University of South Florida

DIGITAL COMMONS
@ UNIVERSITY OF SOUTH FLORIDA

Digital Commons @ University of South Florida

USF Tampa Graduate Theses and Dissertations

USF Graduate Theses and Dissertations

12-11-2009

Synthesis of Local Thermo-Physical Models Using Genetic Programming

Ying Zhang
University of South Florida

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation

Zhang, Ying, "Synthesis of Local Thermo-Physical Models Using Genetic Programming" (2009). *USF Tampa Graduate Theses and Dissertations.*
https://digitalcommons.usf.edu/etd/103

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.

Synthesis of Local Thermo-Physical Models Using Genetic Programming

by

Ying Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Chemical and Biomedical Engineering
College of Engineering
University of South Florida

Major Professor: Aydin K. Sunol, Ph.D.
John A. Llewellyn, Ph.D.
Scott W. Campbell, Ph.D.
Luis H. Garcia-Rubio, Ph.D.
Rafael Perez, Ph.D.

Date of Approval:
December 11, 2008

Keywords: data mining, symbolic regression, function identification, parameter regression, statistic analysis, process simulation

© Copyright 2009, Ying Zhang

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to Dr. Aydin K. Sunol, for his continuous guidance and encouragement throughout my Ph.D. experience.

I would also like to thank Dr. John A. Llewellyn, Dr. Scott W. Campbell, Dr. Luis H. Garcia-Rubio and Dr. Rafael Perez for being my committee members and taking time from their busy schedules. Finally, I would like to thank my family for their help.

TABLE OF CONTENTS

LIST OF TABLES iv

LIST OF FIGURES v

ABSTRACT viii

CHAPTER ONE: INTRODUCTION 1

CHAPTER TWO: LITERATURE REVIEW 6
- 2.1 Review of local models for phase partition coefficients 6
- 2.2 Review of data mining techniques in the knowledge discovery process 14
- 2.3 Elements of structural regression using genetic programming 21
  - 2.3.1 Terminal set, function set and initial representation 23
  - 2.3.2 Fitness measures for models of varying complexity 25
    - 2.3.2.1 Fitness function with no penalty for the model complexity 25
    - 2.3.2.2 Fitness function with model complexity control 26
      - 2.3.2.2.1 Minimum description length 27
      - 2.3.2.2.2 Parsimony pressure 28
    - 2.3.2.3 Fitness function using external validation 30
  - 2.3.3 Genetic operators 31
    - 2.3.3.1 Reproduction and crossover 31
    - 2.3.3.2 Mutation 33
  - 2.3.4 Selection strategy 35
- 2.4 Parametric regression: review of objective functions and optimization methods 36
- 2.5 Applications of intelligent system in chemical engineering 43

CHAPTER THREE: A HYBRID SYSTEM FOR STRUCTURAL AND PARAMETRIC OPTIMIZATION 46
- 3.1 The system structure 46
- 3.2 Data and data preparation 48
- 3.3 The regression strategy 49
- 3.4 Implementation with MATLAB based genetic search toolbox 53
  - 3.4.1 The population architecture 54
  - 3.4.2 The population size 57

3.4.3 The principal component analysis and the selection of terminal & function sets 59
3.4.4 Genetic operator 62
3.4.5 Fitness evaluation 64
3.4.6 Selection strategy 67
3.4.7 Decimation strategy 69
3.4.8 Result designation and termination criteria 72
3.4.9 Summary 73
3.5 Model evaluation 74
3.5.1 Statistical analysis 74
3.5.1.1 Graphical methods 74
3.5.1.2 Goodness of fit statistics 76
3.5.1.3 Cross validation and prediction 76
3.5.2 Steady state and dynamic simulation using local models 78

CHAPTER FOUR: RESULTS AND DISCUSSIONS 79
4.1 Local composition models and their impact 79
4.1.1 Developed models for mixtures that form ideal or near-ideal solutions and their statistical analysis 80
4.1.2 Performance of models for separation of ideal and near ideal mixtures 84
4.1.3 Developed models for mixtures that form non-ideal solutions and their statistical analysis 89
4.1.4 Performance of models for separation of non-ideal solutions 93
4.2 Discussion 99
4.2.1 Ideal gaseous and liquid mixture 100
4.2.2 Non-ideal gaseous mixture and ideal liquid mixture 101
4.2.3 Ideal gaseous and non-ideal liquid mixture 102
4.2.4 Non-ideal gaseous mixture and non-ideal liquid mixture 103
4.2.5 Linear model vs. nonlinear model 104
4.2.6 Extrapolation 105

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS 107

REFERENCES 110

APPENDICES 115
Appendix A. User Manual for GP Package 116
A.1 MATLAB files 116
A.2 Architecture 116
A.3 How to use the GP package 118

Appendix B. Statistical Analysis 122
B.1 Tests for outliers 122
B.2 Evaluation of final model: cross-validation 124

ABOUT THE AUTHOR End Page

LIST OF TABLES

Table 4.1	Result Summary for Propylene (1)-Propane (2)	81
Table 4.2	Standard Error Analysis for Generated K_1 and K_2 Model	84
Table 4.3	Propane-Propylene Distillation Column Profile	88
Table 4.4	Result Summary for Acetone (1) -Water (2)	89
Table 4.5	Standard Error Analysis for Local Model for Acetone-Water System	92
Table 4.6	Tower Profile at Different Runtime	98
Table 4.7	R^2 of Different Models for K_1 in Group Tests	99
Table 4.8	R^2 of Extrapolation Test	106
Table B.1	MATLAB Code for Outlier Test	122
Table B.2	The Result for Outlier Test	123
Table B.3	MATLAB Code for Data Set Partition	125
Table B.4	The Result for Data Set Partition	126

LIST OF FIGURES

Figure 2.1	Flowchart for an Algorithm for Isothermal Flash Calculation Algorithm that Uses Composition Dependent Local Models	8
Figure 2.2	Flowchart for an Algorithm for Isothermal Flash Calculation Algorithm that Uses Equation of State	9
Figure 2.3	Architecture of Local Model Strategy	10
Figure 2.4	Knowledge Discovery Process	15
Figure 2.5	Example of a Regression Tree	17
Figure 2.6	A Simple Multilayer Neural Network Model with Two Hidden Nodes	18
Figure 2.7	A Flowchart for Computation through Genetic Programming	23
Figure 2.8	Functional Representation of $a + b \cdot T + c \cdot T^2 (C_p)$ Using a Tree Structure	24
Figure 2.9	An Example of Reproduction Operator	32
Figure 2.10	Crossover Operation for an Algebraic Equation Manipulation	34
Figure 2.11	An Example of Mutation Operation, Type I	34
Figure 2.12	An Example of Mutation Operation, Type II	35
Figure 3.1	A Hybrid System Structure for Structural and Parametric Optimization	47
Figure 3.2	The Structure for Linear-Nonlinear Regression Strategy	51
Figure 3.3	A Schematic Diagram of the Genetic Search Methodology	54
Figure 3.4	A Flowchart of Genetic Search, Steady State Population	55

Figure 3.5 A Flowchart of Genetic Search, Generational Population 56

Figure 3.6 Fitness vs. Generation Using Different Population Sizes 59

Figure 3.7 Fitness vs. Generation Using Different Mutation and Crossover Probabilities 63

Figure 3.8 The Standard Deviation of Fitness Using Different Mutation and Crossover Probabilities 64

Figure 3.9 Model's Accuracy vs. Generation Using Different Fitness Functions 66

Figure 3.10 Model's Complexity vs. Generation Using Different Fitness Functions 66

Figure 3.11 Model Fitness vs. Generation Using Different Tournament Sizes 68

Figure 3.12 The Standard Deviation of Fitness vs. Generation Using Different Tournament Sizes 68

Figure 3.13 Model's Accuracy vs. Generation Using Different Deletion Strategies 71

Figure 3.14 The Standard Deviation of Fitness vs. Generation Using Different Deletion Strategies 72

Figure 4.1 K_1 Model vs. Experimental Data for Propylene (1)-Propane (2) 82

Figure 4.2 Residual Plot of K_1 for Propylene (1)-Propane (2) 82

Figure 4.3 K_2 Model vs. Experimental Data for Propylene (1)-Propane (2) 83

Figure 4.4 Residual Plot of K_2 for Propylene (1)-Propane (2) 83

Figure 4.5 P-X_1-Y_1 at Low Pressures for Propylene (1)-Propane (2) 85

Figure 4.6 P-X_1-Y_1 at Medium to High Pressures for Propylene (1)-Propane (2) 85

Figure 4.7 K_1 vs. Liquid Composition for Different Temperatures for Propylene (1)-Propane (2) System 86

Figure 4.8 K_1 vs. Pressure for Different Temperatures for Propylene (1)-Propane (2) 86

Figure 4.9 Temperature Profile of Propylene-Propane Distillation Column 88

Figure 4.10 Relative Volatility Profile for Propylene-Propane Distillation Column 89

Figure 4.11 K_1 Model vs. Experimental Data for Acetone (1)-Water (2) 90

Figure 4.12 Residual Plot of K_1 for Acetone (1)-Water (2) 90

Figure 4.13 K_2 Model vs. Experimental Data for Acetone (1)-Water (2) 91

Figure 4.14 Residual Plot of K_2 for Acetone (1)-Water (2) 91

Figure 4.15 T-X_1-Y_1 at Different Temperatures for Acetone (1)-Water (2) 93

Figure 4.16 K_1 vs. Liquid Composition at Different Pressures for Acetone (1)-Water (2) 94

Figure 4.17 K_1 vs. Temperature at Different Pressures for Acetone (1)-Water (2) 94

Figure 4.18 Temperature Profile of Acetone-Water Distillation Column 96

Figure 4.19 Relative Volatility Profile of Acetone-Water Distillation Column 96

Figure 4.20 Distillation Total Flow Rate 97

Figure 4.21 Distillation Column Pressure 98

Figure A.1 Architecture of MATLAB Code 117

SYNTHESIS OF LOCAL THERMO-PHYSICAL MODELS USING GENETIC PROGRAMMING

Ying Zhang

ABSTRACT

Local thermodynamic models are practical alternatives to computationally expensive rigorous models that involve implicit computational procedures and often complement them to accelerate computation for real-time optimization and control. Human-centered strategies for development of these models are based on approximation of theoretical models. Genetic Programming (GP) system can extract knowledge from the given data in the form of symbolic expressions. This research describes a fully data driven automatic self-evolving algorithm that builds appropriate approximating formulae for local models using genetic programming. No a-priori information on the type of mixture (ideal/non ideal etc.) or assumptions are necessary.

The approach involves synthesis of models for a given set of variables and mathematical operators that may relate them. The selection of variables is automated through principal component analysis and heuristics. For each candidate model, the model parameters are optimized in the inner integrated nested loop. The trade-off between accuracy and model complexity is addressed through incorporation of the Minimum Description Length (MDL) into the fitness (objective) function.

Statistical tools including residual analysis are used to evaluate performance of models. Adjusted R-square is used to test model's accuracy, and F-test is used to test if the terms in the model are necessary. The analysis of the performance of the models generated with the data driven approach depicts theoretically expected range of compositional dependence of partition coefficients and limits of ideal gas as well as ideal solution behavior. Finally, the model built by GP integrated into a steady state and dynamic flow sheet simulator to show the benefits of using such models in simulation. The test systems were propane-propylene for ideal solutions and acetone-water for non-ideal. The result shows that, the generated models are accurate for the whole range of data and the performance is tunable. The generated local models can indeed be used as empirical models go beyond elimination of the local model updating procedures to further enhance the utility of the approach for deployment of real-time applications.

CHAPTER ONE

INTRODUCTION

Approaches to modeling of chemical processes have changed significantly in the past three decades. In general, these approaches are divided into two generic categories. One is mechanistic modeling, which is mainly based on first principles and fundamental knowledge. The other is empirical modeling, which is data driven. In the latter, the model structure and its associated parameters are selected to represent the process data accurately for a given range and aim to bring ease through simplified model development stage as well as reduced computational load.

Data driven modeling techniques have been popular for many decades. They are easier to develop than the mechanistic models, particularly for practitioners. This is especially true when mechanistic first principles models and their associated thermo-physical properties are not adequate in representing the real world problems. Furthermore, these mechanistic models are highly nonlinear and complex, which makes them difficult to identify [Ramirez 1989] and implement particularly on-real-time applications. Currently, the most of the data driven modeling methods fall under statistical methods and artificial neural networks headings [Pöyhönen, 1996]. Neural networks usually provide models that are accurate in representing the data, but they don't provide any insight into represented phenomena. Usually, neural networks are black

boxes, and one cannot abstract the underlying physical relationships between input and output data. It is often desirable to gain some insight into the underlying structures, as well as make accurate numeric predictions. Application of Genetic Programming (GP) based approaches are known to produce input-output models with relatively simple and transparent structures and the associated procedures are coined with "symbolic regression" terminology.

Genetic Programming allows synthesis of data driven models when model elements are represented as a tree structure. This tree structure is of variable length and consists of nodes. The terminal nodes can be input variables, parameters or constants while thee non-terminal nodes are standard library functions, like addition, subtraction, multiplication and division. Each tree structure may possibly describe an equation.

Genetic programming works by emulating natural evolution to generate an optimum model structure that best maximizes some fitness function. Model structures evolve through the action of operators known as reproduction, crossover and mutation. Crossover involves interchange of the branches from two parent structures. Mutation is random creation of a completely new branch. At each generation, a population of model structures undergoes crossover, mutation and selection and then a fitness function is evaluated. These operators improve the general fitness of the population. Based on fitness, the next generation is selected from the pool of old and new structures. The process repeats itself until some convergence criterion is satisfied and a model is generated.

One primary classification used for property and process models in Chemical Engineering is based on algebraic versus differential equation models [Franks 1967]. The mathematical models are either comprised of a set of algebraic equations for steady-state operation or by a set of ordinary differential equations (ODE) coupled with algebraic equations for dynamic (time-dependent) models, or partial differential equations (PDE) for distributed models. The majority of algebraic mathematical models for physical or engineered systems can be classified in one of the following three types [Englezos 2001]:

- Type I: A model with a single dependent variable and a single independent variable. For example, heat capacity model for ideal gas is a function of temperature.
- Type II: A model with a dependent variable and several independent variables, for example, a pressure-explicit equation of states (EOS) which is enable the calculation of fluid phase equilibrium and thermo-physical properties such as enthalpy, entropy, and density necessary in the design of chemical processes. Mathematically, a pressure-explicit EOS expresses the relationship among pressure, volume, temperature, and composition for a fluid mixture.
- Type III: A model with multiple dependent variables and several independent variables. A typical group of applications is modeling of reaction kinetics where possible mechanism is depicted as multiple reactions that are coupled through concentration of species.

The objective of this dissertation is to develop a methodology, which uses genetic operations in order to find a symbolic relationship between a single dependent variable and multiple independent variables, i.e., Type II. The approach was demonstrated for Type I problems by Zhang [2004] earlier.

The structure and hence the complexity of the model or the equation is not specified like in the conventional regression, which seeks to find the best set of parameters for a pre-specified model. The goal is to seek a mathematical expression, in symbolic form, which fits or approximates a given sample of data using genetic programming (GP). The approach is called "Symbolic Regression".

The nested two tier approach is proposed in this research where parameter regression method is embedded within GP. The GP is employed to optimize the structure of a model, while classical numerical regression is employed to optimize its parameters for each proposed structure. The model structure and its parameters are unknown, and determined for each step through the algorithm. Model's adequacy is tested through post analysis.

The approach is tested for a practical and significant problem: development of local and/or empirical partition coefficient models for vapor liquid separation.

For accurate chemical process design and effective operation, a correct estimate of physical and thermodynamic properties is a prerequisite. The estimation of these properties through first principle but complex implicit models for pure components and mixtures is computationally costly. The computational time is critical particularly in real time applications. The phase equilibrium calculations are the most computationally

intensive of these properties due to implicit nature of procedures with more complex property models, especially when used with rigorous separation models [Leesley 1977].

Local thermodynamic models are explicit functions that approximate more rigorous models that involve implicit computational procedures in equilibrium calculations. Computations with these functions are fast and non-iterative at times, but are only valid in a limited region where the functions are accurate. Therefore, local models need to be updated as the simulation moves into new regions in the state spaces. Since the late seventies, many functional forms with differing independent variable sets for these models were suggested and some have been implemented within flow sheet simulator packages [Perregaard 1992, Storen 1994, and Storen 1997].

This introductory chapter is followed by Chapter Two, where local models, data mining applications and technologies, evolutionary algorithms and their applications in chemical engineering, and optimization methods and their objective functions are reviewed. Chapter Three describes the proposed system structure, guidelines for determination of GP controlling parameters, and the details of implementing the approach. Results and discussion are given in Chapter Four. Finally, in Chapter Five, conclusion and recommendations are presented.

CHAPTER TWO

LITERATURE REVIEW

This chapter includes the review of local models for vapor-liquid partition coefficient (K value), data mining tasks and techniques, evolutionary algorithms and optimization methods. In the first section, the development of local models is summarized. The review on data mining technologies is given in the second section. The third section describes the development of evolutionary algorithms and the comparison of different algorithms. More emphasis is given to genetic programming. A brief summary of applications of intelligent system in chemical engineering is also given at the end of this section. In the fourth section, some popular optimization methods and pertinent objective functions (criteria) are reviewed.

2.1 Review of local models for phase partition coefficients

A correct estimate of physical and thermodynamic properties is a prerequisite for the accurate chemical process design and operation. The calculation of those properties of pure components and mixtures contributes the major cost in computer time. Local thermodynamic models are practical alternatives to computationally expensive, more rigorous macroscopic or molecular models that involve implicit computational

procedures, and often complement them to accelerate computation for run time optimization and control. Since vapor-liquid equilibrium constant K is among the most computationally expensive one [Leesley 1977], the research efforts on developing local models focus on the vapor-liquid equilibrium constant K.

Since the late seventies, several research groups developed local thermodynamic models and accompanying procedures to be implemented within flow sheet simulator packages. The objective is to replace, or assist more rigorous thermodynamic models with local alternatives to reduce the computer time while maintaining the thermodynamic accuracy at an acceptable level. Local thermodynamic models have been used to accelerate steady state calculation [Leesley et al. 1977, Chimowitz et al. 1983, Perregaard 1993], dynamic simulation [Chimowitz et al. 1984, Perregaard 1993], and dynamic optimization [Storen 1997]. The more rigorous thermodynamic models are nonlinear equation sets which involve iterative calculations for vapor-liquid equilibrium constant (K) model. The local models are in explicit form, and linear with respect to its parameters. Their calculation procedures are shown in Figure 2.1 while the flowchart of isothermal, isobaric flash calculation using an equation of state is shown in Figure 2.2. As can be seen, the explicit local models are much easier and faster to evaluate, but they are only valid locally. The local models must be updated, if the simulation proceeds out of the region where the local model is valid.

The implementation of the idea involves three major components: local model formulation, error monitor and parameter update, as shown in Figure 2.3.

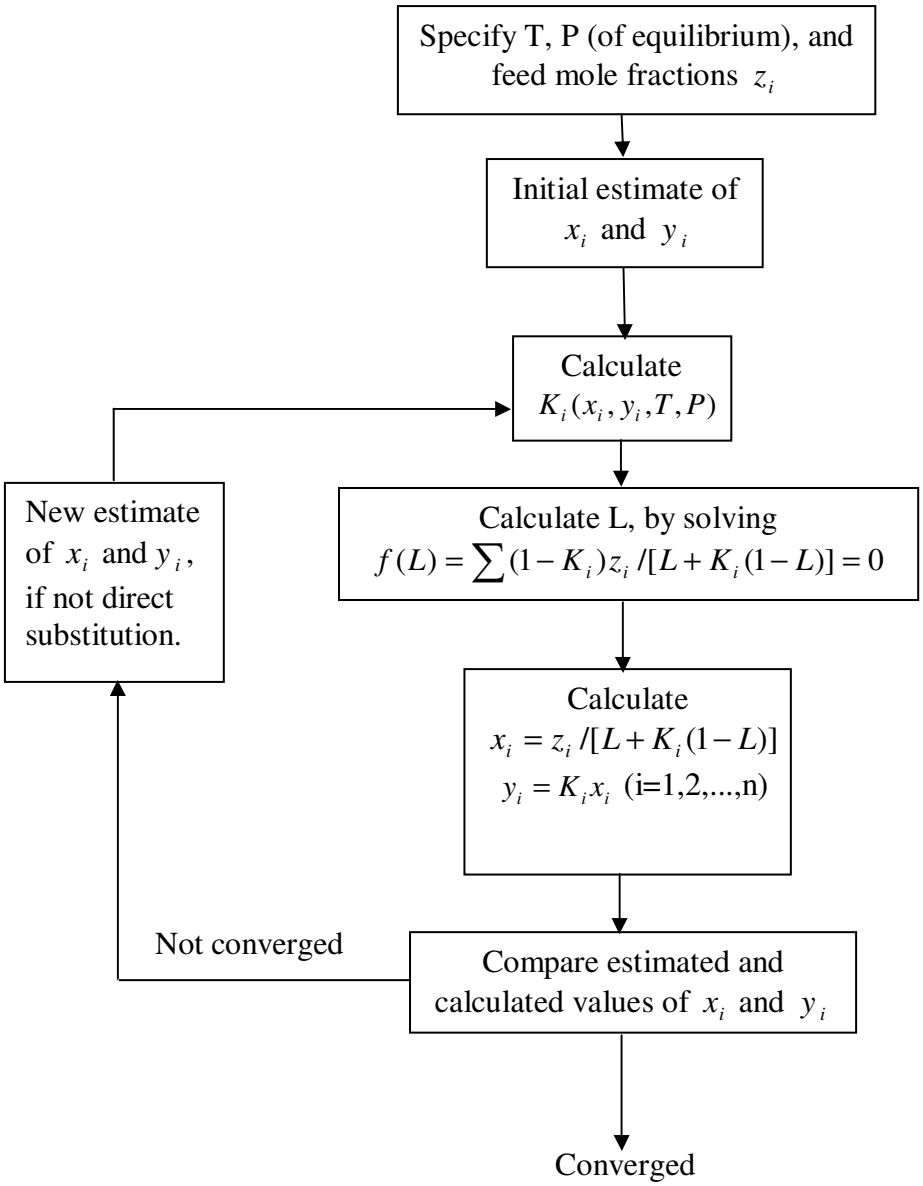


Figure 2.1 Flowchart for an Algorithm for Isothermal Flash Calculation Algorithm that Uses Composition Dependent Local Models

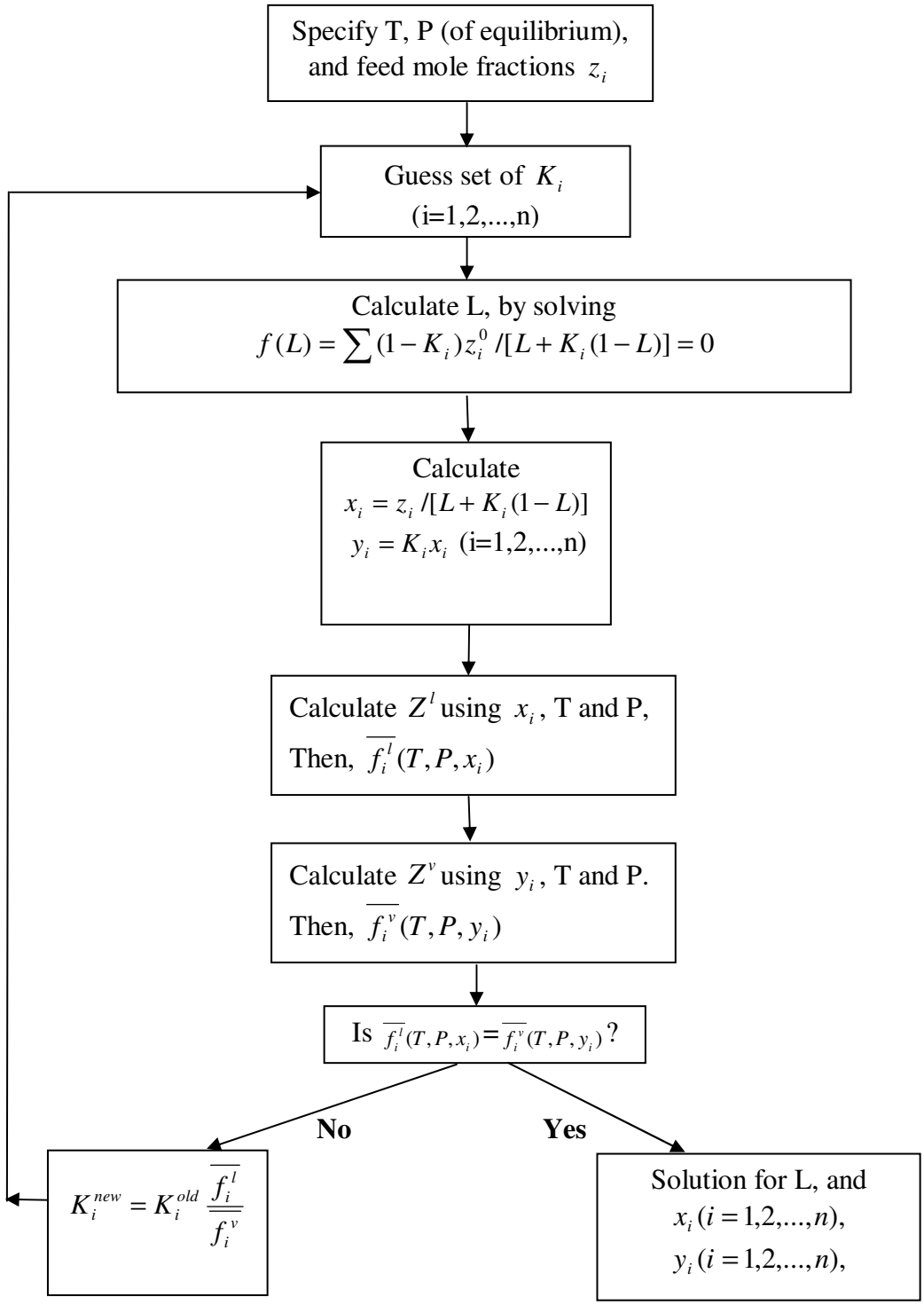


Figure 2.2 Flowchart for an Algorithm for Isothermal Flash Calculation Algorithm that Uses Equation of State

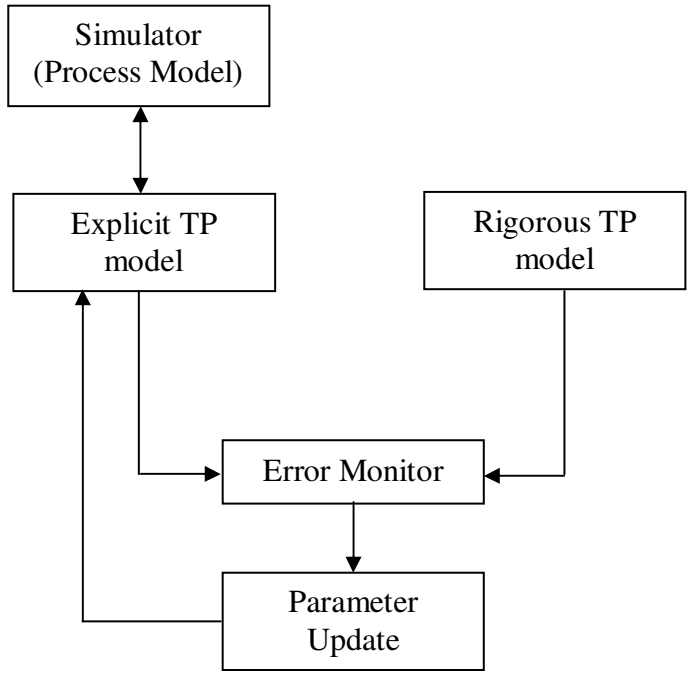


Figure 2.3 Architecture of Local Model Strategy

The first component of local model based system development is to formulate the approximate local function. Leesley [1977] developed several local models for ideal solutions, which didn't include composition dependence. He derived the local K model from the complete form:

$$K_i = \frac{y_i}{x_i} = \frac{\gamma_i^0 \Phi_i^v P_i^s \exp\left\{ \int_0^P \frac{\overline{V}_i^L dP}{RT} - \int_0^{P_i^s} \frac{V_i^L dP}{RT} \right\}}{\hat{\Phi}_i^V P} \tag{2.1}$$

After simplification, through ideal solution assumption and avoiding complex functional forms, an approximation to Eq. (2.1) can be developed for low pressure.

$$\ln K_i = A_{1,i} \ln P_i^s(T) + A_{2,i} - \ln P \tag{2.2}$$

Eq. (2.2) reproduces the temperature dependence of K-values fairly well over the range of 50-100 °C. However, the relation is too approximate to be useful above the pressures of 2-3 bars. Thus, a third adjustable coefficient has been introduced in the approximation formula for high pressure applications:

$$\ln K_i = \frac{A_{1,i}}{T} + A_{2,i} - A_{3,i} \ln P \tag{2.3}$$

Chimowitz [1983] extended the local models to non-ideal solutions for multi-component vapor-liquid system. One of the essential ideas has been to treat multi-component mixtures as pseudo-binary solutions. The functional form used to model the K values, which is composition-dependent for each pseudo-binary, has been also derived from basic thermodynamic considerations. In Chimowitz's work, he presented a local model for non-ideal solutions:

$$\ln K_i = \frac{A}{RT}(1 - x_i)^2 + \ln P_i^s - \ln P \tag{2.4}$$

Lender [1994] used a sequential least squares procedure to build approximating formulae from a general model that contains all the terms necessary to represent any particular mixture:

$$\ln K_i = A_1 + A_2 \ln P_i^s(T) + \frac{A_3}{T} + A_4 \frac{P}{T} + A_5 \ln P + \sum_{i=1}^{n} A_{5+i}(1 - x_i)^2 + \sum_{i=1}^{n} A_{5+n+i} x_i \tag{2.5}$$

The problem with this formula is that it has too many parameters (5+2n) to be efficient. To eliminate the unnecessary terms, Eq. (2.5) is rewritten in the form of:

$$A = (Q^T Q)^{-1} Q^T F \tag{2.6}$$

Eq. (2.6) is the least squares solution of the Eq. (2.5), where A is the vector of local model parameters, F is the vector of ln(K) obtained from experimental data, Q is the matrix of terms. If one lets $C = (Q^T Q)^{-1}$, then,

$$Corr_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} C_{jj}}} \tag{2.7}$$

If this correlation is found to be higher than a specified tolerance, one of the two parameters will be eliminated from the corresponding line and column in matrix C. Stepwise regression strategy is applied. For each parameter introduction, the parameters are re-computed and the residuals are examined. If they are satisfactory, the local model is accepted. If not, the parameter is eliminated before introducing the next one. When all parameters have been examined, the ones that gave the lowest residuals are accented.

