 CONCLUSIONS AND FUTURE WORK

A new methodology using artificial neural networks was designed to predict the activity coefficients based on the knowledge of the structural formula, temperature, pressure and liquid molar composition for the compounds in a mixture. The proposed methodology is a potential tool to estimate the activity coefficients of chemical compounds when experimental data are not available. The neural network method was compared to the UNIFAC method and it was found that the performance of the ANN method was slightly better for the nineteen studied systems.

The mean squared prediction error criterion was used to identify the appropriate structure of the feedforward artificial neural network. The introduced methodology basically consists of selecting a neural network structure for a single neuron in the hidden layer and then the number of neurons increases until the mean squared prediction error has been accomplished. Finally, the neural network structure that provides the minimum variance prediction with the least number of neurons is selected as the best neural network structure.

The neural network method exhibits a high variability in the results. However, the optimum initial point determination and ensemble prediction technique was able to stabilize the output of the neural network method and increase its prediction accuracy. A practical tool was developed as an alternative method that may be used to obtain a reliable prediction of activity coefficients for a given chemical mixture and the MatLab code is presented in Appendix C.

As future work, it is recommended to expand the experimental database to other compound families not included in this work in order to cover a wider range of functional group interactions. Also the ANN method should be trained with other thermodynamic models (i.e. Wilson, van Laar, NRTL and UNIQUAC) and the program should have the flexibility for the user to choose the thermodynamic model for predicting the activity coefficient. The program for the ANN method should be capable of measuring the contribution to the activity coefficient of each functional group interaction. A comparison between the ANN and UNIFAC methods to predict the activity coefficient at infinite dilution γ^{∞} should be performed.

References

Abrams, D.S., and Prausnitz, J.M. (1975), Statistical Thermodynamics of Liquid Mixtures: A New Expression for the Excess Gibbs Energy of Partial or Completely Miscible Systems. *AIChE Journal*, 21, (1) 116-128.

Álvarez, V. H, and Valderrama, J. O. (2004). Método Modificado Lydersen-Joback-Reid para Estimar Propiedades Críticas de Biomoléculas, *Alimentaria*, 354, 55-66.

Bazaraa, M.S., Sherali, H.D., and Shetty, C.M. (1993), *Nonlinear Programming: Theory and Algorithms*, 2nd Ed., John Wiley & Sons, Inc., New York.

Biglin, M., (2004), Isobaric vapour–liquid equilibrium calculations of binary systems using a neural network, *J. Serb. Chem. Soc.,* 69, (8-9) 669–674.

Box, G.E.P., and Jenkins, G.M. (1976), *Time Series Analysis: Forecasting and Control*, Holden-Day, Oakland CA.

Cartensen, J.T. (1995), *Drug Stability, Principles and Practice.* Second Ed., Marcel Dekker Inc.

Chow, H., Chen, H., Ng, T., Myrdal, P., and Yalkowsky, S. H. (1995), Using Backpropagation Networks for the Estimation of Aqueous Activity Coefficients of Aromatic Organic Compounds, *J. Chem. Inf. Comput. Sci.*, 35, (4) 723-728.

Constantinou, L., and Gani, R. (1994), New Group Contribution Method for Estimating Properties of Pure Compounds, *AIChE Journal*, 40, (10) 1697-1710.

Dehghani, M. R., Modarress, H. and Bakhshi, A. (2006), Modeling and prediction of activity coefficient ratio of electrolytes in aqueous electrolyte solution containing amino acids using artificial neural network, *Fluid Phase Equilibria*, 244, 153-159.

Derr, E. L. and Deal, C. H. (1969), Proceedings Int. Dist. Symp. Brighton.

Fredenslund, A., Jones, R.L., and Prausnitz, J.M. (1975), Group Contribution Estimation of Activity Coefficients in Nonideal Liquid Mixtures, *AIChE Journal*, 21, (6) 1086-1099.

Gao, X. Z. and Ovaska, S. J. (2001), Soft computing methods in motor fault diagnosis, *Applied Soft Computing*, 1, 73-81.

Gmehling, J. (2003), Potential of group contribution methods for the prediction of phase equilibria and excess properties of complex mixtures, *Pure Appl. Chem.*, 75, (7) 875-888.

Golob, M. (2001), Decomposed fuzzy proportional-integral-derivative controllers, *Applied Soft Computing*, 18, 73-81.

Hagan, M.T., Demuth, H.B., and Beal, M. (1996), *Neural Network Design*, PWS Publishing Company, Boston.

Hayter, A.J. (1996), *Probability and Statistics for Engineers and Scientists*, PWS Publishing Company, Boston.

Inaoka, M., Kato, K., Sakawa, M. and Ushiro, S. B. (2001), Operation planning of district heating and cooling plans using genetic algorithms for mixed integer programming, *Applied Soft Computing*, 1, 139-150.

Kadirkkamanathan, V. and Regunath, S. (2001), Design of a pH control system using fuzzy non-uniform grid scheduling and evolutionary programming**,** *Applied Soft Computing*, 1, 91-104.

Knox, E. D. (1987), A One-Parameter Group-Contribution Model for Liquid Mixtures, *Journal of Solution Chemistry*, 16, (8) 625-634.

Kojima, K., and Tochigi, K. (1979), *Prediction of vapor-liquid equilibria by the ASOG method*, Elsevier, Amsterdam-Oxford-New York.

Koppen, M. and Ruiz-del-Solar, J. (2001), Steady-state image processing, *Applied Soft Computing*, 1, 53-62.

Larsen, B. L., Rasmussen, P., and Fredenslund, A. (1987), A Modified UNIFAC Group-Contribution Model for Prediction of Phase Equilibria and Heats of Mixing. *Ind. Eng. Chem. Res.*, 26, (7) 2274-2286.

Levine I. N. (1988), *Physical Chemistry*, Third Edition, McGraw-Hill, New York.

Mitchell B. E. and Jurs P. C. (1998a), Prediction of Infinite Dilution Activity Coefficients of Organic Compounds in Aqueous Solution from Molecular Structure, *J. Chem. Inf. Comput. Sci.*, 38, 200-209.

Mitchell B. E. and Jurs P. C. (1998b), Prediction of Aqueous Solubility of Organic Compounds from Molecular Structure, *J. Chem. Inf. Comput. Sci.*, 38, 489-496.

Ohe, S. (1989), *Vapor-Liquid Equilibrium Data*, Elsiever Science Publisher, Amsterdan and New York.

Pandit, S.M. and Wu, S.M. (1983), *Times Series and System Analysis with Applications,* John Wiley and Sons, New York.

Petersen, R., Fredenslund, A., and Rasmussen, P. (1994), Artificial neural networks as a predictive tool for vapor-liquid equilibrium, *Computers and Chemical Engineering,* 18, Suppl., S63 – S67.

Ramirez-Beltran, N.D., Sridhar, L.N., and Rodriguez, H. (1998), A Nonlinear Model to Estimate Chemical Interaction Constants, *Computing Research Conference,* CRC'98, December 4, Mayagüez, Puerto Rico.

Ramirez-Beltran, N.D., and Jackson, H. (1999), Application of Neural Networks to Chemical Process Control, *Computers and Industrial Engineering*, 37, (1-2) 387-390.

Ramirez-Beltran, N. D. and Montes, J. A. (2002), Neural Networks to model dynamic systems with time delays, *IIE Transactions*, 34, 313-327.

Rani, K. Y., Dutt, N. V. K. (2002), Estimation of Activity Coefficients at Infinite Dilution of Halocarbons in Water and Organic Compounds in Hydroflouoroparaffins Using Neural Networks, *Chem. Eng. Comm.*, 189, (3) 372-390.

Reid, R. C., Prausnitz, J. M., Poling, B. E. (1987), *The Properties of Gases and Liquids*, McGraw-Hill, New York.

Renon, H., and Prausnitz, J.M. (1968), Local compositions in thermodynamic excess functions for liquid mixtures, *AIChE Journal*, 14, (1) 135-144.

Sherman, S. R., Trampe, D. B., Bush, D. M., Schiller, M., Eckert, C. A., Dallas, A. J., Li, J., Carr, P. W. (1996), Compilation and Correlation of Limiting Activity Coefficients of Nonelectrolytes in Water, *Ind. Eng. Chem. Res.*, 35, 1044-1058.

Skander, N. and Chitour, C.E. (2002), A New Group-Contribution Method for the Estimation of Physical Properties of Hydrocarbons, *Oil & Gas Science and Technology – Rev. IFP*, 57, (4) 369-376.

Smith, J.M., van Ness, H.C., and Abbott, M.M. (1996), *Introduction to Chemical Engineering Thermodynamics*, McGraw-Hill, New York.

Sorensen, J.M., and Arlt, W. (1979), *DECHEMA Chemistry Data Series*, Frankfurt.

Taskinen J. and Yliruusi J. (2003), Prediction of physicochemical properties based on neural network modeling, *Advanced Drug Delivery Reviews*, 55, 1163-1183.

Urata, S., Takada, A., Murata, J., Hiaki, T. and Sekiya, A. (2002), Prediction of vapor–liquid equilibrium for binary systems containing HFEs by using artificial neural network, *Fluid Phase Equilibria*, 199, 63-78.

Valderrama, J. O. and Álvarez, V. H. (2006), A New Group Contribution Method based on Equation of State Parameters to Evaluate the Critical Properties of Simple and Complex Molecules. *Can. J. Chem. Eng.*, 84, (4) 431-446.

Voracek, J. (2001), Prediction of mechanical properties of cast irons, *Applied Soft Computing*, 1, 119-125.

Wei, W.W.S (1990), *Time Series Analysis: Univariate and Multivariate Methods,* Addison-Wesley Publishing Co., Redwood City, CA.

Weldlich, U. and Gmehling, J. (1987), A Modified UNIFAC Model. Prediction of VLE, h^E and γ^∞, *Ind. Eng. Chem. Res.*, 26, (7) 1372-1381.

Wilks, D. S. (2006), *Statistical Methods in the Atmospheric Sciences 2nd Edition*, Elsiever Inc., New York.

Wu, H. S. and Sandler, S. I. (1991a), Use of ab initio Quantum Mechanics Calculations in Group Contribution Methods 1. Theory and the Basis for Group Identification, *Ind. Eng. Chem. Res.*, 30, (5) 881-889.

Wu, H. S. and Sandler, S. I. (1991b), Use of ab initio Quantum Mechanics Calculations in Group Contribution Methods 2. Test of New Groups in UNIFAC, *Ind. Eng. Chem. Res.*, 30, (5) 889-897.

Appendix A Graphical Comparison Between ANN and UNIFAC

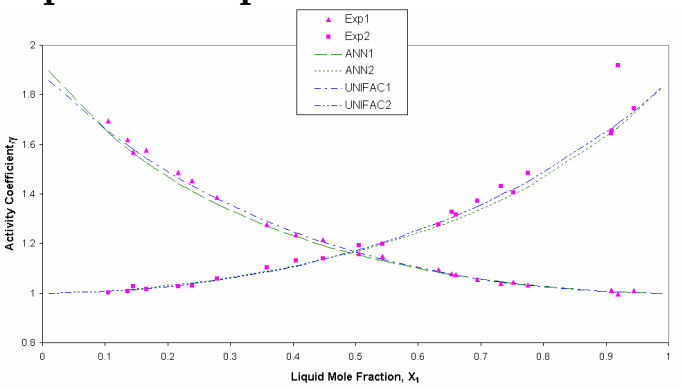


Figure 12. System 1: Acetone (1) + Methanol (2)
This system is included in the 921 systems compiled from DECHEMA. None of the prediction methods were able to predict the deviation peak of γ_2 at a low x_2 value.

Table 14. Activity Coefficients Results for System 1

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	Exp
55	0.1046	1.6481	1.6532	1.6940	1.0085	1.0072	1.0016
55	0.1357	1.5833	1.5956	1.6200	1.0140	1.0121	1.0080
55	0.1452	1.5649	1.5789	1.5692	1.0160	1.0138	1.0260
55	0.1663	1.5260	1.5431	1.5768	1.0207	1.0181	1.0148
55	0.2173	1.4425	1.4637	1.4866	1.0344	1.0309	1.0266
55	0.2390	1.4110	1.4328	1.4525	1.0412	1.0375	1.0286
55	0.2787	1.3587	1.3803	1.3860	1.055	1.0511	1.0566
55	0.3579	1.2715	1.2895	1.2755	1.0882	1.0850	1.1023
55	0.4050	1.2285	1.2434	1.2346	1.1115	1.1096	1.1300
55	0.4480	1.1939	1.2058	1.2151	1.1353	1.1353	1.1396
55	0.5052	1.1540	1.1619	1.1576	1.1710	1.1743	1.1930
55	0.5432	1.1307	1.1362	1.1469	1.1976	1.2035	1.1995
55	0.6332	1.0844	1.0856	1.0940	1.2712	1.2844	1.2776
55	0.6538	1.0754	1.0759	1.0767	1.2905	1.3055	1.3277
55	0.6605	1.0726	1.0729	1.0741	1.2970	1.3125	1.3163
55	0.6945	1.0592	1.0586	1.0542	1.3318	1.3500	1.3731
55	0.7327	1.0458	1.0445	1.0375	1.3748	1.3958	1.4307
55	0.7525	1.0395	1.0380	1.0428	1.399	1.4210	1.4070
55	0.7752	1.0328	1.0313	1.0320	1.4283	1.4514	1.4841
55	0.9080	1.0059	1.0052	1.0099	1.6460	1.6632	1.6438
55	0.9088	1.0058	1.0051	1.0111	1.6476	1.6646	1.6545
55	0.9197	1.0045	1.0039	0.9969	1.6698	1.6850	1.9194
55	0.9448	1.0022	1.0019	1.0106	1.7242	1.7337	1.7461

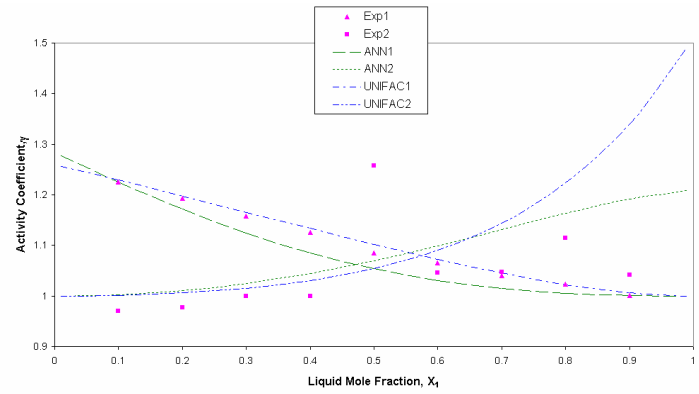


Figure 13. System 2: Methanol (1) + 3-Methylbutanol
This system is included in the 921 systems compiled from DECHEMA.

