2004

Symbolic regression of thermophysical model using genetic programming

Ying, Zhang
University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd
Part of the American Studies Commons

Scholar Commons Citation

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Symbolic Regression of Thermo-Physical Model Using Genetic Programming

by

Ying Zhang

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Chemical Engineering
Department of Chemical 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.

Date of Approval:
April 6, 2004

Keywords: machine learning, function identification, statistic analysis, modeling, nonlinear regression

© Copyright 2004, Ying Zhang
I would like to thank Dr. Aydin K. Sunol for the opportunity and support he extended to me throughout this work. I would also like to thank Dr. John A. Llewellyn and Dr. Scott W. Campbell for being my committee members and taking time from their busy schedules. Lastly, I would like to thank my family and friends for their love and unconditional help.
# TABLE OF CONTENTS

| LIST OF TABLES | iii |
| LIST OF FIGURES | iv |
| ABSTRACT | vii |
| CHAPTER ONE: INTRODUCTION | 1 |
| CHAPTER TWO: LITERATURE REVIEW | 6 |
| 2.1 An Overview of Data Mining | 6 |
| 2.1.1 Data Pre-processing | 8 |
| 2.1.2 Mining Data Techniques | 9 |
| 2.2 An Overview of Evolutionary Algorithms | 11 |
| 2.2.1 Evolutionary Programming | 12 |
| 2.2.2 Genetic Algorithm | 13 |
| 2.2.3 Genetic Programming | 16 |
| 2.2.3.1 Terminal Set, Function Set and Initial Representation | 19 |
| 2.2.3.2 Genetic Operators | 20 |
| 2.2.3.2.1 The Reproduction and Crossover Operations | 20 |
| 2.2.3.2.2 Mutation | 24 |
| 2.2.3.3 Fitness Measure | 25 |
| 2.2.3.4 Selection Strategy | 26 |
| 2.2.4 Application of Intelligent System in Chemical Engineering | 27 |
| 2.3 An Overview of Optimization Methods and Objective Functions | 30 |
| 2.4 Test of Model Adequacy | 35 |
| 2.4.1 Residuals | 36 |
| 2.4.2 Goodness of Fit Statistics | 37 |
| CHAPTER THREE: A HYBRID SYSTEM FOR STRUCTURAL AND PARAMETRIC OPTIMIZATION | 39 |
| 3.1 System Structure | 39 |
| 3.2 Data and Data Preparation | 40 |
| 3.3 Regression Strategy | 41 |
| 3.4 Implement with MATLAB Genetic Search Toolbox | 43 |
| 3.4.1 Terminal Set and Function Set | 45 |
3.4.2 Parameters for Controlling Runs 45
3.4.3 Selection Strategy 46
3.4.4 Decimation Strategy 47
3.4.5 Result Designation and Termination Criteria 47

CHAPTER FOUR: RESULTS AND DISCUSSIONS 48
  4.1 Results 48
  4.2 Discussions 73

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS 78
  5.1 Conclusions 78
  5.2 Recommendations 79