The second component is the error monitor to estimate the range of validity of the local models for a set of parameters and identify when to update the local model. Leesley and Heyen [1977] fixed upper and lower values of the two independent variables, T and P. The bounds defined an interval, which included the two data points used in calculating the parameters. Hillestad et al. [1989] and Storen [1994] developed different error models for predicting the deviation between local and rigorous thermodynamic property models.

The third component is parameter estimation for updating models as the range of model have to change. Macchietto [1986] and Hillestad et al. [1989] applied recursive least-squares methods. The objective here is to preserve information from past data in the covariance matrix for the parameters. When a new data point is introduced, the covariance matrix can be updated and new values for the parameters can be obtained.

Storen [1994] used a simplified scheme with correction factors. Ledent [1994] presented a sequential least squares procedure.

In summary, two major approaches were developed to synthesize local thermodynamic models. One method is to derive a relationship based on a thermodynamic insight. Assumptions are made to simplify the relationship [Leesley 1977, Chimowitz 1983]. Each formula is suitable for a particular type of solutions (ideal/non ideal etc.). The final structure of formula mostly includes one constant term, one term that accounts for the temperature influence, and one term accounts for the pressure influence. In the case of non-ideal solutions, one or more terms may be added, to account for the composition influence. The other approach to the empirical formulation is on evaluated statistical basis, i.e. provide a general form of local model, which includes all the terms described above and their combinations, and then eliminate the redundant terms by examining the correlation for every pair of parameters [Ledent 1994].

Both approaches are human-centered strategies, and they share a common task of developing local models. An initial function structure is proposed first, and then the function structure is simplified and reduced by applying different strategies. The human-centered approaches may bring some limitations to the final structure of local models due to the over-simplified structure introduced by inappropriate assumptions made for the procedure of simplification, or, the insufficient description of the studied system introduced by proposed initial structure.

The form of local model is important because it is closely related to the correlation capabilities of the local model. It's also very important that the local model

ensures a fast, robust and consistent evaluation of the parameters. For this study, we are interested in a fully automatic algorithm, which can develop formula sufficiently and flexibly to all solution mixture types and functional forms. This can be obtained with symbolic regression through genetic programming.

2.2 Review of data mining techniques in the knowledge discovery process

In this study, genetic programming is used as a data mining tool for knowledge discovery in data.

Knowledge discovery in database (KDD) is the nontrivial process of searching valid, novel, potentially useful and ultimately understandable patterns or models in data. It involves a number of steps [Thuraisingham 1999]. For the sake of simplicity, these steps can be grouped as three major stages: data pre-processing, data mining, and post-processing. The simplified flowchart is shown in Figure 2.4.

Data mining, here, refers to a particular step in overall knowledge discovery process. As the core stage of KDD process, it focuses on applying discovery algorithms to find understandable and useful relationships from observed data. The data mining tasks can be divided into three major categories [Hand et al. 2001]: model building, discovering pattern and rules, and retrieval by content.

The tasks of model building can be categorized further based on objectives. The first is descriptive modeling. The goal of a descriptive model is to describe all of the data. Examples of such descriptions include models for the overall probability distribution of

the data (density estimation), partitioning of the n-dimensional space into groups (cluster analysis), and models describing the relationship between variables (dependency modeling). The second one is predictive modeling. The aim of predictive modeling is to build a model that will allow the value of one variable to be predicted from the known values of other variables. The key distinction between prediction and description is that prediction has, as its objective, one or more than one specifically targeted variables, while in descriptive problems no single variable is central to the model. Classification and regression are two of the most popular applications in predictive modeling. In classification, the variable being predicted is categorical, while in regression the variable is quantitative.

From a data mining viewpoint, this study can be set in the category of predictive modeling, which involves both model structure and parameter regression to build a local model for vapor-liquid equilibrium coefficient K.

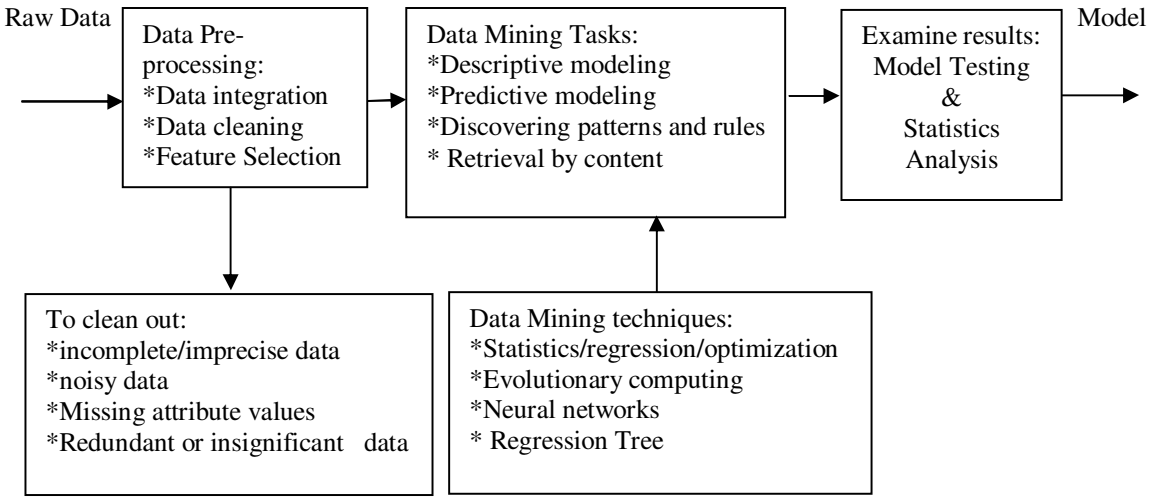


Figure 2.4 Knowledge Discovery Process

There are many different data mining techniques. In this section, only a few of techniques that are applicable for predictive modeling are summarized and compared. These include linear regression, regression tree, neural network, genetic algorithm and genetic programming.

Since the structure of the linear model is simple, easy to interpret, and estimation of parameters for linear models is straightforward, linear regression holds a special place among data-driven data analysis methods.

A linear regression model can be represented as:

$$\hat{y} = a_0 + \sum_{i=1}^{n} a_i X_i \tag{2.8}$$

where the a_i s are parameters that need to be estimated by fitting the model to the given data set. X_i can simply be original predictor variables x_i, or more generalized form of $f(x_i)$, i.e., transformations of the original x variables. $f(x_i)$ could be smooth function, such as log, square-root, or cross-product terms of $x_i s$ for polynomial models which allows interaction among the $x_i s$ in the model.

The parameter estimation for linear regression model is straightforward through least square fitting. However, selecting a proper model structure to fit the data is a challenge. This is because the selected model is generally empirical, rather than first principle. The model may not include all of the predictor variables, or certain functions of the predictor variables, that are needed for correct prediction.

Regression tree (RT) can be viewed as a variant of decision trees. It's designed for approximating real-valued functions, instead of being used for classification as what traditional decision tree does. Regression tree has representation as Figure 2.5:

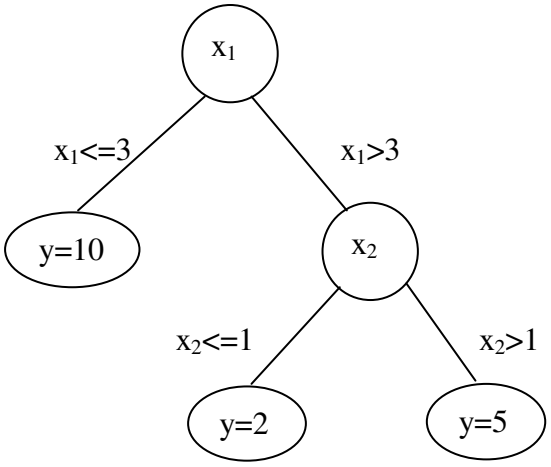


Figure 2.5 Example of a Regression Tree

Regression tree is built through a process known as binary recursive partitioning. This is an iterative process that splits the data into partitions, and then splitting it up further on each of the branches. In the structure of regression tree, each intermediate node is decision node that contains a test on one predictor variable's value. The terminal nodes of the tree contain the predicted output variable values.

The objective function for building an optimum tree structure, i.e. the minimized function, is the mean absolute. The process of regression tree induction usually has two phases: building a tree structure that covers the training data, and pruning the tree to the best size using validation data set. In training process, at each node, the best split that

minimizes the mean absolute distance is selected. Partitioning continues until a pre-specified minimum number of training data are covered by a node, or until the mean absolute distance within a node is zero. "Pruning" involves chopping off nodes from the bottom up so that there are fewer and fewer branches in the tree, so, the regression tree is pruned to avoid the over-fitting.

In terms of performance, regression tree is extremely effective in finding the key attributes in high dimensional applications. In most applications, these key features are only a small subset of the original feature set. On the negative side, regression trees cannot represent compactly many simple functions, for example linear functions. A second weakness is that the regression tree model is discrete, yet predicts a continuous variable. For function approximation, the expectation is a smooth continuous function, but a decision tree provides discrete regions that are discontinuous at the boundaries. For its explanatory capability, regression tree cannot describe the relationship between output variable and predictor variables in a form of functions.

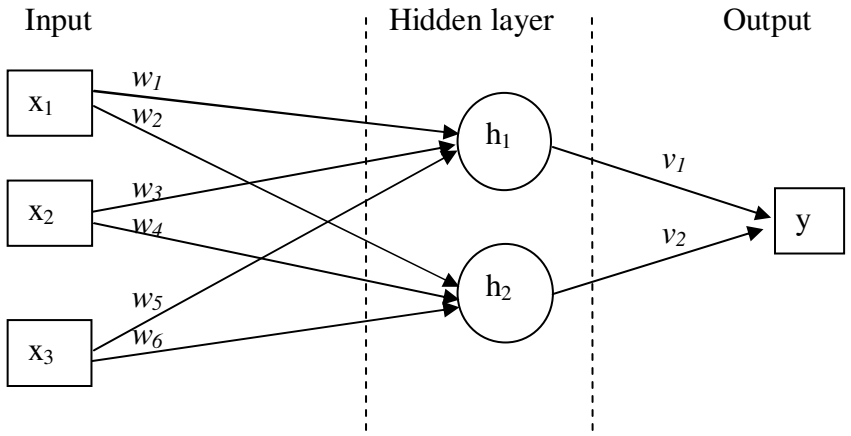


Figure 2.6 A Simple Multilayer Neural Network Model with Two Hidden Nodes

Neural networks have been found to be useful because of their learning and generalization abilities. Model structure presented by neural network is multiple layers of nonlinear transformations of weighted sums of the input variables. In a single hidden layer network as shown in Figure 2.6, w_i and v_i are weight factors, h_i is nonlinear transformation of sum of weighted input variables **x**. The output variable y is sum of weighted h_i. Therefore, in general, output variable y is a nonlinear function of the input variables **x**. As a result, neural network can be used as a nonlinear model for regression. If there is more than one hidden layers, the outputs from one layer, which is the transformed linear combinations of nodes in previous layer, serve as inputs to the next layer. In this next layer, the inputs are combined in exactly the same way, i.e., each node forms a weighted sum that is then nonlinearly transformed. The number of layers and the number of nodes per layer are important decisions. There is no limit to the number of layers that can be used, though it can be proven that a single hidden layer (with enough nodes in that layer) is sufficient to model any continuous functions [Hand 2001]. Once a network has been structured for a particular application, this network is ready to be trained. The weights w_i and v_i are the parameters of this model and must be determined from the data in training process.

The fact that neural network is highly parameterized makes it very flexible, so that it can accurately model relatively small irregularities in functions. On the other hand, such flexibility means that there is a serious danger of over fitting. In recent years, strategies have been developed for overcoming this problem. Due to the multiple layers of nonlinear transformation of weighted sum, the relationship between output variable y

and input variables **x** is hard to be presented in a single explicit form of mathematical model, the neural network is usually used as a black box for predictive modeling.

Evolutionary algorithms (EAs) provide an effective avenue for structural and parametric regression. EAs are originally divided into three major categories, namely evolutionary programming (EP) [Fogel et al. 1966], evolution strategy [Rechenberg 1973] and genetic algorithms (GAs) [Holland 1975]. In the 1990s, a new branch called genetic programming (GP) was added to the group which was introduced by John Koza [Koza 1992, 1994]. GP is an extension of John Holland's GA in which the genetic population consists of models of varying complexities and structures.

GA uses binary string to represent possible solutions to a problem, whereas GP uses tree structure as knowledge representation. Both GA and GP guide the search by using some genetic operators and the principle of "survival of the fittest". The major difference between GA and GP is their coding used to represent possible solutions for a problem. In GA, the solution is presented in a form of fixed length binary string, and its output is a quantity. The aim of such coding is to allow the possible solutions to be manipulated with those genetic operators in evolutionary process. Sometimes, it's a challenge to encode the possible solutions in a structure of binary string. GP uses tree structure with variable sizes, which allows the solution to be manipulated in their current form. Therefore, GP can be used as a tool for symbolic regression, i.e. structural regression. The details of GP will be explained in the next section.

2.3 Elements of structural regression using genetic programming

The objective of this research is to find the approximate function for K, which includes parametric and structural regression. As mentioned earlier, Genetic programming (GP) is an extension of the genetic algorithm in which the genetic population consists of possible solutions (that is, compositions of primitive functions and terminals).

Koza [1992] demonstrated a surprising result that, genetic programming is capable of symbolic regression. To accomplish this, genetic programming starts with a pool of randomly generated mathematical models and genetically breeds the population using the Darwinian principle of survival of the fittest and an analog of naturally occurring genetic crossover (sexual recombination) operation. In other words, genetic programming provides a way to search the space of possible model structures to find a solution that fits, or approximately fits, a given data set.

Genetic programming is a domain independent method that genetically breeds populations of models to fit the given data set by executing the following three steps that are also shown in Figure 2.7:

- Generate an initial population of random individuals (mathematical models) composed of the primitive functions and terminals of the problem.
- Iteratively perform the following intermediate-steps until the termination criterion has been satisfied:

- Execute each individual in the population and assign it a fitness value according to how well it solves the problem.
- Create a new population of individuals by applying the following three primary operations. The operations are applied to individual(s) in the population selected with a probability based on fitness (i.e., the fitter the individual, the more likely it is to be selected).
    - Reproduction: Copy an existing individual to the new population.
    - Crossover: Create two new offspring individuals for the new population by genetically recombining randomly chosen parts of two existing individuals. The genetic crossover (sexual recombination) operation (described below) operates on two parental individuals and produces two offspring individuals using parts of each parent.
    - Mutation: randomly alteration in existing individuals, and produces one offspring individuals.

- The single best individual in the population produced during the run is designated as the result of the run of genetic programming. This result may be the solution (or approximate solution) fitted to the given data set.

The description on GP's components will be given in the following subsections, which includes terminal set, function set, fitness function, genetic operators and selection strategies.

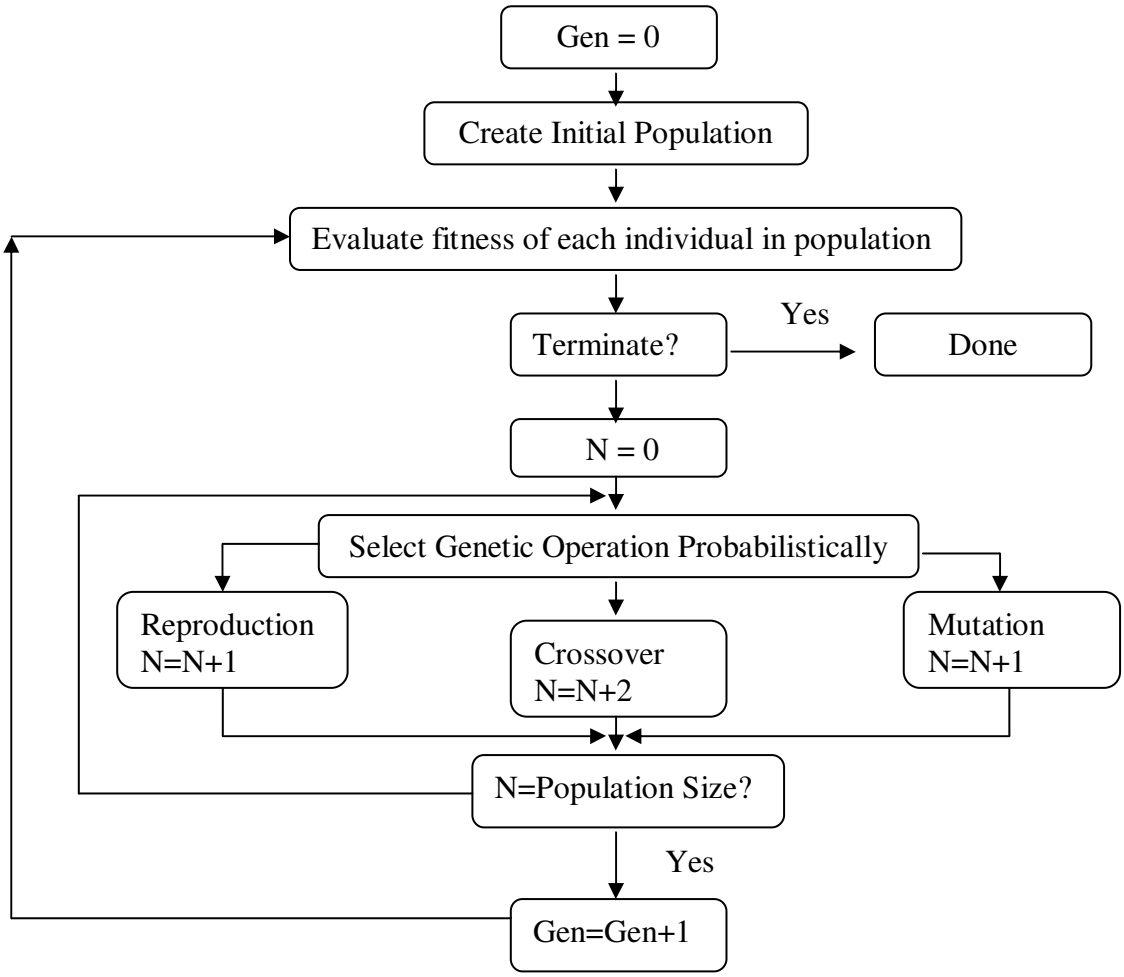


Figure 2.7 A Flowchart for Computation through Genetic Programming

2.3.1 Terminal set, function set and initial representation

In genetic programming, any explicit mathematical equations can be represented by a tree that intermediate nodes are mathematical operators (functions), and terminal nodes (leaves) are input variables and parameters.

As shown in Figure 2.8, the tree corresponding to the equation of ideal heat capacity $C_p = a + b \cdot T + c \cdot T^2$ can be represented as:

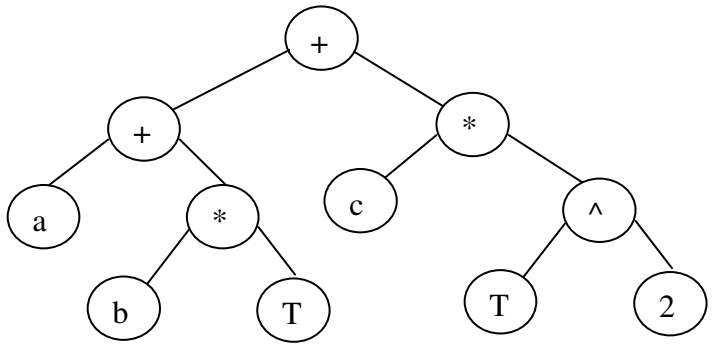


Figure 2.8 Functional Representation of $a + b \cdot T + c \cdot T^2 (C_p)$ Using a Tree Structure

In this graphical depiction, the function set consists of intermediate nodes of the tree that are labeled with several mathematical operators, such as +,* and ^. The terminal set consists of terminal nodes (leaves) of the tree that are labeled with input variables T, parameters *a*, *b, c* and constant "2".

The terminal and function sets are important components of genetic programming. The terminal and function sets contain the primitive elements of the mathematical model to be composed. The sufficiency property requires that the set of terminals and the set of primitive functions should be capable of expressing a solution to the problem.

2.3.2 Fitness measures for models of varying complexity

The most difficult and most important concept of genetic programming is the fitness function. The fitness function determines how well a generated model is fit to the data. Fitness is the driving force of genetic programming. In genetic programming, each individual model in a population is assigned a fitness value.

2.3.2.1 Fitness function with no penalty for the model complexity

The basic fitness function is a function of the difference between the model predicted value and the data. Widely used basic fitness functions include the raw fitness, adjusted fitness and normalized fitness. The raw fitness is the sum of squared errors. In particular, the raw fitness *r (i, t)* of an individual model *i* in the population of size *M* at any generation *t* is

$$r(i,t) = \sum_{j=1}^{N_e} \left[S(i,j) - C(j) \right]^2 \tag{2.9}$$

where *S (i, j)* is the value returned by individual model *i* for data case *j* (of N_e data cases) and *C (j)* is the data value for data case *j*. The closer this sum of squared errors is to zero, the better the model.

Another popular raw fitness is absolute error. Since squared error gives greater weight to extreme differences between the predicted value and the data than absolute error does, the quality of the model is perhaps more appropriately reflected in absolute

error for some cases. Each raw fitness value can be adjusted (scaled) to produce an adjusted fitness measure *a (i, t)*. The adjusted fitness value is

$$a(i,t) = \frac{1}{(1 + r(i,t))} \tag{2.10}$$

where *r (i, t)* is the raw fitness for individual model *i* at generation *t*. Unlike raw fitness, the adjusted fitness is larger for better individuals in the population. Moreover, the adjusted fitness lies between 0 and 1.

Each such adjusted fitness value *a (i, t)* is then normalized. The normalized fitness value *n (i, t)* is

$$n(i,t) = \frac{a(i,t)}{\sum_{j=1}^{M} a(j,t)} \tag{2.11}$$

The normalized fitness not only ranges between 0 and 1 and is larger for better individuals in the population, but the sum of the normalized fitness values is 1. Thus, normalized fitness is a probability value.

2.3.2.2 Fitness function with model complexity control

It was noted that, after a certain number of generations, the average size of the mathematical models in a population would start growing at a rapid pace. However, the increase in complexity of model doesn't show significant improvement on fitness. This behavior displayed by GP is called bloat. Bloat often occurs in symbolic regression problems where GP runs start from a population of small size individuals, and then grows

in complexity to be able to comply with all data. In practice, bloat affects the efficiency of GP significantly. The over-complicated model structures are computationally expensive to evolve or use, it also can be hard to interpret, and may display poor ability of generalization, i.e. overfitting problem. Over the years, many theories have been proposed to explain bloat from different aspects, but none of them is universally accepted as a unified theory to explain the various observations on bloat. Therefore, several strategies for control of complexity were proposed with different theoretical foundations.

2.3.2.2.1 Minimum description length

As described earlier, the problem of over fitting is a common problem to every application in GP. Besides the "complexity" of the generated model by GP, the presence of noise in the data is another possible cause of over fitting. Good results with noisy data are only achievable at the cost of precision on the entire data distribution. The issue of selecting a model of appropriate complexity to overcome the over fitting problem is, therefore, always a key concern in any GP application.

One of proposed strategies is the Minimum Description Length (MDL), which provides a trade-off between the accuracy and the complexity of the model by including a structure estimation term for the model. The final model (with the minimal MDL) is optimum in the sense of being a consistent estimate of the complexity of model while achieving the minimum error.

There are a number of criteria that have been proposed for MDL, which compare models based on a measure of goodness of fit penalized by model complexity. The most popular and widely used criteria are Akaike's Information Criterion (AIC) (Eq. (2.12)) and Schwarz Bayesian Information Criterion (BIC) (Eq. (2.13)).

Akaike's information criterion (AIC)

$$n/2 \ \ln(MSE) + k. \tag{2.12}$$

Schwarz Bayesian Information Criterion (BIC)

$$n/2 \ \ln(MSE) + k \ \ln(n)/2. \tag{2.13}$$

where *MSE* is the mean squared prediction error, $MSE = [1/n]\ SSE$, n is the number of the data points used for the identification of the model, i.e. the sample size.

Both AIC and BIC take the form of a penalized maximized likelihood, and their first term can be interpreted as the evaluation on model's accuracy, the second term is the penalty term, which can be interpreted as the complexity of the model, which is a function of the number of parameters and the depth of the tree structure that represents the model. These two criteria utilize different penalties: AIC adds 1 for each additional variable included in a model, while BIC adds ln (n)/2.

2.3.2.2.2 Parsimony pressure

A variety of practical techniques have been proposed to control complexity bloat. Among these techniques, parsimony pressure method is a simple and frequently used

method to control bloat in genetic programming [Zhang et al. 1993, Zhang et al. 1995]. In this method, the parsimony pressure is applied to the original fitness function:

$$f_p(x) = f(x) - c \cdot l(x) \tag{2.14}$$

$f_p(x)$ is the new fitness function with parsimony pressure term. $f(x)$ is the original fitness of model x, as mentioned above. c is a constant, known as the parsimony coefficient. $l(x)$ is the size of model x, counted as the number of intermediate nodes in the tree representation, i.e., the number of mathematical operators appeared in the model. This new fitness function $f_p(x)$ minimizes model size by using the penalty term as a mild constraint. The penalty is simply proportional to model size. The fitness of models will decrease with the increase on model size. The strength of control over bloat is determined by the parsimony coefficient c. The value of this coefficient is very important: if "c" has a small value, GP runs will still bloat wildly; if the value is too large, GP will take the size of the minimization model as its main target and will almost ignore fitness, which incurs the loss of model accuracy, consequently, weaken the prediction ability of model. However, the proper values of parsimony coefficient highly depend on specific problem being solved, the choice of functions and terminals, and various GP parameter settings. Very few theories have been proposed to help setting the parsimony coefficient, and trial and error method was widely used before Poli [2008] introduced a simple, effective, and theoretically sound solution to this problem.

The strategies introduced above are ones focusing on the fitness against complexity bloat, and here-upon, over fitting problem. Other than those anti-bloat selection rules, numerous empirical techniques have also been proposed to control

complexity bloat, which are based on GP algorithm's improvements. Briefly, these techniques can be summarized into two major categories: size and depth limits [Koza 1992, and anti-bloat genetic operators [Kinnear 1993, Langdon 1998, Langdon 2000, and Crawford-Marks et al. 2002].

2.3.2.3 Fitness function using external validation

Section 2.3.2.2 introduces two of the well-accepted strategies on the complexity control. Although based on different theories, they both combine multiple objectives into a scalar fitness function. A different strategy for choosing models is sometimes used, not based on adding a penalty term, but instead based on external validation of the model. The basic idea is to randomly split the data into two parts, a training set and a validation set. The training set is used to construct the models and estimate the parameters. Then, the fitness function is recalculated using the validation set. These validation scores are used to select models. In the validation context, since the training set and validation data set are independently and randomly selected, for a given model the validation score provides an unbiased estimate of the fitness value of that model for new data points, therefore, the difference in validation scores can be used to choose between models. This general idea of validation has been extended to the notion of cross-validation. This external validation method will be discussed further in section 3.5.1.3.

2.3.3 Genetic operators

In this section, three genetic operators will be described in detail. In the first section, reproduction and crossover will be introduced as two primary operations, and the mutation, including its two different types, will be introduced as a secondary operation.

2.3.3.1 Reproduction and crossover

The two primary genetic operations in GP for modifying the structures are fitness proportionate reproduction, as shown in Figure 2.9, and crossover, as shown in Figure 2.10.

The operation of fitness proportionate reproduction for the genetic programming is an asexual operation in that it operates on only one parental individual (model). The result of this operation is one offspring individual (model). In this operation, if $f(i, t)$ is the fitness of an individual i in the population M at generation t, the individual i will be copied into the next generation with probability

$$\frac{f(i,t)}{\sum_{j=1}^{M} f(j,t)} \tag{2.15}$$

The operation of fitness proportionate reproduction does not create anything new in the population. It increases or decreases the number of occurrences of individuals already in the population, and improves the average fitness of the population (at the expense of the genetic diversity of the population). To the extent, it increases the number

of occurrences of more fit individuals and decreases the number of occurrences of less fit individuals.