Table 15. Activity Coefficients Results for System 2

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
60	0.1	1.2252	1.2288	1.2251	1.0025	1.0013	0.9703
60	0.2	1.1717	1.1974	1.1926	1.0104	1.0060	0.9774
60	0.3	1.1247	1.1655	1.1586	1.0243	1.0151	0.9994
60	0.4	1.0853	1.1336	1.1253	1.0442	1.0305	1.0000
60	0.5	1.0540	1.1023	1.0853	1.0694	1.0545	1.2579
60	0.6	1.0306	1.0725	1.0653	1.0990	1.0908	1.0456
60	0.7	1.0147	1.0452	1.0401	1.1311	1.1445	1.0468
60	0.8	1.0053	1.0223	1.0228	1.1630	1.2238	1.1138
60	0.9	1.0010	1.0062	1.0003	1.1910	1.3408	1.0420

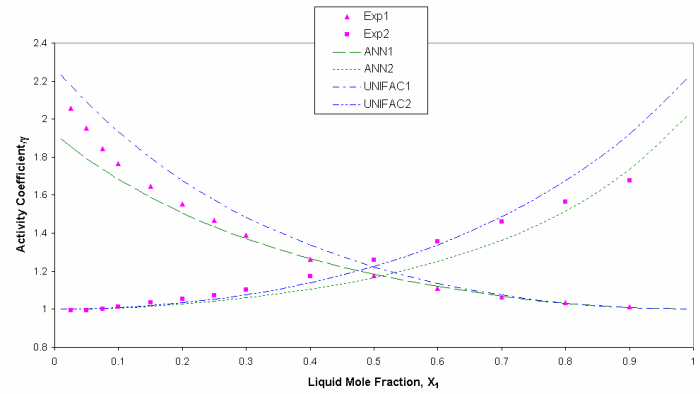


Figure 14. System 3: Acetone (1) + Ethanol (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 16. Activity Coefficients Results for System 3

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
40	0.025	1.8567	2.1767	2.0581	1.0005	1.0005	0.9954
40	0.050	1.7939	2.0896	1.9530	1.0018	1.0021	0.9957
40	0.075	1.7361	2.0085	1.8453	1.0040	1.0048	1.0025
40	0.100	1.6827	1.9330	1.7657	1.0070	1.0085	1.0129
40	0.150	1.5875	1.7968	1.6483	1.0154	1.0190	1.0366
40	0.200	1.5053	1.6782	1.5519	1.0269	1.0339	1.0530
40	0.250	1.4338	1.5747	1.4672	1.0415	1.0532	1.0741
40	0.300	1.3713	1.4842	1.3904	1.0592	1.0771	1.1013
40	0.400	1.2677	1.3358	1.2630	1.1050	1.1400	1.1733
40	0.500	1.1859	1.2226	1.1755	1.1670	1.2255	1.2583
40	0.600	1.1211	1.1375	1.1097	1.2502	1.3385	1.3565
40	0.700	1.0703	1.0754	1.0642	1.3627	1.4856	1.4615
40	0.800	1.0327	1.0330	1.0334	1.5177	1.6762	1.5654
40	0.900	1.0087	1.0082	1.0127	1.7361	1.9240	1.6764

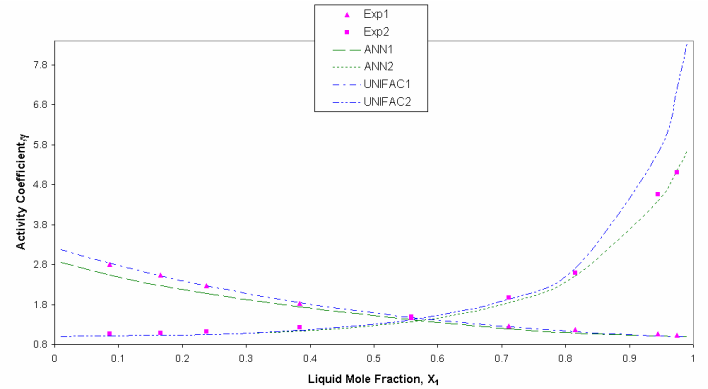


Figure 15. System 4: Tetrachloromethane (1) + 2-Propanol (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 17. Activity Coefficients Results for System 4

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
70	0.086	2.5354	2.8327	2.8057	1.0060	1.0059	1.0641
70	0.166	2.2720	2.5160	2.5406	1.0220	1.0234	1.0821
70	0.238	2.0734	2.2670	2.2771	1.0460	1.0508	1.1110
70	0.384	1.7384	1.8499	1.8277	1.1334	1.1527	1.2301
70	0.559	1.4144	1.4744	1.4780	1.3671	1.4154	1.4922
70	0.711	1.1957	1.2352	1.2669	1.8399	1.9353	1.9684
70	0.815	1.0865	1.1128	1.1656	2.5117	2.7205	2.5806
70	0.945	1.0086	1.0137	1.0574	4.3935	5.5942	4.5631
70	0.975	1.0018	1.0031	1.0355	5.1647	7.2058	5.1094

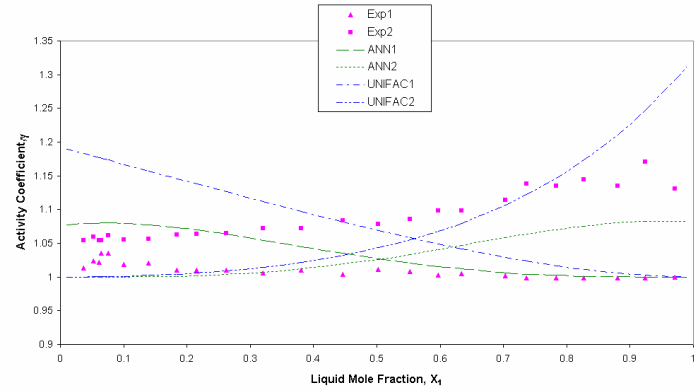


Figure 16. System 5: Methanol (1) + 1-Propanol (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 18. Activity Coefficients Results for System 5

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
60.02	0.0368	1.0792	1.1830	1.0128	0.99996	1.0001	1.0537
60.02	0.0528	1.0797	1.1789	1.0236	0.99994	1.0003	1.0592
60.02	0.0611	1.0799	1.1768	1.0212	0.99993	1.0004	1.0538
60.02	0.0649	1.0799	1.1759	1.0354	0.99993	1.0005	1.0537
60.02	0.0764	1.0799	1.1740	1.0354	0.99992	1.0007	1.0615
60.02	0.1007	1.0794	1.1668	1.0184	0.99997	1.0012	1.0550
60.02	0.1393	1.0774	1.1570	1.0210	1.0002	1.0023	1.0562
60.02	0.1842	1.0734	1.1457	1.0100	1.0010	1.0042	1.0624
60.02	0.2151	1.0698	1.1379	1.0100	1.0018	1.0059	1.0640
60.02	0.2625	1.0635	1.1261	1.0103	1.0037	1.0092	1.0648
60.02	0.3206	1.0547	1.1117	1.0062	1.0071	1.0146	1.0718
60.02	0.3806	1.0452	1.0972	1.0098	1.0121	1.0218	1.0722
60.02	0.4459	1.0350	1.0819	1.0040	1.0191	1.0321	1.0834
60.02	0.5022	1.0269	1.0691	1.0111	1.0263	1.0432	1.0781
60.02	0.552	1.0205	1.0583	1.0078	1.0336	1.0551	1.0858
60.02	0.597	1.0154	1.0489	1.0032	1.0406	1.0678	1.0981
60.02	0.6338	1.0117	1.0417	1.0044	1.0466	1.0798	1.0986
60.02	0.7034	1.0063	1.0290	1.0022	1.0580	1.1068	1.1143
60.02	0.737	1.0043	1.0235	0.9987	1.0633	1.1223	1.1379
60.02	0.783	1.0023	1.0166	0.9984	1.0700	1.1464	1.1346
60.02	0.8275	1.0010	1.0109	0.99896	1.0756	1.1734	1.1448
60.02	0.8804	1.0002	1.0055	0.99903	1.0807	1.2110	1.1353
60.02	0.9238	1.0000	1.0023	0.99873	1.0829	1.2469	1.1705
60.02	0.9707	0.99998	1.0004	0.99953	1.0830	1.2918	1.1305

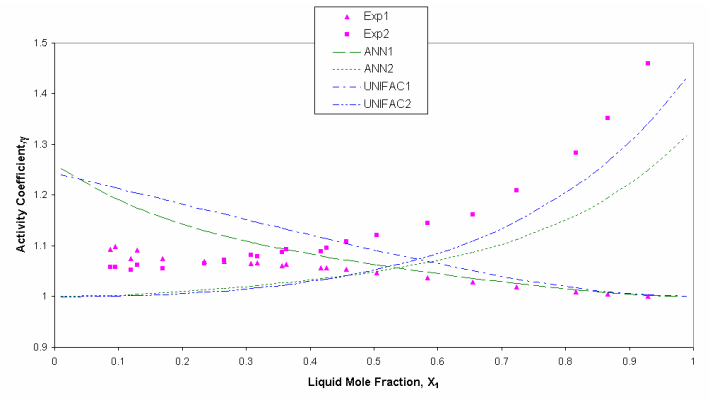


Figure 17. System 6: Methanol (1) + 1-Butanol (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 19. Activity Coefficients Results for System 6

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
25	0.0871	1.1979	1.2169	1.0929	1.0022	1.0010	1.0585
25	0.0953	1.193	1.2144	1.0980	1.0026	1.0012	1.0579
25	0.1200	1.1793	1.2069	1.0743	1.0040	1.0020	1.0520
25	0.1295	1.1743	1.2040	1.0919	1.0047	1.0023	1.0625
25	0.1690	1.1556	1.1919	1.0748	1.0075	1.0041	1.0554
25	0.2340	1.1299	1.1720	1.0693	1.0132	1.0084	1.0646
25	0.2654	1.1194	1.1624	1.0696	1.0163	1.0111	1.0719
25	0.3072	1.107	1.1496	1.0650	1.0209	1.0156	1.0813
25	0.3173	1.1042	1.1465	1.0664	1.0221	1.0169	1.0786
25	0.3557	1.0942	1.1349	1.0603	1.0268	1.0222	1.0868
25	0.3628	1.0925	1.1327	1.0631	1.0277	1.0232	1.0928
25	0.4172	1.0800	1.1165	1.0562	1.0352	1.0328	1.0891
25	0.4258	1.0782	1.1139	1.0565	1.0365	1.0345	1.0952
25	0.4569	1.0718	1.1048	1.0543	1.0414	1.0413	1.1081
25	0.5048	1.0624	1.0909	1.0467	1.0499	1.0535	1.1214
25	0.5845	1.0481	1.0689	1.0367	1.0672	1.0796	1.1441
25	0.6544	1.0363	1.0509	1.0281	1.0870	1.1100	1.1621
25	0.7237	1.0255	1.0348	1.0182	1.1127	1.1488	1.2094
25	0.8164	1.0129	1.0168	1.0088	1.1604	1.2188	1.2827
25	0.8660	1.0074	1.0094	1.0046	1.1944	1.267	1.3519
25	0.9290	1.0023	1.0028	1.001	1.2491	1.3422	1.4593

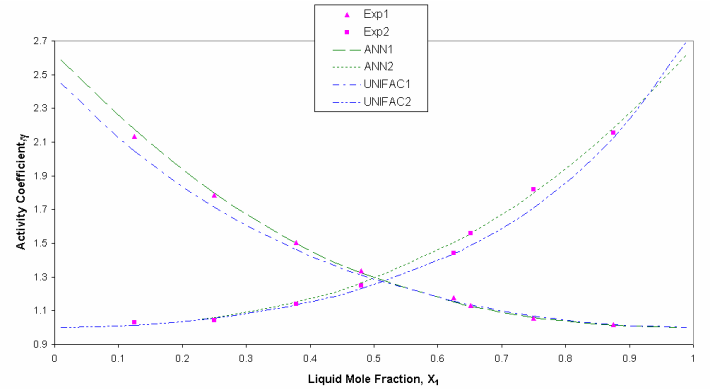


Figure 18. System 7: Methyl Acetate (1) + Methanol (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 20. Activity Coefficients Results for System 7

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
55	0.125	2.1741	2.0455	2.1343	1.0131	1.0133	1.0284
55	0.250	1.7972	1.7131	1.7839	1.0590	1.0557	1.0426
55	0.378	1.5006	1.4617	1.5055	1.1505	1.1354	1.1404
55	0.480	1.3243	1.3113	1.3372	1.2638	1.2321	1.2466
55	0.625	1.1510	1.1568	1.1778	1.5025	1.4387	1.4433
55	0.652	1.1274	1.1346	1.1318	1.5586	1.4888	1.5592
55	0.750	1.0608	1.0692	1.0573	1.7972	1.7117	1.8197
55	0.875	1.0137	1.0175	1.0185	2.1856	2.1232	2.1559

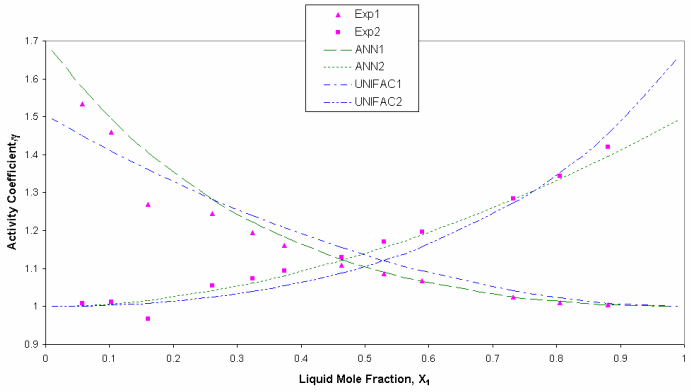


Figure 19. System 8: Acetone (1) + Benzene (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 21. Activity Coefficients Results for System 8

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
40	0.0570	1.5766	1.4501	1.5333	1.0021	1.0011	1.0089
40	0.1027	1.4944	1.4091	1.4592	1.0068	1.0036	1.0126
40	0.1608	1.4050	1.3600	1.2695	1.0163	1.0090	0.9662
40	0.2615	1.2824	1.2824	1.2442	1.0413	1.0250	1.0557
40	0.3240	1.2229	1.2390	1.1951	1.0619	1.0398	1.0728
40	0.3734	1.1832	1.2072	1.1605	1.0809	1.0544	1.0947
40	0.4636	1.1244	1.1547	1.1081	1.1212	1.0887	1.1297
40	0.5300	1.0906	1.1207	1.0859	1.1553	1.1213	1.1698
40	0.5892	1.0663	1.0937	1.0670	1.1889	1.1566	1.1964
40	0.7325	1.0256	1.0417	1.0253	1.2821	1.2726	1.2834
40	0.8051	1.0130	1.0228	1.0107	1.3357	1.3527	1.3445
40	0.8807	1.0047	1.0088	1.0042	1.3959	1.4564	1.4197

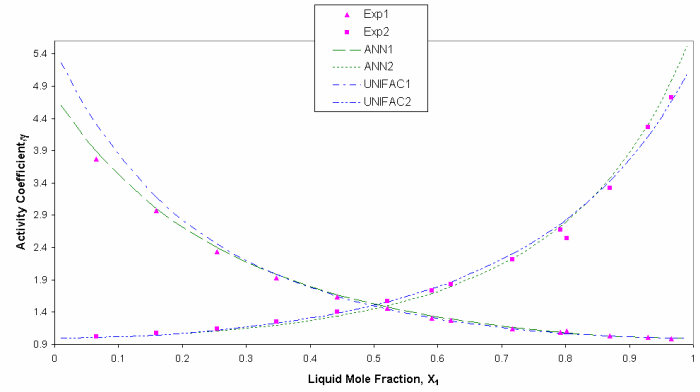


Figure 20. System 9: Acetone (1) + Hexane (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 22. Activity Coefficients Results for System 9

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
45	0.0651	3.8874	4.3018	3.7694	1.0068	1.0080	1.0175
45	0.1592	3.0064	3.1826	2.9722	1.0399	1.0470	1.0764
45	0.2549	2.3976	2.4547	2.3331	1.1031	1.1203	1.1431
45	0.3478	1.9805	1.9822	1.9260	1.1979	1.2283	1.2537
45	0.4429	1.6703	1.6479	1.6352	1.3391	1.3859	1.4005
45	0.5210	1.4776	1.4493	1.4539	1.5009	1.5618	1.5664
45	0.5907	1.3414	1.3140	1.3105	1.6942	1.7656	1.7287
45	0.6202	1.2923	1.2663	1.2709	1.7941	1.8686	1.8321
45	0.7168	1.1615	1.1430	1.1446	2.2260	2.2978	2.2188
45	0.7923	1.0876	1.0760	1.0878	2.7256	2.7672	2.6746
45	0.8022	1.0796	1.0689	1.1058	2.8058	2.8404	2.5458
45	0.8692	1.0355	1.0302	1.0274	3.4712	3.4271	3.3264
45	0.9288	1.0108	1.0090	1.0116	4.3058	4.1224	4.2642
45	0.9658	1.0025	1.0021	0.9884	4.9897	4.6660	4.7251

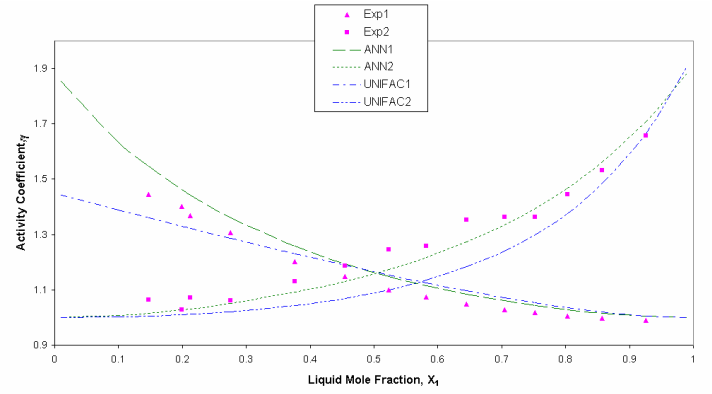


Figure 21. System 10: Acetone (1) + Toluene (2)
This system is included in the 921 systems compiled from DECHEMA.