REFERENCES 81
## LIST OF TABLES

<p>| Table 4.1 | Input and Result Summary for Case I | 48 |
| Table 4.2 | Input and Result Summary for Case II | 51 |
| Table 4.3 | Input and Result Summary for Case III | 53 |
| Table 4.4 | Input and Result Summary for Case IV | 55 |
| Table 4.5 | Input and Result Summary for Case V | 58 |
| Table 4.6 | Input and Result Summary for Case VI | 60 |
| Table 4.7 | Input and Result Summary for Case VII | 63 |
| Table 4.8 | Input and Result Summary for Case VIII | 65 |
| Table 4.9 | Input and Result Summary for Case IX | 68 |
| Table 4.10 | Input and Result Summary for Case X | 70 |
| Table 4.11 | Parameter Comparison between IUPAC and Case III | 76 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Data Mining Procedure</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Flowchart of Genetic Programming</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Functional Representation of $a \cdot x + b$ Using Tree Structure</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Example of Reproduction Operator</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Crossover Operation for Algebraic Equation Manipulation</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of Mutation Operation, (a) Type I &amp; (b) Type II</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>Hybrid System Structure for Structural and Parametric Optimization</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Flowchart for Marquardt Regression</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Schematic Diagram of the Genetic Search Methodology</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Flowchart of Genetic Search</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Heat Capacity $C_p$ Model Fit for Case I</td>
<td>49</td>
</tr>
<tr>
<td>4.2</td>
<td>Residual of Model Testing as a Function of Temperature for Case I</td>
<td>50</td>
</tr>
<tr>
<td>4.3</td>
<td>Residual of Model Prediction as a Function of Temperature for Case I</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Heat Capacity $C_p$ Model Fit for Case II</td>
<td>51</td>
</tr>
<tr>
<td>4.5</td>
<td>Residual of Model Testing as a Function of Temperature for Case II</td>
<td>52</td>
</tr>
<tr>
<td>4.6</td>
<td>Residual of Model Prediction as a Function of Temperature for Case II</td>
<td>53</td>
</tr>
<tr>
<td>4.7</td>
<td>Heat Capacity $C_p$ Model Fit for Case III</td>
<td>54</td>
</tr>
<tr>
<td>4.8</td>
<td>Residual of Model Testing as a Function of Temperature for Case III</td>
<td>54</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>Residual of Model Prediction as a Function of Temperature for Case III</td>
<td>55</td>
</tr>
<tr>
<td>4.10</td>
<td>Heat Capacity $C_p$ Model Fit for Case IV</td>
<td>56</td>
</tr>
<tr>
<td>4.11</td>
<td>Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case IV</td>
<td>57</td>
</tr>
<tr>
<td>4.12</td>
<td>Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case IV</td>
<td>58</td>
</tr>
<tr>
<td>4.13</td>
<td>Heat Capacity $C_p$ Model Fit for Case V</td>
<td>59</td>
</tr>
<tr>
<td>4.14</td>
<td>Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case V</td>
<td>59</td>
</tr>
<tr>
<td>4.15</td>
<td>Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case V</td>
<td>60</td>
</tr>
<tr>
<td>4.16</td>
<td>Heat Capacity $C_p$ Model Fit for Case VI</td>
<td>61</td>
</tr>
<tr>
<td>4.17</td>
<td>Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VI</td>
<td>62</td>
</tr>
<tr>
<td>4.18</td>
<td>Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case VI</td>
<td>62</td>
</tr>
<tr>
<td>4.19</td>
<td>Heat Capacity $C_p$ Model Fit for Case VII</td>
<td>64</td>
</tr>
<tr>
<td>4.20</td>
<td>Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VII</td>
<td>64</td>
</tr>
<tr>
<td>4.21</td>
<td>Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case VII</td>
<td>65</td>
</tr>
<tr>
<td>4.22</td>
<td>Heat Capacity $C_p$ Model Fit for Case VIII</td>
<td>66</td>
</tr>
<tr>
<td>4.23</td>
<td>Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VIII</td>
<td>67</td>
</tr>
<tr>
<td>4.24</td>
<td>Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case VIII</td>
<td>67</td>
</tr>
<tr>
<td>4.25</td>
<td>Heat Capacity $C_p$ Model Fit for Case IX</td>
<td>69</td>
</tr>
</tbody>
</table>
Figure 4.26  Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case IX  

Figure 4.27  Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case IX  

Figure 4.28  Heat Capacity $C_p$ Model Fit for Case X  

Figure 4.29  Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case X  

Figure 4.30  Residual Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case X
SYMBOLIC REGRESSION OF THERMO-PHYSICAL MODEL USING GENETIC PROGRAMMING

Ying Zhang

ABSTRACT

The symbolic regression problem is to find a function, in symbolic form, that fits a given data set. Symbolic regression provides a means for function identification. This research describes an adaptive hybrid system for symbolic function identification of thermo-physical model that combines the genetic programming and a modified Marquardt nonlinear regression algorithm.

Genetic Programming (GP) system can extract knowledge from the data in the form of symbolic expressions, i.e. a tree structure, that are used to model and derive equation of state, mixing rules and phase behavior from the experimental data (properties estimation).