The crossover (recombination) operation for the genetic programming starts with two parental individuals (models). Both parents are selected from the population with a probability equal to its normalized fitness. The result of the crossover operation is two offspring individuals (models). Unlike fitness proportionate reproduction, the crossover operation creates new individuals in the populations.

Parent: $a \cdot T + b$ Offspring: $a \cdot T + b$

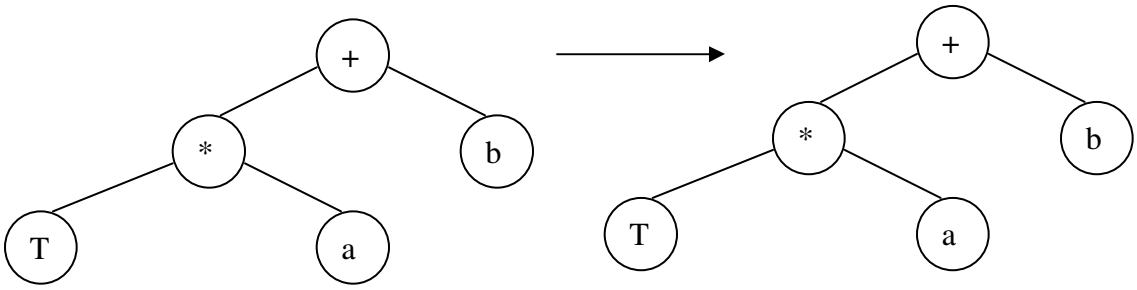


Figure 2.9 An Example of Reproduction Operator

The operation begins by randomly and independently selecting one point in each parent using a specified probability distribution (discussed below). The number of points in two parental individuals typically is not equal to each other. As will be seen, the crossover operation is well-defined for any two individuals. That is, for any two individuals and any two crossover points, the resulting offspring are always valid individuals in the population. Each offspring contains some traits from its parent.

The crossover fragment for a particular parent is the rooted sub-tree whose root is the crossover point for that parent and where the sub-tree consists of the entire sub-tree lying below the crossover point (i.e., more distant from the root of the original tree).

The first offspring is produced by deleting the crossover fragment of the first parent and then impregnating the crossover fragment of the second parent at the crossover point of the first parent. The second offspring is produced in a symmetric manner. Since entire sub-trees are swapped, this genetic crossover (recombination) operation produces syntactically and semantically valid individuals as offspring regardless of which point is selected in either parent. For example, consider the parental individuals, i.e., algebraic equation $a \cdot T + b$ and $T^2 + a \cdot b$. These two models can be depicted graphically as rooted, point-labeled trees with ordered branches.

The two parental models are shown in Figure 2.10. Suppose that the crossover points are randomly selected for each parent individual. The crossover points are therefore the * in the first parent and the + in the second parent. The places from which the crossover fragments were removed are identified with dash line.

2.3.3.2 Mutation

Mutation is another important feature of genetic programming. Two types of mutations are possible. In the first type, a function can only replace a function or a terminal can only replace a terminal. In the second type, an entire sub-tree can replace another sub-tree. Figures 2.11 and 2.12 explain the concept of mutation:

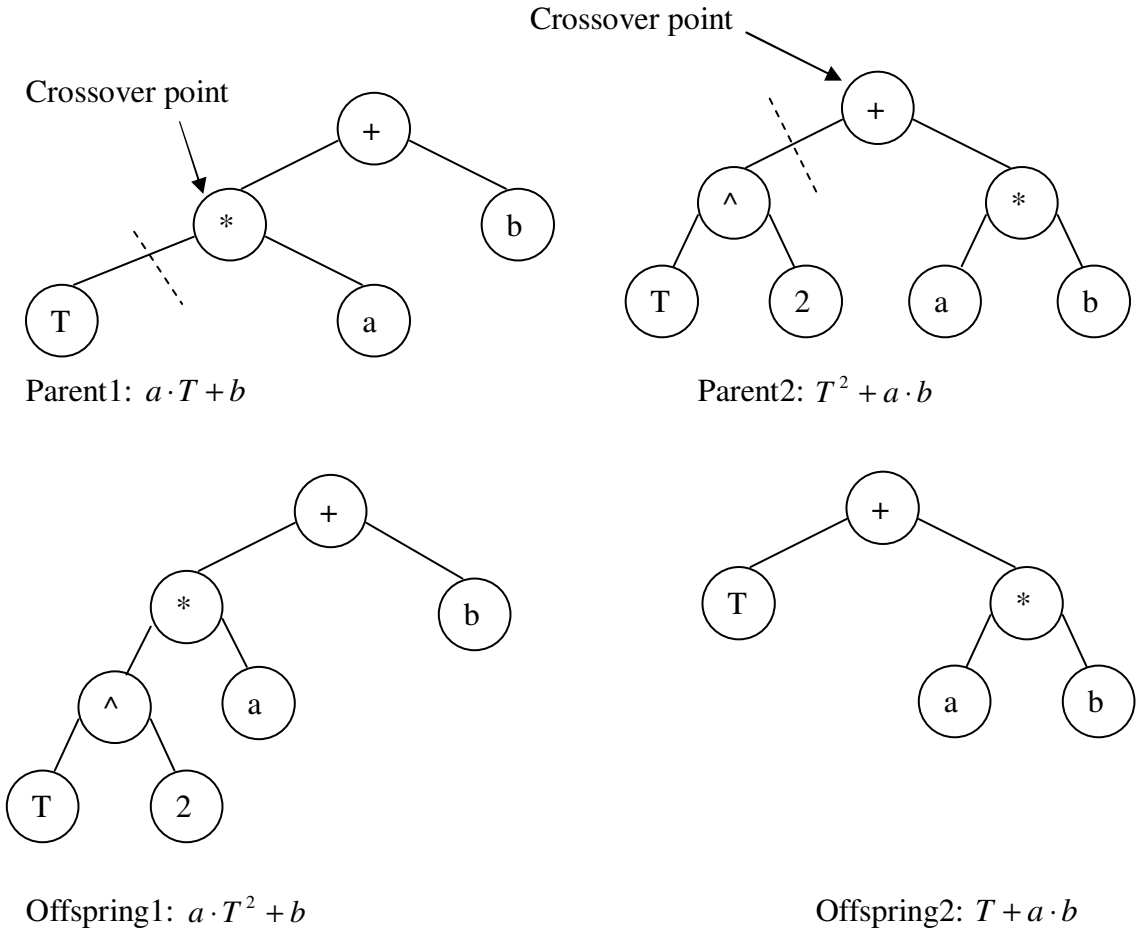


Figure 2.10 Crossover Operation for an Algebraic Equation Manipulation

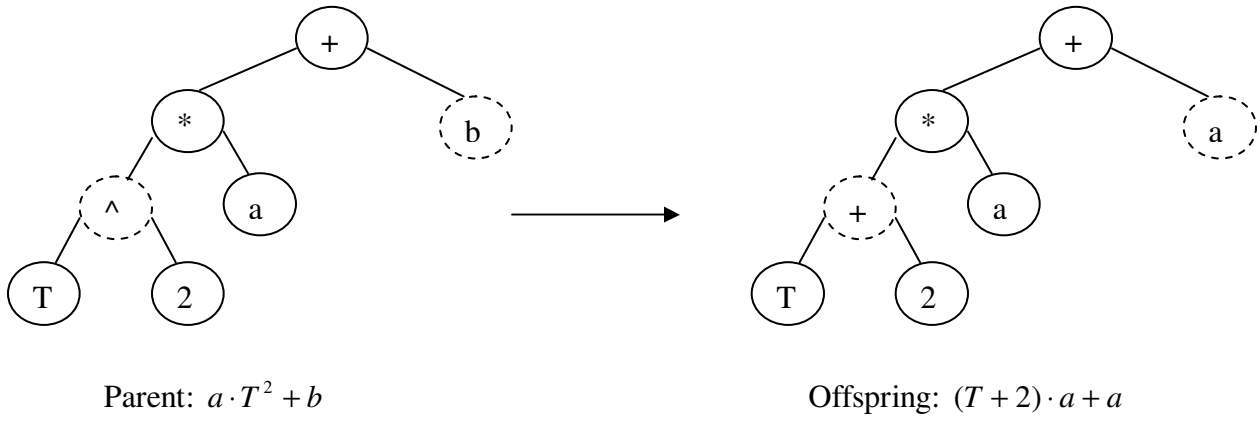
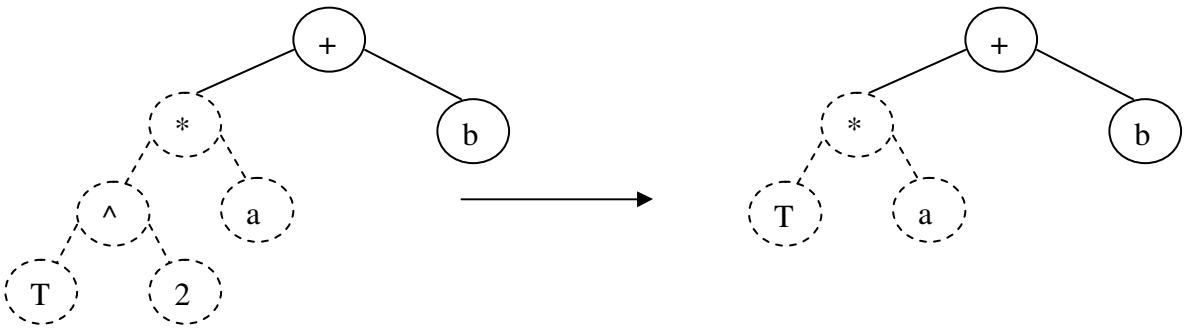


Figure 2.11 An Example of Mutation Operation, Type I



Parent: $a \cdot T^2 + b$ Offspring: $a \cdot T + b$

Figure 2.12 An Example of Mutation Operation, Type II

2.3.4 Selection strategy

There are many different selection methods based on fitness. The most popular is fitness-proportionate selection. If *f (i, t)* is the fitness of individual *i* in the population at generation *t*, then, under fitness-proportionate selection, the probability that individual *i* will be selected to process genetic operation is:

$$\frac{f(i,t)}{\sum_{j=1}^{M} f(j,t)} \tag{2.16}$$

where *M* is the population size. Among the alternative selection methods are tournament selection and rank selection [Goldberg 1989]. In rank selection, selection is based on the rank (not the numerical value) of the fitness values of the individuals in the population. Rank selection reduces the potentially dominating effects of comparatively high-fitness individuals in the population by establishing a predictable, limited amount of selection pressure in favor of such individuals. At the same time, rank selection exaggerates the

difference between closely clustered fitness values so that the better ones can be sampled more.

In tournament selection, a specified group of individuals (typically two) are chosen at random from the population and the one with the better fitness (i.e., the lower standardized fitness) is then selected.

2.4 Parametric regression: review of objective functions and optimization methods

Optimization techniques are used to find a set of values for design variables that best meet an objective. In parameter estimation, the optimum parameter values are searched with the aid of a selected optimization method, to minimize or maximize a well-defined objective function that depends on the parameters, measurements and the model.

The objective function is a suitable measure of the overall departure of the model performance from the observed measurements. A widely used objective function in parameter estimation is the least squares formulation.

For the model equation $Y = f(x,b)$, b is denoted as a vector that is an estimate of the parameter vector β, x is a vector of independent variables, the sum of squared residuals is:

$$\Phi = \varepsilon'\varepsilon = (Y^* - Y)'(Y^* - Y) \tag{2.17}$$

where Y^* is vector of experimental observations of the dependent variables. The least squares method is used to evaluate the unknown vector b by minimizing the sum of squared residuals Φ.

In linear regression analysis, the equation is a linear function of the parameters, which is in the form of:

$$Y^* = X\beta + \mu \tag{2.18}$$

Eq. (2.17) can be:

$$\Phi = \varepsilon'\varepsilon = (Y^* - Xb)'(Y^* - Xb) \tag{2.19}$$

In order to calculate the vector b, which minimizes Φ, the partial derivative of Φ with respect to b is taken, and set equal to zero:

$$\frac{\partial \Phi}{\partial b} = (-X)'(Y^* - Xb) + (Y^* - Xb)'(-X) = 0 \tag{2.20}$$

Eq. (2.20) can be simplified and rearranged to yield:

$$b = (X'X)^{-1} X' Y^* \tag{2.21}$$

Therefore, the value of the parameter vector b can be obtained directly from the Eq. (2.21) given above.

In nonlinear regression analysis, the model equation $Y = f(x,b)$ is a relation that is nonlinear with respect to the parameters. There are several techniques for minimization of the sum of squared residuals described by Eq. (2.17). These techniques are broadly classified into two categories: gradient methods and direct search methods. The gradient search methods require derivatives of the objective functions whereas the direct methods are derivative-free and rely solely on function evaluations. The gradient search methods are efficient for smooth functions, and still efficient if there are some discontinuities in the derivatives. Direct search techniques, which use function values, are more efficient for highly discontinuous functions.

The major gradient search methods include: Gauss-Newton, steepest descent, Marquardt and Newton's. All of these methods are local methods that provide global solution only for convex models.

Gauss-Newton method can be used to convert nonlinear problem into a linear one by approximating the function Y by a Taylor series expansion around an estimated value of the parameter vector b:

$$Y(x,b) = Y(x,b^m + \Delta b) = Y(x,b^m) + \left.\frac{\partial Y}{\partial b}\right|_{b^m} \Delta b = Y + J\Delta b \tag{2.22}$$

where the Taylor series has been truncated after the second term. Eq. (2.22) is linear in Δb.

Therefore, the problem has been transformed from finding b to that of finding the correction to b, that is Δb, which must be added to an estimate of b to minimize the sum of squared residuals.

$$J = \begin{bmatrix} \frac{\partial Y_1}{\partial b_1} \cdots\cdots \frac{\partial Y_1}{\partial b_k} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ \frac{\partial Y_n}{\partial b_1} \cdots\cdots \frac{\partial Y_n}{\partial b_k} \end{bmatrix}, \tag{2.23}$$

and $\Delta b = (J'J)^{-1} J'(Y^* - Y)$ (2.24)

$$b^{m+1} = b^m + \Delta b \tag{2.25}$$

where m is the iteration counter.

In Gauss-Newton method, the drawback is the fact that the incremental changes, namely the Δb as described previously, can be estimated very poorly due to computation

of the partial derivative matrix $(J'J)^{-1}$ when it is close to singular. The result is that the convergence may be very slow with a large number of iterations being required. Even wrong signs may occur on the Δb s, and then the procedure will move in the wrong direction. The method may not converge at all with the residual sum of squares continuing to increase. Also, the closer a model is to behaving like a linear model, the more likely it is to converge in a small number of iterations from a reasonable starting point and, more important, the zone of ability of converge is greater for a close-to-linear model than a far-from-linear one. Since this objective function is a quadratic one in nature, Gauss-Newton method is susceptible.

In the steepest descent method, the gradient of a scalar objective function gives the direction of the greatest objective function decrease at any. Therefore, the initial vector of parameters estimates are corrected in the direction of the negative gradient of Φ:

$$\Delta b = -K\left(\frac{\partial \Phi}{\partial b}\right) \tag{2.26}$$

where Φ is the sum of squared residuals and K is a suitable constant factor and Δb is the correction vector to be applied to the estimated value of b to obtain a new estimate of the parameter vector, same as before:

$$b^{m+1} = b^{m} + \Delta b \tag{2.27}$$

where m is the iteration counter.

Then, Δb can be calculated from:

$$\Delta b = 2KJ'(Y^{*} - Y) \tag{2.28}$$

where J is the Jacobian matrix of partial derivatives of Y with respect to b evaluated at all n points where experimental observations are available, as shown in Gauss-Newton method.

The steepest descent method moves toward the minimum sum of squares without diverging, provided that the value of K, which determines the step size, is small enough. The value of K may be a constant throughout the calculations, which may change at calculation step. However, the rate of convergence to the minimum decreases as the search approaches this minimum.

Marquardt method is a compromise between the Gauss-Newton and the steepest descent methods. This interpolation is achieved by adding the diagonal matrix (λI) to the matrix *(J'J)* in the function of Δb in Gauss-Newton method above:

$$\Delta b = (J'J + \lambda I)^{-1} J'(Y^* - Y) \tag{2.29}$$

The value of λ is chosen, at each iteration, so that the corrected parameter vector will result in a lower sum of squares in the following iteration. We can see from Δb equation above, as $\lambda \to 0$, Marquardt method approaches the Gauss-Newton method; while as $\lambda \to \infty$, this method is identical to steepest descent, with the exception of a scale factor that does not affect the direction of the parameter correction vector but that gives a small step size. From this aspect, by selecting appropriate value of λ, an indicator of compromising between Gauss-Newton and Steepest Descent method, Marquardt method can combine the best feature of those two methods: almost always converges and does not "slow down" [Draper et al. 1981].

In Newton's method, similar to Gauss-Newton method that approximates the function Y by a Taylor series expansion to the second term, the sum of squared residuals Φ is also expanded by Taylor series up to the third term:

$$\Phi(x,b) = \Phi(x,b^{(m)}) + \left(\frac{\partial \Phi}{\partial b}\right)^{(m)} \Delta b + \frac{1}{2}\Delta b' \left(\frac{\partial^2 \Phi}{\partial b^2}\right)^{(m)} \Delta b \tag{2.30}$$

Taking the partial derivative of both sides of Eq. (2.30) with respect to b gives

$$\left(\frac{\partial \Phi}{\partial b}\right) = \left(\frac{\partial \Phi}{\partial b}\right)^{(m)} + \left(\frac{\partial^2 \Phi}{\partial b^2}\right)^{(m)} \Delta b = -2J'(Y^* - Y) + H\Delta b \tag{2.31}$$

where H is Hessian matrix of the second-order partial derivative of Φ with respect to b evaluated at all n points where experimental observations are available:

$$H = \begin{bmatrix} \frac{\partial^2 \Phi}{\partial b_1^2} \cdots\cdots\cdots \frac{\partial^2 \Phi}{\partial b_1 \partial b_k} \\ \frac{\partial^2 \Phi}{\partial b_2 \partial b_1} \cdots\cdots\cdots \frac{\partial^2 \Phi}{\partial b_2 \partial b_k} \\ \frac{\partial^2 \Phi}{\partial b_k \partial b_1} \cdots\cdots\cdots \frac{\partial^2 \Phi}{\partial b_k^2} \end{bmatrix}, \tag{2.32}$$

The above description of optimization methods is based on least squares as the objective function. Other than least squares, maximum likelihood estimation is also popularly used as an objective function for parameter estimation purpose, which is based on statistical principles and account of data quality.

In general, statistically based parameter estimation reduces the problem of the determination of parameters in the mechanistic model to assessing the correspondence between the residuals generated by a particular set of parameter values and the assumptions made about the residual distribution.

It is assumed that every vector of residuals ε for an experiment is a random vector following a probability density function of specified form $p_i(\varepsilon_i, b)$, where b is the unknown vector of parameters. The experimental outcome of a ε_i vector is regarded as a random sample out of the distribution defined by p_i. The combination of all p_i for all ε_i results in the likelihood function:

$$L(b; \varepsilon_1, \varepsilon_2, \varepsilon_3, ...)$$

which, for correct specification of the joint probability density function for all ε's and known true b values, represents the probability density of getting just that set of ε_i vectors obtained experimentally.

In the parameter estimation situation, b is not known. So in Maximum Likelihood Estimation, those unknown values are searched with the aid of an optimization method, which maximize this function L. This means that the "optimal" values b obtained are those parameter values which generate the residual pattern for which the probability density is highest.

Maximum Likelihood (ML) method can be used with any joint probability density functional form of residuals, while one specific distribution properties of residuals has to be assumed before ML method is applied. In most of applications, the normal distribution is used. However, often these assumptions are not fulfilled at optimal parameter values determined due to random measurement errors, systematic measurement errors, such as drift, calibration, measurement technique, deterministic model inadequacies, errors in values assumed to be precisely known dependent variable.

Other than these measurement errors, there are three types of computation related errors: the truncation error, the round off error, and the propagation error. The truncation error is a function of the number of terms that are retained in the approximation of the solution from the infinite series expansion. Since computers carry number using a finite number of significant figures, a round off error is introduced in the calculation when the computer rounds up or down (or just chops) the number to n significant figures. Meanwhile, the truncation and round off errors may accumulate and propagate, creating the propagation error, which may grow in exponential or oscillatory pattern. Thus, these errors may cause the calculated solution to deviate drastically from the correct solution.

All errors explained above may affect the distribution properties of residuals, in other words, the assumptions made on the probability density function of the residuals may be violated. In this case, the optimal parameter values may be not trustable.

2.5 Applications of intelligent system in chemical engineering

Development of intelligent systems in process engineering has been mainly focused in the following six areas [Stephanopoulos 1987, 1994]:

- Process design: Select thermodynamic models and estimate physical properties: select the best thermodynamic model(s) for the problem [Fredenslund 1980, Banares-Alcantara et al. 1985, and Gani 1989]. If no explicit models are available, the system could select a method and estimate the value of the physical property [Friese 1998]. Use process data with

engineering heuristics to recommend the optimal processing method for the task at hand [Bamicki 1990].

- Fault diagnosis: process troubleshooting, i.e., determining the origins of process problems and recommending solutions [Frank 1997, Ozyurt 1998, Ruiz et al. 2000].
- Process control: improving process control through utilization of qualitative process information, trend analysis, neural networks, etc.
- Planning and operations: scheduling, developing procedures, assessing safety concerns, executing complex inter-related procedures, and aiding maintenance [Csukas 1998].
- Modeling and simulation: using qualitative reasoning and symbolic computing to model and simulate chemical processes [Cao et al. 1999, McKay et al. 1997, Csukas 1998, Greeff 1998, Gao et al. 2001, Hinchliffe 2003, Grosman et al. 2004].
- Product design, development, and selection: recommending chemical formulations, compositions, materials, process procedures, etc., required to design, develop, or select a new or existing product that achieves specified objectives [Xu 2005].

The foundation of accurate chemical process design and simulation is a correct estimate of physical and thermodynamic properties. In this exploratory project, an automatic procedure will be developed to identify a thermo-physical model from a set of

given data. This model is expected to be used in further investigation of the physical system or to validate the structure of an existing model developed in some other way.

CHAPTER THREE

A HYBRID SYSTEM FOR STRUCTURAL AND PARAMETRIC OPTIMIZATION

In this chapter, the structure of hybrid system and its implementation is introduced. The structure of hybrid system is presented in the first section. The data used throughout this research and its preparation are given in the second section. In the third section, the regression strategies applied in this research is introduced. The fourth section describes the determination of genetic programming (GP) controlling parameters. In the last section, the model verification strategies are summarized.

3.1 The system structure

The purpose of this research is to investigate the feasibility of designing a general-purpose machine function identification system which can automatically build a function model to fit the given experimental data. The approach is to solve the function identification problems is through coupling the symbolic computing method (Genetic Programming) and a parameter regression method. The parameter regression process involves either linear or nonlinear depending on the problem definition. The two-layer structure is shown in Figure 3.1. The parameter regression is embedded in the genetic

programming as an inner layer. For the given data set, the structural regression system searches the space of mathematical models, dynamically creates new generation of mathematical models using genetic programming, and optimizes the parameters of the generated models using the linear or nonlinear regression algorithm in an effort to develop best model and associated parameter that represent (fit) the data. The complete procedure ends with a statistical analysis which is used to evaluate the model and its performance.

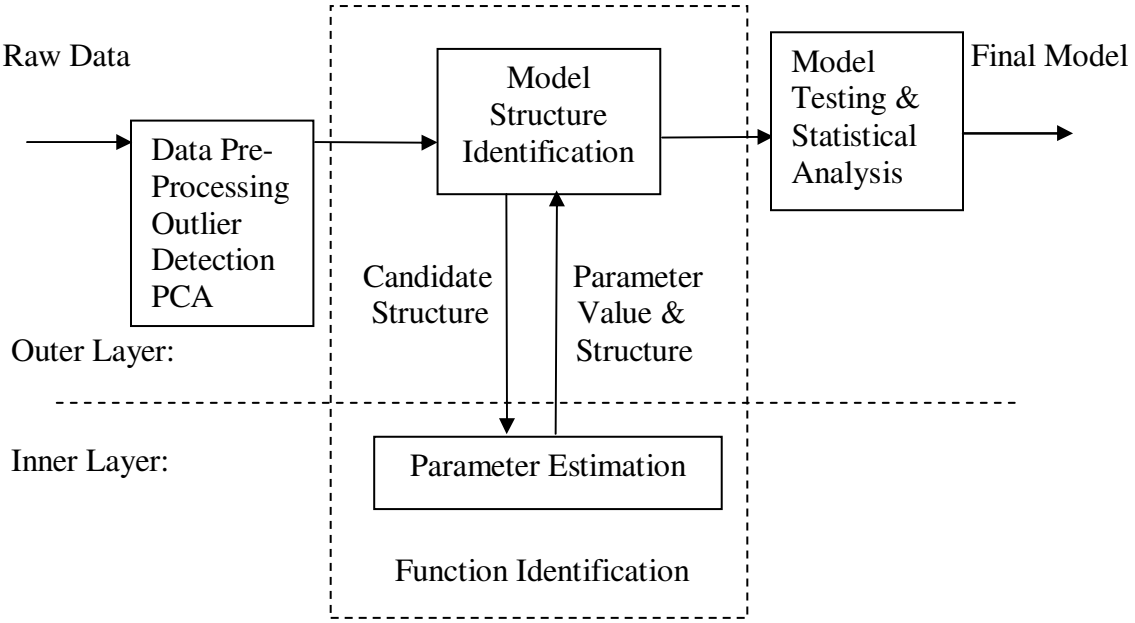


Figure 3.1 A Hybrid System Structure for Structural and Parametric Optimization

3.2 Data and data preparation

Generally, the data to be mined is voluminous, incomplete or imprecise, noisy, has missing values, redundant or insignificant. To get a better mining result, the data will be preprocessed to eliminate those data points. In other words, pre-processing is a sequence of operations converting raw data into data representation suitable for processing tasks. Data preparation is one of the most important steps in the model development process. From the simplest analysis to the most complex model, the quality of the data used is vital to the success of the modeling. Once the data is cleaned, then, the data set is worthy of modeling.

The widely applied data pre-processing approaches include data integration, data cleaning and feature selection [Freitas 2002]. Data integration is necessary if the data to be mined comes from several different sources. This step involves, for instance, removing inconsistencies in variable names or variable value names between data sets of different sources. For data cleaning, it is important to make sure that the data to be mined is as accurate as possible. This step may involve detecting and correcting errors in the data, filling in missing values, etc. Feature selection (select the variable) consists of selecting a subset of features (variables) relevant for mining the data among all original features.

Data on vapor-liquid equilibrium can be obtained from various sources. In this research, the equilibrium data for propylene-propane and acetone-water systems are taken from DECHEMA. Since the linear regression method is sensitive to outliers, outliers are

particularly need to be detected, and deleted if there is any. Outliers can be detected from studentized residuals that are defined for the *i*th observation with:

$$r_i = \frac{y_i - \hat{y}_i}{\sqrt{MSE(1-h_{ii})}} \tag{3.1}$$

where h_{ii} is the *i*th diagonal element of the "hat" matrix H that is defined as: $H_{n\times n} = X(X'X)^{-1}X'$. MSE is the mean square error. When working with a sufficient number of observations, e.g. (n-p-1)>20, where n is the number of observations, and p is the number of parameters, a $|r_i|$>2.0 indicates that the *i*th observation might be an outlier. Similarly, a $|r_i|$>2.5 is a strong indicator of a likely outlier.

3.3 The regression strategy

The developed GP package includes both linear regression and nonlinear regressions options for parameter estimation. The regression procedure in this package is shown in Figure 3.2. Regression strategy can be selected depending on the definition of the problem. All individuals in the population will be checked for their linearity automatically. Depending on the characteristics of the problem, user needs to assign a value to the indicator of model's type before running the program. The assigned values can be -1 for linear model only, 0 for both linear and nonlinear models, and 1 for nonlinear model only. If the final model is expected to be linear, and the indicator's value is set to -1, then, the program will delete the nonlinear models and apply the linear

regression. If the final model is expected to be nonlinear, the indicator's value is set to 1, then, the program will delete the linear models, and get into the nonlinear regression procedure. If the final model could be either linear or nonlinear, user predefines the indicator's value as 0, then, after checking the linearity of individual model, the program will apply the corresponding regression strategy considering both types of models.

To identify the linearity of model, the first order derivative of individual model with respect to each parameter is taken. If all derivatives of model with respect to parameters are constant, the model is linear on parameters. For example:

$$y = a + b \cdot x \tag{3.2}$$

$$y = a + b^2 \cdot x \tag{3.3}$$

Eq. (3.2) is linear, while Eq. (3.3) is nonlinear.

For Eq. (3.2), since $\frac{\partial y}{\partial a} = 1$, $\frac{\partial y}{\partial b} = x$, and there are no parameters a and b that appears in the derivative terms, Eq. (3.2) is linear with respect to parameters a and b.

Similarly for Eq. (3.3), $\frac{\partial y}{\partial a} = 1$, and $\frac{\partial y}{\partial b} = 2bx$, and although the derivative of parameter a is constant, the partial derivative with respect to b is a function of b. Therefore, Eq. (3.3) is not linear with respect to parameter b.

In identification of linearity, MATLAB symbolic math toolbox is used.

Linear least squares regression is a very popular tool for empirical process modeling because of its effectiveness and completeness. The estimates of the unknown parameters obtained from linear least squares regression are unique and are regarded as the global optimal values. Furthermore, linear least squares makes very efficient use of

data and is easy to implement. Good results can be obtained with relatively small data sets.