Table 23. Activity Coefficients Results for System 10

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
35	0.1469	1.5461	1.3612	1.4453	1.0152	1.0051	1.0650
35	0.1988	1.4624	1.3309	1.4012	1.0271	1.0098	1.0273
35	0.2126	1.4424	1.3230	1.3677	1.0308	1.0114	1.0714
35	0.2757	1.3608	1.2868	1.3066	1.0503	1.0205	1.0624
35	0.3761	1.2581	1.2309	1.2022	1.0908	1.0428	1.1293
35	0.4541	1.1954	1.1891	1.1469	1.1311	1.0688	1.1853
35	0.5234	1.1492	1.1536	1.1007	1.1746	1.1003	1.2444
35	0.5821	1.1158	1.1251	1.0738	1.2183	1.1350	1.2570
35	0.6450	1.0850	1.0963	1.0492	1.2736	1.1828	1.3524
35	0.7043	1.0603	1.0713	1.0283	1.3359	1.2409	1.3634
35	0.7519	1.0434	1.0530	1.0178	1.3947	1.2994	1.3632
35	0.8026	1.0282	1.0357	1.0055	1.4679	1.3769	1.4446
35	0.8578	1.0152	1.0199	0.9967	1.5628	1.4848	1.5315
35	0.9257	1.0043	1.0060	0.9893	1.7078	1.6649	1.6577

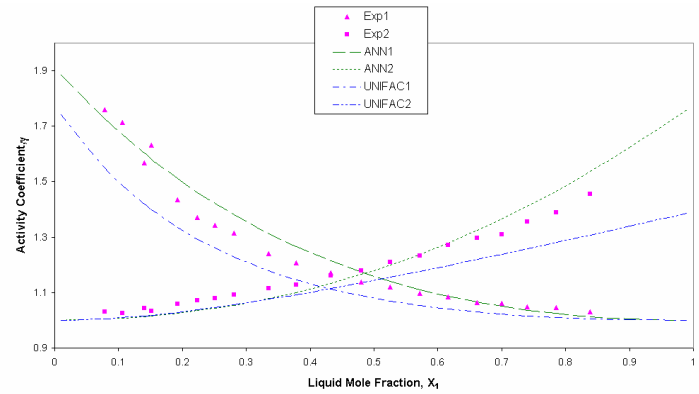


Figure 22. System 11: 2,4-Dimethylpentane (1) + Benzene (2)

This system was not included in the 921 systems compiled from DECHEMA.

Table 24. Activity Coefficients Results for System 11

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
77.3	0.078	1.7276	1.5491	1.7593	1.0041	1.0054	1.0312
77	0.106	1.6693	1.4863	1.7142	1.0076	1.0097	1.0257
76.5	0.14	1.6034	1.4203	1.5672	1.0133	1.0162	1.0430
76.3	0.152	1.5814	1.3992	1.6312	1.0157	1.0188	1.0327
76.1	0.192	1.5122	1.3359	1.4359	1.0252	1.0287	1.0585
75.9	0.224	1.4614	1.2925	1.3714	1.0344	1.0377	1.0711
75.7	0.251	1.4215	1.2601	1.3418	1.0434	1.0460	1.0802
75.5	0.281	1.3801	1.2280	1.3134	1.0546	1.0559	1.0908
75.4	0.335	1.3130	1.1789	1.2407	1.0783	1.0753	1.1155
75.4	0.378	1.2658	1.1467	1.2073	1.1004	1.0919	1.1277
75.2	0.432	1.2136	1.1133	1.1715	1.1323	1.1142	1.1610
75.4	0.48	1.1734	1.0889	1.1375	1.1648	1.1349	1.1781
75.3	0.525	1.1404	1.0700	1.1211	1.1988	1.1552	1.2085
75.5	0.572	1.1105	1.0536	1.0977	1.2381	1.1769	1.2333
75.5	0.616	1.0865	1.0409	1.0852	1.2784	1.1979	1.2698
75.9	0.662	1.0650	1.0300	1.0632	1.3243	1.2200	1.2968
76.1	0.700	1.0500	1.0227	1.0611	1.3652	1.2387	1.3089
76.3	0.74	1.0366	1.0163	1.0495	1.4110	1.2585	1.3541
76.6	0.785	1.0243	1.0106	1.0446	1.4661	1.2810	1.3897
77.2	0.839	1.0131	1.0056	1.0303	1.5369	1.3079	1.4555

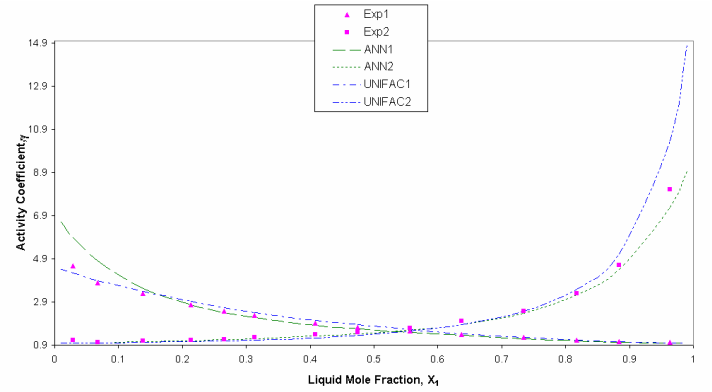


Figure 23. System 12: Cyclohexane (1) + Isopropyl Alcohol (2)
This system is included in the 921 systems compiled from DECHEMA as an isothermal system at a different temperature.

Table 25. Activity Coefficients Results for System 12

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
67.3	0.029	5.8966	4.2363	4.5675	1.0025	1.0010	1.1364
67.0	0.068	4.7907	3.8929	3.7706	1.0131	1.0053	1.0328
63.1	0.138	3.5330	3.3805	3.2967	1.0488	1.0225	1.0833
61.2	0.213	2.7561	2.9176	2.7533	1.1052	1.0557	1.1213
60.1	0.266	2.3999	2.6403	2.4872	1.1543	1.0901	1.1672
59.1	0.313	2.1646	2.4239	2.2774	1.2038	1.1294	1.2508
58.3	0.408	1.8285	2.0534	1.9303	1.3234	1.2412	1.3775
58.0	0.475	1.6574	1.8386	1.7335	1.4303	1.3552	1.5011
57.8	0.556	1.4901	1.6203	1.5438	1.6021	1.5512	1.6984
57.8	0.637	1.3495	1.4395	1.3758	1.8560	1.8488	2.0065
57.9	0.734	1.2068	1.2640	1.2500	2.3732	2.4592	2.4882
58.5	0.818	1.1066	1.1433	1.1363	3.2129	3.4869	3.2916
59.1	0.884	1.0471	1.0688	1.0686	4.4163	5.1331	4.6018
61.9	0.963	1.0054	1.0093	1.0192	7.2844	10.364	8.1042

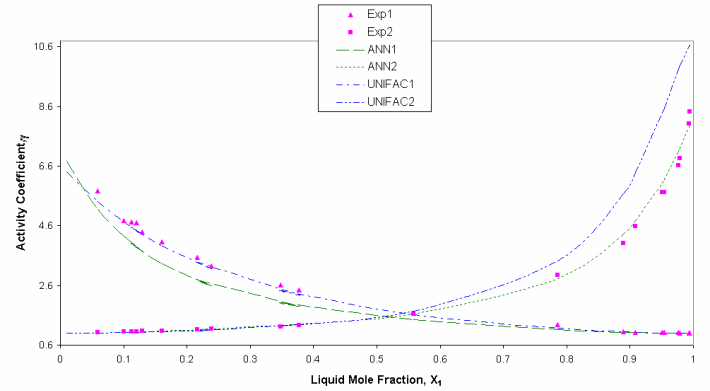


Figure 24. System 13: Ethylcyclohexane (1) + Isopropyl Alcohol (2)
This system was not included in the 921 systems compiled from DECHEMA.

Table 26. Activity Coefficients Results for System 13

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
66.5	0.059	5.1585	5.4078	5.7770	1.0099	1.0063	1.0344
66.25	0.100	4.2455	4.7320	4.7842	1.0270	1.0180	1.0583
66.34	0.120	3.8974	4.4444	4.7150	1.0379	1.0259	1.0538
66.3	0.130	3.7423	4.3109	4.3936	1.0439	1.0304	1.0666
66.3	0.113	4.0132	4.5422	4.7387	1.0339	1.0230	1.0558
66.4	0.161	3.3276	3.9310	4.0579	1.0650	1.0467	1.0770
66.5	0.239	2.6015	3.1712	3.2597	1.1321	1.1043	1.1418
66.4	0.216	2.7790	3.3709	3.5390	1.1103	1.0847	1.1188
67.7	0.377	1.9077	2.2803	2.4396	1.2976	1.2778	1.2537
67.5	0.348	2.0160	2.4316	2.6274	1.2575	1.2322	1.2134
68.8	0.558	1.4502	1.6084	1.6792	1.6506	1.7364	1.6517
70.8	0.786	1.1244	1.1613	1.2921	2.8320	3.4287	2.9582
77.5	0.89	1.0369	1.0485	1.0434	4.3350	5.6805	4.0146
78.3	0.909	1.0258	1.0344	1.0380	4.7710	6.3847	4.5875
84.1	0.951	1.0079	1.0106	1.0335	6.0371	8.3865	5.7276
85.1	0.954	1.0070	1.0094	1.0266	6.1476	8.5371	5.7134
92.1	0.977	1.0018	1.0024	1.0165	7.1086	9.8605	6.6209
92.9	0.979	1.0015	1.0020	1.0046	7.2028	9.9750	6.8637
101.6	0.994	1.0001	1.0002	1.0141	7.9729	10.6650	8.4350
101.1	0.993	1.0002	1.0002	1.0081	7.9178	10.6120	8.0210

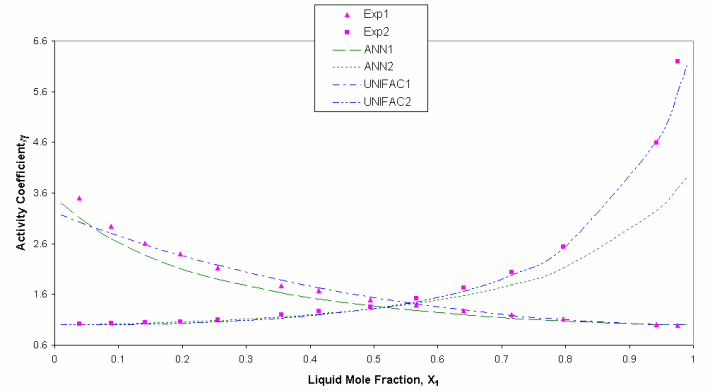


Figure 25. System 14: Benzene (1) + Isopropyl Alcohol (2)
This system is included in the 921 systems compiled from DECHEMA as an isothermal system at a different temperature.

Table 27. Activity Coefficients Results for System 14

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
69.5	0.039	3.1057	3.0254	3.5011	1.0025	1.0013	1.0210
67.1	0.089	2.6948	2.8018	2.9422	1.0122	1.0069	1.0392
65.4	0.142	2.3655	2.5814	2.6093	1.0295	1.0183	1.0500
63.9	0.197	2.1047	2.3733	2.3984	1.0543	1.0366	1.0652
62.9	0.255	1.8925	2.1739	2.1215	1.0874	1.0642	1.1086
61.8	0.355	1.6262	1.8768	1.7724	1.1620	1.1364	1.2058
61.0	0.414	1.5083	1.7275	1.6761	1.2179	1.1979	1.2677
60.9	0.495	1.3772	1.5476	1.4992	1.3137	1.3134	1.3528
60.3	0.566	1.2835	1.4141	1.3972	1.4228	1.4562	1.5141
60.2	0.640	1.2016	1.2949	1.2816	1.5730	1.6655	1.7310
60.1	0.716	1.1315	1.1930	1.1976	1.7860	1.9809	2.0360
60.3	0.797	1.0715	1.1061	1.1206	2.1170	2.5075	2.5360
63.0	0.942	1.0067	1.0105	1.0104	3.2457	4.6252	4.5911
64.7	0.976	1.0012	1.0019	0.9953	3.6938	5.5995	6.2019

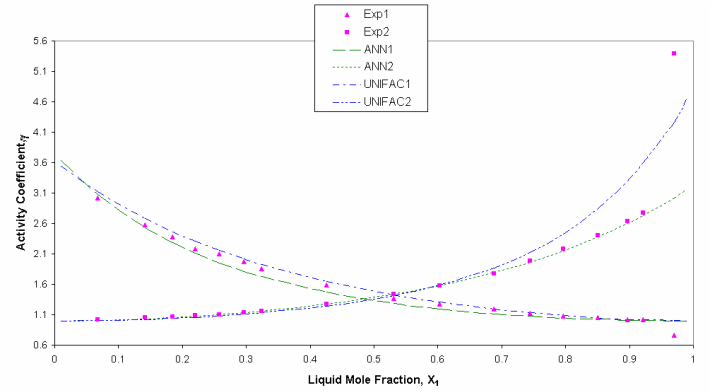


Figure 26. System 15: Toluene (1) + Isopropyl Alcohol (2)

This system was not included in the 921 systems compiled from DECHEMA. None of the prediction methods was able to predict the deviation peak of γ_{21} at a low X_2 value.

Table 28. Activity Coefficients Results for System 15

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
81.6	0.067	3.0777	3.1255	3.0223	1.0069	1.0052	1.0237
81.2	0.142	2.5265	2.6786	2.5740	1.0302	1.0236	1.0529
81.2	0.185	2.2816	2.4636	2.3817	1.0510	1.0404	1.0696
81.4	0.220	2.1118	2.3070	2.1813	1.0718	1.0578	1.0854
81.5	0.258	1.9523	2.1544	2.1034	1.0986	1.0807	1.0997
81.5	0.296	1.8148	2.0178	1.9673	1.1297	1.1082	1.1353
81.8	0.324	1.7254	1.9252	1.8546	1.1557	1.1316	1.1525
82.2	0.426	1.4680	1.6437	1.5846	1.2731	1.2439	1.2749
83.2	0.531	1.2856	1.4234	1.3654	1.4376	1.4178	1.4330
84.0	0.603	1.1949	1.3046	1.2723	1.5820	1.5873	1.5742
85.4	0.688	1.1151	1.1918	1.1935	1.7941	1.8664	1.7752
86.6	0.744	1.0758	1.1321	1.1279	1.9641	2.1188	1.9817
88.5	0.797	1.0469	1.0853	1.0778	2.1521	2.4286	2.1768
91.0	0.851	1.0250	1.0475	1.0584	2.3762	2.8441	2.4000
94.4	0.897	1.0119	1.0234	1.0244	2.5980	3.3017	2.6305
96.6	0.922	1.0068	1.0136	1.0242	2.7325	3.6029	2.7742
104.6	0.970	1.0010	1.0021	0.7619	3.0227	4.2603	5.3889

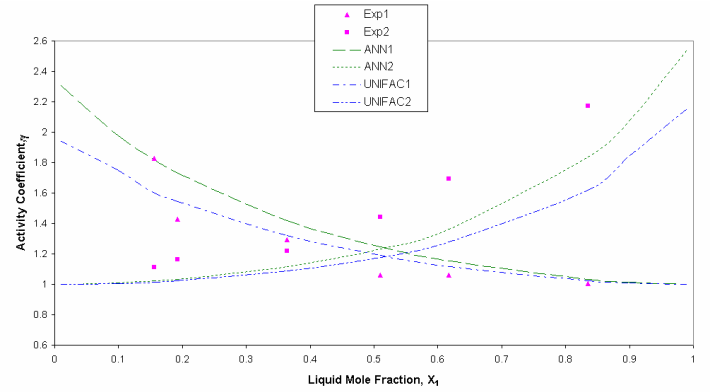


Figure 27. System 16: Ethyl Ether (1) + Acetone (2)
This system is included in the 921 systems compiled from DECHEMA as an isothermal system at a different temperature. The prediction for the activity coefficient of molecule 2 from the UNIFAC and ANN methods were close to each other but far from the experimental values.

Table 29. Activity Coefficients Results for System 16

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
0	0.156	1.8182	1.6010	1.8281	1.0214	1.0154	1.1136
0	0.192	1.7304	1.5429	1.4276	1.0321	1.0233	1.1643
0	0.364	1.4183	1.3220	1.2933	1.1140	1.0861	1.2191
0	0.510	1.2447	1.1901	1.0588	1.2330	1.1786	1.4402
0	0.617	1.1516	1.1178	1.0588	1.3634	1.2780	1.6908
0	0.835	1.0306	1.0238	1.0028	1.8394	1.6189	2.1738

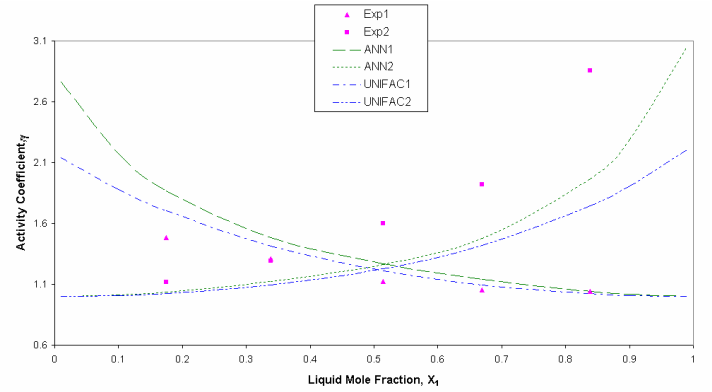


Figure 28. System 17: Acetone (1) + Isopropyl Alcohol (2)
This system is included in the 921 systems compiled from DECHEMA even though was taken from a different experimental database.