During the automatic evolution process of GP, the function structure of generated individual module could be highly complicated. To ensure the convergence of the regression, a modified Marquardt regression algorithm is used. Two stop criteria are attached to the traditional Marquardt algorithm to enforce the algorithm repeat the regression process before it stops.

Statistic analysis is applied to the fitted model. Residual plot is used to test the goodness of fit. The $\chi^2$-test is used to test the model’s adequacy.
Ten experiments are run with different form of input variables, number of data points, standard errors added to data set, and fitness functions. The results show that the system is able to find models and optimize for its parameters successfully.
CHAPTER ONE
INTRODUCTION

The approaches for modeling of chemical processes have undergone significant changes in the last three decades. These approaches are mainly divided into two generic categories. One is mechanistic modeling, where substantial amount of first principles knowledge is used as the basis. The other one is empirical modeling, i.e. data driven modeling, where the model structure and parameters are selected so as to represent the process data accurately.

Data driven modeling techniques have been popular for many decades. They are more cost effective compared to the development of mechanistic models, especially when first principles are not sufficient to represent real world problems. Furthermore, these mechanistic models are highly non-linear and complex, which makes them difficult to identify [Ramirez 1989]. Currently, the majority of data driven modeling methods can be further categorized under two headings [Pöyhönen 1996]: statistical methods and artificial neural networks. Neural networks usually provide models that are accurate in representing the data, but they don't give any insight into the structure of the process. They are commonly classified black box, and one can not abstract the underlying physical relationships between input and output data. It is often desirable to gain some insight into the underlying process structures, as well as make accurate numeric predictions.
Therefore, the objective of this work is to produce input-output models with relatively simple, transparent structures using a novel approach which offers a useful alternative to popular data based modeling methodologies.

A primary classification used for property and process models in chemical engineering is algebraic versus differential equation models [Franks 1967]. The mathematical models are either comprised of a set of algebraic equations for steady-state properties, or by a set of ordinary differential equations (ODE) coupled with algebraic equation for dynamic models, or partial differential equations (PDE) for distributed models. The majority of mathematical models for physical or engineered systems can be classified in one of these three categories.

Algebraic equation models may cover many cases, such as [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 single pressure-explicit equation of state (EOS) which is employed for the calculation of fluid phase equilibrium and thermo-physical properties required in the design of processes involving non-ideal fluid mixtures. Mathematically, a pressure-explicit EOS expresses the relationship among pressure, volume, temperature, and composition for a fluid mixture.
• Type III: The model with multiple dependent variables and several independent variables. For example, in thermo-physical models, the derived thermo-physical properties, such as Enthalpy (H) and entropy (S), are coupled with pressure-explicit EOS. Another example is chemical kinetics models where multiple reactions are coupled through concentration of species.

The aim of this work is to develop a methodology, which uses genetic operations in order to find a symbolic relationship between a single response and single independent variable, Type I, that describes the relationship between input and output data. 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. This new technique is called symbolic regression, which is seeking a mathematical expression, in symbolic form, which fits or approximates a given sample of data.

Symbolic regression differs from conventional regression method in the way that the model is specified in advance by the user. In conventional regression, a given set of values of various independent variable(s), model and the corresponding values for the dependent variable(s). One finds the best set of parameters that represent the data. The model is selected a priori, and its adequacy is tested through post analysis that allows model discrimination.

Genetic Programming allows optimization of a tree structure or function structure. This tree structure is of variable length and is constructed of nodes. The terminal nodes
can be input variables, parameters or constants. The non-terminal nodes are standard library functions, like +, -, *, /.

Each structure is one possible description of the 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 the branches from two parent structures being interchanged. Mutation means the creation of a completely new branch determined at random. 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 until some convergence criterion is satisfied.