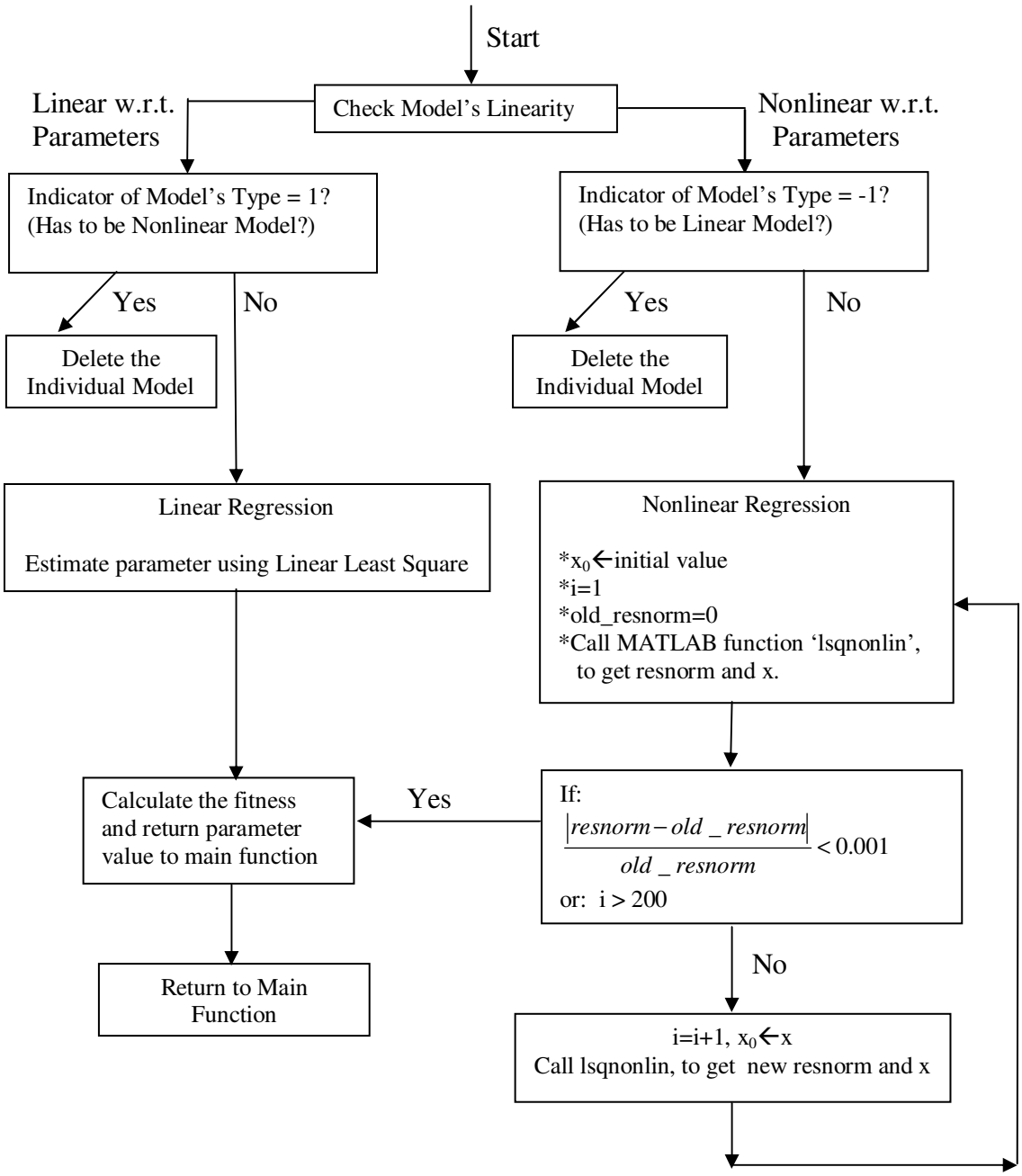


Figure 3.2 The Structure for Linear-Nonlinear Regression Strategy

note: ‘i’ is the iteration number; ‘x_0’is the initial value for parameters; ‘x’ is the parameter values obtained from Marquardt method; ‘resnorm’ is sum of the least square at (i+1)th iteration; ‘old_resnorm’ is sum of least square at ith iteration; ‘lsqnonlin’ is MATLAB nonlinear regression function “Marquardt”.

The evolution of the model is an automatic process. The individual models generated in the process can be very complicated and its structure is neither predicted nor easily simplified. This increases the difficulty in convergence during parameter regression stage, if the nonlinear regression strategy is applied. Nonlinear least squares regression may involve iterative computation to estimate the parameters. With functions that are linear in the parameters, the least squares estimates of the parameters can always be obtained analytically, while that is generally not the case with nonlinear models. The use of iterative numerical procedures for nonlinear regression requires the user to provide initial values for the unknown parameters before the optimization process starts. The initial values should generally be reasonably close to the real parameter values or the optimization procedure may not converge. Different initial values will result in convergence to a local minimum rather than the global minimum unless the problem is convex.

To ensure convergence of parameter regression robustly and more precisely, an automated re-start operation and two stop criteria are implemented. These enforce the algorithm to repeat the regression process before it stops.

The automated re-start operation initializes the parameter value with the regressed parameter values in last run.

Two termination criteria are: the relative change of sum of the least square is less than a setup value, i.e.,

$$\frac{|resnorm - old_resnorm|}{old_resnorm} < 0.001 \tag{3.4}$$

or the number of iterations i exceeds a pre-defined number, as shown in Figure 3.2.

3.4 Implementation with MATLAB based genetic search toolbox

The Genetic Search Toolbox provides an integrated environment for performing a genetic search, including a collection of genetic operations used to implement genetic search methods.

A schematic representation of the genetic search process is shown in Figure 3.3. Important functional elements of a genetic search are: a population with an associated fitness evaluation methodology, one or more selection and creation strategies, and a decimation strategy. The core component of every genetic search is the population. In a genetic search, each member of a population needs to be evaluated and assigned a fitness. Members of a population can be selected for genetic operations through a variety of criteria. In order to manage the size of the population in a genetic search, members of a population also can be selected for deletion through different decimation criteria.

Genetic programming is controlled by several parameters for its components. In the following sections, several tests for different population sizes, mutation and crossover probabilities, fitness functions, tournament sizes, and deletion strategies will be analyzed.

The sensitivity is analyzed when the values of the controlling parameters are changed within a range, one at a time with all other parameters being fixed. Thus, although the resulting parameters, population size etc. are not optimal for a given specific problem, the analysis provides a good feasible set and is robust for most systems.

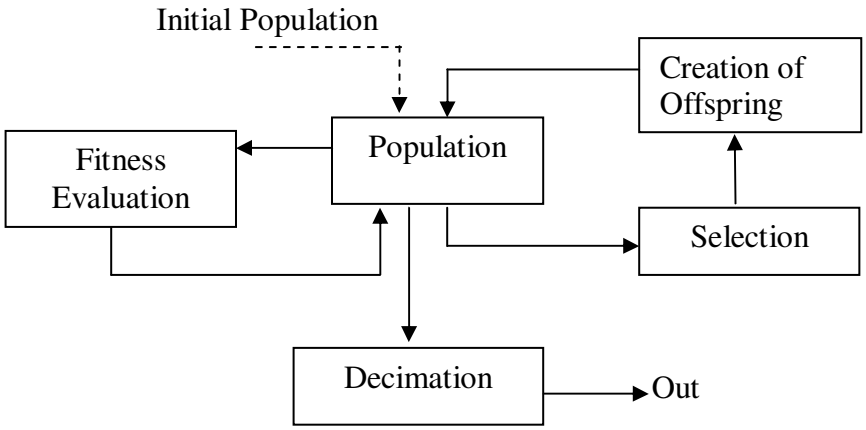


Figure 3.3 A Schematic Diagram of the Genetic Search Methodology

3.4.1 The population architecture

The genetic search process can be implemented in two distinct ways. In the first, the genetic search uses a “steady state” population, i.e., one or two individuals are selected and manipulated at a time, and one or two new offspring are generated, as shown in Figure 3.4. The term "generation" will refer to a single iteration of creating one or two new off-spring. The second, as shown in Figure 3.5, is the generational approach, individuals are selected and entire population is manipulated, many new offspring are

created at each generation; one generation may represent almost complete population turnover (some members may be retained unmodified through the "reproduction").

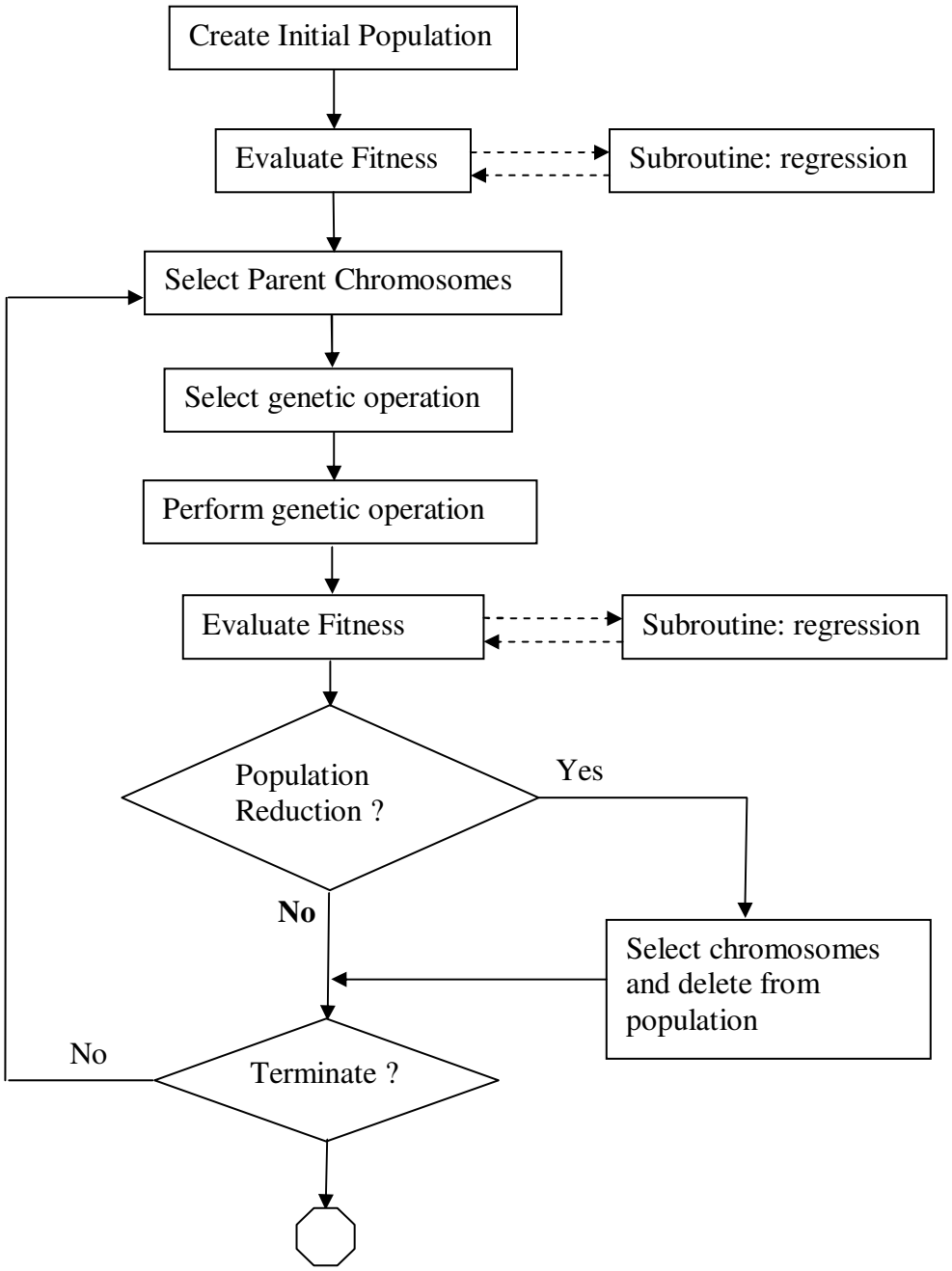


Figure 3.4 A Flowchart of Genetic Search, Steady State Population

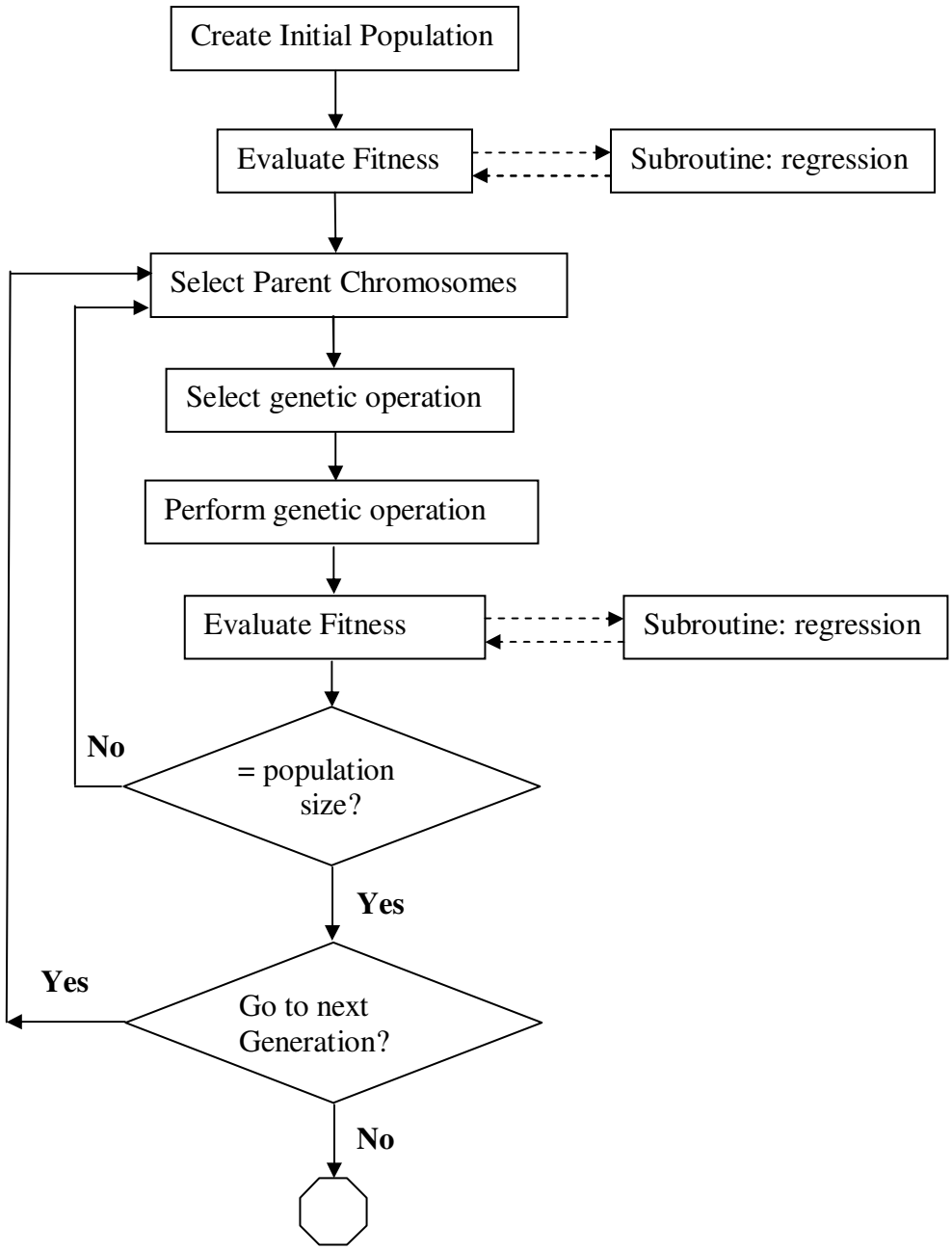


Figure 3.5 A Flowchart of Genetic Search, Generational Population

In this research, the “steady state” population is used. Specifically, the following occurs, as shown in Figure 3.4: An individual is first selected according to some selection strategy, and then the genetic operator is selected. According to the selected genetic operator, a second individual may be selected and perform the genetic operation to create one or two off-spring. Finally, the decimation strategy allows selection of the individual which is going to be deleted from the population, and then the new child replaces the individual deleted.

As a steady state optimizer, when GP operates on just one individual at a time, the number of cycles within a given run can be high, perhaps 25,000 or more. In order to make results more comparable to a generational optimizer, the number of cycles is divided by the size of the population to give the approximate number of generations. The theoretical understanding of the relationship between steady state and generation optimizers is not strong. In order to generate reliable statistics, we ran each test multiple times; typically ten times. From these runs, we then calculated the average performance for each selection scheme.

3.4.2 The population size

Genetic programming is an optimization technique that uses a population of candidate individuals to search the solution space. Since a population consists of multiple individuals, several locations in the solution space are examined in parallel. The use of a large population has several advantages. First, this allows the evolutionary algorithm to examine a large number of positions in the solution space simultaneously. Second, a large

population is more resistant to the loss of diversity in the population. Diversity can be lost in a population when, due to evolutionary pressure, a large number of individuals become similar to the best individuals of the population. When this happens, the search will be restricted to the small area of the solution space containing these similar individuals. Consequently, finding new solutions becomes more difficult. When using a large population, the diversity of the population will persist longer. The disadvantage of using a large population is that more individuals have to be evaluated every generation. Often, the fitness evaluation is the most time-consuming step in genetic programming, and reducing the number of fitness evaluations can significantly speed up the search.

Since the population size is one of the major control parameters, some analytic methods are available for suggesting optimal population sizes for runs of the genetic algorithm on particular problems. However, the practical reality is that researchers generally do not use any such analytic method to choose the population size. Instead, researchers determine the population size such that genetic programming can execute a reasonably large number of generations within the amount of computer time people are willing to devote to the problem [Koza et al. 2003]. Therefore, in this research, a group of tests on the speed of GP convergence are run for different population sizes of 250 (P250), 500 (P500) and 1000 (P1000). As shown in Figure 3.6, the GP runs with the population of 1000 and 500 display a similar behavior, both of which reach the approximately optimal solution at about eighteenth generation, while the population of 250 needs about twenty-five generations to find an approximate solution. Compared with P250, P1000 and P500 have better fitness than P250 does. A larger population size does help to speed

up the searching, but also has impact on improving the accuracy level. The population size was set to 500 based on the results.

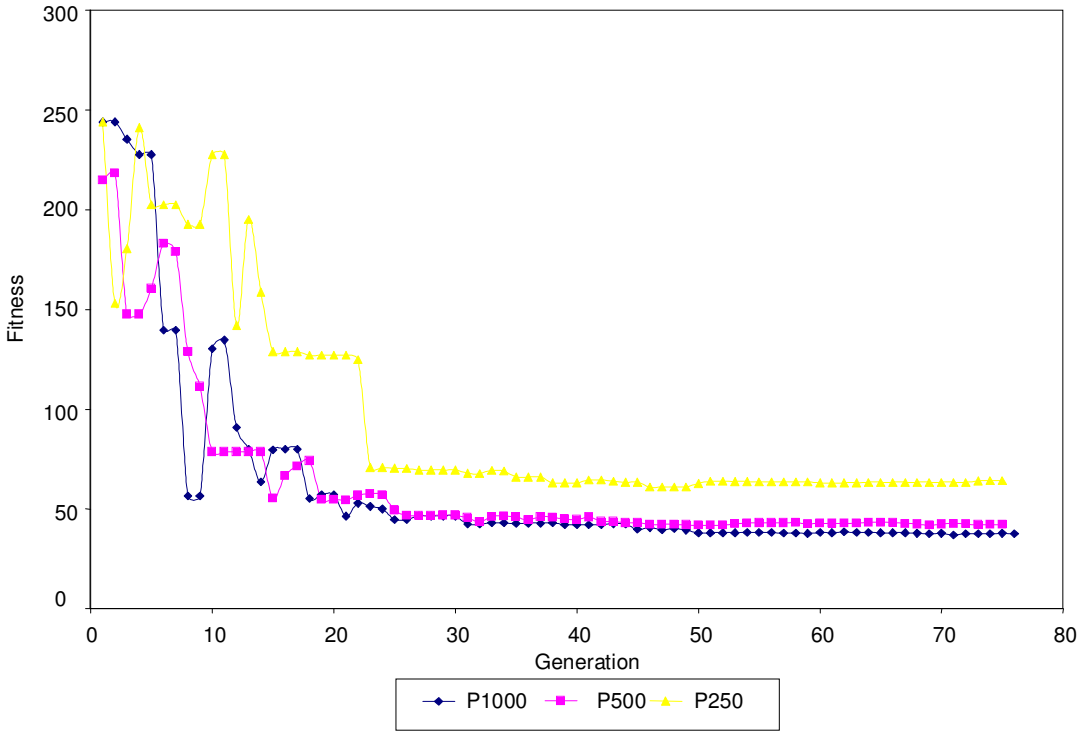


Figure 3.6 Fitness vs. Generation Using Different Population Sizes

3.4.3 The principal component analysis and the selection of terminal & function sets

The complexity of the mathematical model evolved by genetic programming depends on the choice of the function set and terminal set. The chance of discovering a specific formula in a finite number of generations is a decreasing function of its model complexity [Chen 1997]. A larger function set and terminal set will help reduce the complexity of a mathematical model. From this perspective, it seems that a larger

terminal set and function set can enhance search efficiency. In fact, there are empirical evidences that suggest that this is indeed the case [Johnson 2000]. The influence of search space and population size has to be taken into the consideration.

The search space includes all potential individuals and its size grows exponentially with the size of terminal and function sets. The probability of finding the solution in a finite number of generations depends on population size ratio s, i.e. $s = \frac{G}{S}$, where, G is population size, and S is the size of search space. If population size doesn't grow exponentially with the size of function and terminal set, then the population size ratio will be close to zero, i.e., the probability of finding the specific formula in a finite number of generations is nearly impossible.

Since in practice the population size cannot grow in proportion to the size of function and terminal set, reducing the complexity of a mathematical function by enlarging terminal and function set may help gain little efficiency. Therefore, constrained by the population size ratio, the size of function and terminal set has to be optimized.

There are several strategies to optimize the size of function and terminal set. Principal Component Analysis (PCA) is one way to help optimize the size of terminal set.

PCA is a multivariate procedure where the data is transformed such that the maximum variabilities are projected onto the axes. Essentially, a set of correlated variables are transformed into a set of uncorrelated variables which are ordered by reduced variability. The uncorrelated variables are linear combinations of the original variables. The first principal component is the combination of variables that explains the greatest amount of variation. The second principal component defines the next largest

amount of variation and is independent to the first principal component and so on, with the last of these variables can be removed with minimum loss of real data. The main use of PCA is to reduce the dimensionality of a data set, i.e. the size of the terminal set, while retaining as much information as is possible. PCA computes a compact and optimal description of the data set.

The method, proposed by Jolliffe [1986], uses the principal components as the basis for the feature selection. A high absolute value of the i'th coefficient of one of the principal components (PC) implies that the i'th original variable is very dominant in that PC. By choosing the variables corresponding to the highest coefficients of each of the first several selected PC's, the same projection as that computed by PCA is approximated. This method is a very intuitive and computationally feasible method.

Due to the low dimensionality of the vapor-liquid equilibrium data set, where the degrees of freedom is only two, application of PCA is not necessary in this case. However, the developed GP package aims to be more general. Therefore, PCA is included in the package as an option to aid selection of terminal set. A more practical way than PCA, which is also popular among GP users, is to incorporate some physically meaningful variables and their parameters into the function and the terminal set. In this research, according to the pre-knowledge on thermodynamics, we believe that some composition of the functions and terminals supplied here can yield a solution to the problem. The algorithm performs symbolic regression on the experimental data to extract functional representation of the data. The genetic programming module starts with a set of primitive functions, including +, -, *, /, exp, square root (sqrt), log, power (^) and so

on. Temperature, pressure, the vapor and liquid composition are selected to be the elements of terminal set.

3.4.4 Genetic operator

The genetic programming paradigm is controlled by two major numerical parameters, i.e., the population size and the maximum number of generations. These two parameters depend on the difficulty of the problem involved. The population size has been addressed in section 3.4.2. Throughout this research, unless the specific declaration is made, the maximum number of generations is set to seventy-five. As shown in Figure 3.6, seventy-five generations is enough for GP to converge to an approximate solution in this research. Other minor numerical parameters include the probability of crossover, mutation, which will be defined in this section.

Since the population is steady state, the individual will be preserved in the population until it is selected for deletion by the decimation strategy. Therefore, genetic operator "reproduction" is wasteful, only crossover and mutation are applied.

Good values for the mutation and crossover probabilities depend on the problem and must be manually tuned based on experience as there are few theoretical guidelines on how to do this. For some problems performance can be quite sensitive to these values while in others their values do not make much difference.

Different mutation and crossover probabilities are tested. The ratio between mutation and crossover is set to 0.5:0.5, 0.4:0.6 and 0.2:0.8 respectively. Figure 3.7 is the

fitness plot over different mutation and crossover probabilities, and Figure 3.8 is the plot of the standard deviation of fitness value. The standard deviation of fitness value is an index of the diversity of individuals in the population. Figure 3.7 shows that, different mutation and crossover probabilities can reach equivalent accuracy levels. Meanwhile, the diversity in population follows a similar trend. These results show that, in this research, GP is robust and not sensitive to the mutation and crossover probabilities.

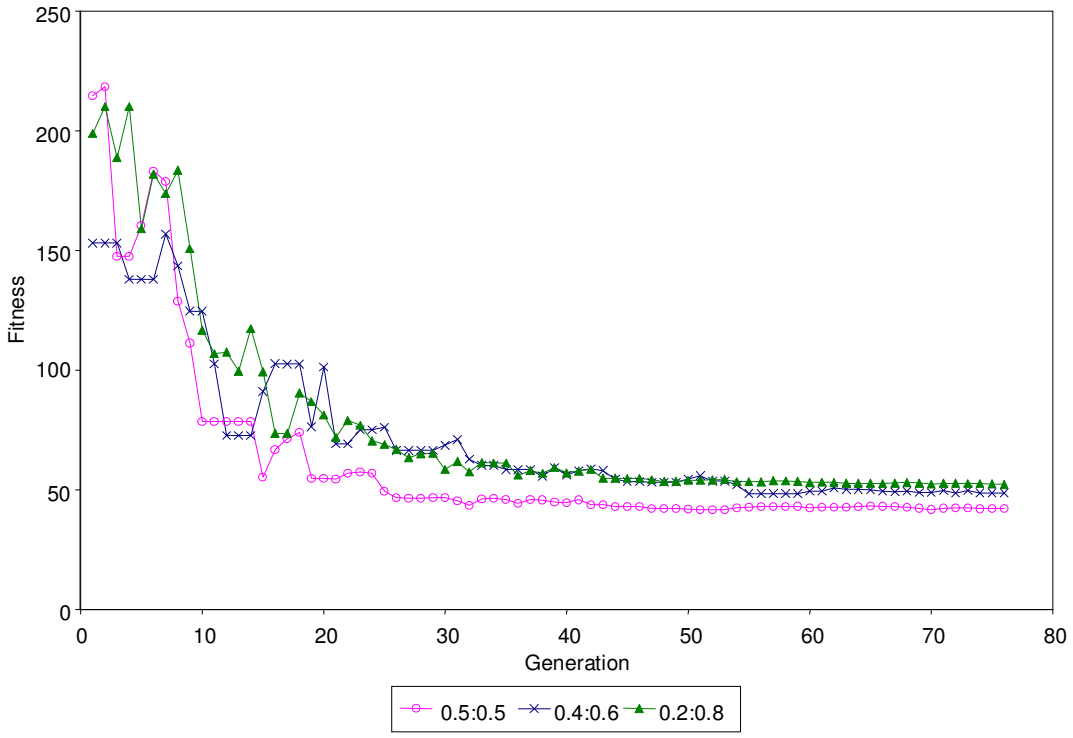


Figure 3.7 Fitness vs. Generation Using Different Mutation and Crossover Probabilities

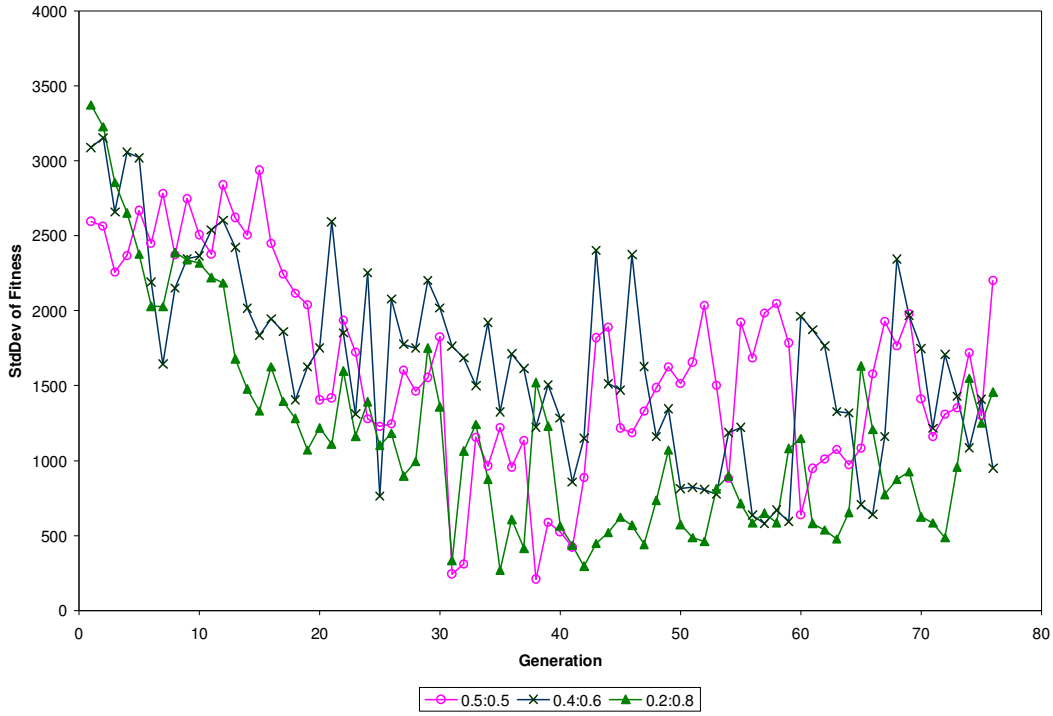


Figure 3.8 The Standard Deviation of Fitness Using Different Mutation and Crossover Probabilities

3.4.5 Fitness evaluation

The fitness function is the driving force of GP, which should reflects the goodness of a potential solution which is proportional to the probability of the selection of the individual. Usually, the fitness function is based on the sum of square error (SSE) between the calculated and the measured output values:

$$SSE = \sum_{j=1}^{N_e} \left[S(i, j) - C(j) \right]^2 \tag{3.5}$$

where $S(i, j)$ is the value returned by equation i for fitness case j (of N_e fitness cases) and $C(j)$ is the correct value for fitness case j. The closer this sum of distances is to zero, the better the program.