Table 30. Activity Coefficients Results for System 17

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
25	0.175	1.8681	1.7034	1.4823	1.0384	1.0238	1.1202
25	0.339	1.4848	1.4127	1.3108	1.1225	1.0923	1.2898
25	0.514	1.2678	1.2090	1.1242	1.2620	1.2265	1.6016
25	0.669	1.1395	1.0938	1.0516	1.4751	1.4180	1.9182
25	0.839	1.0397	1.0220	1.0409	1.9681	1.7471	2.8548

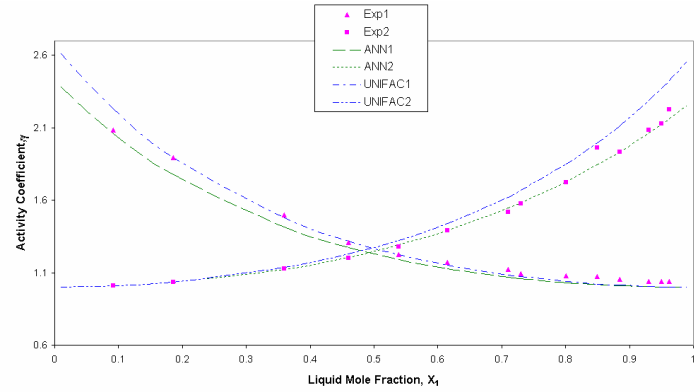


Figure 29. System 18: Isopropyl Alcohol (1) + Ethyl Acetate (2)

This system is included in the 921 systems compiled from DECHEMA even though was taken from a different experimental database.

Table 31. Activity Coefficients Results for System 18

Temp (ºC)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
40	0.0915	2.0621	2.2288	2.0862	1.0078	1.0086	1.0101
40	0.1860	1.7757	1.8928	1.8919	1.0324	1.0355	1.0331
40	0.3595	1.4155	1.4788	1.5003	1.1238	1.1357	1.1293
40	0.4600	1.2757	1.3197	1.3103	1.2078	1.2290	1.2020
40	0.5385	1.1922	1.2245	1.2235	1.2921	1.3243	1.2771
40	0.6150	1.1286	1.1515	1.1706	1.3924	1.4399	1.3895
40	0.7100	1.0699	1.0836	1.1233	1.5460	1.6225	1.5160
40	0.7300	1.0601	1.0721	1.0954	1.5829	1.6675	1.5749
40	0.8010	1.0319	1.0387	1.0771	1.7290	1.8492	1.7242
40	0.8500	1.0178	1.0219	1.0740	1.8446	1.9977	1.9641
40	0.8845	1.0105	1.0129	1.0532	1.9341	2.1156	1.9337
40	0.9300	1.0038	1.0047	1.0395	2.0634	2.2905	2.0824
40	0.9500	1.0019	1.0024	1.0372	2.1246	2.3752	2.1279
40	0.9620	1.0011	1.0014	1.0407	2.1627	2.4285	2.2248

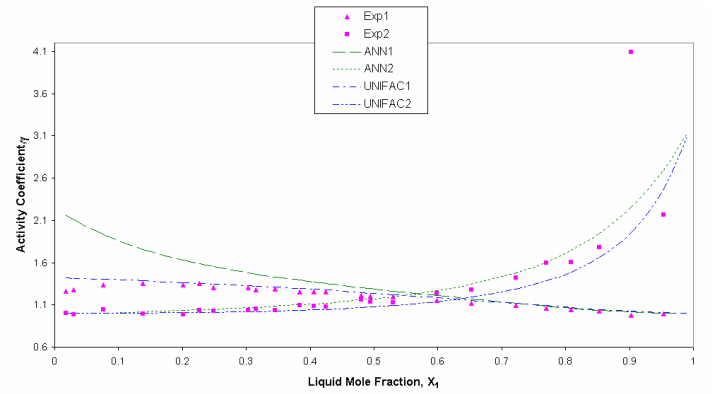


Figure 30. System 19: Chloroform (1) + n-Butyl Alcohol (2)
This system was not included in the 921 systems compiled from DECHEMA. None of the prediction methods was able to predict the deviation peak of γ_{21} at a low X_2 value.

Table 32. Activity Coefficients Results for System 19

Temp (°C)	x_1	γ_1			γ_2		
		ANN	UNIFAC	exp	ANN	UNIFAC	exp
115.1	0.0180	2.1604	1.4168	1.2632	1.0004	1.0000	0.9987
113.8	0.0300	2.1075	1.4136	1.2811	1.0010	1.0001	0.9855
106.4	0.0764	1.9334	1.4038	1.3392	1.0058	1.0009	1.0404
99.7	0.1388	1.7581	1.3855	1.3514	1.0173	1.0031	0.9932
93.4	0.2016	1.6285	1.3652	1.3401	1.0333	1.0073	0.9835
90.3	0.2272	1.5853	1.3571	1.3528	1.0409	1.0096	1.0354
89.3	0.2484	1.5527	1.3485	1.3062	1.0477	1.0119	1.0304
84.8	0.3036	1.4790	1.3272	1.3063	1.0672	1.0195	1.0352
84.4	0.3156	1.4647	1.3219	1.2770	1.0719	1.0215	1.0502
82.2	0.3452	1.4315	1.3096	1.2849	1.0841	1.0271	1.0330
80.0	0.3838	1.3917	1.2925	1.2541	1.1018	1.0360	1.0942
78.7	0.4060	1.3702	1.2825	1.2510	1.1130	1.0420	1.0894
77.5	0.4244	1.3531	1.2741	1.2564	1.1230	1.0475	1.0802
75.1	0.4798	1.3043	1.2470	1.2144	1.1576	1.0680	1.1595
74.6	0.4940	1.2922	1.2398	1.2065	1.1678	1.0743	1.1376
73.0	0.5300	1.2625	1.2216	1.1980	1.1968	1.0926	1.1293
70.8	0.598	1.2084	1.1854	1.1511	1.2668	1.1387	1.2364
69.2	0.6524	1.1670	1.1556	1.1225	1.3428	1.1906	1.2764
67.2	0.7220	1.1171	1.1171	1.0920	1.4788	1.2874	1.4204
66.3	0.7698	1.0857	1.0905	1.0591	1.6085	1.3843	1.5972
65.4	0.8084	1.0626	1.0698	1.0462	1.7435	1.4909	1.6087
64.5	0.8528	1.0393	1.0471	1.0269	1.9442	1.6614	1.7834
63.9	0.9032	1.0183	1.0241	0.9762	2.2540	1.9565	4.0908
62.6	0.9536	1.0045	1.0067	0.9932	2.6918	2.4646	2.1655

Appendix B Functional Groups Table

Table 33. Organic Functional Groups Table Based on Quantum Mechanical Calculations

Family name	Group #	Functional Group	Family name	Group #	Functional Group
Alkane	G1	C	Nitrile	G95	CH2CN
	G2	CH		G96	CH3CN
	G3	CH2	Nitro	G97	CNO2
	G4	CH3		G98	CHNO2
	G5	CH4		G99	CH2NO2
Alkene	G6	C=		G100	CH3NO2
	G7	CH=	Thiol	G101	CSH
	G8	CH2=		G102	CHSH
Alkyne	G9	Cº		G103	CH2SH
	G10	CHº		G104	CH3SH
Alcohol	G11	COH	Chloroalkane	G105	CCl
	G12	CHOH		G106	CHCl
	G13	CH2OH		G107	CH2Cl
	G14	CH3OH		G108	CH3Cl
Glycol	G15	GCOH	Fluoroalkane	G109	CHF
	G16	GCHOH		G110	CH2F2
	G17	gCH2OH		G111	CF
Ether	G18	COC		G112	CF2
	G19	CHOCH		G113	CF3
	G20	CH2OCH2		G114	CHF2
	G21	CH3OCH3		G115	CHF3
	G22	COCH		G116	CH3F
	G23	COCH2		G117	CH2F
	G24	COCH3	Chlorofluoroalkane	G118	CClF2
	G25	CHOCH2		G119	CCl2F
	G26	CHOCH3		G120	CClF3
	G27	CH2OCH3		G121	CCl3F
	G28	CH3OC		G122	CHClF
	G29	CH3OCH		G123	CHCl2
	G30	CH3OCH2		G124	CHCl3
	G31	CH2OC		G125	CCl2
	G32	CH2OCH		G126	CCl3
	G33	CHOC		G127	CHClF2

Family name	Group #	Functional Group	Family name	Group #	Functional Group
Aldehyde	G34	HCHO		G128	CHCl2F
	G35	CCHO		G129	CClF
	G36	CHCHO		G130	CH2ClF
	G37	CH2CHO		G131	CH2Cl2
	G38	CH3CHO		G132	CCl2F2
Ketone	G39	CCOC		G133	CCl4
	G40	CHCOCH	Arene	G134	aC
	G41	CH2COCH2		G135	aCH
	G42	CH3COCH3		G136	aCOH
	G43	CCOCH		G137	(aCH)2aCNH2
	G44	CCOCH2		G138	(aCH)2aCNO2
	G45	CCOCH3		G139	(aCH)2aCCl
	G46	CHCOCH2		G140	aCHNaCH
	G47	CHCOCH3	Amide(1)	G141	CCONH2
	G48	CH2COCH3		G142	CHCONH2
	G49	CH3COC		G143	CH2CONH2
	G50	CH3COCH		G144	CH3CONH2
	G51	CH3COCH2	Amide(2)	G145	CCONHC
	G52	CH2COC		G146	CHCONHCH
	G53	CH2COCH		G147	CH2CONHCH2
	G54	CHCOC		G148	CH3CONHCH3
Carboxylic Acid	G55	HCOOH		G149	CCONHCH
	G56	CCOOH		G150	CCONHCH2
	G57	CHCOOH		G151	CCONHCH3
	G58	CH2COOH		G152	CHCONHCH2
	G59	CH3COOH		G153	CHCONHCH3
Ester(1)	G60	HCOOC		G154	CH2CONHCH3
	G61	HCOOCH		G155	CH3CONHC
	G62	HCOOCH2		G156	CH3CONHCH
	G63	HCOOCH3		G157	CH3CONHCH2
Ester(2)	G64	CCOOC		G158	CH2CONHC
	G65	CHCOOCH		G159	CH2CONHCH

Family name	Group #	Functional Group	Family name	Group #	Functional Group
	G66	CH2COOCH2		G160	CHCONHC
	G67	CH3COOCH3	Amide(3)	G161	CCONC2
	G68	CCOOCH		G162	CHCON(CH)2
	G69	CCOOCH2		G163	CH2CON(CH2)2
	G70	CCOOCH3		G164	CH3CON(CH3)2
	G71	CHCOOCH2		G165	CCON(CH)2
	G72	CHCOOCH3		G166	CCON(CH2)2
	G73	CH2COOCH3		G167	CCON(CH3)2
	G74	CH3COOC		G168	CHCON(CH2)2
	G75	CH3COOCH		G169	CHCON(CH3)2
	G76	CH3COOCH2		G170	CH2CON(CH3)2
	G77	CH2COOC		G171	CH3CONC2
	G78	CH2COOCH		G172	CH3CON(CH)2
	G79	CHCOOC		G173	CH3CON(CH2)2
Amine(1)	G80	CNH2		G174	CH2CONC2
	G81	CHNH2		G175	CH2CON(CH)2
	G82	CH2NH2		G176	CHCONC2
	G83	CH3NH3	Water	G177	H2O
Amine(2)	G84	C2NH	Furan	G178	Furan
	G85	(CH)2NH	DMF	G179	DMF
	G86	(CH2)2NH	Carbon Monoxide	G180	CO
	G87	(CH3)2NH	Carbon Dioxide	G181	CO2
Amine(3)	G88	C3N	Carbon Disulfide	G182	CS2
	G89	(CH)3N	Pyrrolidone	G183	CH2NHC=0
	G90	(CH2)3N	Vinyl Chloride	G184	CH2=CHCl
	G91	(CH3)3N	3,3,3-Trifluoropropene	G185	CF3CH=CH2
Hydrogen Cyanide	G92	HCN	Methoxide	G186	OCH3
Nitrile	G93	CCN			
	G94	CHCN			

Appendix C MatLab Programs

The first MatLab program developed was called ***Create.m*** and it purpose is to organize the vapor-liquid equilibrium experimental data within the information matrix. The following diagram shows the algorithm developed to perform the mentioned task.

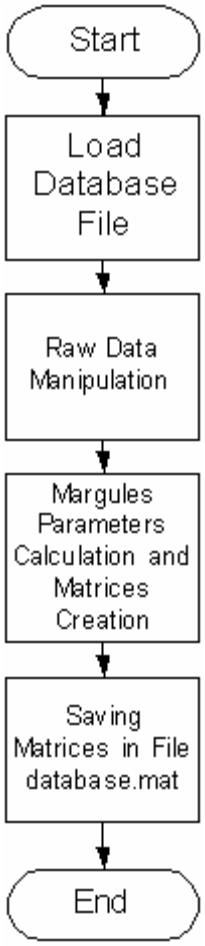


Figure 31. Algorithm flowchart for Create.m

The vapor-liquid equilibrium experimental data was entered into a spreadsheet using the format specified in the following table.

Table 34. Experimental raw data spreadsheet format

System #	0	0	0	0	0	0	0
0	Temp	A12	A21	alfa12	0	0	0
0	A1	B1	C1	0	0	0	0
0	A2	B2	B3	0	0	0	0
0	G11	G12	G13	G14	G15	G16	G17
0	n11	n12	n13	n14	n15	n16	n17
0	G21	G22	G23	G24	G25	G26	G27
0	n21	n22	n23	n24	n25	n26	n27
0	P1	X1	Y1	T1	□U11	□U21	0
0	P2	X2	Y2	T2	□U12	□U22	0
0	P3	X3	Y3	T3	□U13	□U23	0
0	P4	X4	Y4	T4	□U14	□U24	0
0	P5	X5	Y5	T5	□U15	□U25	0
0	P6	X6	Y6	T6	□U16	□U26	0
0	Pn	Xn	Yn	Tn	□U1n	□U2n	0
0	0	0	0	0	0	0	0

This table presents the format used to organize the experimental vapor-liquid equilibrium data taken from DECHEMA.

where,

System # is the data system number.

Temp is the temperature of the system if isothermal in ºC.

A12, A21and alfa12 are the NRTL model constants.

A1, B1 and C1 are the Antoine constants for molecule 1.

A2, B2 and C2 are the Antoine constants for molecule 2.

G11 to G17 are the number of the functional groups for molecule 1.

n11 to n17 are the quantity of each functional group for molecule 1.

G21 to G27 are the number of the functional groups for molecule 2.

n21 to n27 are the quantity of each functional group for molecule 2.

P1 to Pn are the system pressure for each data point.

X1 to Xn are the liquid mol fraction for molecule 1 for each data point.

Y1 to Yn are the vapor mol fraction for molecule 1 for each data point.

T1 to Tn are the system temperature for each data point.

$\gamma U11$ *to* $\gamma U1n$ are the activity coefficient predicted by UNIFAC for molecule 1 for each data point.

$\gamma U21$ *to* $\gamma U2n$ are the activity coefficient predicted by UNIFAC for molecule 2 for each data point.