Genetic Programming system can extract knowledge from the data in the form of symbolic expressions, i.e. a tree structure, that are used to model and derive equation of state, mixing rules and phase behavior from the experimental data (properties estimation). Symbolic regression is the process of discovering both the functional form of a target function and all of its necessary coefficients, or at least an approximation to these. This is distinct from other forms of regression such as polynomial of a pre-specified order. Thus, with GP method, symbolic regression has the advantage that no a priori modeling assumptions have to be made for the unknown system.

The two tier approach proposed in this research also combines the evolutionary search of symbolic regression using genetic programming (GP) with the parameter optimization of numerical regression method Marquardt. The idea of the algorithm is to
embed Marquardt method into GP, where GP is employed to optimize the structure of a model, while Marquardt is employed to optimize its parameter. In other words, the approach mainly consists of two steps: one is the evolutionary modeling process used to optimize the structure of models based on GP, and the other is the parameter optimization process used to optimize the parameters of a model based on Marquardt. The number of parameters is unknown, and determined for each step within the algorithm.

Chapter 2 gives an overview on data mining, statistic analysis, nonlinear regression method, evolutionary algorithms, specifically, genetic programming and the application of intelligent system in chemical engineering. Chapter 3 describes the system structure and the implement procedure. Results are demonstrated in Chapter 4. Finally, conclusion and recommendations are presented in Chapter 5.
CHAPTER TWO
LITERATURE REVIEW

This chapter is divided into four sections. The first section begins with an overview on data mining. The second section describes the development of evolutionary algorithms and comparison of different evolutionary algorithms. More emphasis is given to genetic programming. At the end of this section, a brief summary of applications of intelligent system in chemical engineering is given. In the third section, some popular optimization methods and pertinent objective functions (criteria) are described. In the last section of this chapter, the statistic analysis for model identification and discrimination is given.

2.1 An Overview of Data Mining

Data mining is the analysis of data sets to find new and novel relationships. They are useful in consolidation and summary of the data in an effort to enable their understandable and useful representation. Based on different objectives, it can be categorized into the following types of tasks [Hand et al. 2001]:

- Exploratory data Analysis (EDA)—the goal here is simply to explore the data without any clear ideas of what we are looking for. Typically, EDA techniques are interactive and visual, and there are many effective
graphical display methods for relatively small, low-dimensional data sets. As the dimensionality increases, it becomes much more difficult to visualize the cloud of points in n-space. So, the projection techniques that produce informative low-dimensional projections of the data are needed.

- **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 segmentation), and models describing the relationship between variables (dependency modeling).

- **Predictive modeling: Classification and Regression**—The aim here is to build a model that will permit the value of one variable to be predicted from the known values of other variables. In classification, the variable being predicted is categorical, while in regression the variable is quantitative. The key distinction between prediction and description is that prediction has as its objective a unique variable, while in descriptive problems no single variable is central to the model.

- **Discovering Patterns and Rules**—the three types of tasks listed above are concerned with model building. Other data mining applications are concerned with pattern detection.
• Retrieval by Content—here the user has a pattern of interest and wishes to find similar patterns in the data set. This task is most often used for test and image data sets.

Data mining is the process of seeking relationships within a data set—of seeking accurate, convenient, and useful summary representations of some aspect of the data. It involves a number of steps [Thuraisingham 1999]. For the sake of simplicity, these steps can be roughly categorized into: data pre-processing, mining data, and post-processing.

The simplified flowchart is shown in Fig 2.1.

![Data Mining Procedure Flowchart]

**Figure 2.1 Data Mining Procedure**

### 2.1.1 Data Pre-processing

Generally, the data to be mined may be characterized as the huge volume, incomplete or imprecise, noisy, 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 (like prediction). 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.

Data pre-processing includes the following steps [Freitas 2002]:

- Data integration—this is necessary if the data to be mined comes from several different sources. This step involves, for instance, removing inconsistencies in attribute names or attribute value names between data sets of different sources.
- 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.
- Attribute selection (select the variable)—this step consists of selecting a subset of attributes (variables) relevant for mining the data, among all original attributes.

2.1.2 Mining Data Techniques

Data mining techniques are numerous. These include Bayesian method, machine learning, neural network, evolutionary techniques and others [Thuraisingham 1999].