A good model is not only accurate but simple, transparent and interpretable. In addition, a complex over-parameterized model decreases the general estimation performance of the model. Since GP is self-evolving process and can result in overly complex models, there is a need for a fitness function that ensures a trade-off between complexity and model accuracy.

To evaluate the efficiency of different fitness functions, including the least square, Akaike's Information Criterion (AIC) and Schwarz Bayesian Information Criteria (BIC), several GP tests are run for the propylene-propane system as an example, and the generated model's accuracy and complexity are compared, as shown in Figure 3.9 and Figure 3.10.

In Figure 3.9, it shows that, compared with the least square, both AIC and BIC keep a relatively comparable accuracy level in this case. Figure 3.10 is the comparison of the model's complexity. It shows that, without applying the parsimony rule, GP tends to increase the complexity of generated model consistently with the generations. Between two parsimony rules, i.e., AIC and BIC, BIC has a better control on the generated model's complexity, and tends to select more parsimonious models, because the BIC places a larger penalty on the number of predictors. Theoretical and simulation studies done by another researcher [Hansen 2001] also concludes that, mostly in the regression

case, when the underlying model is finite-dimensional (specified by finite number of parameters), BIC should be preferred.

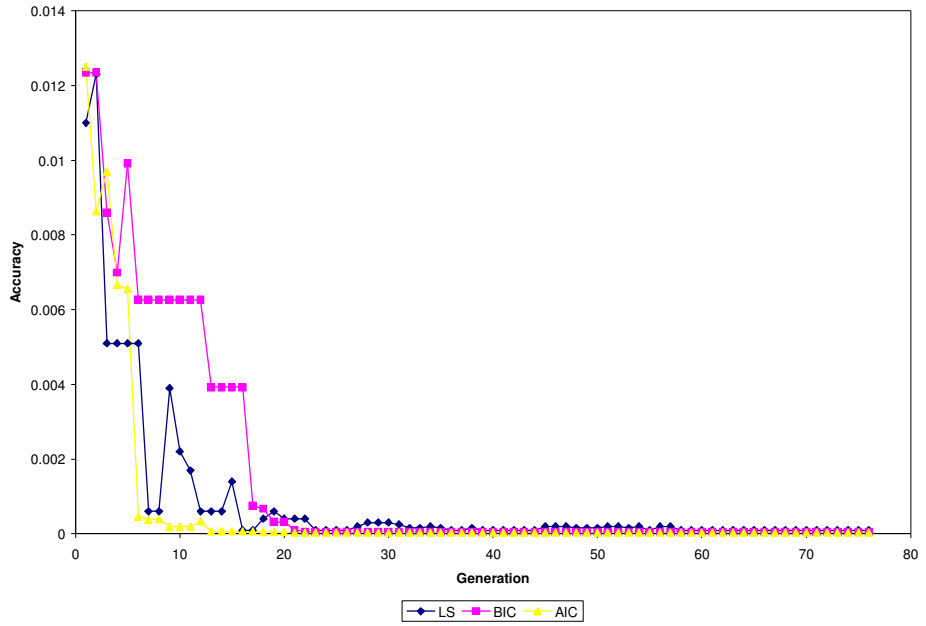


Figure 3.9 Model's Accuracy vs. Generation Using Different Fitness Functions

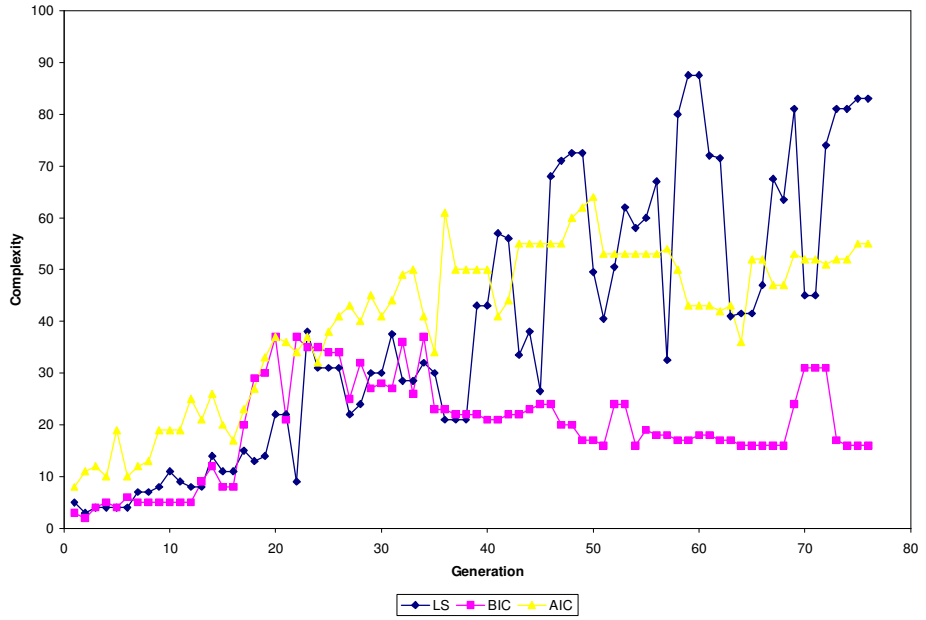


Figure 3.10 Model's Complexity vs. Generation Using Different Fitness Functions

3.4.6 Selection strategy

One practical difficulty in symbolic regression is the problem of crowding. Crowding is a phenomenon where some individuals that are more fit than others in the population are quickly reproduced. Then, copies of these individuals and similar individuals take over a large fraction of the population. The crowding reduces the diversity of the population, thereby slowing further progress by the GP. Several strategies have been explored for reducing crowding. One approach is to change the selection function, using criteria such as tournament selection or rank selection instead of fitness proportionate selection.

For the selection strategy, the commonly used tournament selection is applied. Under tournament selection, a group of individuals is randomly selected from the population, then, the individual with the highest fitness in this subset is returned. The size of the group is called the tournament size. It is clear that, the larger this group is, the more likely a highly fit individual from the population is selected. In the tests, tournament sizes ranging from 3 to 6 are used, in order to examine a range of selection intensities.

When reporting test results, the following notation is applied: TOUR3 means tournament selection with a tournament size of 3. Similar notations are used for TOUR4, TOUR6 and so on.

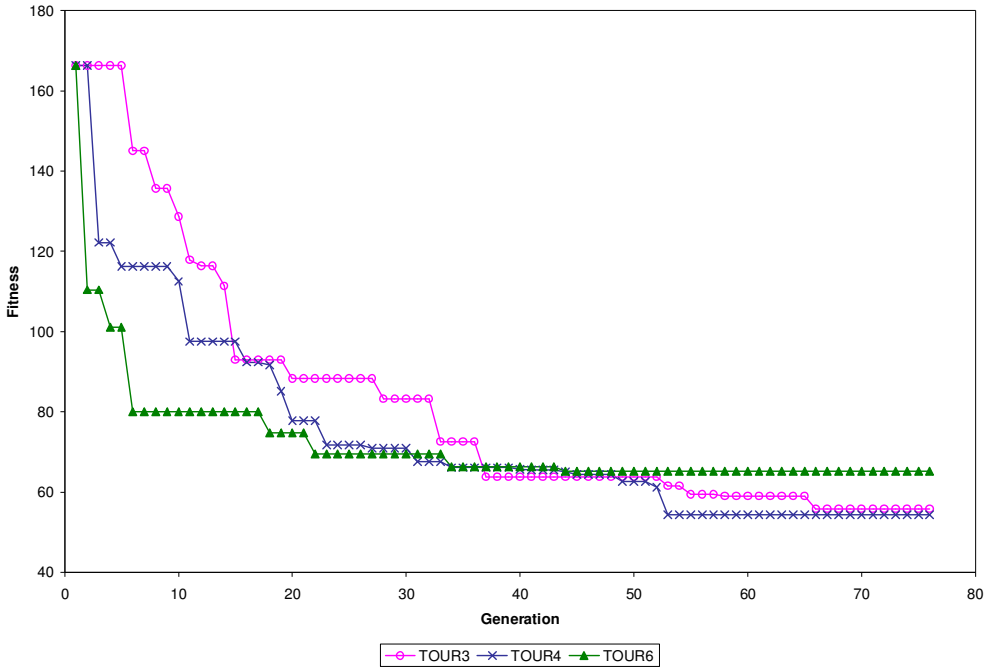


Figure 3.11 Model Fitness vs. Generation Using Different Tournament Sizes

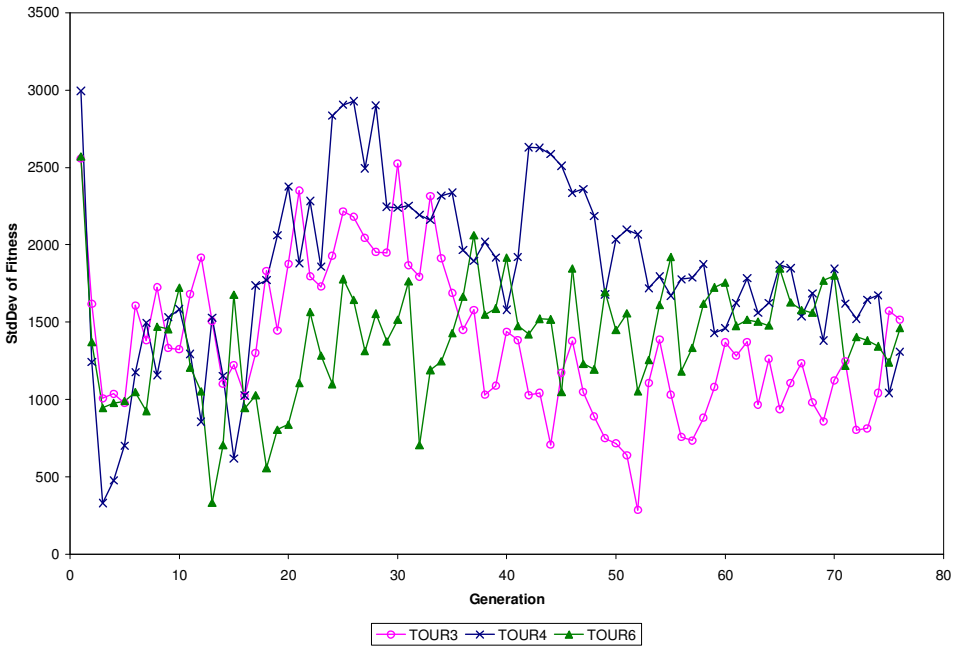


Figure 3.12 The Standard Deviation of Fitness vs. Generation Using Different Tournament Sizes

Figure 3.11 is the comparison of model's fitness using different tournament sizes. Figure 3.12 shows the impact of tournament size on preserving the diversity in population.

In Figure 3.11, it shows that, compared to TOUR4 and TOUR6, TOUR3 converges relatively slowly. It takes sixty-six generations for TOUR3 to find an approximate solution, while TOUR4 takes fifty-three generations, and TOUR6 needs 34 generations to reach the approximate solution. From TOUR3 to TOUR6, with the tournament size increasing, the slope of convergence curve sets steeper. For a given number of generations, the tournament size is larger, i.e., the intensity of selections are increased, the possibility of fittest solution to be chosen will be increased; the number of generations needed to find an approximate solution is less. However, too fast or too slow convergence is not the case expected. If the algorithm converges too fast, it may imply premature termination and the solution may be a local optimal solution, as shown for TOUR6. TOUR6 converges the fastest, but its fitness is the worst one among those of three different tournament sizes.

TOUR3 converged a little bit slowly while TOUR6 converged prematurely and became stuck. TOUR4 appears to be about the correct tournament size for this problem. TOUR3 could be an alternative size for this problem.

3.4.7 Decimation strategy

In order to manage the size of the population in a genetic search, as well as to keep the better individuals in the population, the decimation strategy for deleting excess

individuals in the population need to be decided. As analogous to the selection strategies described and compared in the section 3.4.6, different decimation strategies are also compared in this section, as shown in Figure 3.13 and Figure 3.14. The maximum population size is set to 500, if the population size is bigger than this limit, then, the individuals chosen by the decimation strategy will be deleted from the population.

A common problem experienced with population based optimization methods is the gradual decline in population diversity that tends to occur over time. This can slow a search progress or even halt it completely, if the population converges on a local optimum from which it cannot escape.

With steady state population optimizers, the standard decimation strategy used is simply random deletion. The reason for this is that it is neutral in the sense that it does not skew the distribution of the population in any way. Thus, whether the population tends toward high or low fitness etc. is solely a function of the selection scheme and its parameters, in particular the selection intensity. This issue was discussed in the previous section.

The other option for decimation strategy is that the individuals are deleted from the population based on their fitness. The worse the fitness is, the higher the probability of the particular individual being selected for deletion. Figure 3.13 depicts the comparison between two different decimation strategies in terms of model accuracy. Figure 3.14 shows the standard deviation of fitness. As mentioned earlier, the standard deviation of fitness value in the population is an index of the diversity of individuals in this population. As shown in Figure 3.13, fitness- proportional decimation strategy allows

GP algorithm to converge at the fifteenth generation while random deletion strategy result in convergence at the twenty-seventh generation. On the other hand, fitness-proportional deletion strategy allows GP run converge faster than random deletion. However, the fitness- proportional deletion strategy leads to less diversity in the population. As shown in Figure 3.14, using the same selection strategy (tournament selection with size 4), the diversity of population using random deletion is maintained well over the generations, however, the diversity of population using fitness-proportional deletion is decreased tremendously after the second generation. From this group of tests, we conclude that, the diversity of population with random deletion is best for the whole process.

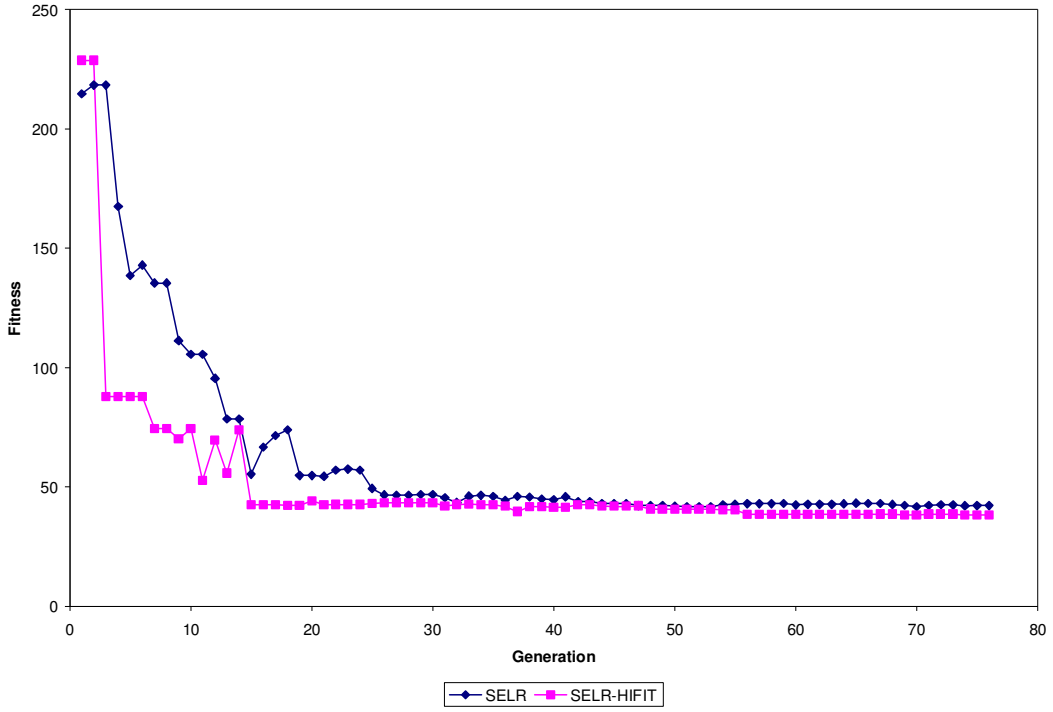


Figure 3.13 Model's Accuracy vs. Generation Using Different Deletion Strategies

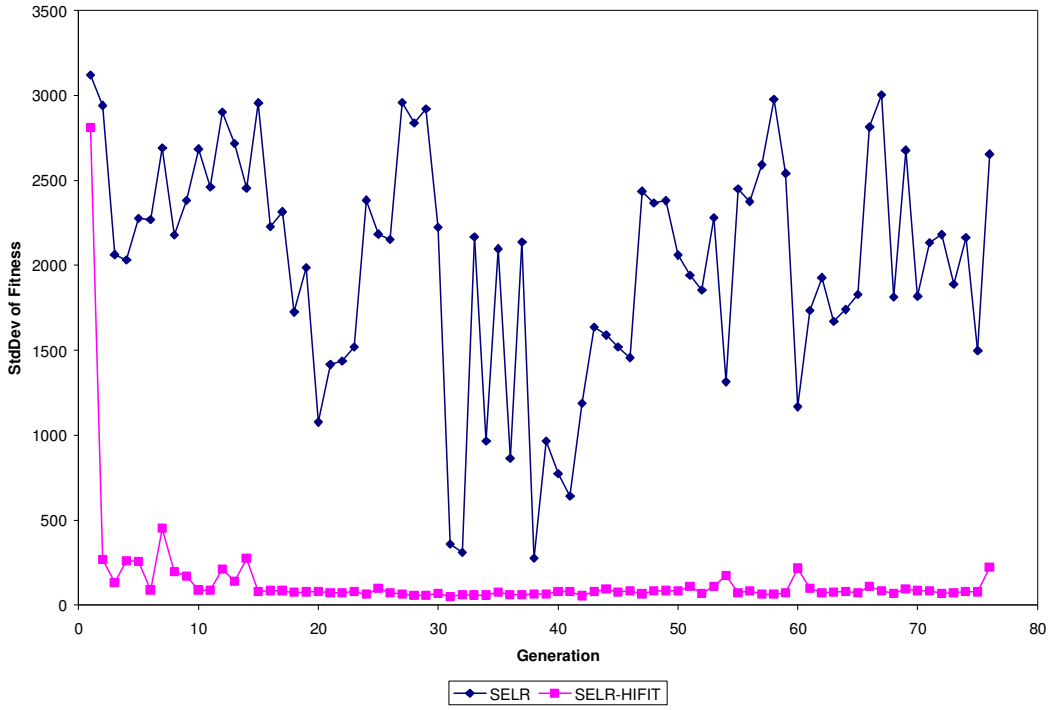


Figure 3.14 The Standard Deviation of Fitness vs. Generation Using Different Deletion Strategies

3.4.8 Result designation and termination criteria

Each computer experiment is terminated when the maximum number of GP generation is reached. To have a better evaluation of the number of maximum GP generations, computational experiments with different population sizes, mutation and crossover probabilities, fitness evaluation functions, selection strategies and decimation strategies are performed. The results were shown in the sections from 3.4.2 to 3.4.7. Model's accuracy vs. generation using different population sizes is shown in Figure 3.6. Figure 3.7 depicts model's accuracy vs. generation using different mutation and

crossover probabilities. Figure 3.9 is the same plot using different fitness functions, Figure 3.11 depicts model's accuracy vs. generation using different selection intensities, and Figure 3.13 is using different deletion strategies. All these plots of model's accuracy using different GP controlling parameters show that GP program will reach its approximate solution before forty generations. The accuracy doesn't improve as the number of generations is increased beyond that point. To ensure that GP program has enough run time and the generated model fits the data at a satisfied level, the maximum number of generation is set to 75 throughout this research. In each generation, the best so-far model is designated.

3.4.9 Summary

Based on the theoretical analysis and experimental tests shown in previous sections, the input values for the GP controlling parameters used throughout the research are summarized as follows:

- Population Size: 500
- Terminal Set: temperature, pressure, liquid phase composition (x_i), parameters (maximum eight parameters)
- Function Set: +, -, ×, ÷, ^, exponential, log
- Mutation and Crossover Probability: 0.5:0.5
- Fitness Function: Schwarz Bayesian Information Criteria (BIC)
- Selection Strategy: Tournament, Size 4.

- Maximum Generations: 75
- Deletion Strategy: Random

3.5 Model evaluation

Genetic Programming (GP) is inherently probabilistic in nature, and thus for symbolic regression problems, each time the algorithm is used, it's expected that one will arrive at different approximate solutions. Therefore, one should perform multiple experiments to develop alternate models, employ a statistical analysis to further prune the list of model candidates, and evaluate model performance through steady state and dynamic simulation of columns that utilize these models.

3.5.1 Statistical analysis

The GP generated models with optimized parameters were evaluated using graphical and numerical statistical methods.

3.5.1.1 Graphical methods

A scatter plot of the predicted response versus the observed response provides simple and visual depiction of model performance. Any model that can explain most of the variation in the observed responses will produce a plot with points clustered around a

45° line. Better models yield less scatter about this $\hat{y} = y$ line. Moreover, the scatter of points about the $\hat{y} =$ y line should remain roughly constant with magnitude. That is, a poor model that is less accurate at larger values of $\hat{y}$ will produce increasing scatter with larger values of y. A scatter plot of the residuals is also useful in evaluating a regression model. Here, the model residuals or errors ($r = y - \hat{y}$) are plotted against the model predictions ($\hat{y}$). Residual plots are used to visually verify some of the basic assumptions underlying regression analysis and observe model data mismatch in term of bias and accuracy. The residuals (errors) between the model predictions and observed responses should have a mean of zero and a constant variance. Hence, the scatter in the residuals should be fairly uniform and centered about $\varepsilon = 0$. A good regression model will produce a scatter in the residuals that is roughly constant with $\hat{y}$. Unsatisfactory models yield a scatter in these residuals that change with $\hat{y}$.

All of the information on lack of fit is contained in the residuals. Assuming the model you fit to the data is correct, the residuals approximate the random errors. Therefore, if the residuals appear to behave randomly, it suggests that the model fits the data well. However, if the residuals display a systematic pattern, it is a clear sign that the model has a bias in representing data.

If the residuals appear randomly scattered around zero, the model describes the data accurately and without systematic bias.

3.5.1.2 Goodness of fit statistics

To express the quality of fit between a regression model and the sample data, the coefficient of multiple determination (R^2) is typically used. However, adding any regressor variable to a multiple regression model, even an irrelevant regressor, yields a smaller SSE and greater R^2. For this reason, R^2 by itself is not a good measure of the quality of fit. To overcome this deficiency in R^2, and adjusted value is used. The adjusted coefficient of multiple determinations ($\overline{R}^2$) is defined as:

$$\overline{R}^2 = 1 - \left(\frac{n-1}{n-p}\right)(1 - R^2) \tag{3.6}$$

Since a number of model coefficients (p) is used in computing $\overline{R}^2$, the value will not necessarily increase with the addition of any regressor. Hence, $\overline{R}^2$ is a more reliable indicator of model quality.

3.5.1.3 Cross validation and prediction

Finally, it's often important to evaluate the ability of a fitted regression model to predict future events. The best way to do this is to gather additional, new data and compare these observed responses with predictions from the model. However, when this is not possible, the data used to fit the model can be split and a cross validation is performed. A good regression model can be fit to part of the original data set and still accurately predict the other observations. To perform a double cross-validation:

- Partition the data into two subsets (say, A and B) with an equal number of observations in each. Assigning individual observations to subset A or B must be done randomly.
- Using the same model form, fit the model using the data from subset A. use this model to predict the observations in subset B.
- Compute the predicted $R^2_{p,A}$ for the model that fits to subset A, as defined in Eq. (3.8) below.
- Similarly, fit the model to data in subset B and use this to predict the observations in subset A.
- Compute the predicted $R^2_{p,B}$ for the model fit to subset B.

A good model will produce high values of R^2_p for both subsets and these values will be approximately equal ($R^2_{p,A} \cong R^2_{p,B}$). The predicted R^2_p for a model fit to subset A ($R^2_{p,A}$) is computed with:

$$R^2_{p,A} = 1 - \frac{\sum_{i=1}^{n_B} (y_{iB} - \hat{y}_{iA})^2}{\sum_{i=1}^{n_B} (y_{iB} - \overline{y_B})^2} \tag{3.7}$$

where n_B and y_B are the number and mean of the observed responses (y_{iB}) in the random subset B. Using the model fit to subset A, predictions of the observations in subset B are made to give $\hat{y}_{iA}$. The predicted R^2_p for a model that fits to data in subset B ($R^2_{p,B}$) is computed the same way, with the use of subsets A and B reversed.

3.5.2 Steady state and dynamic simulation using local models

To have a further evaluation on the model's stability and accuracy, the generated model is integrated with chemical engineering process simulation package CHEMCAD. In CHEMCAD, users are allowed to supply their own K-values in three major ways: users may input tabular K-values that CHEMCAD will then interpolate for use during the calculations. The second option is that, the K-values are assumed to be a function of temperature in the following polynomial forms:

$$f(K) = a + bT + cT^3 + dT^4 \tag{3.8}$$

Users need to provide the coefficient a, b, c and d. The third option allows users to create their own added modules. The User Added Modules (UAM) gives users the ability to create new thermodynamic routines. The evolved K-value model by GP can be linked with CHEMCAD as a UAM, and then evaluated by process simulation. UAM's are programmed in Visual C++ and have access to a large number of internal CHEMCAD routines.

CHAPTER FOUR

RESULTS AND DISCUSSION

The structure and implementation of the GP package for symbolic regression was addressed in Chapter Three. In this chapter, the application of the GP package for development of local K value models for propylene-propane (ideal or near ideal solution) and acetone-water systems (non-ideal solution) is presented. The results and their discussion are given in sections 4.1 and 4.2 respectively. In section 4.1, generated K value models with associated regressed parameter values, the phase equilibrium plots, distillation column simulation profiles (steady state, or dynamic simulation) are presented along with residual plots. In section 4.2, the performance of the models for ideal and non-ideal solutions is discussed. The approach is applied to both propylene-propane and acetone-water systems, using different terminal sets and for different pressure ranges of data. Furthermore, the results will be discussed and the performance of models will be compared to rigorous models as well as data.

4.1 Local composition models and their impact

Macroscopic vapor-liquid equilibrium coefficient “K value” is a function of temperature, pressure and compositions of liquid as well as vapor phase. However,

simple composition independent models are very popular due to reduced computation burden, since iterative procedures with composition dependent models are avoided. These models work well for ideal solutions. In this section, the equilibrium K value models developed using GP for ideal (or near ideal) propylene-propane binary system and strongly non-ideal acetone-water binary system are studied respectively. The data for propylene-propane system ranges from low to higher pressures for different temperatures, and for acetone-water system, the data ranges from low temperatures to higher temperature for different pressures. For each binary system, the final K value model and its parameter are summarized, and the fit of the models are evaluated statistically while the model performance is evaluated with process simulation package CHEMCAD.

4.1.1 Developed models for mixtures that form ideal or near-ideal solutions and their statistical analysis

In Table 4.1, the final structure of K model for propylene-propane solution are summarized.

Table 4.1 Result Summary for Propylene (1) –Propane (2)

Result for K_1:
Evolutionary Model: $$\log K_1 = \frac{v_1 \cdot (1-x_1)^2}{T^2} + \frac{v_2 \cdot P \cdot (1-x_1)^2}{T} + \frac{v_3 \cdot (1-x_1)}{T}$$ Adjusted R^2: 0.997
Result for K_2:
Evolutionary Model: $$\log K_2 = \frac{v_1 \cdot (1-x_2)}{T} + \frac{v_2 \cdot (1-x_2)}{T^2} + \frac{v_3 \cdot (1-x_2)^2}{T^3} + \frac{v_4 \cdot (1-x_2)}{T \cdot P} + (1-x_2)$$ Adjusted R^2: 0.996

The $y = \hat{y}$ plot for generated K_1 and K_2 models are given in Figures 4.1 and 4.3 respectively. The points are distributed along 45° line. No significant deviation from 45° line is observed. The residual plots for K_1 and K_2 model are given in Figures 4.2 and 4.4. The mean value for both residual plots are close to zero.