Filling all unused cells of the spreadsheet data table with zeros is required by MatLab to load the file. The MatLab program reads the data file, process the experimental raw data and creates the information matrix. The Create.m program uses the MatLab function ***fminunc*** and the function ***Createfun_Margules3.m*** to calculate the Margules equation parameters. The information matrix is saved in a file to be used by the second MatLab program. The MatLab code for program ***Create.m*** is presented below.

```
% Program Create.m
   % This program will create the data base for the activity
   coefficients
   % estimation program.
   % The raw data was taken from DECHEMA data series.

   clear
   clc

   % Load the data file where experimental data is contain.
   filename = input('Enter the experimental data file name:','s');
   s = ['load ' filename '.txt'];
   disp('A large data file is loading. This may take a few minutes.')
   disp('Please wait!')
   eval(s);
   disp('Data had been loaded!')
   g = 186;
   MW = [12 13    14    15    16    12    13    14    12    13    29
      30    31    32    29    30    31    40    42    44    46    41
      42    43    43    44    45    43    44    45    42 43 41 30 41 42
      43 44 52    54    56    58    53    54    55    55    56    57
      55    56    57    54    55    53    46    59    58    59    60
      57    58    59    60    68    70    72    74    69    70    71
      71    72    74    71    72    73    70    71    69    28    29
```

```
   30    31    39    41    43    45    50    52    56    59    27
   38    39    40    41    58    59    60    61    45    46    47
   48    47.5  48.5  49.5  50.5  32    52    31    50    69    51
   70    34    33    85.5  102   104.5 137.5 67.5  84    119.5 83
   118.5 86.5  103   66.5  68.5  85    121   154   12    13    29
   54    84    73.5  40    56    57    58    59    67    69    71
   73    68    69    70    71    72    73    70    71    72    69
   70    68    66    81    85    87    80    82    84    83    85
   86    81    83    85    80    82    79    18    68    73    28
   44    32    57    62.5  96 31];
s2 = ['data = ' filename ';'];
eval(s2)
% Raw data manipulation
systems = [nonzeros(data(:,1)) find(data(:,1))];
systems_Margules3 = [nonzeros(data(:,1)) find(data(:,1))];
systems_Margules3 = [systems_Margules3; length(systems_Margules3)+1
length(data(:,1))];
X = [];
X1mol = [];
X2mol = [];
Y1mol = [];
Y2mol = [];
GF1 = [];
GF2 = [];
Antoine1 = [];
Antoine2 = [];
Gammas1E = [];
Gammas2E = [];
Gammas1U = [];
Gammas2U = [];
P_exp = [];
T_exp = [];
Max = max(systems(:,1));
for i = 1:length(systems(:,1))
   S = int2str(i);
   M = num2str(Max);
   %clc
   sysnumber = ['Manipulating data of system #' S ' of ' M '.'];
   disp(sysnumber);

%  X, Y and Gammas matrices creation

 % Margules equation parameters calculation
   X1_Margules3 = [];
   X2_Margules3 = [];
   Y1_Margules3 = [];
   Y2_Margules3 = [];
   T_Margules3 = [];
   P_Margules3 = [];
   Antoine1_Margules3 = [];
   Antoine2_Margules3 = [];
   points_Margules3(i) = systems_Margules3(i+1,2) -
systems_Margules3(i,2) - 8;
   X1_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,3)';
   X2_Margules3 = 1-X1_Margules3;
   Y1_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,4)';
```

```
   Y2_Margules3 = 1-Y1_Margules3;
   T_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,5)';
   P_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,2)';
   Gamma1U =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,6)';
   Gamma2U =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,7)';
   Antoine1_Margules3 =
[nonzeros(data(systems_Margules3(i,2)+2,:))'];
   Antoine2_Margules3 =
[nonzeros(data(systems_Margules3(i,2)+3,:))'];
   if
or(or(X1_Margules3(1)==0,Y1_Margules3(1)==0),or(X1_Margules3(1)==1,Y
1_Margules3(1)==1)) == 1
      X1_Margules3(1) = [];
      X2_Margules3(1) = [];
      Y1_Margules3(1) = [];
      Y2_Margules3(1) = [];
      T_Margules3(1) = [];
      P_Margules3(1) = [];
      Gamma1U(1) = [];
      Gamma2U(1) = [];
      points_Margules3(i) = points_Margules3(i)-1;
   else
   end
   if
or(or(X1_Margules3(points_Margules3(i))==0,Y1_Margules3(points_Margu
les3(i))==0),or(X1_Margules3(points_Margules3(i))==1,Y1_Margules3(po
ints_Margules3(i))==1)) == 1
      X1_Margules3(points_Margules3(i)) = [];
      X2_Margules3(points_Margules3(i)) = [];
      Y1_Margules3(points_Margules3(i)) = [];
      Y2_Margules3(points_Margules3(i)) = [];
      T_Margules3(points_Margules3(i)) = [];
      P_Margules3(points_Margules3(i)) = [];
      Gamma1U(points_Margules3(i)) = [];
      Gamma2U(points_Margules3(i)) = [];
      points_Margules3(i) = points_Margules3(i)-1;
   else
   end
 % Calculate Gamma i experimental
   Gamma1E_Margules3 =
(Y1_Margules3.*P_Margules3)./(X1_Margules3.*10.^(Antoine1_Margules3(
1)-(Antoine1_Margules3(2)./(T_Margules3+Antoine1_Margules3(3)))));
   Gamma2E_Margules3 =
(Y2_Margules3.*P_Margules3)./(X2_Margules3.*10.^(Antoine2_Margules3(
1)-(Antoine2_Margules3(2)./(T_Margules3+Antoine2_Margules3(3)))));
   GeRT_exp_Margules3 = (X1_Margules3.*log(Gamma1E_Margules3)) +
(X2_Margules3.*log(Gamma2E_Margules3));
   Gamma1E_Gamma2E = [Gamma1E_Margules3 Gamma2E_Margules3];
   save expdata X1_Margules3 X2_Margules3 T_Margules3
Gamma1E_Margules3 Gamma2E_Margules3 GeRT_exp_Margules3
Gamma1E_Gamma2E
   options = optimset('MaxFunEvals',1000,'MaxIter',1000);
```

```
   A0_Margules3 = [1;1;1];
   A_Margules3_fminunc(i,:) =
fminunc('Createfun_Margules3',A0_Margules3);
   Antoine1 = [Antoine1;Antoine1_Margules3];
   Antoine2 = [Antoine2;Antoine2_Margules3];
   Gammas1E = [Gammas1E;Gamma1E_Margules3'];
   Gammas2E = [Gammas2E;Gamma2E_Margules3'];
   X1mol = [X1mol;X1_Margules3'];
   X2mol = [X2mol;X2_Margules3'];
   Y1mol = [Y1mol;Y1_Margules3'];
   Y2mol = [Y2mol;Y2_Margules3'];
   Gammas1U = [Gammas1U;Gamma1U'];
   Gammas2U = [Gammas2U;Gamma2U'];
   P_exp = [P_exp;P_Margules3'];
   T_exp = [T_exp;T_Margules3'];
   P = data(systems(i,2)+8,2);
   points = 0;
   point = 0;
   while P ~= 0
      xpoint = 1;
      ypoint = 1;
      if data(systems(i,2)+point+8,3) == 0
         xpoint = 0;
      end
      if data(systems(i,2)+point+8,4) == 0
         ypoint = 0;
      end
      if data(systems(i,2)+point+8,3) == 1
         xpoint = 0;
      end
      if data(systems(i,2)+point+8,4) == 1
         ypoint = 0;
      end
      if and(xpoint,ypoint) == 1
         points = points + 1;
      else
      end
      point = point + 1;
      P = data(systems(i,2)+8+point,2);
   end
   systems(i,3) = points;
   NGF1 = length(nonzeros(data(systems(i,2)+4,:)));
   NGF2 = length(nonzeros(data(systems(i,2)+6,:)));
   [o,p] = size(X);
   k = 1;
   l = 0;

 % Suma de la cantidad total de interacciones.
   sumINT(o+1) = 0;
   sumINT(o+2) = 0;
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            sumINT(o+1) = sumINT(o+1) +
(data(systems(i,2)+5,r+1)*data(systems(i,2)+7,s+1));
            sumINT(o+2) = sumINT(o+2) +
(data(systems(i,2)+5,r+1)*data(systems(i,2)+7,s+1));
         else
         end
      end
```

```
   end

 % 1)Posisionar el numero de la interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            GF1 = [GF1;data(systems(i,2)+4,r+1)];
            GF2 = [GF2;data(systems(i,2)+6,s+1)];
            X(o+1,l+k) = g * (data(systems(i,2)+4,r+1) - 1) +
data(systems(i,2)+6,s+1);
            X(o+2,l+k) = g * (data(systems(i,2)+6,s+1) - 1) +
data(systems(i,2)+4,r+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
 % 2)Posisionar la Temperatura del sistema.
   if systems(i,1) > 943
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(T_Margules3);
               X(o+2,l+k) = mean(T_Margules3);
               l = l + 8;
               else
               end
            end
         end
   elseif systems(i,1) > 921
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(T_Margules3)-273.15;
               X(o+2,l+k) = mean(T_Margules3)-273.15;
               l = l + 8;
            else
            end
         end
      end
   else
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(T_Margules3);
               X(o+2,l+k) = mean(T_Margules3);
               l = l + 8;
            else
            end
         end
      end
   end
   k = k + 1;
   l = 0;
 % 3)Posisionar la Presión del sistema.
   if systems(i,1) > 943
      for r = 1:NGF1
         for s = 1:NGF2
```

```
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(P_Margules3);
               X(o+2,l+k) = mean(P_Margules3);
               l = l + 8;
            else
            end
         end
      end
   elseif systems(i,1) > 921
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(P_Margules3) * 7.500617;
               X(o+2,l+k) = mean(P_Margules3) * 7.500617;
               l = l + 8;
            else
            end
         end
      end
   else
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
               X(o+1,l+k) = mean(P_Margules3);
               X(o+2,l+k) = mean(P_Margules3);
               l = l + 8;
            else
            end
         end
      end
   end
   k = k + 1;
   l = 0;
 % 4)Posisionar la cantidad de GF de la molécula #1 en la
interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            X(o+1,l+k) = data(systems(i,2)+5,r+1);
            X(o+2,l+k) = data(systems(i,2)+7,s+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
 % 5)Posisionar la cantidad de GF de la molécula #2 en la
interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            X(o+1,l+k) = data(systems(i,2)+7,s+1);
            X(o+2,l+k) = data(systems(i,2)+5,r+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
```

```
   l = 0;
   sumMW1 = 0;
   sumMW2 = 0;
 % 6)Posisionar el peso molecular del grupo funcional de la molécula
#1.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            X(o+1,l+k) =
data(systems(i,2)+5,r+1)*MW(data(systems(i,2)+4,r+1));
            X(o+2,l+k) =
data(systems(i,2)+7,s+1)*MW(data(systems(i,2)+6,s+1));
            l = l + 8;
         else
         end
      end
      sumMW1 = sumMW1 +
data(systems(i,2)+5,r+1)*MW(data(systems(i,2)+4,r+1));
   end
   k = k + 1;
   l = 0;
 % 7)Posisionar el peso molecular del grupo funcional de la molécula
#2.
   for s = 1:NGF2
      for r = 1:NGF1
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            X(o+1,l+k) =
data(systems(i,2)+7,s+1)*MW(data(systems(i,2)+6,s+1));
            X(o+2,l+k) =
data(systems(i,2)+5,r+1)*MW(data(systems(i,2)+4,r+1));
            l = l + 8;
         else
         end
      end
      sumMW2 = sumMW2 +
data(systems(i,2)+7,s+1)*MW(data(systems(i,2)+6,s+1));
   end
   k = k + 1;
   l = 0;
 % 8)Posisionar 1
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+4,r+1) ~= data(systems(i,2)+6,s+1)
            X(o+1,l+k) = 1;
            X(o+2,l+k) = 1;
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
end
disp('End of experimental data manipulation process!')
disp('Matrices X Y and Gammas had been created succesfully!')
disp('Saving data matrices to file Databasenew_Margules3.mat')
save Databasenew_Margules3 X X1mol X2mol Y1mol Y2mol systems
Antoine1 Antoine2 Gammas1E Gammas2E Gammas1U Gammas2U sumINT
A_Margules3_fminunc P_exp T_exp
disp('Data had been save succesfully!')
```

```
% Program Createfun_Margules3.m
   % This program will determine the Margules constants A, B and C for
   liquid
   % binary systems.

   function SSE = Createfun_Margules3(A)
   load expdata
   A1 = A(1);
   B = A(2);
   C = A(3);
   X1 = X1_Margules3;
   X2 = X2_Margules3;
   GeRT = (A1+B.*(X1-X2)+C.*(X1-X2).^2);
   Gamma1M = exp(((A1+3*B+5*C).*(X2.^2))-
   ((4*(B+4*C)).*(X2.^3))+((12*C).*(X2.^4)));
   Gamma2M = exp(((A1-3*B+5*C).*(X1.^2))+((4*(B-
   4*C)).*(X1.^3))+((12*C).*(X1.^4)));
   Gamma1_Gamma2 = [Gamma1M Gamma2M];
   delta = Gamma1_Gamma2 - Gamma1E_Gamma2E;
   SSE = diag(delta * delta');
```

The second MatLab program developed was called ***NNvsUNIFAC.m*** and it purpose is to search and choose the required vapor-liquid equilibrium experimental data within the information matrix, create the training matrices, train the neural network, predict the Margules parameters and, calculate the activity coefficients using the four-suffix Margules equation. The following diagram shows the algorithm developed to perform the mentioned tasks.