One popular class of data mining techniques has come to be called market basket analysis. These are techniques that group items together, for example, which items go
together, who travels with whom together, etc. The actual techniques employed to do
market basket analysis are intelligent searching and pruning the search. Many of the
intelligent search techniques that were developed for artificial intelligence are being
employed for market basket analysis.

Another data mining technique is decision trees. This is a machine learning
technique and is used extensively for classification. Data are divided into groups based on
some attribute value. Subsequently, a tree structure is formed with leaves at the end; the
decision tree is then used for training. Subsequently, as new data appears to be analyzed,
the training examples are used to classify the data.

Neural networks have been found to be useful because of their learning and
generalization abilities. Neural networks are another popular data mining technique that
has been around for while. A neural network is essentially a collection of inputs signals,
nodes, and output signals. These systems are first trained with training sets and examples.
Once the learning is over, new patterns are given to the network. The network then uses
its training experience to analyze the new data. It may be used for clustering, identifying
entities, deviation analysis and various other data mining tasks.

Inductive logic programming is a machine learning technique. It originated from
logic programming. Instead of deducing new data from existing data and rules, inductive
logic programming is all about inducing rules from analyzing data. The theory is well
developed. Inductive logic programming uses a variation of the resolution principle in
theorem proving for discovering rules.
Several other data mining techniques are in use today. They include automatic cluster detection techniques, which are a collection of techniques to find clusters; nearest neighbor techniques, which are a collection of techniques that analyzes new data based on its neighbors. Evolutionary techniques are an essential tool for structural and parametric optimization. The three important evolutionary algorithms are: evolutionary programming [Fogel et al. 1966], genetic algorithm [Holland 1975] and genetic programming [Koza 1992]. The detail of Evolutionary Computing will be given in next section.

2.2 An Overview of Evolutionary Algorithms

Recently, due to the merits of self-adaptation, self-organization, self-learning and generality, evolutionary algorithms (EAs) have been successfully applied in a wide range of economic, engineering and scientific computations [Goldberg, 1989; Mitchell, 1996]. EAs are adaptive methods for solving computational problems in many fields, which mimic the process of biological evolution and the mechanisms of natural selection and genetic variation. They use suitable codings to represent possible solutions to a problem, and guide the search by using some genetic operators and the principle of “survival of the fittest”. EAs originally consist of two notable branches, namely evolutionary programming (EP) [Fogel et al. 1966] 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 computer programs of varying sizes and shapes.
2.2.1 Evolutionary Programming

Evolutionary Programming, originally conceived by Lawrence J. Fogel in 1960’s, is a stochastic optimization strategy similar to genetic algorithm, but instead places emphasis on the behavioral linkage between parents and their off-spring, rather than seeking to emulate specific genetic operators as observed in nature.

For EP, like GAs, there is an underlying assumption that a fitness landscape can be characterized in terms of variables, and that there is an optimum solution (or multiple such optima) in terms of those variables.

The basic EP method involves 3 steps (Repeat until a threshold for iteration is exceeded or an adequate solution is obtained):

- Choose an initial population of trial solutions at random. The number of solutions in a population is highly relevant to the speed of optimization, but no definite answers are available as to how many solutions are appropriate and how many solutions are just wasteful.

- Each solution is replicated into a new population. Each of these off-spring solutions are mutated according to a distribution of mutation types, ranging from minor to extreme with a continuum of mutation types between. The severity of mutation is judged on the basis of the functional change imposed on the parents.
Each off-spring solution is assessed by computing its fitness. Typically, a stochastic tournament is held to determine N solutions to be retained for the population of solutions, although this is occasionally performed deterministically. There is no requirement that the population size be held constant or that only a single off-spring be generated from each parent.

There are two important ways in which EP differs from GAs.

First, there is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the genome. In EP, the representation follows from the problem.

Second, the mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behavior of the off-spring as highly probable and substantial variations as increasingly unlikely. Further, the severity of mutations is often reduced as the global optimum is approached.

2.2.2 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics.