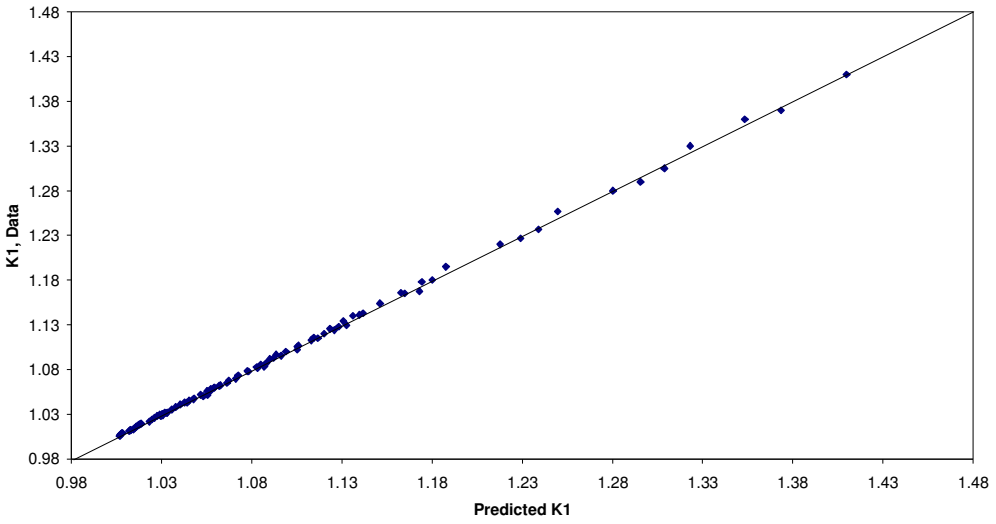


Figure 4.1 K_1 Model vs. Experimental Data for Propylene (1)-Propane (2)

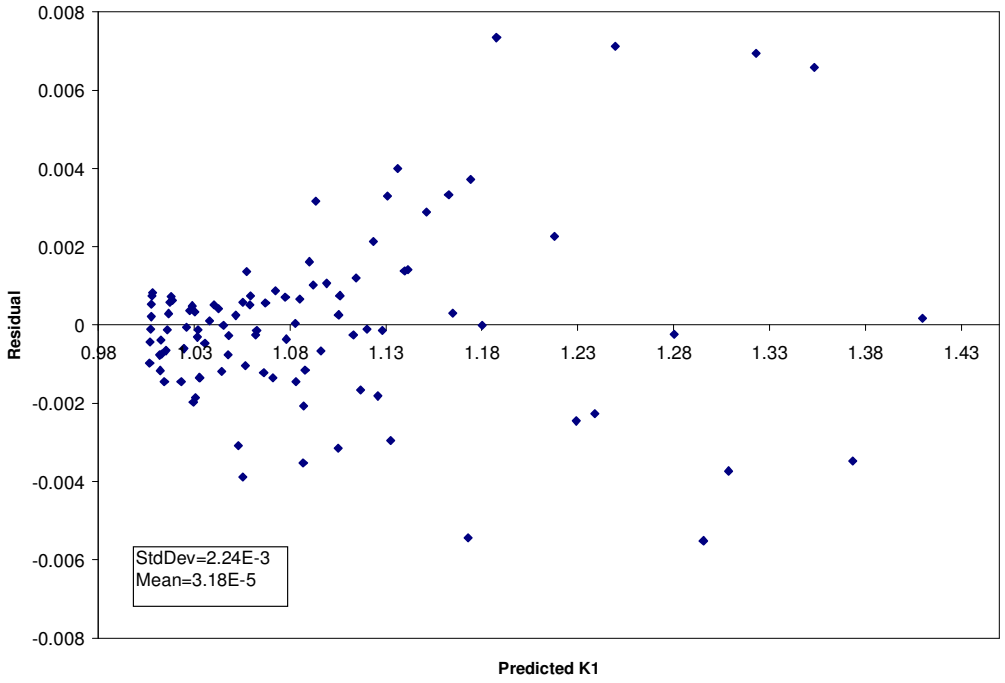


Figure 4.2 Residual Plot of K_1 for Propylene (1)-Propane (2)

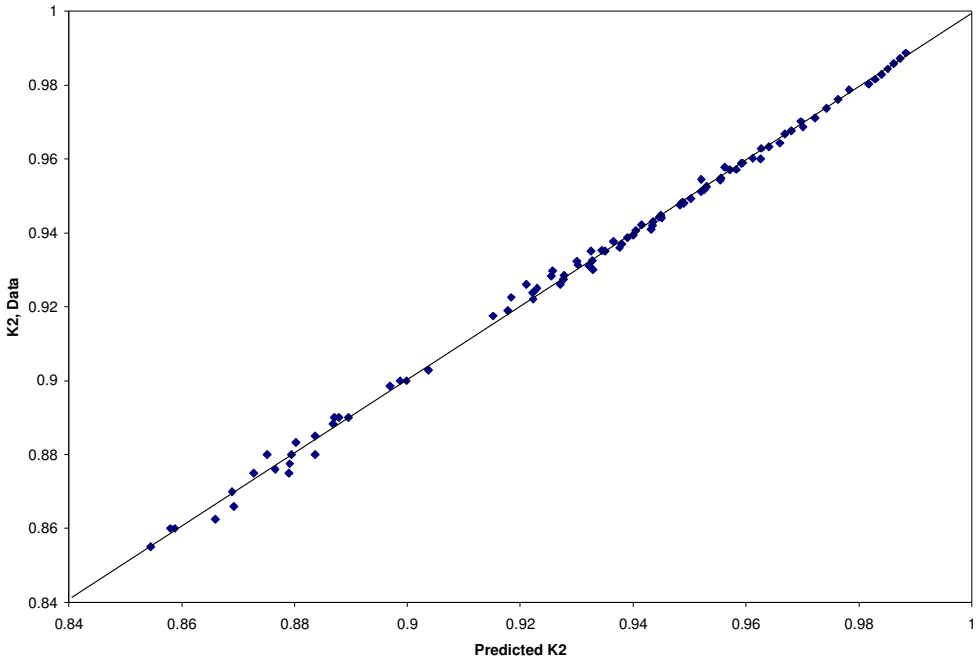


Figure 4.3 K_2 Model vs. Experimental Data for Propylene (1)-Propane (2)

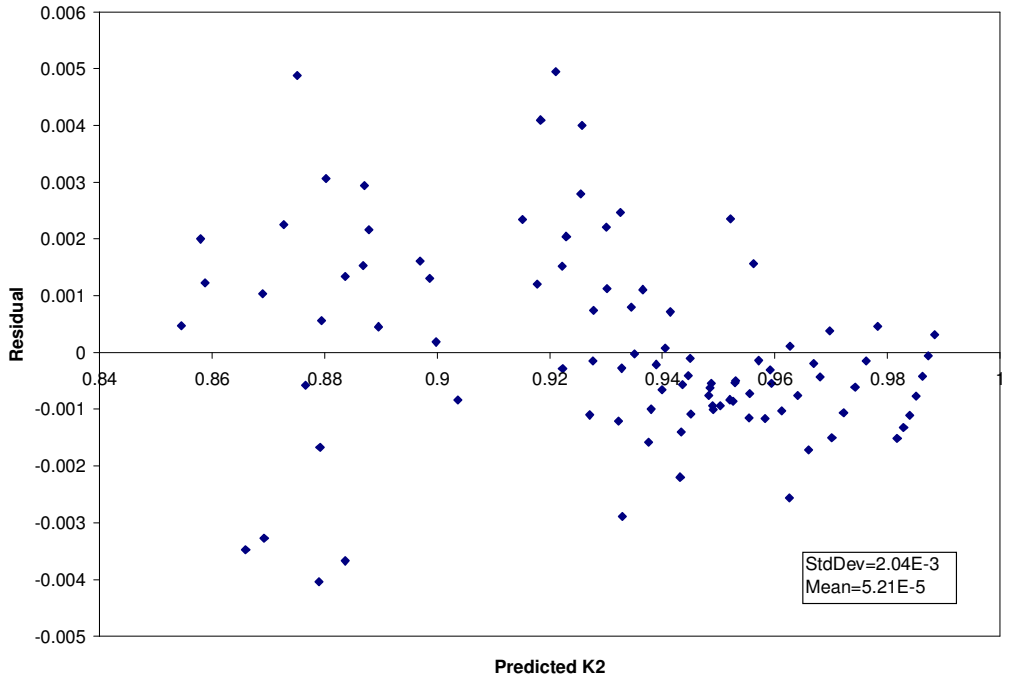


Figure 4.4 Residual Plot of K_2 for Propylene (1)-Propane (2)

|ΔK| and |ΔK|/K (%) for K_1 and K_2 at five different temperatures are list in Table 4.2. Maximum and average values of |ΔK|/K (%) for K_1 and K_2 are small, which indicate the models for K_1 and K_2 fit experimental data well.

Table 4.2 Standard Error Analysis for Generated K_1 and K_2 Model

T, K	|ΔK|/K, % $K_1(K_2)$		|ΔK| $K_1(K_2)$	
	Max.	Avg.	Max.	Avg.
223.75	0.614 (0.255)	0.292 (0.168)	0.00734 (0.00235)	0.00351 (0.00150)
239.35	0.484 (0.417)	0.158 (0.269)	0.00659 (0.00367)	0.00186 (0.00242)
310.93	0.315 (0.534)	0.101 (0.232)	0.00372 (0.00495)	0.00112 (0.00239)
327.59	0.121 (0.130)	0.0659 (0.0679)	0.00138 (0.00121)	0.000716 (0.000648)
344.26	0.286 (0.106)	0.147 (0.0557)	0.00316 (0.00101)	0.00156 (0.000533)

4.1.2 Performance of models for separation of ideal and near ideal mixtures

The generated K_1 and K_2 models are integrated with CHEMCAD as user added modules. P-x-y for five different temperatures 223.75K, 239.35K, 310.93K, 327.59K and 344.26K are plotted and compared with those from Peng-Robinson (PR) EOS, as shown in Figures 4.5 and 4.6. As observed, the generated local models fit the entire data range well. They also perform as well as the PR EOS that is usually the default model for these tasks.

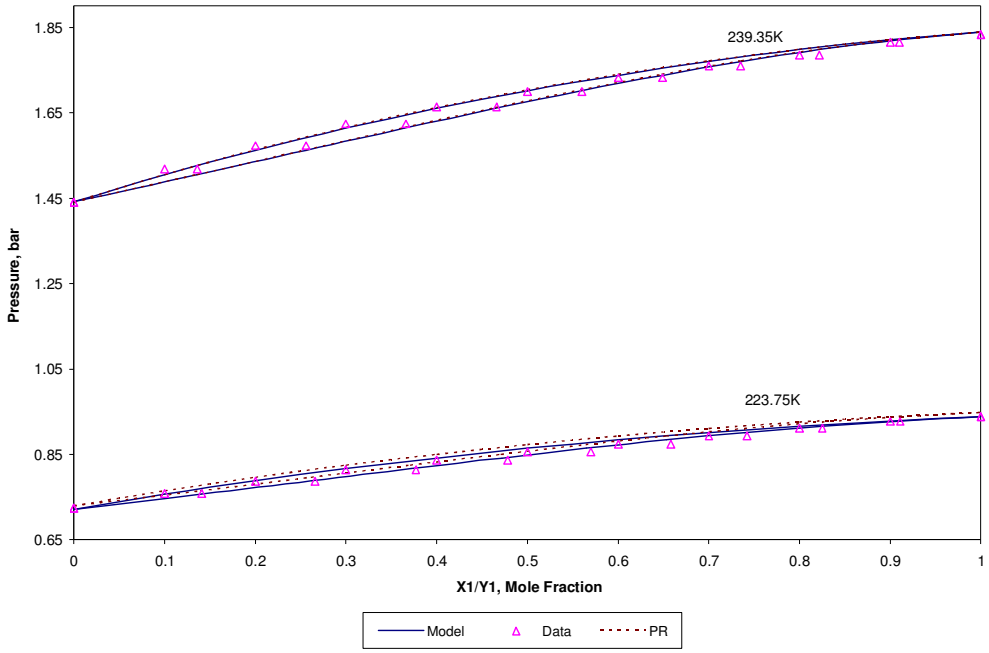


Figure 4.5 P-X_1-Y_1 at Low Pressures for Propylene (1)-Propane (2)

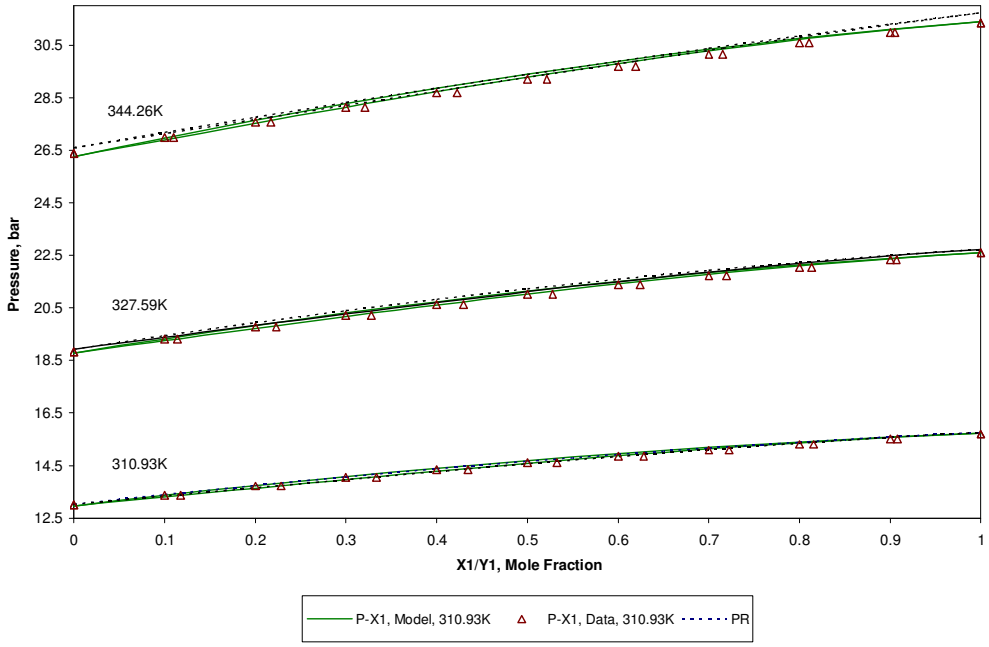


Figure 4.6 P-X_1-Y_1 at Medium to High Pressures for Propylene (1)-Propane (2)

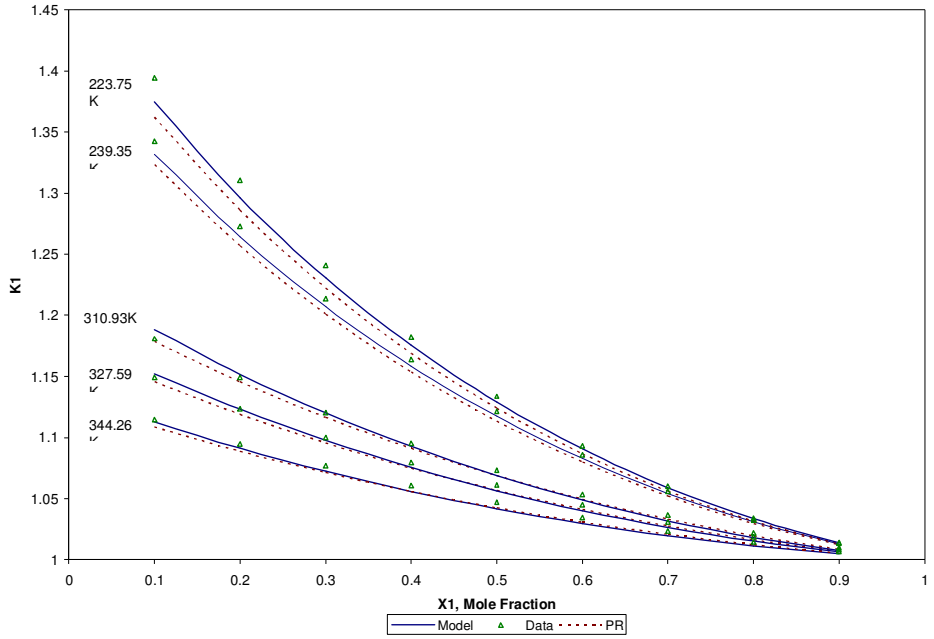


Figure 4.7 K_1 vs. Liquid Composition for Different Temperatures for Propylene (1)-Propane(2) System

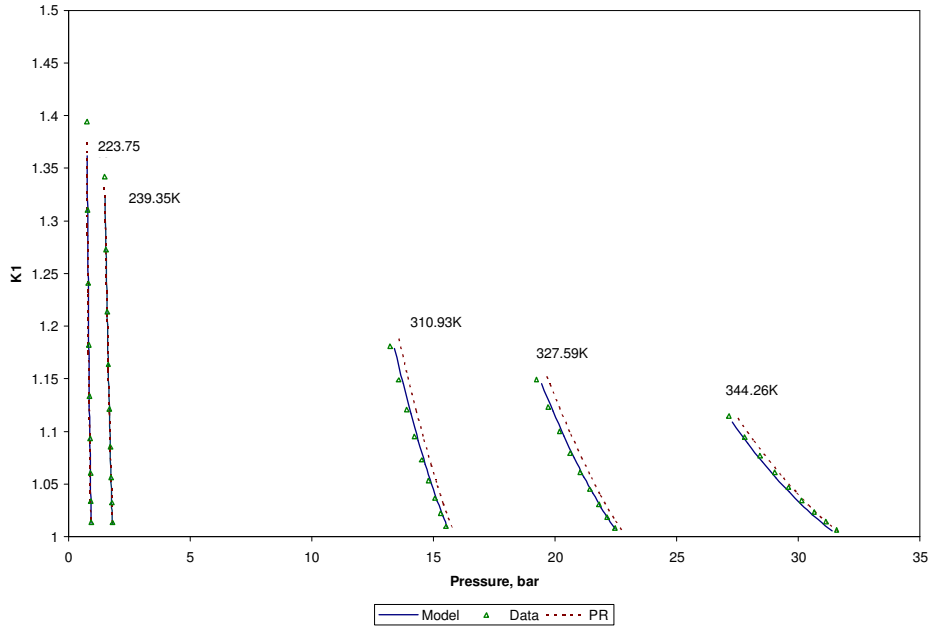


Figure 4.8, K_1 vs. Pressure for Different Temperatures for Propylene (1)-Propane (2)

The results show that the generated K models are a function of pressure, composition and temperature, therefore, the K-composition and K-pressure analysis of models are given in Figures 4.7 and 4.8. K_1-composition plot in Figure 4.7 shows the local model has a better fit over composition than rigorous model PR does at whole data range. In K_1-pressure plot, local model show slight deviation at high pressure range.

Propylene is the most important building block in any petrochemical industry. The cost of producing propylene is much associated with the propylene-propane separation step, conventionally performed by low temperature or high-pressure distillation. This step is energy intensive, therefore costly, because of the low relative volatility; the separation process requires large distillation columns with more than 210 trays and high reflux ratios. This makes the propylene-propane system one of the most energy-expensive separations.

To test local model's reliability further, steady state simulation is run. A 210-plate distillation column is simulated. The feed stream containing 800 lbmol/hr propylene and 200 lbmol/hr propane is fed to tray 125 at pressure 19 bar as saturated liquid. The column is operated under 17.5 bar on the top with 1.8 bar pressure drop, the mole fraction of propylene on the top and at the bottom is 0.95 and 0.05 respectively. The tower temperature profile is presented in Figure 4.9 and relative volatility is presented in Figure 4.10. The vapor and liquid flow rate at different tray is given in Table 4.3.

Table 4.3 Propane-Propylene Distillation Column Profile

		Model	Data	PR
Stage 10	Vapor, lbmol/h	8481.52	8092.83	8534.49
	Liquid, lbmol/h	7650.25	7261.44	7703.27
Stage 100	Vapor, lbmol/h	8634.44	8227.33	8688.34
	Liquid, lbmol/h	7802.67	7395.33	7856.68
Stage 200	Vapor, lbmol/h	8861.79	8438.48	8918.01
	Liquid, lbmol/h	9031.94	8608.7	9088.10
Condenser, MMBtu/h		-46.03	-44.02	-46.43
Reboiler, MMBtu/h		45.86	43.84	46.27

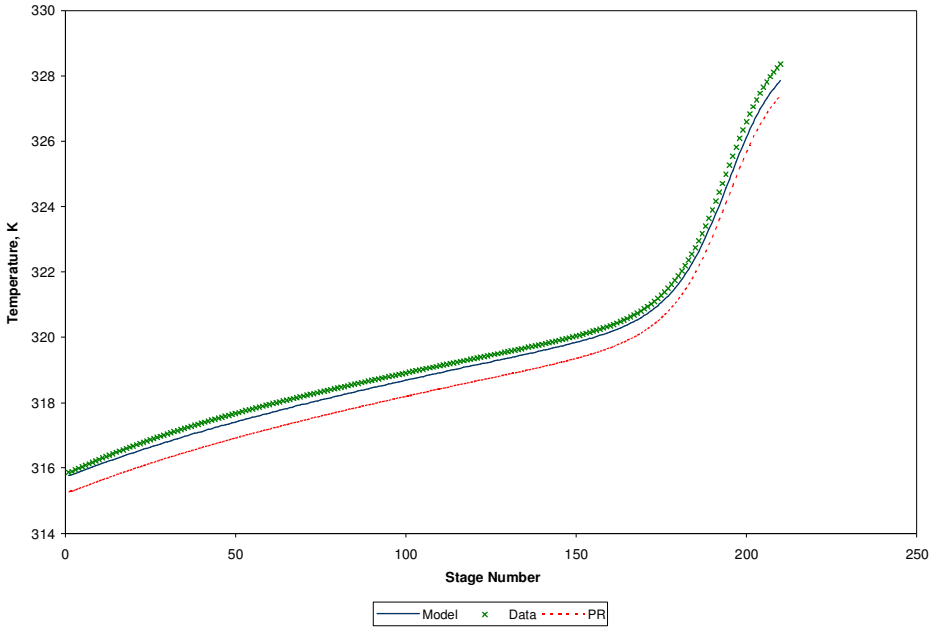


Figure 4.9 Temperature Profile of Propylene-Propane Distillation Column

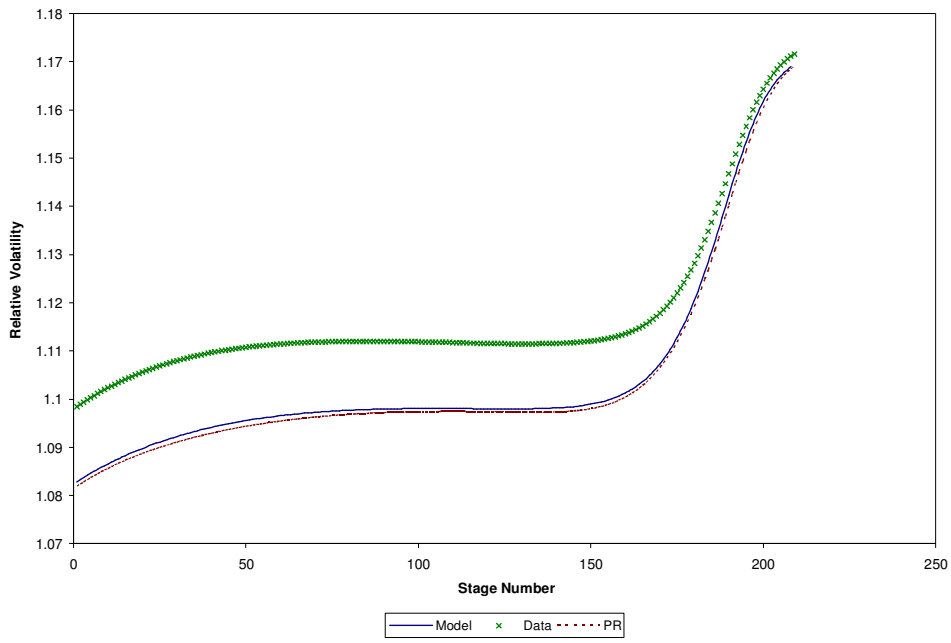


Figure 4.10 Relative Volatility Profile for Propylene-Propane Distillation Column

4.1.3 Developed models for mixtures that form non-ideal solutions and their statistical analysis

Table 4.4 Result Summary for Acetone (1) -Water (2)

Result for K_1:
Evolutionary Model: $\log K_1 = v_1 \cdot (1-x_1) + v_2 \cdot T \cdot (1-x_1) + v_3 \cdot T \cdot (1-x_1) \cdot x_1^{\frac{(1-x_1)}{T}} + v_4 \cdot \log(P)$ Adjusted R^2: 0.994
Result for K_2:
Evolutionary Model: $\log K_2 = \frac{v_1 \cdot (1-x_2)^2}{T} + v_2 \cdot [(1-x_2)^3 + \log(P) + 1] + \frac{v_3}{T^2} + \frac{v_4}{T^3} + \log(T) + 1$ Adjusted R^2: 0.990

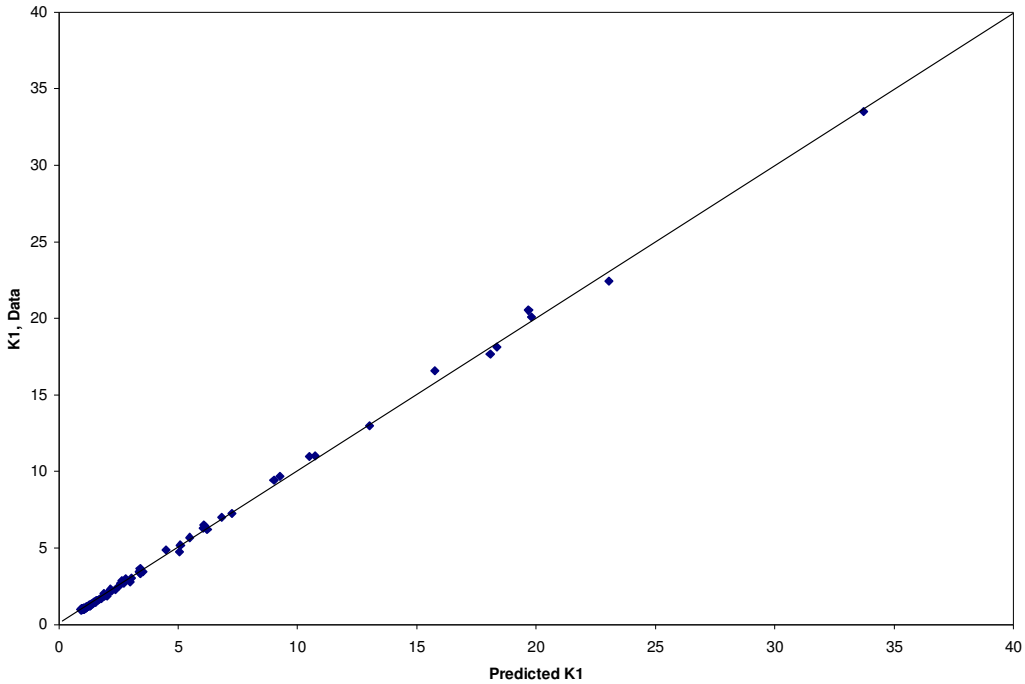


Figure 4.11 K_1 Model vs. Experimental Data for Acetone (1)-Water (2)

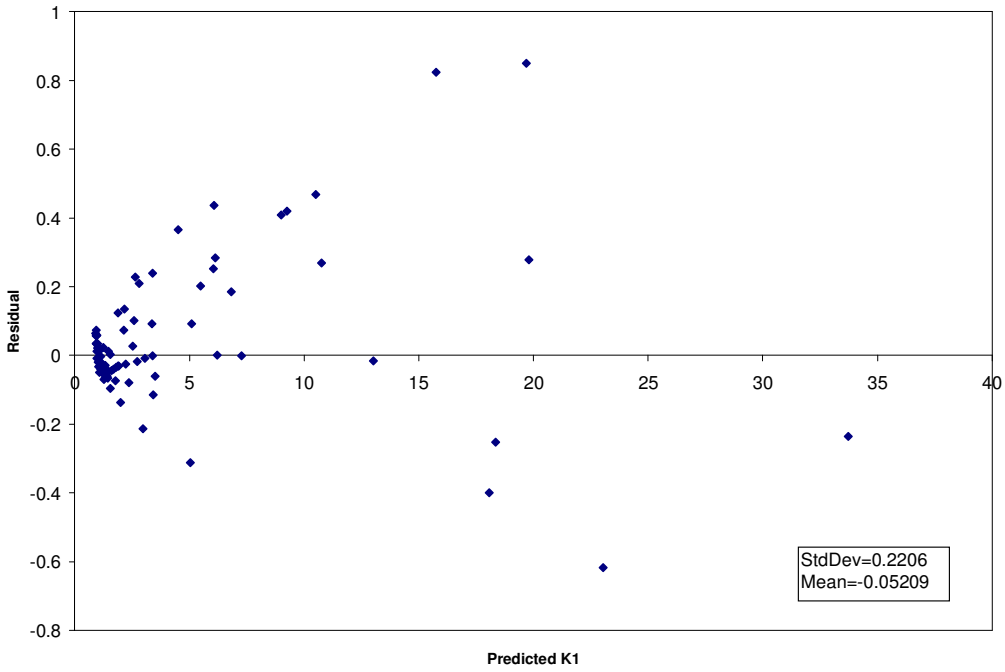


Figure 4.12 Residual Plot of K_1 for Acetone (1)-Water (2)