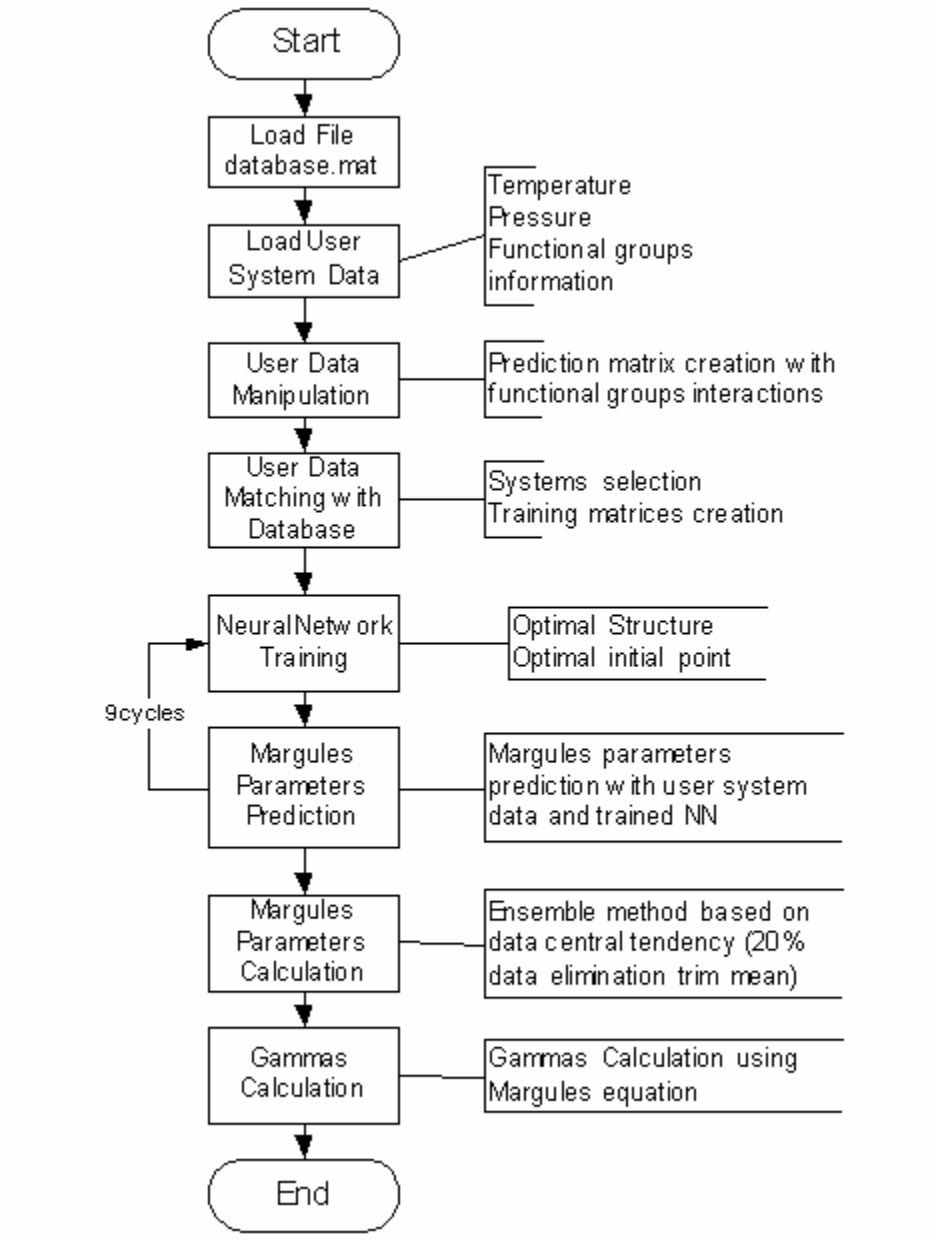


Figure 32. Algorithm flowchart for NNvsUNIFAC.m

The program saves the predicted activity coefficients and plots a graph comparing the results given by the ANN and UNIFAC methods with the experimental data. The MatLab code for program ***NNvsUNIFAC.m*** is presented below.

```
%Program NNvsUNIFAC.m
   % This program will compare the results of predicting the activity
   coefficients using the UNIFAC
   % and the Neural Networks methods based on the experimental data
   taken from DECHEMA series.

     clear
     clc
     close all
     warning off all
     totalsys = 1000;
     Predsys1 = [];
     Predsys2 = [];

   % Specify system(s) to be evaluated
   % evalsys = [960 954 955 957 944 950 951 948 949];
   % evalsys = [22 50   74    202   358   366   550   736   771   772];
     evalsys = input('Enter the experimental data system number:');
     for loop = 1:length(evalsys)
        sys = evalsys(loop);
        tic % Start time counter.
      % Load the experimental database file from DECHEMA.
        load Databasenew_Margules3.mat
        sse3 = [];
        [f,c] = size(X1mol);

      % Determine systems data position in Matrix X.
        systems(1,4) = 1;
        systems(1,5) = 1;
        for i = 2:length(systems(:,1))
           systems(i,4) = systems(i-1,4) + 2;
           systems(i,5) = systems(i-1,5) + systems(i-1,3);
        end
        A_Margules3_eval = [];
        tfsys = [];
        nneusys = [];
        Gammaspred = [];
        GammasT = [];
        errorssys = [];
        Xevalc = [];
        Xeval = [];
        Xexp = [];
        Yeval = [];
        Y1exp = [];
        Y2exp = [];
        Peval = [];
        Pexp = [];
        Teval = [];
        Texp = [];
        Gamma1Ueval = [];
        Gamma2Ueval = [];
        Antoine1eval = [];
        Antoine2eval = [];
        TPeval = [];
        TPexp = [];
        X1eval = [];
        X2eval = [];
        X1X2exp = [];
        Gammaseval = [];
        Gammasexp = [];
```

```
     systemseval = [];
     systemsexp = [];
     IntXeval1 = [];
     IntXeval2 = [];
     S = num2str(sys);
     sysnumber = ['Evaluating system #' S '.'];
     disp(sysnumber);
     Xeval = X(systems(sys,4):systems(sys,4) + 1,:); % Matriz de
evaluacion = sistema extraido.
     IntXeval = Xeval(:,1:8:length(Xeval(1,:)));
     Xevalc = X(systems(sys,4):systems(sys,4) +
1,1:8*length(nonzeros(IntXeval(1,:)))); % Matriz de evaluacion con
la cantidad de interacciones del sistema a ser extraido.
     Xexp = X(1:2*921, :);
     Y1eval = Y1mol(systems(sys,5):systems(sys,5) + systems(sys,3) -
1);   % Vector de fracciones molares de fase vapor del sistema a
extraer.
     Y2eval = Y2mol(systems(sys,5):systems(sys,5) + systems(sys,3) -
1);   % Vector de fracciones molares de fase vapor del sistema a
extraer.
     Y1exp = Y1mol(1:f-740);
     Y2exp = Y2mol(1:f-740);
     Peval = X(systems(sys,4),3);
     Peval_Y = P_exp(systems(sys,5):systems(sys,5) + systems(sys,3)-
1);
     Pexp = X(1:2*921,3);
     Teval = X(systems(sys,4),2);
     Teval_Y = T_exp(systems(sys,5):systems(sys,5) + systems(sys,3)-
1);
     Texp = X(1:2*921,2);
     Gamma1Ueval = Gammas1U(systems(sys,5):systems(sys,5) +
systems(sys,3) - 1);
     Gamma2Ueval = Gammas2U(systems(sys,5):systems(sys,5) +
systems(sys,3) - 1);
     Antoine1eval = Antoine1(sys,:);
     Antoine2eval = Antoine2(sys,:);
     A_Margules3_eval = A_Margules3_fminunc(sys,:);
     sumINTexp = sumINT(1:2*921);   % Vector de cantidad de
interacciones del sistema a extraer.
     sumINTeval = sumINT(systems(sys,4):systems(sys,4) + 1);
     X1eval = X1mol(systems(sys,5):systems(sys,5) + systems(sys,3) -
1);
     X2eval = X2mol(systems(sys,5):systems(sys,5) + systems(sys,3) -
1);
     Gammas1Eeval = Gammas1E(systems(sys,5):systems(sys,5) +
systems(sys,3) - 1); % Vector de coeficientes de actividad
experimentales del sistema a extraer.
     Gammas2Eeval = Gammas2E(systems(sys,5):systems(sys,5) +
systems(sys,3) - 1); % Vector de coeficientes de actividad
experimentales del sistema a extraer.
     Gammas1Eexp = Gammas1E(1:f-740);
     Gammas2Eexp = Gammas2E(1:f-740);
     systemseval = systems(sys,:); % Matriz de informacion del
sistema a extraer.
     systemsexp = systems(1:length(systems(:,1))-39,:);
     IntXeval1 = Xevalc(1,1:8:length(Xevalc(1,:))); % Interacciones
en Xevalc parte de Gamma1.
     IntXeval1 = reshape(IntXeval1,1,prod(size(IntXeval1))); %
Arreglo de las interacciones en un solo vector.
```

```
     IntXeval1 = unique(nonzeros(IntXeval1)); % Interacciones
presentes en el sistema de interes para determinar Gamma1.
     IntXeval2 = Xevalc(2,1:8:length(Xevalc(2,:))); % Interacciones
en Xevalc parte de Gamma2.
     IntXeval2 = reshape(IntXeval2,1,prod(size(IntXeval2))); %
Arreglo de las interacciones en un solo vector.
     IntXeval2 = unique(nonzeros(IntXeval2)); % Interacciones
presentes en el sistema de interes para determinar Gamma2.

   % Training Matrices Creation
     message = ['Creating Training Matrices for system ' S '!'];
     disp(message);
     Xe1 = []; % Experimental Data Training Matrix for Gamma1.
     TG1 = []; % Targets Matrix for Gamma1.
     TA1 = [];
     TY1 = [];
     T1 = [];
     P1 = [];
     sumINTe1 = [];
     sumINTe2 = [];
     monitor1 = [];
     Xe2 = []; % Experimental Data Training Matrix for Gamma2.
     TG2 = []; % Targets Matrix for Gamma2.
     TA2 = [];
     TY2 = [];
     T2 = [];
     P2 = [];
     monitor2 = [];
     [f,c] = size(Xexp); % Experimental Matrix rows and columns.
     int = c/8; % Maximum number of interactions for systems in the
Experimental Matrix.
     [fp,cp] = size(Xevalc); % Evaluation Matrix rows and columns.
     intp = cp/8; % Number of interactions for the system of
interest.

   % Training Matrix Creation for Gamma1
     monitor1 = zeros(1,length(IntXeval1));
     L = 1;
     o = 0;
     p = 0;
     s = 0;
     t = 0;
     SysXe1 = []; % Chosen systems matrix to determine Gamma1.
     SysXe2 = []; % Chosen systems matrix to determine Gamma2.
     for i = 1:length(systemsexp(:,1)) % Evaluation of all available
systems in the Experimental Matix.
         s = s + 1;
         t = t + 1;
         while s <= 2
            p = 0;
            for j = 1:intp
               L = 1;
               if s <= 2
                  for m = 1:int
                     if and(Xexp(t,L) ==
IntXeval1(j),monitor1(j)<totalsys)
                        o = o + 1;
                        monitor1(j) = monitor1(j)+1;
                        SysXe1(o,:) = systemsexp(i,:);
                        Xe1(o,:) = Xexp(t,:);
```

```
                        TG1(o) = Gammas1Eexp(t);
                        TA1(o,:) = A_Margules3_fminunc(i,:);
                        TY1(o) = Y1exp(t);
                        T1(o) = Texp(t);
                        P1(o) = Pexp(t);
                        sumINTe1(o) = sumINTexp(t);
                        L = 1;
                        p = 1;
                        break
                     else
                        L = L + 8;
                     end
                  end
               else
                  break
               end
               if p == 1
                  break
               end
            end
            s = s + 1;
            t = t + 1;
         end
         s = 0;
         t = t - 1;
      end

   % Training Matrix Creation for Gamma2
      monitor2 = zeros(1,length(IntXeval2));
      L = 1;
      o = 0;
      p = 0;
      s = 0;
      t = 0;
      for i = 1:length(systemsexp(:,1)) % Evaluation of all available
systems in the Experimental Matix.
         s = s + 1;
         t = t + 1;
         k = 1;
         while s <= 2
            p = 0;
            k = 1;
            for j = 1:intp
               L = 1;
               if s <= 2
                  for m = 1:int
                     if and(Xexp(t,L) ==
IntXeval2(j),monitor2(j)<totalsys)
                        o = o + 1;
                        monitor2(j) = monitor2(j)+1;
                        SysXe2(o,:) = systemsexp(i,:);
                        Xe2(o,:) = Xexp(t,:);
                        TG2(o) = Gammas2Eexp(t);
                        TA2(o,:) = A_Margules3_fminunc(i,:);
                        TY2(o) = Y2exp(t);
                        T2(o) = Texp(t);
                        P2(o) = Pexp(t);
                        sumINTe2(o) = sumINTexp(t);
                        L = 1;
                        p = 1;
```

```
                    break
                 else
                    L = L + 8;
                 end
              end
           else
              break
           end
           if p == 1
              break
           end
        end
        s = s + 1;
        t = t + 1;
     end
     s = 0;
     t = t - 1;
  end

  % Training Matrices manipulation and re-arrangement
    if isempty(Xe2) == 0
       Xe12 = [Xe1 Xe2];
       Xeval12 = [Xevalc(1,:) Xevalc(2,:)];
       sumINTe12 = sumINTe1 + sumINTe2;
       sumINTeval12 = sumINTeval(1) + sumINTeval(2);
       IntXe12 = Xe12(:,1:8:length(Xe12(1,:))); % Interactions in
Xe12.
       IntXe12 = reshape(IntXe12,1,prod(size(IntXe12))); % Re-
arrangement in one vector for interaction in Xe12.
       IntXeval12 = Xeval12(1,1:8:length(Xeval12(1,:))); %
Interactions in Xeval12 to determine Margules equation parameters.
       IntXeval12 = reshape(IntXeval12,1,prod(size(IntXeval12))); %
Re-arrangement in one vector for interactions in Xeval12.
       Intjun12 =
unique(union(nonzeros(IntXe12)',nonzeros(IntXeval12)')); % All
interactions toghether.
       Xe12mod = zeros(length(Xe12(:,1)),length(Intjun12)*8); %
Initialization for Modified Experimental Matrix.
       Xeval12mod = zeros(1,length(Intjun12)*8); % Initialization
for Modified Evaluation Matrix.

     % Verifing that all interactions from the system of interest
are
     % present in the Experimental Matrix
       IntXe12 = unique(nonzeros(IntXe12)); % Interactions present
in the experimental data to determine Margules parameters.
       IntXeval12 = unique(nonzeros(IntXeval12)); % Interactions
present in the system of interest to determine Margules parameters.
       tf12 = ismember(IntXeval12,IntXe12); % Interactions from the
system of interest present in the experimental data to determine
Margules parameters.
       Intfound12 = sum(tf12); % Interactions from the system of
interest found in the experimental data to determine Margules
parameters.
       Ints12 = length(tf12); % Interactions from the system of
interest available to determine Margules parameters.
       Match12(sys,:) = [Intfound12 Ints12];
       s12 = 0;

     % Modifying evaluation and experimental variables to align
```

```
      % interactions data
        for i = 1:length(nonzeros(SysXe1(:,1)))
           for j = 1:8:length(Xe12(1,:))
              u12 = find(ismember(Intjun12,Xe12(s12+1,j)));
              if Xe12(s12+1,j) == 0
              else
                 Xe12mod(s12+1,8*u12-7:8*u12) = Xe12(s12+1,j:j+7);
              end
           end
           s12 = s12 + 1;
        end
        for j = 1:8:length(Xeval12(1,:))
           ueval12 = find(ismember(Intjun12,Xeval12(1,j)));
           if Xeval12(1,j) == 0
           else
              Xeval12mod(:,8*ueval12-7:8*ueval12) =
Xeval12(1,j:j+7);
           end
        end
        datapoints(loop) = sum(SysXe1(:,3));

      % Extracting excess data from Experimental Training Matrices
        j = 0;
        PG12 = []; % Training Matrix with only information to be
used to determine Mergules parameters.
        PG12 = [T1',P1'];
        for i = 1:length(Intjun12)
           PG12 =
[PG12,(Xe12mod(:,j+4).*Xe12mod(:,j+5))./sumINTe12'];
           j = j + 8;
        end

      % Extracting excess data from Evaluation Training Matrices
        j = 0;
        SG12 = []; % Evaluation Matrix with only information to be
used to determine Margules parameters.
        SG12 = [Teval,Peval];
        for i = 1:length(Intjun12)
           SG12 =
[SG12,(Xeval12mod(:,j+4).*Xeval12mod(:,j+5))./sumINTeval12(1)'];
           j = j + 8;
        end
        TGeval1 = Gammas1Eeval(1:systemseval(1,3)); % Targets Matrix
for Gamma1.
        TGeval2 = Gammas2Eeval(1:systemseval(1,3)); % Targets Matrix
for Gamma2.
        TYeval1 = Y1eval(1:systemseval(1,3));
        TYeval2 = Y2eval(1:systemseval(1,3));
        PG12 = PG12';
        SG12 = SG12';
        TA1 = TA1';

      % Neural Networks Training
        [fPG1 cPG1] = size(PG12);
        [fTA1 cTA1] = size(TA1);
        P11 = PG12(:,1:ceil(cPG1/2));
        P21 = PG12(:,ceil(cPG1/2)+1:cPG1);
        T11 = TA1(:,1:ceil(cTA1/2));
        T21 = TA1(:,ceil(cTA1/2)+1:cTA1);
        P31 = SG12;
```

```
        T31 = A_Margules3_eval';
        n2 = 3;
        pp1 = [P11 P21];
        tt1 = [T11 T21];
        mmax1 = minmax(PG12);
        pack
        b111 = [];
        W111 = [];
        b121 = [];
        W121 = [];
        e = [];
        a31 = [];
        max_n = 5;
      % NN Optimum Structure Search
        tf = {'logsig' 'purelin'};
        for i = 1:9
           disp(i)
           e1 = [];
           for j = 1:max_n % Hidden layer optimum number of neurons
search.
              nneu = [j n2];
              net1 = newff(mmax1,nneu,tf);
              net1.trainParam.show = NaN; %Instruction to disable
training graphics.
              net1.trainParam.mem_reduc = 4;  %Instruction to reduce
memory requirement for Jacobian Matrix calculation.
              [net1 tr1] = train(net1,P11,T11);
              a1 = sim(net1,P21);
              e1(j,:) = mean(T21 - a1);
              pack  % Instruction to reduce memory space used for
variables.
           end
           SSE1 = diag(e1 * e1');
           [m1 pos1] = min(SSE1);
           nno1(loop) = pos1;
           nneu1 = [nno1(loop) n2];
           a1 = [];
           app1 = [];
           e1 = [];
         % Optimum initial point search
           for k = 1:20
              net1 = newff(mmax1,nneu1,tf);
              net1.trainParam.show = NaN;
              net1.trainParam.mem_reduc = 4;
              [net1 tr1] = train(net1,pp1,tt1);
              eval(['b11',num2str(k),' = net1.b{1,1};'])
              eval(['b21',num2str(k),' = net1.b{2,1};'])
              eval(['W11',num2str(k),' = net1.iw{1,1};'])
              eval(['W21',num2str(k),' = net1.lw{2,1};'])
              a1(:,:,k) = sim(net1,pp1);
              e1(k,:) = mean(tt1 - a1(:,:,k));
              pack
           end
           SSE1 = diag(e1 * e1');
           [m1 pio1] = min(SSE1);
           net1 = newff(mmax1,nneu1,tf);
           eval(['net1.b{1,1} = b11',num2str(pio1),';'])
           eval(['net1.b{2,1} = b21',num2str(pio1),';'])
           eval(['net1.iw{1,1} = W11',num2str(pio1),';'])
           eval(['net1.lw{2,1} = W21',num2str(pio1),';'])
```

```
            net1.trainParam.show = NaN;
            net1.trainParam.mem_reduc = 4;
            [net1 tr1] = train(net1,pp1,tt1);
            app1 = sim(net1,pp1);
            a31(i,:) = sim(net1,P31);
         end

      % Final results determination based on data central tendency.
         aSG1 = sort(a31);
         for e = 1:2
            aSG1(1,:) = [];
         end
         for e = 7:-1:6
            aSG1(e,:) = [];
         end
         aSG1med = mean(aSG1);
         A_NN = aSG1med(1);
         B_NN = aSG1med(2);
         C_NN = aSG1med(3);
         Gamma1M = exp(((A_NN+3*B_NN+5*C_NN).*(X2eval.^2))-
((4*(B_NN+4*C_NN)).*(X2eval.^3))+((12*C_NN).*(X2eval.^4)));
         Gamma2M = exp(((A_NN-3*B_NN+5*C_NN).*(X1eval.^2))+((4*(B_NN-
4*C_NN)).*(X1eval.^3))+((12*C_NN).*(X1eval.^4)));
         eSG1 = TGeval1 - Gamma1M;
         eSG2 = TGeval2 - Gamma2M;

      % Data Plot and Saving
         timett1 = 1:length(TG1);
         time31 = 1:length(TGeval1);
         eU1 = TGeval1 - Gamma1Ueval;
         eU2 = TGeval2 - Gamma2Ueval;
         Y1predANN = ((X1eval.*10.^(Antoine1eval(:,1)'-
(Antoine1eval(:,2)'./((Teval_Y)+Antoine1eval(:,3)')))).*Gamma1M)./(P
eval_Y);
         Y2predANN = ((X2eval.*10.^(Antoine2eval(:,1)'-
(Antoine2eval(:,2)'./((Teval_Y)+Antoine2eval(:,3)')))).*Gamma2M)./(P
eval_Y);
         Y1predU = ((X1eval.*10.^(Antoine1eval(:,1)'-
(Antoine1eval(:,2)'./((Teval_Y)+Antoine1eval(:,3)')))).*Gamma1Ueval)
./(Peval_Y);
         Y2predU = ((X2eval.*10.^(Antoine2eval(:,1)'-
(Antoine2eval(:,2)'./((Teval_Y)+Antoine2eval(:,3)')))).*Gamma2Ueval)
./(Peval_Y);
         eval(['figure(',num2str(sys),num2str(totalsys),')'])
         subplot(2,2,1)

plot(X1eval,TYeval1,'k',X1eval,Y1predANN,'g',X1eval,Y1predU,'r')
         title(['System: Methanol(1) + n-Propyl Acetate (2)'])
         legend('Exp','ANN','UNIFAC')
         xlabel('X1');
         ylabel('Y1');
         subplot(2,2,2)

plot(X1eval,TGeval1,'k',X1eval,Gamma1M,'g',X1eval,Gamma1Ueval,'r')
         legend('Exp','ANN','UNIFAC')
         xlabel('X1');
         ylabel('Gamma1');
         subplot(2,2,3)

plot(X2eval,TYeval2,'k',X2eval,Y2predANN,'g',X2eval,Y2predU,'r')
```

```
        legend('Exp','ANN','UNIFAC')
        xlabel('X2');
        ylabel('Y2');
        subplot(2,2,4)

plot(X2eval,TGeval2,'k',X2eval,Gamma2M,'g',X2eval,Gamma2Ueval,'r')
        legend('Exp','ANN','UNIFAC')
        xlabel('X2');
        ylabel('Gamma2');
        toc
        time(loop) = toc;
        eval(['save system',num2str(sys),'_Margules3.txt Gamma1Ueval
TGeval1 X1eval aSG1med Gamma1M Gamma2Ueval TGeval2 X2eval aSG1
Y1predANN Y2predANN Y1predU Y2predU time -ascii'])
        Predsys1 =
[Predsys1;sys.*ones(length(Gamma1M),1),Gamma1M,Gamma1Ueval,TGeval1];
        Predsys2 =
[Predsys2;sys.*ones(length(Gamma2M),1),Gamma2M,Gamma2Ueval,TGeval2];
     end
  end
```