GAs simulate the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA. Each individual represents a point in a search space and a possible solution. The individuals in the
population are then made to go through a process of evolution. GAs are based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals using the following foundations:

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
- Genes from 'good' individuals propagate throughout the population so that two good parents will sometimes produce off-springs that are better than either parent.
- Thus, each successive generation will become more suited to their environment.

A population of individuals is maintained within search space for a GA, each representing a possible solution to a given problem. Each individual is coded as a fixed-length vector of components, or variables, in terms of some alphabet, usually the binary alphabet \{0,1\}. Although a substantial amount of research have been performed on variable-length strings and other structures. The majority of work with genetic algorithms is focused on fixed-length ones. To continue the genetic analogy, these individuals are analogous to chromosomes and the variables are analogous to genes. Thus, a chromosome (solution) is composed of several genes (variables). A fitness score is assigned to each solution representing the abilities of an individual to 'compete'. The
individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective ‘breeding’ of the solutions to produce ‘off-spring’ better than the parents by combining information from the chromosomes.

The GA maintains a population of n chromosomes (solutions) with associated fitness values. Parents are selected to mate, on the basis of their fitness, producing off-spring via a reproductive plan. Consequently, highly fit solutions are given more opportunities to reproduce, so that off-spring inherits characteristics from each parent. As parents mate and produce off-spring, room must be made for the new arrivals since the population is kept at a static size. Individuals in the population die and are replaced by the new solutions, eventually, the algorithm creates a new generation once all mating opportunities in the old population have been exhausted. In this way, it is hoped that over successive generations better solutions will thrive while the least fit solutions die out.

New generations of solutions are produced containing, on average, more of the good genes than of a typical solution from previous generations. Each successive generation will contain more good ‘partial solutions’ than previous generations. Eventually, once the population has converged and is not producing off-spring noticeably different from those in previous generations, the algorithm itself is said to have converged to a set of solutions to the problem at hand.
2.2.3 Genetic Programming

Genetic programming (GP) is an extension of the genetic algorithm in which the genetic population consists of computer programs (that is, compositions of primitive functions and terminals).

GP differs from GAs by utilizing the following:

- Tree structure, variable length chromosomes rather than GAs’ chromosomes of fixed length and structure. The output of the genetic algorithm is a quantity, while the output of the genetic programming is another computer program.

- Chromosomes coded in a problem specific fashion that can usually be executed in their current form, rather than binary strings.

Koza [1992] demonstrated a surprising and counter-intuitive result, namely that computers can be programmed by means of natural selection. Specifically, genetic programming is capable of evolving a computer program for solving, or approximately solving, a surprising variety of problems from a wide variety of fields. To accomplish this, genetic programming starts with a pool of randomly generated computer programs composed of available programmatic ingredients 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 computer programs to find a program that solves, or approximately solves, a problem.
Genetic programming is a domain independent method that genetically breeds populations of computer programs to solve problems by executing the following three steps:

- Generate an initial population of random computer programs composed of the primitive functions and terminals of the problem.
- Iteratively perform the following sub-steps until the termination criterion has been satisfied:
  - Execute each program in the population and assign it a fitness value according to how well it solves the problem.
  - Create a new population of programs by applying the following three primary operations. The operations are applied to program(s) in the population selected with a probability based on fitness (i.e., the fitter the program, the more likely it is to be selected).
    - Reproduction: Copy an existing program to the new population.
    - Crossover: Create two new off-spring programs for the new population by genetically recombining randomly chosen parts of two existing programs. The genetic crossover (sexual recombination) operation (described below) operates on two parental computer programs and produces two off-spring programs using parts of each parent.
    - Mutation: randomly alteration in existing programs, and produces one off-spring programs.
• The single best computer program in the population produced during the run is designated as the result of the run of genetic programming. This result may be solution (or approximate solution) to the problem.

A flowchart showing the above steps is given in Fig 2.2.