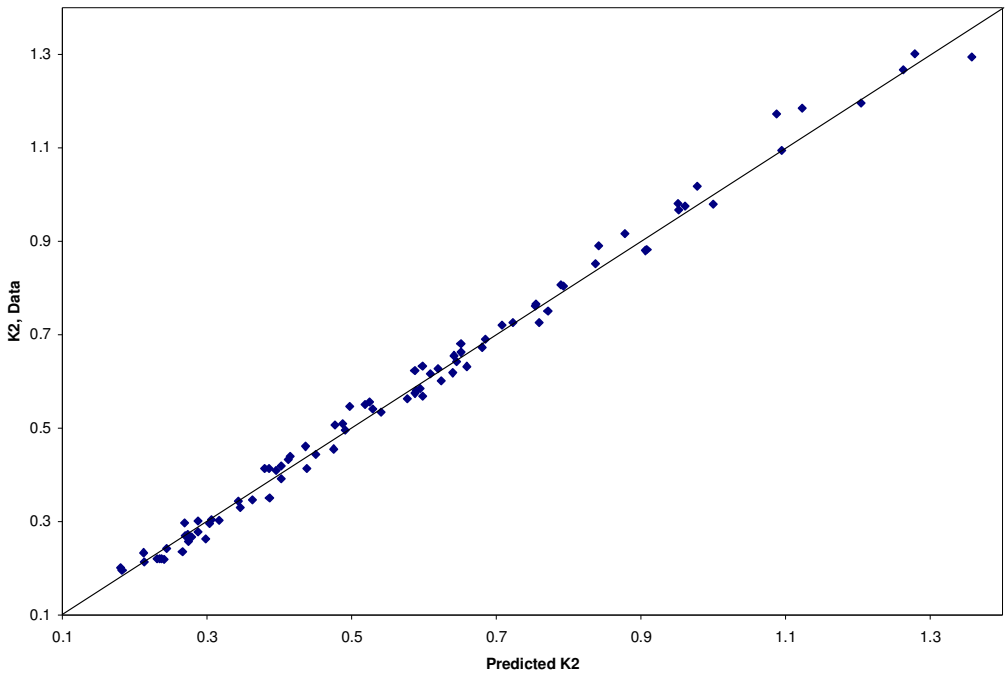


Figure 4.13 K_2 Model vs. Experimental Data for Acetone (1)-Water (2)

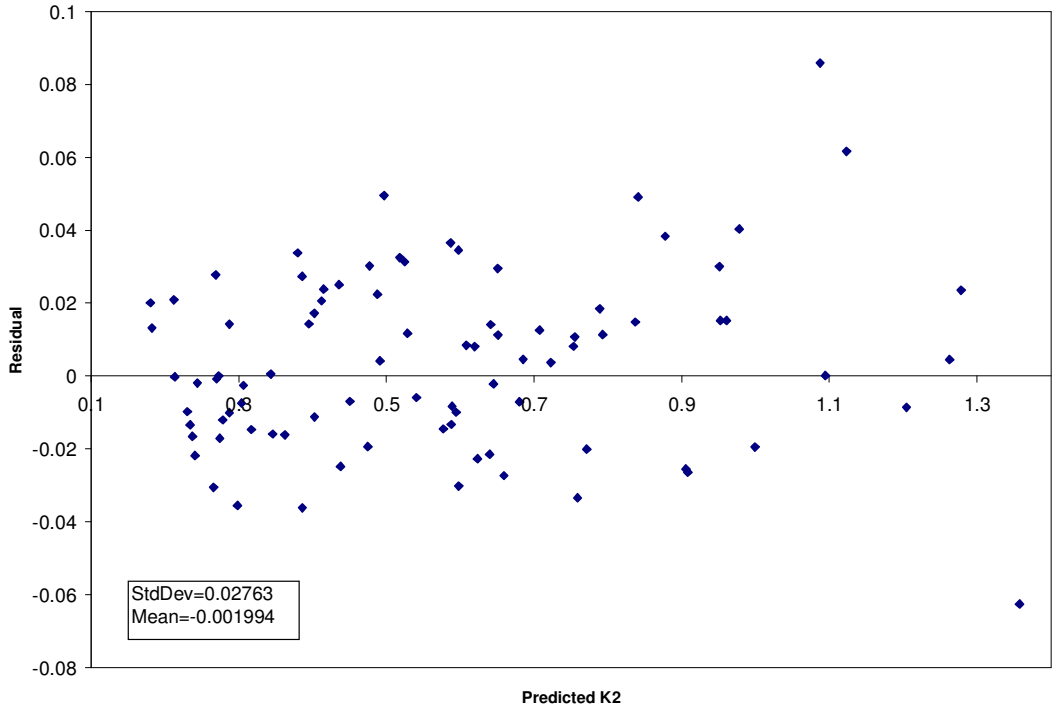


Figure 4.14 Residual Plot of K_2 for Acetone (1)-Water (2)

The $y = \hat{y}$ plot for generated K_1 and K_2 models for acetone-water system are given in Figures 4.11 and 4.13 respectively. The points are distributed along 45° line. No significant deviation from 45° line is observed. The residual plots for K_1 and K_2 models are given in Figures 4.12 and 4.14. The mean values for both residual plots are very close to zero.

|ΔK| and |ΔK|/K (%) for K_1 and K_2 at three different pressures are listed in Table 4.5. Maximum and average values of |ΔK|/K (%) for K_1 and K_2 are small, which indicate that the models for K_1 and K_2 fit the experimental data well. It's also observed that the model's deviation at medium (17.24 bar) and high (34.47 bar) pressure are larger than those at low pressure, 1.013 bar.

Table 4.5 Standard Error Analysis for Local Model for Acetone-Water System

P, bar	|D_K|/K, % K_1 (K_2)		|D_K| K_1 (K_2)	
	Max.	Avg.	Max.	Avg.
1.013	1.92 (4.0)	0.962 (2.78)	0.374 (0.0325)	0.0564 (0.0147)
17.24	4.99 (4.32)	2.61 (2.26)	0.643 (0.0273)	0.102 (0.0173)
34.47	4.97 (5.20)	3.01 (2.89)	0.245 (0.0625)	0.0661 (0.0298)

4.1.4 Performance of models for separation of non-ideal solutions

The generated K_1 and K_2 models for acetone-water solution are integrated with CHEMCAD as an user added module. T-x-y for three different pressure levels of 1.013bar, 17.24 bar and 34.47 bar are plotted and compared with activity coefficient model NRTL, as shown in Figure 4.15. As shown, the generated local models fit the entire range of data set relatively well. It also shows that the generated local models have larger deviations as the pressure increases.

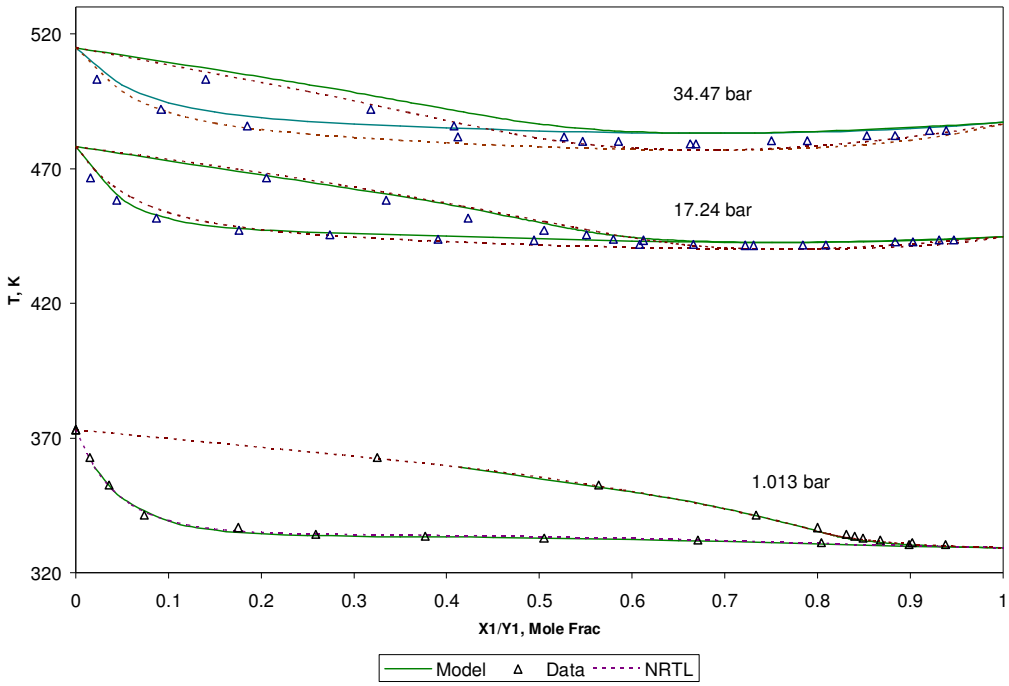


Figure 4.15 T-X_1-Y_1 at Different Temperatures for Acetone (1)-Water (2)

The K-composition and K-pressure analysis of models are given in Figures 4.16 and 4.17. K_1-composition plot in Figure 4.16 shows that the local model has a slightly better fit over composition than NRTL does at the entire data range. In K_1-pressure plot,

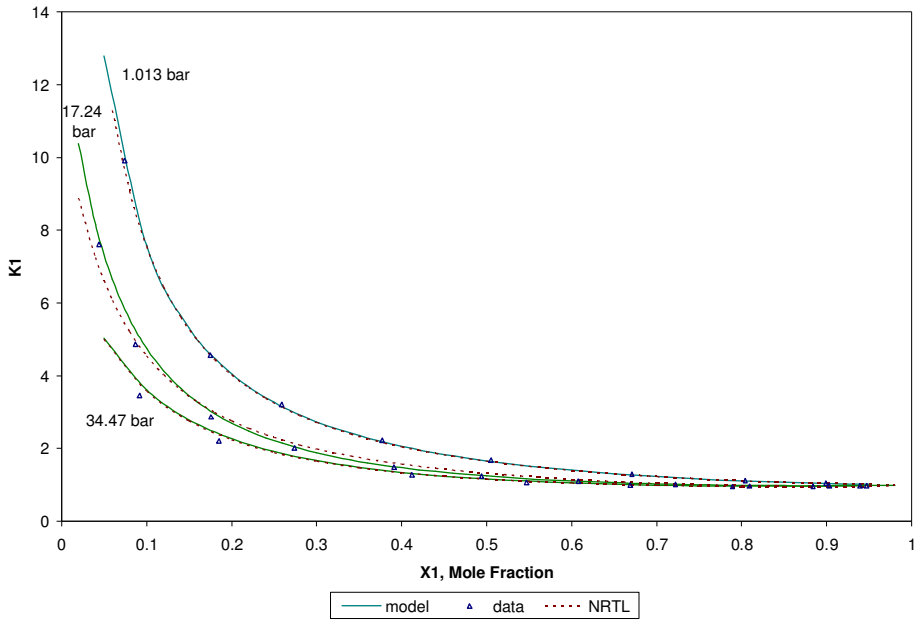


Figure 4.16 K_1 vs. Liquid Composition at Different Pressures for Acetone (1)-Water(2)

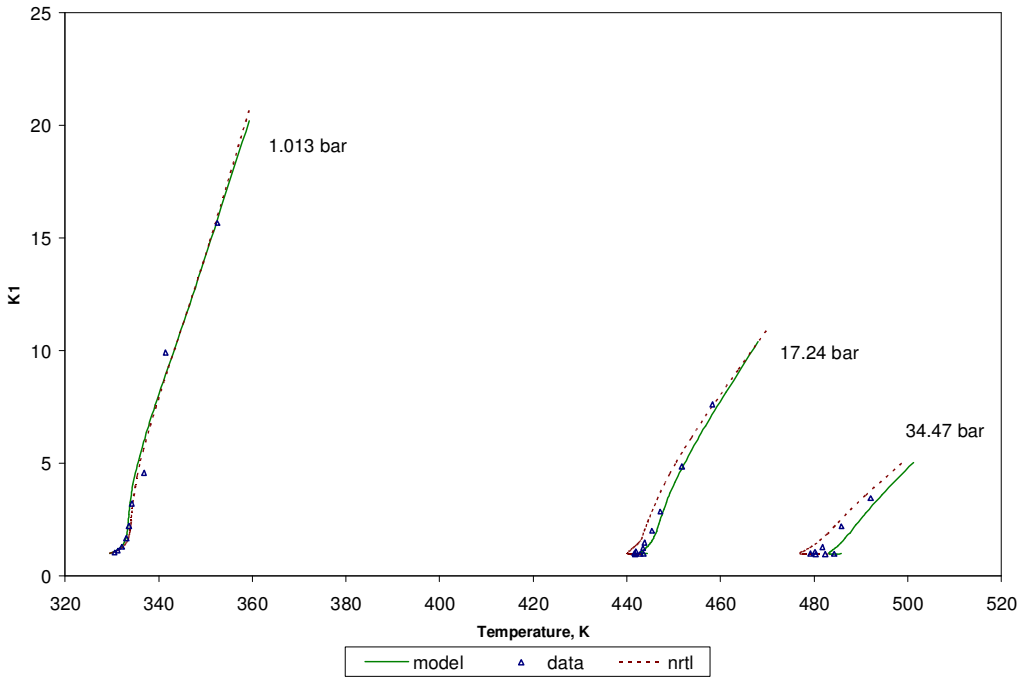


Figure 4.17 K_1 vs. Temperature at Different Pressures for Acetone (1)-Water (2)

the local model shows a larger deviation at high pressure due to the azeotropy. As can be seen from Figures 4.16 and 4.17, the dependency of the K-values on temperature and composition are different. Figure 4.17 shows that the relative deviation of temperature dependency is increased with the pressure increased, but still within the acceptable level.

In the examples given here, the local models were used at pressures up to 34.47 bar with reliable results. From our experience with mixtures that we have examined, the general conclusions are that, for non-ideal mixtures, the generated model is capable of correlating the equilibrium ratios to a relative acceptable accuracy over a wide range of compositions, temperature and pressure.

To test local model's reliability further, steady state simulation is run. A 56-plate acetone-water distillation column is simulated. The feed stream containing 400 lbmol/hr acetone and 600 lbmol/hr water is fed to the bottom of the column (tray 55) at pressure of 2 bar as saturated liquid. The column is operated under 1.013 bar on the top with 0.5 bar pressure drop, the mole fraction of acetone on the top and at the bottom is 0.95 and 0.05 respectively. The tower temperature profile is presented in Figure 4.18 and relative volatility is presented in Figure 4.19. Figures 4.18 and 4.19 show that, compared with the data and NRTL model, the generated local model's performance is excellent under the specified operating conditions.

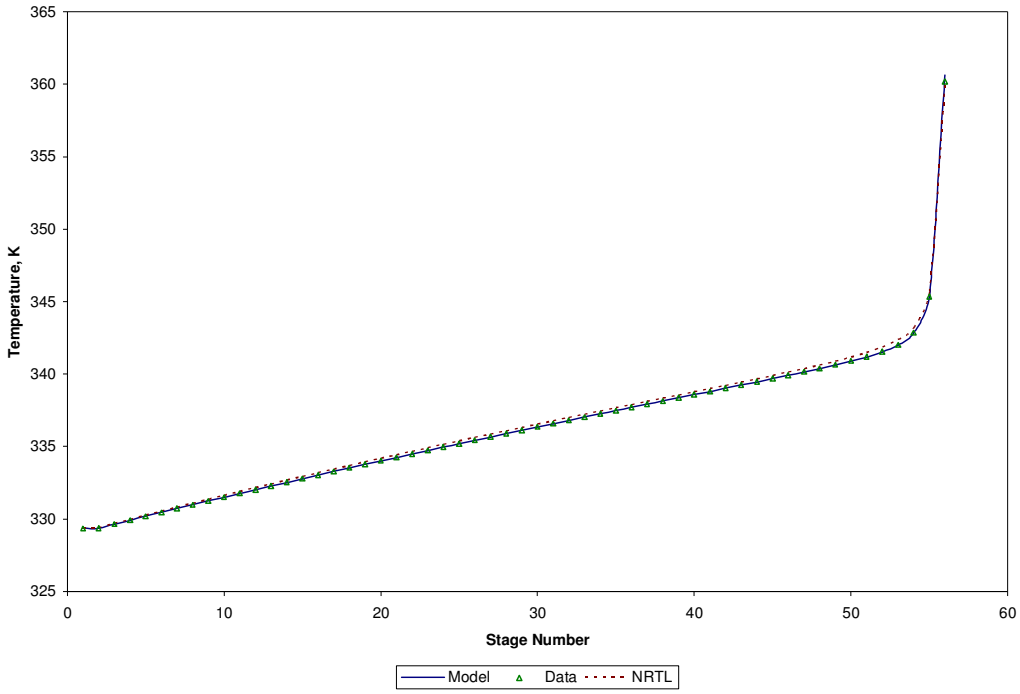


Figure 4.18 Temperature Profile of Acetone-Water Distillation Column

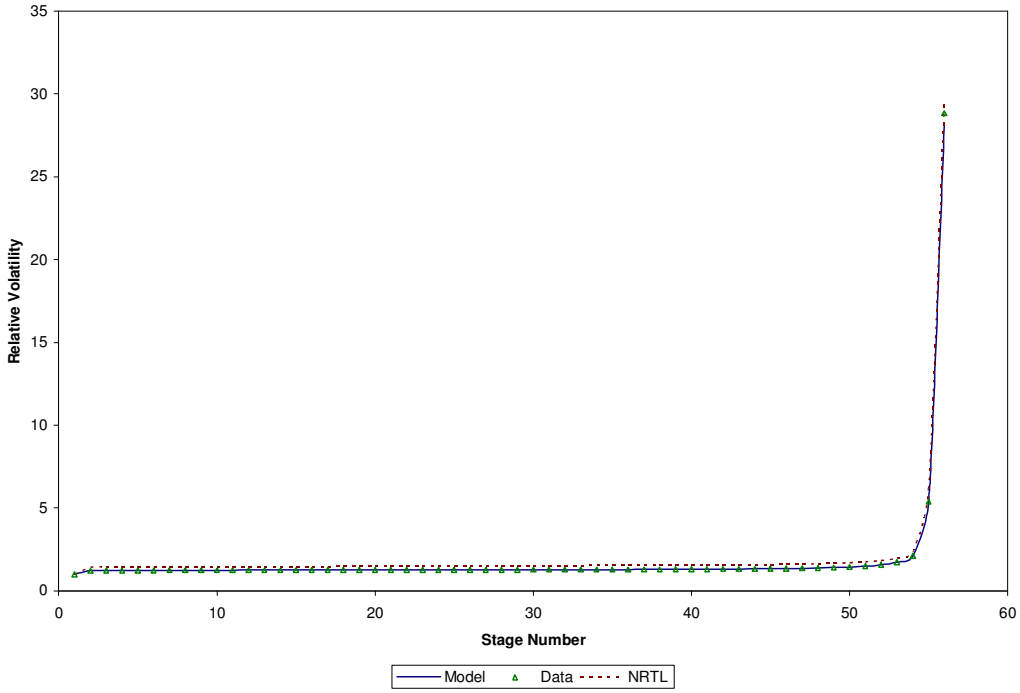


Figure 4.19 Relative Volatility Profile of Acetone-Water Distillation Column

In dynamic simulation, the process starts at steady state, a disturbance of 10% feed flow rate is introduced at the 50th minute of the simulation. The resulting tower vapor and liquid flow rate profiles at different simulation times are given in Table 4.6. The distillation flow rate fluctuation due to the 10% feed flow rate increase is shown in Figure 4.20. The pressure response to the disturbance over time is shown in Figure 4.21. All figures for dynamic simulation show that the generated model gives a reliable result in the simulation.

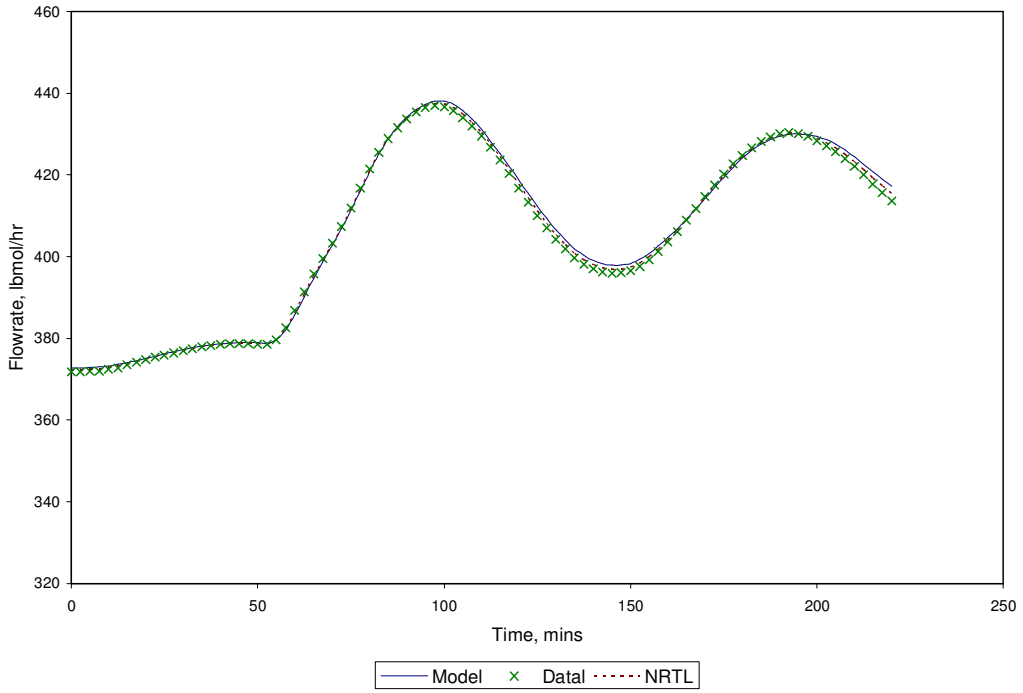


Figure 4.20 Distillation Total Flow Rate

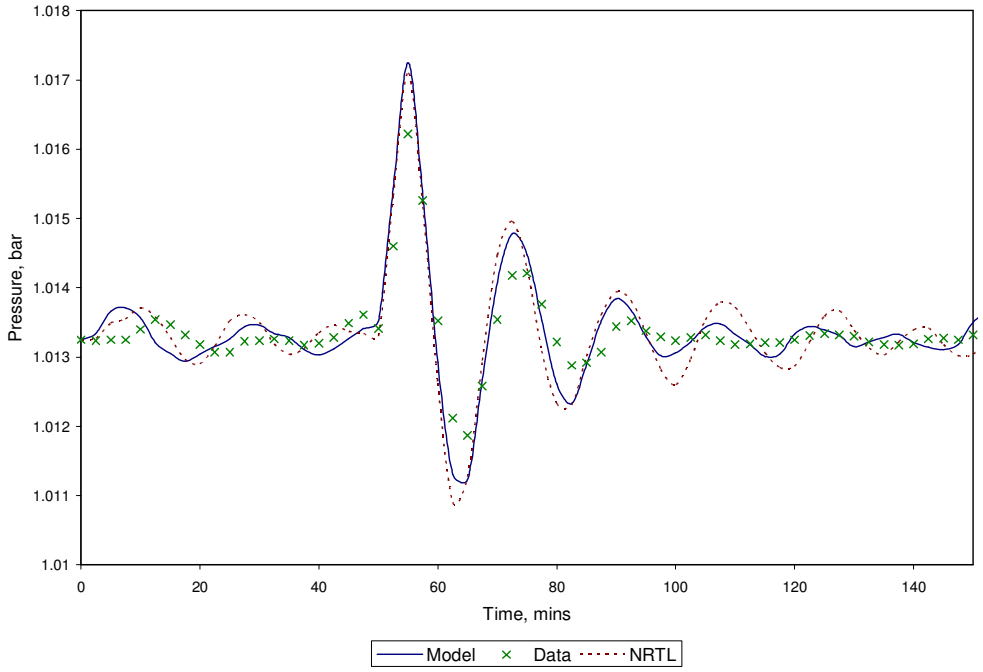


Figure 4.21 Distillation Column Pressure

Table 4.6 Tower Profile at Different Runtime

		0 (min.)			200 (min.)		
		Feed: 1000 lbmol/h			Feed: 1100 lbmol/h		
		Model	Data	NRTL	Model	Data	NRTL
Stage 5	Vapor lbmol/h	1630.8	1611.3	1523.6	1793.9	1772.4	1686.0
	Liquid lbmol/h	1257.8	1238.3	1150.2	1383.6	1362.2	1275.3
Stage 25	Vapor lbmol/h	1621.4	1602.0	1507.8	1783.6	1762.3	1668.6
	Liquid lbmol/h	1248.7	1229.3	1134.8	1373.6	1352.3	1258.3
Stage 50	Vapor lbmol/h	1598.2	1578.7	1485.1	1758.0	1736.5	1643.6
	Liquid lbmol/h	1220.9	1201.5	1109.5	1343.0	1321.6	1230.4
Condenser, MMBtu/h		-21.03	-20.78	-19.64	-23.13	-22.86	-21.76
Reboiler, MMBtu/h		21.51	21.25	20.06	23.66	23.37	22.26

4.2 Discussion

The linear models of vapor-liquid partition coefficient K for propane-propylene and acetone-water systems presented in Section 4.1 evolved from full data set that covers the entire pressure range. For acetone-water system, as shown in Figures 4.15 to 4.17, the T-X-Y diagram at 1.013 bar, 17.34 bar and 34.47 bar respectively, the deviation of the models increases with the pressure.

Table 4.7 R^2 of Different Models for K_1 in Group Tests

Data Pressure (P) Range for Linear and Non-linear Models		Propylene-Propane			Acetone-Water		
		PT	PTX	PTXY	PT	PTX	PTXY
Low P	Linear	0.995	0.996	0.996	0.85	0.995	0.996
	Nonlinear	0.995	0.996	0.996	0.87	0.995	0.996
High P	Linear	0.992	0.995	0.996	0.82	0.991	0.995
	Nonlinear	0.994	0.996	0.996	0.87	0.993	0.997
Full Data	Linear	0.993	0.994	0.995	0.82	0.991	0.996
	Nonlinear	0.994	0.995	0.995	0.87	0.993	0.996

In an effort to study pressure induced effects, the full data set was divided into low pressure and high pressure sets and additional tests were performed. The model was forced to be linear and nonlinear respectively, for both low and high pressures, as discussed in the next section.

4.2.1 Ideal gaseous and liquid mixture

Low pressure models developed for propylene-propane system with different terminal sets provide very good data representation. Models with PT, PTX and PTXY terminal sets have R^2 values of 0.995, 0.996 and 0.996 respectively. Including composition term in evolved local model doesn't improve model's goodness of fit since PT models represent the data very well. This result is expected as can be seen from the discussion to follow in this section.

The basic vapor-liquid equilibrium relation can be expressed as:

$$\bar{f}_i^l(T,P,\underline{x}) = x_i P \bar{\phi}_i^l(T,P,\underline{x}) = y_i P \bar{\phi}_i^v(T,P,\underline{y}) = \bar{f}_i^v(T,P,\underline{y}) \tag{4.1}$$

The description of vapor-liquid equilibrium using an equation of state for both liquid and vapor phase is referred to as the $\phi-\phi$ method, then,

$$K_i \equiv \frac{y_i}{x_i} = \frac{\bar{\phi}_i^l(T,P,\underline{x})}{\bar{\phi}_i^v(T,P,\underline{y})} \tag{4.2}$$

The other alternative is to use an activity coefficient model for the liquid phase and an equation of state for the vapor phase, i.e, $\gamma-\phi$ method. Eq. (4.1) can be rewritten in the form of:

$$x_i \gamma_i(T,P,\underline{x}) P_i^{sat} \bar{\phi}_i^{l,sat}(T,P) = y_i P \bar{\phi}_i^v(T,P,\underline{y}) \tag{4.3}$$

$$K_i = \frac{\gamma_i(T,P,\underline{x})P_i^{sat}\,\underline{\phi}_i^{l,sat}(T,P)}{P\,\underline{\phi}_i^{v}(T,P,\underline{y})} \tag{4.4}$$

At low pressures, the vapor phase can be treated as an ideal gas, and all fugacity coefficient corrections are negligible. Thus, Eq. (4.4) can be simplified to:

$$K_i = \frac{\gamma_i(T,P,\underline{x})P_i^{sat}}{P} \tag{4.5}$$

Furthermore, propylene-propane mixture forms an ideal solution in the liquid phase, i.e. $\gamma_i = 1$ for both propylene and propane. Then, eq. (4.5) can be further simplified to:

$$K_i = \frac{P_i^{sat}(T)}{P} \tag{4.6}$$

Eq. (4.6) depicts that the vapor-liquid partition coefficient K for propylene-propane system at low pressure is only the function of temperature and pressure, not composition. Therefore, the models with terminal sets of PTX and PTXY don't improve the models' accuracy significantly.

4.2.2 Non-ideal gaseous mixture and ideal liquid mixture

For propylene-propane system at high pressure, as shown in Table 4.7, R^2 for the local models using different terminal sets of PT, PTX and PTXY are 0.992, 0.995 and 0.996 respectively. Compared with the local model using the terminal set of PT, the one with PTX has better accuracy, whereas the model using PTXY has about same R^2 value.

At higher pressures, the behavior of propylene-propane vapor phase deviates from ideal gas state and thus fugacity coefficient cannot be negligible. The liquid phase is not affected by pressure much so it still can be treated as ideal solution. The Eq. (4.4) can be reduced to:

$$K_i = \frac{P_i^{sat} \bar{\phi}_i^{l,sat}(T,P)}{P \bar{\phi}_i^{v}(T,P,\underline{y})} \tag{4.7}$$

Due to the composition dependent characteristic of fugacity coefficient correction, the introduced composition term of the evolved local model can and do improve model's accuracy.