The third MatLab program was called ***Gamma_ANN_user.m*** and it purpose is to provide the users with the tool for using the ANN method for predicting activity coefficients for their own systems. This program uses the function ***Create_Gamma_user.m*** that creates the required matrices from the system of interest data to be used as input for predicting the activity coefficients using the ANN method. The instructions for using the program ***Gamma_ANN_user.m*** are as follow:

1) Create the system of interest data file in a Microsoft Excel spreadsheet following the format specified in Table 35. Use the functional groups table in Appendix B to assign the functional groups numbers.

Table 35. System of interest data spreadsheet format

System #	0	0	0	0	0	0	0
0	A1	B1	C1	0	0	0	0
0	A2	B2	B3	0	0	0	0
0	G11	G12	G13	G14	G15	G16	G17
0	n11	n12	n13	n14	n15	n16	n17
0	G21	G22	G23	G24	G25	G26	G27
0	n21	n22	n23	n24	n25	n26	n27
0	T1	P1	X1	0	0	0	0
0	T2	P2	X2	0	0	0	0
0	T3	P3	X3	0	0	0	0
0	T4	P4	X4	0	0	0	0
0	T5	P5	X5	0	0	0	0
0	T6	P6	X6	0	0	0	0
0	Tn	Pn	Xn	0	0	0	0
0	0	0	0	0	0	0	0

This table presents the format used to organize the system of interest data.

Where,

System # is the data system number.

A1, B1 and C1 are the Antoine constants for molecule 1.

A2, B2 and C2 are the Antoine constants for molecule 2.

G11 to G17 are the number of the functional groups for molecule 1.

n11 to n17 are the quantity of each functional group for molecule 1.

G21 to G27 are the number of the functional groups for molecule 2.

n21 to n27 are the quantity of each functional group for molecule 2.

T1 to Tn are the system temperature for each data point.

P1 to Pn are the system pressure for each data point.

X1 to Xn are the liquid mol fraction for molecule 1 for each data point.

2) Run the MatLab program Gamma_ANN_user.m by typing the program name without the "*.m*" extension at the MatLab command window and pressing Enter.
3) The program will ask for the system of interest data file name. Type the system of interest data file name without the "*.xls*" extension and press Enter.
4) The program will ask for the molecules names. Type the name of the first molecule and press Enter. Then, type the name of the second molecule and press Enter.
5) The program will predict the activity coefficients using the ANN method and at the end it will save the results in the same file created to store the data for the system of interest but in a different worksheet called *Output1*. Also the program plots a graph of the predicted activity coefficients vs. the liquid molar fraction and the predicted vapor molar fraction of each molecule vs. its respective liquid molar fraction.

The MatLab code for program ***Gamma_ANN_user.m*** and function ***Create_Gamma_user.m*** is presented below.

```
%Program Gamma_ANN_user.m
   %This program will predict the activity coefficients using the
   %the Neural Networks method.

     clear
     clc
     close all
   % Load the data file where user data is contain.
     userdata = input('Enter your data file name:','s');
     Create_Gamma_user(userdata);
     userdata = [userdata '.xls'];
     load database_user.mat
     disp('User data have been processed and loaded!')
     warning off all
     totalsys = 1000;
     Predsys1 = [];
     Predsys2 = [];
     % Specify system(s) to be evaluated
     evalsys = systems_user(:,1);
```

```
  for loop = 1:length(evalsys)
     Header = [];
     Data = [];
     Molecule1 = [];
     Molecule2 = [];
     Molecule1 = input('Enter the name of molecule (1):','s');
     Molecule2 = input('Enter the name of molecule (2):','s');
     sys = evalsys(loop);
     tic % Start time counter.
    % Load the experimental database file from DECHEMA.
     load Databasenew_Margules3.mat
     sse3 = [];
     [f,c] = size(X1mol);
    % Determine systems data position in Matrix X.
     systems(1,4) = 1;
     systems(1,5) = 1;
     systems_user(1,4) = 1;
     systems_user(1,5) = 1;
     for i = 2:length(systems(:,1))
        systems(i,4) = systems(i-1,4) + 2;
        systems(i,5) = systems(i-1,5) + systems(i-1,3);
     end
     if length(evalsys) > 1
        for i = 2:length(systems_user(:,1))
           systems_user(i,4) = systems_user(i-1,4) + 2;
           systems_user(i,5) = systems_user(i-1,5) + systems_user(i-
1,3);
        end
     else
     end
     A_Margules3_eval = [];
     tfsys = [];
     nneusys = [];
     Gammaspred = [];
     GammasT = [];
     errorssys = [];
     Xevalc = [];
     Xeval = [];
     Xexp = [];
     Yeval = [];
     Y1exp = [];
     Y2exp = [];
     Peval = [];
     Pexp = [];
     Teval = [];
     Texp = [];
     Antoine1eval = [];
     Antoine2eval = [];
     TPeval = [];
     TPexp = [];
     X1eval = [];
     X2eval = [];
     X1X2exp = [];
     Gammaseval = [];
     Gammasexp = [];
     systemseval = [];
     systemsexp = [];
     IntXeval1 = [];
     IntXeval2 = [];
     S = num2str(sys);
```

```
     sysnumber = ['Evaluating system #' S '.'];
     disp(sysnumber);
     Xeval = X_user(systems_user(sys,4):systems_user(sys,4) + 1,:);
% Matriz de evaluacion = sistema extraido.
     IntXeval = Xeval(:,1:8:length(Xeval(1,:)));
     Xevalc = X_user(systems_user(sys,4):systems_user(sys,4) +
1,1:8*length(nonzeros(IntXeval(1,:)))); % Matriz de evaluacion con
la cantidad de interacciones del sistema a ser extraido.
     Xexp = X(1:2*921, :);
     Y1exp = Y1mol(1:f-740);
     Y2exp = Y2mol(1:f-740);
     Peval = X_user(systems_user(sys,4),3);
     Peval_Y = P_exp_user(systems_user(sys,5):systems_user(sys,5) +
systems_user(sys,3)-1);
     Pexp = X(1:2*921,3);
     Teval = X_user(systems_user(sys,4),2);
     Teval_Y = T_exp_user(systems_user(sys,5):systems_user(sys,5) +
systems_user(sys,3)-1);
     Texp = X(1:2*921,2);
     Antoine1eval = Antoine1_user(sys,:);
     Antoine2eval = Antoine2_user(sys,:);
     sumINTexp = sumINT(1:2*921);   % Vector de cantidad de
interacciones del sistema a extraer.
     sumINTeval =
sumINT_user(systems_user(sys,4):systems_user(sys,4) + 1);
     X1eval = X1mol_user(systems_user(sys,5):systems_user(sys,5) +
systems_user(sys,3) - 1);
     X2eval = X2mol_user(systems_user(sys,5):systems_user(sys,5) +
systems_user(sys,3) - 1);
     Gammas1Eexp = Gammas1E(1:f-740);
     Gammas2Eexp = Gammas2E(1:f-740);
     systemseval = systems_user(sys,:); % Matriz de informacion del
sistema a extraer.
     systemsexp = systems(1:length(systems(:,1))-39,:);
     IntXeval1 = Xevalc(1,1:8:length(Xevalc(1,:))); % Interacciones
en Xevalc parte de Gamma1.
     IntXeval1 = reshape(IntXeval1,1,prod(size(IntXeval1))); %
Arreglo de las interacciones en un solo vector.
     IntXeval1 = unique(nonzeros(IntXeval1)); % Interacciones
presentes en el sistema de interes para determinar Gamma1.
     IntXeval2 = Xevalc(2,1:8:length(Xevalc(2,:))); % Interacciones
en Xevalc parte de Gamma2.
     IntXeval2 = reshape(IntXeval2,1,prod(size(IntXeval2))); %
Arreglo de las interacciones en un solo vector.
     IntXeval2 = unique(nonzeros(IntXeval2)); % Interacciones
presentes en el sistema de interes para determinar Gamma2.

   % Training Matrices Creation
     message = ['Creating Training Matrices for system ' S '!'];
     disp(message);
     Xe1 = []; % Experimental Data Training Matrix for Gamma1.
     TG1 = []; % Targets Matrix for Gamma1.
     TA1 = [];
     TY1 = [];
     T1 = [];
     P1 = [];
     sumINTe1 = [];
     sumINTe2 = [];
     monitor1 = [];
     Xe2 = []; % Experimental Data Training Matrix for Gamma2.
```

```
     TG2 = []; % Targets Matrix for Gamma2.
     TA2 = [];
     TY2 = [];
     T2 = [];
     P2 = [];
     monitor2 = [];
     [f,c] = size(Xexp); % Experimental Matrix rows and columns.
     int = c/8; % Maximum number of interactions for systems in the
Experimental Matrix.
     [fp,cp] = size(Xevalc); % Evaluation Matrix rows and columns.
     intp = cp/8; % Number of interactions for the system of
interest.

   % Training Matrix Creation for Gamma1
     monitor1 = zeros(1,length(IntXeval1));
     L = 1;
     o = 0;
     p = 0;
     s = 0;
     t = 0;
     SysXe1 = []; % Chosen systems matrix to determine Gamma1.
     SysXe2 = []; % Chosen systems matrix to determine Gamma2.
     for i = 1:length(systemsexp(:,1)) % Evaluation of all available
systems in the Experimental Matix.
        s = s + 1;
        t = t + 1;
        while s <= 2
           p = 0;
           for j = 1:intp
              L = 1;
              if s <= 2
                 for m = 1:int
                    if and(Xexp(t,L) ==
IntXeval1(j),monitor1(j)<totalsys)
                       o = o + 1;
                       monitor1(j) = monitor1(j)+1;
                       SysXe1(o,:) = systemsexp(i,:);
                       Xe1(o,:) = Xexp(t,:);
                       TG1(o) = Gammas1Eexp(t);
                       TA1(o,:) = A_Margules3_fminunc(i,:);
                       TY1(o) = Y1exp(t);
                       T1(o) = Texp(t);
                       P1(o) = Pexp(t);
                       sumINTe1(o) = sumINTexp(t);
                       L = 1;
                       p = 1;
                       break
                    else
                       L = L + 8;
                    end
                 end
              else
                 break
              end
              if p == 1
                 break
              end
           end
           s = s + 1;
           t = t + 1;
```

```
        end
        s = 0;
        t = t - 1;
     end

   % Training Matrix Creation for Gamma2
     monitor2 = zeros(1,length(IntXeval2));
     L = 1;
     o = 0;
     p = 0;
     s = 0;
     t = 0;
     for i = 1:length(systemsexp(:,1)) % Evaluation of all available
systems in the Experimental Matix.
        s = s + 1;
        t = t + 1;
        k = 1;
        while s <= 2
           p = 0;
           k = 1;
           for j = 1:intp
              L = 1;
              if s <= 2
                 for m = 1:int
                    if and(Xexp(t,L) ==
IntXeval2(j),monitor2(j)<totalsys)
                       o = o + 1;
                       monitor2(j) = monitor2(j)+1;
                       SysXe2(o,:) = systemsexp(i,:);
                       Xe2(o,:) = Xexp(t,:);
                       TG2(o) = Gammas2Eexp(t);
                       TA2(o,:) = A_Margules3_fminunc(i,:);
                       TY2(o) = Y2exp(t);
                       T2(o) = Texp(t);
                       P2(o) = Pexp(t);
                       sumINTe2(o) = sumINTexp(t);
                       L = 1;
                       p = 1;
                       break
                    else
                       L = L + 8;
                    end
                 end
              else
                 break
              end
              if p == 1
                 break
              end
           end
           s = s + 1;
           t = t + 1;
        end
        s = 0;
        t = t - 1;
     end

   % Training Matrices manipulation and re-arrangement
     if isempty(Xe2) == 0
        Xe12 = [Xe1 Xe2];
```

```
        Xeval12 = [Xevalc(1,:) Xevalc(2,:)];
        sumINTe12 = sumINTe1 + sumINTe2;
        sumINTeval12 = sumINTeval(1) + sumINTeval(2);
        IntXe12 = Xe12(:,1:8:length(Xe12(1,:))); % Interactions in
Xe12.
        IntXe12 = reshape(IntXe12,1,prod(size(IntXe12))); % Re-
arrangement in one vector for interaction in Xe12.
        IntXeval12 = Xeval12(1,1:8:length(Xeval12(1,:))); %
Interactions in Xeval12 to determine Margules equation parameters.
        IntXeval12 = reshape(IntXeval12,1,prod(size(IntXeval12))); %
Re-arrangement in one vector for interactions in Xeval12.
        Intjun12 =
unique(union(nonzeros(IntXe12)',nonzeros(IntXeval12)')); % All
interactions toghether.
        Xe12mod = zeros(length(Xe12(:,1)),length(Intjun12)*8); %
Initialization for Modified Experimental Matrix.
        Xeval12mod = zeros(1,length(Intjun12)*8); % Initialization
for Modified Evaluation Matrix.
      % Verifing that all interactions from the system of interest
are
      % present in the Experimental Matrix
        IntXe12 = unique(nonzeros(IntXe12)); % Interactions present
in the experimental data to determine Margules parameters.
        IntXeval12 = unique(nonzeros(IntXeval12)); % Interactions
present in the system of interest to determine Margules parameters.
        tf12 = ismember(IntXeval12,IntXe12); % Interactions from the
system of interest present in the experimental data to determine
Margules parameters.
        Intfound12 = sum(tf12); % Interactions from the system of
interest found in the experimental data to determine Margules
parameters.
        Ints12 = length(tf12); % Interactions from the system of
interest available to determine Margules parameters.
        Match12(sys,:) = [Intfound12 Ints12];
        s12 = 0;
      % Modifying evaluation and experimental variables to align
      % interactions data
        for i = 1:length(nonzeros(SysXe1(:,1)))
           for j = 1:8:length(Xe12(1,:))
              u12 = find(ismember(Intjun12,Xe12(s12+1,j)));
              if Xe12(s12+1,j) == 0
              else
                 Xe12mod(s12+1,8*u12-7:8*u12) = Xe12(s12+1,j:j+7);
              end
           end
           s12 = s12 + 1;
        end
        for j = 1:8:length(Xeval12(1,:))
           ueval12 = find(ismember(Intjun12,Xeval12(1,j)));
           if Xeval12(1,j) == 0
           else
              Xeval12mod(:,8*ueval12-7:8*ueval12) =
Xeval12(1,j:j+7);
           end
        end
        datapoints(loop) = sum(SysXe1(:,3));
      % Extracting excess data from Experimental Training Matrices
        j = 0;
        PG12 = []; % Training Matrix with only information to be
used to determine Mergules parameters.
```

```
        PG12 = [T1',P1'];
        for i = 1:length(Intjun12)
           PG12 =
[PG12,(Xe12mod(:,j+4).*Xe12mod(:,j+5))./sumINTe12'];
           j = j + 8;
        end
      % Extracting excess data from Evaluation Training Matrices
        j = 0;
        SG12 = []; % Evaluation Matrix with only information to be
used to determine Margules parameters.
        SG12 = [Teval,Peval];
        for i = 1:length(Intjun12)
           SG12 =
[SG12,(Xeval12mod(:,j+4).*Xeval12mod(:,j+5))./sumINTeval12(1)'];
           j = j + 8;
        end
        PG12 = PG12';
        SG12 = SG12';
        TA1 = TA1';

      % Neural Networks Training
        [fPG1 cPG1] = size(PG12);
        [fTA1 cTA1] = size(TA1);
        P11 = PG12(:,1:ceil(cPG1/2));
        P21 = PG12(:,ceil(cPG1/2)+1:cPG1);
        T11 = TA1(:,1:ceil(cTA1/2));
        T21 = TA1(:,ceil(cTA1/2)+1:cTA1);
        P31 = SG12;
        T31 = A_Margules3_eval';
        n2 = 3;
        pp1 = [P11 P21];
        tt1 = [T11 T21];
        mmax1 = minmax(PG12);
        pack
        b111 = [];
        W111 = [];
        b121 = [];
        W121 = [];
        e = [];
        a31 = [];
        max_n = 5;
      % NN Optimum Structure Search
        tf = {'logsig' 'purelin'};
        for i = 1:9
           disp(i)
           e1 = [];
           for j = 1:max_n % Hidden layer optimum number of neurons
search.
              nneu = [j n2];
              net1 = newff(mmax1,nneu,tf);
              net1.trainParam.show = NaN; %Instruction to disable
training graphics.
              net1.trainParam.mem_reduc = 4;  %Instruction to reduce
memory requirement for Jacobian Matrix calculation.
              [net1 tr1] = train(net1,P11,T11);
              a1 = sim(net1,P21);
              e1(j,:) = mean(T21 - a1);
              pack  % Instruction to reduce memory space used for
variables.
           end
```

```
            SSE1 = diag(e1 * e1');
            [m1 pos1] = min(SSE1);
            nno1(loop) = pos1;
            nneu1 = [nno1(loop) n2];
            a1 = [];
            app1 = [];
            e1 = [];
          % Optimum initial point search
            for k = 1:20
               net1 = newff(mmax1,nneu1,tf);
               net1.trainParam.show = NaN;
               net1.trainParam.mem_reduc = 4;
               [net1 tr1] = train(net1,pp1,tt1);
               eval(['b11',num2str(k),' = net1.b{1,1};'])
               eval(['b21',num2str(k),' = net1.b{2,1};'])
               eval(['W11',num2str(k),' = net1.iw{1,1};'])
               eval(['W21',num2str(k),' = net1.lw{2,1};'])
               a1(:,:,k) = sim(net1,pp1);
               e1(k,:) = mean(tt1 - a1(:,:,k));
               pack
            end
            SSE1 = diag(e1 * e1');
            [m1 pio1] = min(SSE1);
            net1 = newff(mmax1,nneu1,tf);
            eval(['net1.b{1,1} = b11',num2str(pio1),';'])
            eval(['net1.b{2,1} = b21',num2str(pio1),';'])
            eval(['net1.iw{1,1} = W11',num2str(pio1),';'])
            eval(['net1.lw{2,1} = W21',num2str(pio1),';'])
            net1.trainParam.show = NaN;
            net1.trainParam.mem_reduc = 4;
            [net1 tr1] = train(net1,pp1,tt1);
            app1 = sim(net1,pp1);
            a31(i,:) = sim(net1,P31);
          end
        % Final results determination based on data central tendency.
          aSG1 = sort(a31);
          for e = 1:2
             aSG1(1,:) = [];
          end
          for e = 7:-1:6
             aSG1(e,:) = [];
          end
          aSG1med = mean(aSG1);
          A_NN = aSG1med(1);
          B_NN = aSG1med(2);
          C_NN = aSG1med(3);
          Gamma1M = exp(((A_NN+3*B_NN+5*C_NN).*(X2eval.^2))-
((4*(B_NN+4*C_NN)).*(X2eval.^3))+((12*C_NN).*(X2eval.^4)));
          Gamma2M = exp(((A_NN-3*B_NN+5*C_NN).*(X1eval.^2))+((4*(B_NN-
4*C_NN)).*(X1eval.^3))+((12*C_NN).*(X1eval.^4)));
        % Data Plot and Saving
          linex = [0 1];
          liney = [0 1];
          Y1predANN = ((X1eval.*10.^(Antoine1eval(:,1)'-
(Antoine1eval(:,2)'./((Teval_Y)+Antoine1eval(:,3)')))).*Gamma1M)./(P
eval_Y);
          Y2predANN = ((X2eval.*10.^(Antoine2eval(:,1)'-
(Antoine2eval(:,2)'./((Teval_Y)+Antoine2eval(:,3)')))).*Gamma2M)./(P
eval_Y);
          eval(['figure(',num2str(sys),num2str(totalsys),')'])
```

```
            subplot(2,2,1:2)
            plot(X1eval,Gamma1M,'b',X1eval,Gamma2M,'r')
            title(['System: ' Molecule1 ' (1) + ' Molecule2 ' (2)'])
            legend('Gamma1','Gamma2','Best')
            xlabel('X1');
            ylabel('Gamma');
            xlim([0 1]);
            subplot(2,2,3)
            plot(linex,liney,'k',X1eval,Y1predANN,'g')
            xlabel('X1');
            ylabel('Y1pred');
            axis([0 1 0 1]);
            subplot(2,2,4)
            plot(linex, liney,'k',X2eval,Y2predANN,'g')
            xlabel('X2');
            ylabel('Y2pred');
            axis([0 1 0 1]);
            toc
            time(loop) = toc;
            Header =
    {'Temp(ºC)','P(mmHg)','X1','Y1_pred','Gamma1_ANN','Gamma2_ANN'};
            Data = [Teval_Y Peval_Y X1eval Y1predANN Gamma1M Gamma2M];
            eval(['xlswrite(''',userdata, ''', Header, ''Output',
    num2str(sys), ''');']);
            eval(['xlswrite(''',userdata, ''', Data, ''Output',
    num2str(sys), ''', ''A2'');']);
         end
      end


%Function Create_Gamma_user.m
    % This program will create the data base for the activity
    coefficients
    % estimation program.
    function g = Create_Gamma_user(userdata)
    % Load the data file where the user data is contain.
    s = ['data = xlsread(''' userdata '.xls'');'];
    eval(s);
    g = 186;
    MW = [12 13    14    15    16    12    13    14    12    13    29
       30    31    32    29    30    31    40    42    44    46    41
       42    43    43    44    45    43    44    45    42 43 41 30 41 42
       43 44 52    54    56    58    53    54    55    55    56    57
       55    56    57    54    55    53    46    59    58    59    60
       57    58    59    60    68    70    72    74    69    70    71
       71    72    74    71    72    73    70    71    69    28    29
       30    31    39    41    43    45    50    52    56    59    27
       38    39    40    41    58    59    60    61    45    46    47
       48    47.5  48.5  49.5  50.5  32    52    31    50    69    51
       70    34    33    85.5  102   104.5 137.5 67.5  84    119.5 83
       118.5 86.5  103   66.5  68.5  85    121   154   12    13    29
       54    84    73.5  40    56    57    58    59    67    69    71
       73    68    69    70    71    72    73    70    71    72    69
       70    68    66    81    85    87    80    82    84    83    85
       86    81    83    85    80    82    79    18    68    73    28
       44    32    57    62.5  96 31];
    % Raw data manipulation
    systems = [nonzeros(data(:,1)) find(data(:,1))];
    systems_Margules3 = [nonzeros(data(:,1)) find(data(:,1))];
```

```
systems_Margules3 = [systems_Margules3; length(systems_Margules3)+1
length(data(:,1))];
X = [];
X1mol = [];
X2mol = [];
GF1 = [];
GF2 = [];
Antoine1 = [];
Antoine2 = [];
P_exp = [];
T_exp = [];
Max = max(systems(:,1));
for i = 1:length(systems(:,1))
   S = int2str(i);
   M = num2str(Max);
% X matrix creation
   X1_Margules3 = [];
   X2_Margules3 = [];
   T_Margules3 = [];
   P_Margules3 = [];
   Antoine1_Margules3 = [];
   Antoine2_Margules3 = [];
   points_Margules3(i) = systems_Margules3(i+1,2) -
systems_Margules3(i,2) - 8;
   X1_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,4)';
   X2_Margules3 = 1-X1_Margules3;
   T_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,2)';
   P_Margules3 =
data(systems_Margules3(i,2)+8:systems_Margules3(i,2)+8+points_Margul
es3(i)-1,3)';
   Antoine1_Margules3 =
[nonzeros(data(systems_Margules3(i,2)+1,:))'];
   Antoine2_Margules3 =
[nonzeros(data(systems_Margules3(i,2)+2,:))'];
   if or(X1_Margules3(1)==0,X1_Margules3(1)==1) == 1
      X1_Margules3(1) = [];
      X2_Margules3(1) = [];
      T_Margules3(1) = [];
      P_Margules3(1) = [];
      points_Margules3(i) = points_Margules3(i)-1;
   else
   end
   if
or(X1_Margules3(points_Margules3(i))==0,X1_Margules3(points_Margules
3(i))==1) == 1
      X1_Margules3(points_Margules3(i)) = [];
      X2_Margules3(points_Margules3(i)) = [];
      T_Margules3(points_Margules3(i)) = [];
      P_Margules3(points_Margules3(i)) = [];
      points_Margules3(i) = points_Margules3(i)-1;
   else
   end
   Antoine1 = [Antoine1;Antoine1_Margules3];
   Antoine2 = [Antoine2;Antoine2_Margules3];
   X1mol = [X1mol;X1_Margules3'];
   X2mol = [X2mol;X2_Margules3'];
```

```
   P_exp = [P_exp;P_Margules3'];
   T_exp = [T_exp;T_Margules3'];
   P = data(systems(i,2)+7,3);
   points = 0;
   point = 0;
   while P ~= 0
      xpoint = 1;
      ypoint = 1;
      if data(systems(i,2)+point+8,4) == 0
         ypoint = 0;
      end
      if data(systems(i,2)+point+8,4) == 1
         ypoint = 0;
      end
      if and(xpoint,ypoint) == 1
         points = points + 1;
      else
      end
      point = point + 1;
      P = data(systems(i,2)+8+point,3);
   end
   systems(i,3) = points;
   NGF1 = length(nonzeros(data(systems(i,2)+3,:)));
   NGF2 = length(nonzeros(data(systems(i,2)+5,:)));
   [o,p] = size(X);
   k = 1;
   l = 0;
 % Suma de la cantidad total de interacciones.
   sumINT(o+1) = 0;
   sumINT(o+2) = 0;
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            sumINT(o+1) = sumINT(o+1) +
(data(systems(i,2)+4,r+1)*data(systems(i,2)+6,s+1));
            sumINT(o+2) = sumINT(o+2) +
(data(systems(i,2)+4,r+1)*data(systems(i,2)+6,s+1));
         else
         end
      end
   end
 % 1)Posisionar el numero de la interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            GF1 = [GF1;data(systems(i,2)+3,r+1)];
            GF2 = [GF2;data(systems(i,2)+5,s+1)];
            X(o+1,l+k) = g * (data(systems(i,2)+3,r+1) - 1) +
data(systems(i,2)+5,s+1);
            X(o+2,l+k) = g * (data(systems(i,2)+5,s+1) - 1) +
data(systems(i,2)+3,r+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
 % 2)Posisionar la Temperatura del sistema.
      for r = 1:NGF1
```

```
         for s = 1:NGF2
            if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
               X(o+1,l+k) = mean(T_Margules3);
               X(o+2,l+k) = mean(T_Margules3);
               l = l + 8;
            else
            end
         end
      end
   k = k + 1;
   l = 0;
 % 3)Posisionar la Presión del sistema.
      for r = 1:NGF1
         for s = 1:NGF2
            if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
               X(o+1,l+k) = mean(P_Margules3);
               X(o+2,l+k) = mean(P_Margules3);
               l = l + 8;
            else
            end
         end
      end
   k = k + 1;
   l = 0;
 % 4)Posisionar la cantidad de GF de la molécula #1 en la
interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            X(o+1,l+k) = data(systems(i,2)+4,r+1);
            X(o+2,l+k) = data(systems(i,2)+6,s+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
 % 5)Posisionar la cantidad de GF de la molécula #2 en la
interacción.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            X(o+1,l+k) = data(systems(i,2)+6,s+1);
            X(o+2,l+k) = data(systems(i,2)+4,r+1);
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
   sumMW1 = 0;
   sumMW2 = 0;
 % 6)Posisionar el peso molecular del grupo funcional de la molécula
#1.
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
```

```
            X(o+1,l+k) =
data(systems(i,2)+4,r+1)*MW(data(systems(i,2)+3,r+1));
            X(o+2,l+k) =
data(systems(i,2)+6,s+1)*MW(data(systems(i,2)+5,s+1));
            l = l + 8;
         else
         end
      end
      sumMW1 = sumMW1 +
data(systems(i,2)+4,r+1)*MW(data(systems(i,2)+3,r+1));
   end
   k = k + 1;
   l = 0;
 % 7)Posisionar el peso molecular del grupo funcional de la molécula
#2.
   for s = 1:NGF2
      for r = 1:NGF1
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            X(o+1,l+k) =
data(systems(i,2)+6,s+1)*MW(data(systems(i,2)+5,s+1));
            X(o+2,l+k) =
data(systems(i,2)+4,r+1)*MW(data(systems(i,2)+3,r+1));
            l = l + 8;
         else
         end
      end
      sumMW2 = sumMW2 +
data(systems(i,2)+6,s+1)*MW(data(systems(i,2)+5,s+1));
   end
   k = k + 1;
   l = 0;
 % 8)Posisionar 1
   for r = 1:NGF1
      for s = 1:NGF2
         if data(systems(i,2)+3,r+1) ~= data(systems(i,2)+5,s+1)
            X(o+1,l+k) = 1;
            X(o+2,l+k) = 1;
            l = l + 8;
         else
         end
      end
   end
   k = k + 1;
   l = 0;
end
X_user = X;
P_exp_user = P_exp;
T_exp_user = T_exp;
X1mol_user = X1mol;
X2mol_user = X2mol;
systems_user = systems;
Antoine1_user = Antoine1;
Antoine2_user = Antoine2;
sumINT_user = sumINT;
save database_user X_user systems_user P_exp_user T_exp_user
X1mol_user X2mol_user Antoine1_user Antoine2_user sumINT_user
```