4.2.3 Ideal gaseous and non-ideal liquid mixture

As can be seen In Table 4.7, the local models developed for acetone-water system at low pressure improve significantly the goodness of fit after the liquid composition is introduced to the model, R^2 is 0.85 for the terminal set of PT and R^2 is 0.995 for the terminal set of PTX. There is no significant improvement for the terminal set of PTXY where R^2 is 0.996. At low pressures, the vapor phase of acetone-water system approaches ideal gas state, and fugacity coefficient corrections can be omitted.

However, in liquid phase, acetone-water forms a polar solution which displays a strong non-ideal behavior. Therefore, the effect of activity coefficient needs to be taken into account. Eq. (4.5) can be used to describe the vapor-liquid equilibrium for acetone-water system at low pressure.

Since activity coefficient is composition dependent, the evolved local model with composition term improves the goodness of fit.

4.2.4 Non-ideal gaseous mixture and non-ideal liquid mixture

At high pressures, R^2 s' for local models with different terminal set of PT, PTX and PTXY are 0.82, 0.991 and 0.995 respectively for acetone-water system. The models with terminal set of PTX and PTXY have better data fit than the model with the terminal set of only PT. The model with PTXY also has better accuracy than the model with PTX. At higher pressures, both vapor and liquid phase display non-ideal behavior, and therefore fugacity coefficient and activity coefficient are necessary for description of the mixture. The composition has relatively larger impact on the system, in both vapor phase and liquid phase. The rigorous form of K model Eq. (4.4) can be used to describe this binary system at high pressure. From Eq. (4.4), we can see K value is vapor and liquid composition- dependent, therefore, the local models with the terms of composition in vapor and liquid phase can significantly improve the goodness of fit.

It's also observed that, in acetone-water system at low pressure, the model's accuracy with PTX is close to that of the model with PTXY, however, at high pressure, the difference of accuracy between PTX and PTXY is larger. This is because at higher pressures, the non-ideal gas behavior has more impact in vapor phase; therefore, including the composition in the vapor phase can improve the model's accuracy.

4.2.5 Linear model vs. nonlinear model

Using each low and high pressure group data, linear and nonlinear models were developed. The linear models that evolved from low pressure data for both propane-propylene and acetone-water systems were given in Table 4.7. As can be seen in the Table 4.7, nonlinear models for low pressure do not exhibit significant improvement on goodness of fit for propylene-propane system. At low pressure, the propane-propylene system is an ideal or nearly ideal system. The linear model is good enough to describe the characteristics of solution. For the acetone-water system, the nonlinear model improves the prediction accuracy for the one with terminal set of PT, which R^2 is 0.87 compared to 0.85 for linear model, but there is no significant improvement for the linear models that have terminal set of PTX and PTXY. It indicates that composition is a necessary term for the evolved local model to have adequate capability to describe non-ideal system. Once the composition term is included in the model, linear model is also good enough to describe the non-ideal behavior.

For higher pressures, the scenario got changed. When the linear model and the nonlinear model are compared, for propane-propylene system, the nonlinear model has slight improvement on accuracy for the PT model, where R^2 is 0.994 for the nonlinear model compared with 0.992 for the linear model. For models with terminal sets of PTX and PTXY, the nonlinear ones don't show significant improvement on models' accuracy. For the acetone-water system at higher pressures, the nonlinear model shows some improvement for PT, PTX and PTXY terminal sets. The acetone-water system exhibits

strong nonlinearity with high pressure. Therefore, a nonlinear model may describe this behavior better. It also can be further demonstrated in Figures 4.16 and 4.17, with K_1 vs. composition and K_1 vs. pressure plots respectively. At low pressures, with no azeotropy present, the linear local model with PTX terminal set has a good fit to the data. At higher pressures, with an azeotropic behavior, K value exhibits a strong nonlinearity. Therefore, the nonlinear model can fit the data better.

Acetone-water is a non-ideal system with an azeotropic point, which displays a nonlinear behavior with increasing pressure. The vapor-liquid partition coefficient K also displays a stronger nonlinearity than that of propane-propylene system. However, the model that evolved from the data is a linear model, with respective to its parameters. This limits model performance for the acetone-water system.

This group of tests displays that, the models with different terminal sets are consistent with the thermo-physical behavior of ideal (or nearly ideal) and non-ideal systems.

4.2.6 Extrapolation

In an effort to explore the extrapolation capability and validity of the models, models generated utilizing only low pressure data, including both the linear and the nonlinear models, were tested with the high pressure data. Similarly, models generated using high pressure data were used to test model at lower pressure.

The R^2 s for each case are shown in Table 4.8. "***" in Table 4.8 indicates the correspondingly redundant result presented in Table 4.7.

Table 4.8 R^2 of Extrapolation Test

		Propylene-Propane		Acetone-Water	
		fit to low pressure data	fit to high pressure data	fit to low pressure data	fit to high pressure data
model (PTX) generated from low pressure data	linear	***	0.848	***	0.528
	nonlinear	***	0.976	***	0.720
model (PTX) generated from high pressure data	linear	0.946	***	0.919	***
	nonlinear	0.966	***	0.977	***

As can be seen in Table 4.8, the models generated from low pressure data do not represent high pressure data well. This is true whether the model is linear or nonlinear, particularly for acetone-water system since azeotropy information is missing in low pressure. Also one can observe that the nonlinear models have better generalization capability than the linear models which is not surprising giving the non-linear nature of the phenomena particularly as transitions from ideal gas to real or azeotropy occurs. Similarly, the models generated from high pressure data have better generalization capability than those generated from low pressure data. These results prove again that genetic programming method is a fully data-driven method and is reliable in capturing phenomena when applied judiciously.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

This research proposes a hybrid evolutionary modeling algorithm to build the vapor-liquid equilibrium K-factor models automatically. The developed system may also serve as a general-purpose machine function identification package which can automatically build a function model to fit the given experimental data. The main idea of the algorithm is to embed linear or nonlinear regression into GP, where GP is employed to optimize the structure of a model, while linear or nonlinear regression method is employed to optimize its parameters. A distinct advantage of this method is that no a-priori assumptions have to be made about the actual model form: the structure and complexity of the model evolve as part of the problem solution. It may reduce the limitations on function structure that insufficient descriptions on studied system introduced by proposed initial structure, or over-simplified structure introduced by inappropriate assumptions made for function simplification procedure. The developed K model can be inserted in process simulation programs to save computation time, without affecting the accuracy of the simulation.

The results clearly prove that the hybrid system is able to find the explicit form of models that closely approximate the data in a relatively wide range while the accuracy can be tailored to a desired level. The models have also been shown to accurately

represent ideal mixture equilibrium ratios for a binary vapor-liquid system. The greater the accuracy and region of validity of the models, the less frequently the parameters have to be updated. This in turn will require less use of the rigorous thermodynamic-physical property programs for computationally expensive property evaluations. For a non-ideal system with azeotropic point, GP seems to be capable for the whole pressure range at an acceptable level, and if necessary, one can adjust the parameter of the generated model to fit the data well.

The experimental data was obtained from literature, which may contain error. The results revealed that the hybrid system is able to find the explicit model that closely approximated the data. It shows GP's robustness on data.

In some cases, parsimony principle was incorporated through the fitness function. The results show that a simpler structure, but less accuracy can be obtained.

In conclusion, the results presented in this research indicate the potential of GP for developing thermo-physical model and other general purpose function identification. Genetic programming is a robust and efficient paradigm for discovering model structure using the expressiveness of symbolic representation.

In this developed package, Marquardt regression method is used to get the parameter values for a nonlinear model structure. This method results in local as opposed to global solutions. The GP, as well as other evolutionary methods, will terminate early or converge to incorrect solution when the parameters regressed using local solvers are not adequate. Therefore, a global optimization method will resolve the local optimization problem, as well as ensuring a robust hybrid symbolic regression scheme.

As stated before, a primary classification used for property and process models in chemical engineering is algebraic versus differential equation models. K model is an algebraic equation that has multiple dependent variables and a single independent variable. The extended application of the developed package to differential equation can be studied further, which may enable the functionality of the package to be completed.

REFERENCES

Bamicki, S. D. and Fair, J. R. (1990). Separation synthesis: a knowledge-based approach. 1. Liquid Mixture separations. *Ind. Eng. Chem. Res.,* 29: 421-435.

Banares-Alcantara, R., Westerberg, A. W. and Rychener, M. D. (1985). Development of an expert system for physical property predictions. *Computers Chem. Engng,* 9:127-142.

Cao, H. Yu, J., Kang, L. and Chen, Y. (1999). The Kinetic evolutionary modeling of complex systems of chemical reactions. *Computers & Chemistry,* 23:143-151.

Chen S.-H. and Yeh C.-H. (1997)**.** Toward a computable approach to the efficient market hypothesis: an application of genetic programming. *Journal of Economic Dynamics and Control,* 21(4): 1043-1063.

Csukas, B. and Balogh, S. (1998). Combining genetic programming with generic simulation models in evolutionary synthesis. *Computers in Industry,* 36:181-197.

Draper, N. R. and Smith H. (1981). Applied regression analysis. John Wiley, NY.

Englezos, P. and Kalogerakis, N. (2001). Applied parameter estimation for chemical engineers. Marcel Dekker, NY.

Fogel, L. J., Owens, A. J. and Walsh, M. J. (1966). Artificial intelligence through simulated evolution. John Wiley, NY.

Frank, P. and Köppen-Seliger, B. (1997)**.** New developments using AI in fault diagnosis**.** *Engineering Applications of Artificial Intelligence,* 10: 3-14.

Franks, R. G. (1967). Mathematical modeling in chemical engineering, Wiley, New York.

Fredenslund, A., Rasmussen, O. and Mollerup, J. (1980). Thermo physical and transport properties for chemical process design. In *Foundations of Computer-aided Process Design*.

Freitas, A.A. (2002). A survey of evolutionary algorithms for data mining and knowledge discovery. To appear in: A. Ghosh and S. Tsutsui. (Eds.) Advances in Evolutionary Computation. Springer-Verlag.

Friese, T. Ulbig, P. and Schulz, S. (1998). Use of evolutionary algorithms for the calculation of group contribution parameters in order to predict thermodynamic properties. Part I: Genetic Algorithms. *Computers Chem. Engng,* 22:1559-1572.

Gani R. and O'Connell, J. P. (1989). A knowledge-based system for the selection of thermodynamic models. *Computers Chem. Engng,* 13:397-404.

Gao, L. and Loney, N. W. (2001). Evolutionary polymorphic neural network in chemical process modeling. *Computers Chem. Engng,* 25:1403-1410.

Goldberg, D. E. (1989). Genetic algorithms in search optimization and machine learning. Addison Wesley, Reading, MA.

Greeff, D. J. and Aldrich, C. (1998). Empirical modelling of chemical process systems with evolutionary programming**,** *Computers Chem. Engng,* 22:995-1005.

Grosman, B. and Lewin, R. D. (2004). Adaptive genetic programming for steady-state process modeling, *Computers Chem. Engng,* 28:2779-2790.

Hand, D., Mannila, H. and Smyth, P. (2001). Principles of data mining. MIT Press, Cambridge, MA.

Hansen, M. and Yu, B. (2001). Model selection and the principle of minimum description length. *Journal of the American Statistical Association,* 96(454): 746--774.

Holland, J. H. (1975). Adaptation in natural and artificial system, University of Michigan Press, Ann Arbor, MI.

Hutter, M. (2002). Fitness uniform selection to preserve genetic diversity. *In Proc. 2002 congress on evolutionary computation.* 783-788.

Johnson, H. E., Gilbert, R. J., and Winson, K. (2000). Explanatory analysis of the metabolome using genetic programming of simple, interpretable rules. *Genetic Programming and Evolable Machines.* 1(3):243-258.

Jolliffe, I. T (1986). Principal component analysis, Springer-Verlag, New York.

King, R. A. (1986). Expert systems for materials selection and corrosion. *The Chemical Engineer,* December, 42-57.

Kinnear, K. E. Jr. (editor) (1999). Advances in genetic programming III. The MIT Press, Cambridge, MA.

Kinnear, K. E. Jr. (editor) (1994). Advances in genetic programming. The MIT Press, Cambridge, MA.

Kirkwood, R. L., Locke, M. H. and Douglas, J. M. (1988). A prototype expert system for synthesizing chemical process flowsheet. *Computers Chem. Engng,* 12:329-341.

Koza, J. R., Bennett III. F. H., Andre, D. and Keane, M. A. (1999). Genetic programming III: darwinian invention and problem solving. Morgan Kaufmann, San Francisco, CA.

Koza, J. R. (1994). Genetic programming II: automatic discovery of reusable programs. MIT press, Cambridge, MA.

Koza, J. R., Keane, M. A., Streeter, M. J. Mydlowec, W., Yu, J. and Lanza, G. (2003). Genetic programming IV: routine human-competitive machine intelligence. Springer.

Legg, S., Hutter, M., and Kumar, A. (2004). Tournament versus fitness uniform selection. *In Proceeding of the 2004 congress on evolutionary computation.*

McKay, B., Willis, M. and Barton, G. (1997). Steady-state modeling of chemical process systems using genetic programming. *Computers Chem. Engng,* 21:981-996.

Miettinen, K. (editor) (1999). Evolutionary algorithms in engineering and computer science: recent advances in genetic algorithms, evolution strategies, evolutionary programming, genetic programming, and industrial applications, Wiley, NY.

Mitchell, M. (1996). An introduction to genetic algorithms, MIT Press, Cambridge, MA.

Mitchell, T. (1997). Machine Learning. McGraw Hill, NY.

Ozyurt, B. (1998). Chemical process fault diagnosis using pattern recognition and semi-quantitative model based methods. Ph.D. dissertation, University of South Florida.

Poli, R. (2008). Covariant Parsimony Pressure for Genetic Programming. *Computers Chem. Engng,* 21:981-996.

Pöyhönen, H.O. and Savic, D. A. (1996). Symbolic Regression Using Object-Oriented Genetic Programming (in C++), Centre For Systems And Control Engineering, Report No. 96/04, School of Engineering, University of Exeter, Exeter, United Kingdom, 72-84.

Quagliarella, D. (editor) (1998). Genetic algorithms and evolution strategy in engineering and computer science: recent advances and industrial applications, John Wiley & Sons, New York.

Quantrille, T. E. and Liu, Y. A. (1991). Artificial Intelligence in chemical engineering. Academic Press, Inc. San Diego.

Ramirez F. (1989). Computational methods for process simulation, Butterworths, MA.

Ruiz, D., Nougués, J., Calderón, Z., Espuña, A. and Puigjaner. L. (2000). Neural network based framework for fault diagnosis in batch chemical plants**.** *Computers & Chemical Engineering.*24:777-784.

Stephanopoulos, G. (1987). The future of expert systems in chemical engineering. *Chem. Engng. Prog,* 83:44-51.

Stephanopoulos, G. and Han, C. (1994). Intelligent systems in process engineering: a review. In *Proc. 5th Intl. Symp. on Process Systems Engineering, Kyongju, Korea*.

Stephanopoulos, G., Johnston, J. and Kriticos, T. (1987). Design-kit: an object-oriented environment for process engineering. *Computers Chem. Engng* 11(6): 655-678.

Thuraisingham, B. (1999). Data mining technologies, techniques, tools and trends. CRC Press, NY.

Xu, W. (2005). Improved genetic algorithms for deterministic optimization and optimization under uncertainty, *Ind. Eng. Chem. Res,* 44: 7132-7146.

Zhang, B-T. and Muhlenbein, H. (1993). Evolving optimal neural networks using genetic algorithm with Occam's razor, *Complex Systems,* 7:199-220.

Zhang, B-T. and Muhlenbein, H. (1995). Balancing accuracy and parsimony in genetic programming, *Evolutionary Computation,* 3(1):17-38.

Zhang, B-T. (2000). Bayesian methods for efficient GP, *Genetic Programming and evolvable machine learning,* 1, 217-242.

APPENDICES

Appendix A. User Manual for GP Package

A.1 MATLAB files

In the developed GP package, the MATLAB files (.m files) include:

- gp_ main.m: the main code that completes GP operations.
- marquardt.m: the subroutine that performs nonlinear parameter regression using Marquardt method, and calculates the fitness for an individual.
- linear.m: the subroutine that performs linear parameter regression using least square, and calculates the fitness for an individual.

A.2 Architecture

The relationship between the .m files described above is given in the Figure A.1. The expanded structure for main code (gp_main.m) is on left side, and the subroutine is shown on the right hand side.

Appendix A (Continued)

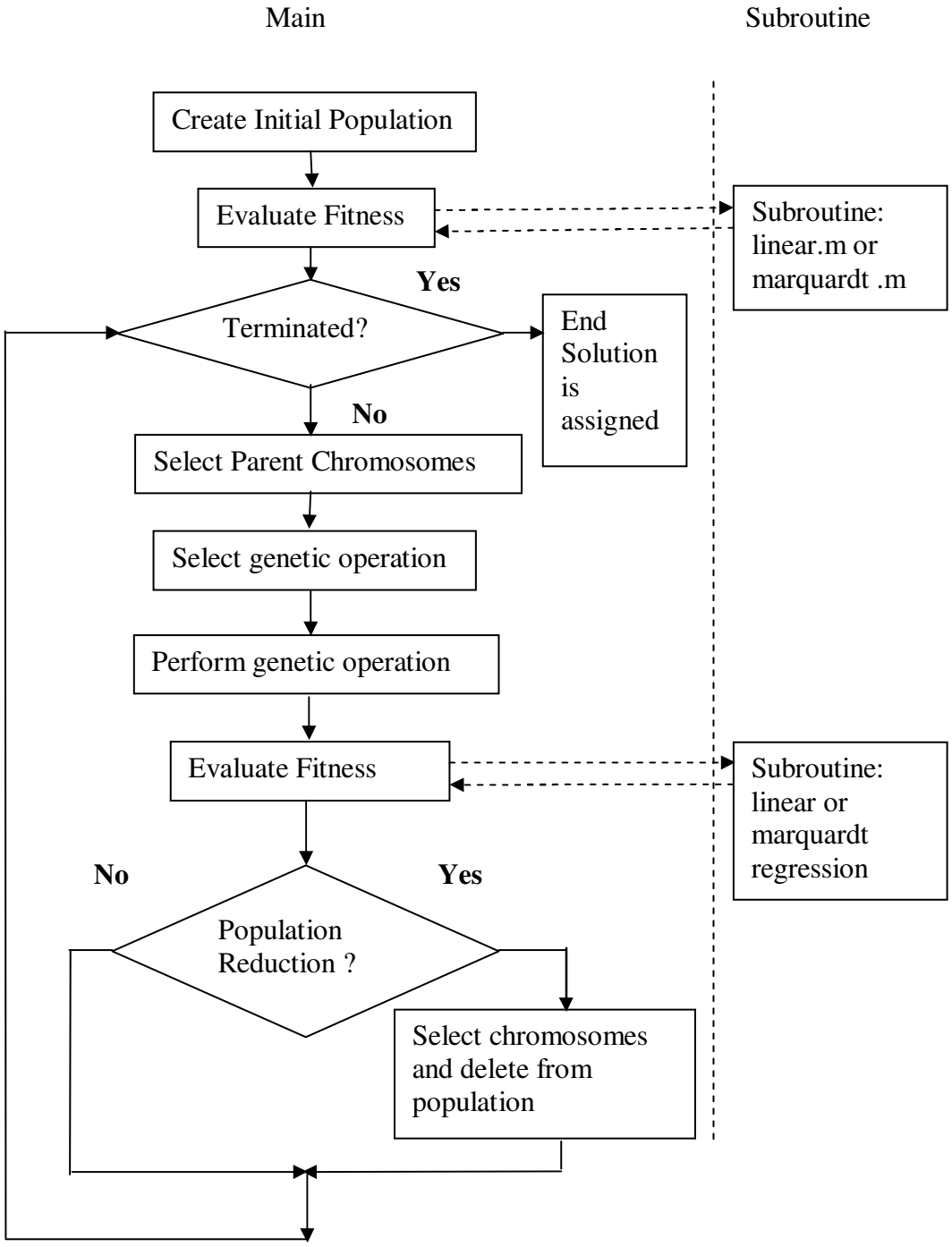


Figure A.1 Architecture of MATLAB Code

Appendix A (Continued)

A.3 How to use the GP package

In this section, we will follow the Figure A.1, and give a detail explanation on how to specify the parameter in the code.

- Step 1: load data set and assign the value to the variables of terminal set. Command: DATA = LOAD (‘FILENAME’)
- Step 2: create initial population. Command: CHROM_ID = GS_NEW (POPULATION, CHROMOSOME), POPULATION is a character string designating the chromosome population in which the new chromosome is created. CHROMOSOME is a character string defining the gene structure of the new chromosome. CHROM_ID is an integer identifying the newly created chromosome. Example: mem_id = gs_new ('Pop1', 'v2/T'); The initial population defined with GS_NEW command needs to include the complete terminal set and function set.
- Step 3: define regression strategy. Command: REGRESSION_TYPE = ‘TYPE_VALUE’. TYPE_VALUE is set to 0 for linear regression, 1 for nonlinear regression, or 1 for both. Default value is set to 0. This command is to define the regression strategy going to be used in the GP.

Appendix A (Continued)

- Step 4: define stop criteria (Maximum Generation). Command: MAXGEN = 'MAXIMUM NUMBER OF GENERATION'. This command defines the maximum generation of GP. The stop criterion is controlled by a while loop.
- Step 5: select parent chromosomes. The selection strategies provided in GP toolbox are:
    - GS_SEL_HIFIT (or GS_SEL_LOFIT): select two chromosomes with the highest (lowest) fitness values and return their ID
    - GS_SEL_LOTRN (or GS_SEL_HITRN): choose chromosomes using tournament selection and return their IDs, with chromosomes with higher (lower) fitness winning the tournament.
    - GS_SELR: select two chromosomes randomly.
    - GS_SELR_HIFIT (or GS_SELR_LOFIT): select two chromosomes randomly, with the probabilities of selection being proportional to their fitness value.
    - GS_SELR_HIRANK (or GS_SELR_LORANK): select two high (low) fitness chromosomes randomly, with the probabilities of selection being based on fitness rank. In this research, tournament method is used. Example: mem_ids = gs_sel_lotrn ('Pop1',4,2).

Appendix A (Continued)

- Step 6: define and perform genetic operators. Command: OP_NAME = GS_SEL_OP (OP_LIST, OP_PROB). Choose a random genetic operation from a list according to specified probabilities. OP_LIST is a cell array containing the names of the operations among which to select, OP_PROB is a corresponding list of probabilities, OP_NAME returns a string identifying the chosen operation, or null (‘ ’) if an error occurs.

  Then run genetic operations using: Command: CHROM_IDS = GS_OP (POPULATION, OPERATION, CHROM_ID1, [CHROM_ID2, MUTPROB]). GS_OP implements a specified genetic operation on specified chromosome(s). POPULATION is a string designating the population from which the chromosomes to be operated. OPERATION is a string identifying the genetic operation. CHROM_ID1 is the first chromosome to be operated on. CHROM_ID2 is the second chromosome to be operated on, and is only required for crossover operations. MUTPROB is the mutation probability, required only for binary mutation. CHROM_IDS returns a vector of identification numbers of the one or two new chromosomes created by the genetic operation. Example: op_name = gs_sel_op ({'mut', 'xovr'}, [0.5, 0.5]); off_ids = gs_op ('Pop1', op_name, 12, 22).

Appendix A (Continued)

- Step 7: define deletion strategy. The commands are same as the ones used for the selection strategies. Command: OUTCOME = GS_DEL (POPULATION, CHROM_ID). POPULATION is a character string designating the chromosome population from which the chromosome is to be deleted. CHROM_ID is an integer identifying the chromosome to be deleted. OUTCOME returns the specified CHROM_ID if the deletion is successful; otherwise, a value 0 is returned. Example: gs_del ('Pop1', off_ids(off)).

Appendix B. Statistical Analysis

B.1 Tests for outliers

Outlier can be detected from student residuals.The MATLAB code is given in Table B.1.

Table B.1 MATLAB Code for Outlier Test

```
nn=length(x1data);
residual=k1_calc1-k1data;

avg_k1=sum(k1data)/nn;

sst=sum((k1data-avg_k1).^2);
sse=sum(residual.^2);
ssr=sum((k1_calc1-avg_k1).^2)

counter=length(param);

mse=sse/(nn-counter)
msr=ssr/(counter-1)

chr1=char('(1+3*x1+x1*log(x1))-v1*T+v2*x1/T-v3*(1-log(x1))-v4*(1-x1)')

term1=char('T');
term2=char('x1./T')
term3=char('1-log(x1)')
term4=char('1-x1')

r=0;
X=[eval(term1) eval(term2) eval(term3) eval(term4)];
H=X*inv(X'*X)*X';
for ii=1:length(k1data)
   r(ii)=abs((k1data(ii)-k1_calc1(ii))/sqrt(mse*(1-H(ii,ii))));
end
```

Appendix B (Continued)

Table B.2 The Result for Outlier Test

	r		r		r		r		r
1	1.667	26	0.014	51	0.848	76	0.446	101	0.1719
2	0.694	27	0.341	52	0.224	77	0.275	102	0.0758
3	1.594	28	0.106	53	0.0003	78	0.022	103	0.0657
4	0.978	29	0.154	54	0.091	79	0.003	104	0.0673
5	0.437	30	0.030	55	0.156	80	0.062	105	0.1176
6	0.087	31	0.005	56	0.201	81	0.113	106	0.1549
7	0.054	32	0.009	57	0.215	82	0.100		
8	0.165	33	0.026	58	0.182	83	0.125		
9	0.259	34	0.081	59	0.129	84	0.090		
10	0.334	35	0.124	60	0.094	85	1.540		
11	1.979	36	0.592	61	2.647	86	1.265		
12	0.099	37	0.491	62	1.371	87	0.963		
13	0.846	38	1.389	63	0.968	88	0.529		
14	0.457	39	0.216	64	0.428	89	0.288		
15	0.598	40	0.170	65	0.106	90	0.046		
16	0.389	41	0.150	66	0.026	91	0.042		
17	0.380	42	0.060	67	0.157	92	0.072		
18	0.119	43	0.058	68	0.194	93	0.085		

Appendix B (Continued)

Table B.2 (Continued)

	r		r		r		r
19	0.021	44	0.057	69	0.213	94	0.108
20	0.028	45	0.061	70	0.172	95	0.039
21	0.088	46	0.040	71	0.141	96	0.022
22	0.155	47	0.006	72	4.143	97	6.162
23	1.001	48	0.013	73	0.828	98	0.056
24	2.510	49	2.592	74	0.803	99	0.268
25	0.520	50	2.276	75	0.530	100	0.206

$|r_i| > 3.0$ is a strong indicator that this data point is a likely outlier. The table shows that, data point #72 and #97 could be an outlier. Looking into the data, it is found that, both #72 and #97 are the equilibrium data points of dilute solution, which cannot be deleted, therefore, in this data set, no outlier was deleted.

B.2 Evaluation of final model: cross-validation

When it's not possible to gather additional new data, and compare these observed data with the predicted value from the model, a cross validation is an alternative evaluation method.

Appendix B (Continued)

The original data set can be divided into two subsets A and B randomly, which is double cross-validation. A good model should be fit to both parts. The MATLAB code for data set partition is given in Table B.3.

Table B.3 MATLAB Code for Data Set Partition

```
r = ceil(106.*rand(53,1))
r=sort(r,1)
alarm=2;

for ivv=1:20
for iii=2:53
if r(iii)==r(iii-1)
 r(iii) = ceil(106.*rand(1,1))
end
end

r=sort(r,1);
end
r=sort(r,1);

A_index=r;
B_index=0;
counter_Bset=0;
for iqq=1:106
alarm0=0;
for imm=1:53
if A_index(imm)==iqq
alarm0=1;
break;
end
end
if alarm0 ==0
counter_Bset=counter_Bset+1;
B_index(counter_Bset)=iqq;
end
end
```

Appendix B (Continued)

Table B.4 The Result for Data Set Partition

Set A (Datapoint_indext_setA) 2 3 7 15 16 19 21 22 25 29 31 33 37 38 41 44 45 46 48 49 50 52 53 54 56 57 58 63 65 66 69 70 72 73 74 76 78 79 80 81 84 87 88 89 90 91 92 95 96 98 99 100 101
Set B (Datapoint_indext_setB) 1 4 5 6 8 9 10 11 12 13 14 17 18 20 23 24 26 27 28 30 32 34 35 36 39 40 42 43 47 51 55 59 60 61 62 64 67 68 71 75 77 82 83 85 86 93 94 97 102 103 104 105 106
Result: $R^2_{p,A} = 1 - \frac{\sum_{i=1}^{n_B}(y_{iB} - \hat{y}_{iA})^2}{\sum_{i=1}^{n_B}(y_{iB} - \overline{y_B})^2} = 0.9952$ $R^2_{p,B} = 1 - \frac{\sum_{i=1}^{n_B}(y_{iA} - \hat{y}_{iB})^2}{\sum_{i=1}^{n_B}(y_{iA} - \overline{y_A})^2} = 0.9948$

$R^2_{p,A}$ and $R^2_{p,B}$ have a high value, and close enough to each other. It concludes that, the evolved regression model is good.

ABOUT THE AUTHOR

Ying Zhang received a Bachelor's Degree in Chemical Engineering from Beijing Union University, Beijing, China in 1995 and a M.S. in Chemical Engineering from University of South Florida in 2004, and continued her Ph.D. program at the University of South Florida until 2008.