![Flowchart of Genetic Programming](image)

**Figure 2.2 Flowchart of Genetic Programming**

The use of automatically defined functions in genetic programming is discussed in Koza [1992, 1994, 1999]. Advances in genetic programming are described in Kinnear
The numerous recent applications of genetic programming were illustrated in Quagliarella [1998] and Miettinen [1999].

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

### 2.2.3.1 Terminal Set, Function Set and Initial Representation

The terminal and function sets are important components of genetic programming. The terminal and function sets are the alphabet of the programs to be made. The terminal set consists of the variables and constants of the programs. The functions are several mathematical functions, such as addition, subtraction, division, multiplication and other more complex functions.

Any computer program can be depicted graphically as a rooted point-labeled tree in a plane whose internal points are labeled with functions, whose external points (leaves) are labeled with terminals, and whose root is labeled with the function appearing just inside the outermost left parenthesis. The tree corresponding to the computer program for the equation $a \cdot x + b$ is shown in Fig 2.3.

![Figure 2.3 Functional Representation of $a \cdot x + b$ Using Tree Structure](image-url)
In this graphical depiction, the two internal points of the tree are labeled with functions (+ and *). The three external points (leaves) of the tree are labeled with terminals, where \( a \) and \( b \) are constant, and \( x \) is the independent variable.

The closure property of the function set and terminal set requires that each of the functions in the function set be able to accept, as its arguments, any value and data type that may possibly be returned by any function in the function set and any value and data type that may possibly be assumed by any terminal in the terminal set. That is, each function in the function set should be well defined and closed for any combination of arguments that it may encounter.

The sufficiency property requires that the set of terminals and the set of primitive functions be capable of expressing a solution to the problem.

### 2.2.3.2 Genetic Operators

In this sub-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.

#### 2.2.3.2.1 The Reproduction and Crossover Operations

The two primary genetic operations for modifying the structures undergoing adaptation are Darwinian fitness proportionate reproduction (Fig 2.4) and crossover (Fig 2.5). They are described below.
The operation of fitness proportionate reproduction for the genetic programming paradigm is the basic engine of Darwinian reproduction and survival of the fittest. It is an asexual operation in that it operates on only one parental program. The result of this operation is one off-spring program. 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

\[
\frac{f(i,t)}{\sum_{j=1}^{M} f(j,t)} \tag{2.1}
\]

Note that 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. It improves the average fitness of the population (at the expense of the genetic diversity of the population). To the extent that it increases the number of occurrences of more fit individuals and decreases the number of occurrences of less fit individuals.

Parent: \( a \cdot x + b \)  
Off-spring: \( a \cdot x + b \)

![Figure 2.4 Example of Reproduction Operator](image)
The crossover (recombination) operation for the genetic programming paradigm is a sexual operation that starts with two parental programs. Both parents are selected from the population with a probability equal to its normalized fitness. The result of the crossover operation is two off-spring programs. Unlike fitness proportionate reproduction, the crossover operation creates new individuals in the populations.

The operation begins by randomly and independently selecting one point in each parent using a specified probability distribution (discussed below). Note that the number of points in two parents typically is not equal to each other. As will be seen, the crossover operation is well-defined for any two programs. That is, for any two programs and any two crossover points, the resulting off-springs are always valid computer programs. Off-spring contains some traits from each 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 off-spring is produced by deleting the crossover fragment of the first parent from the first parent and then impregnating the crossover fragment of the second parent at the crossover point of the first parent. In producing this first off-spring, the first parent acts as the base parent (the female parent) and the second parent acts as the impregnating parent (the male parent). The second off-spring is produced in a symmetric manner. Since entire sub-trees are swapped, this genetic crossover (recombination) operation produces syntactically and semantically valid computer programs as off-spring regardless of which point is selected in either parent.
For example, consider the parental computer program $a \cdot x + b$ and $x^2 + a \cdot b$.

These two computer programs can be depicted graphically as rooted, point-labeled trees with ordered branches.

The two parental computer programs are shown in Figure 2.5. Suppose that the crossover points are randomly selected for each parent individual as shown in Figure 3.3. The crossover points are therefore the * in the first parent and the + in the second parent.

![Figure 2.5 Crossover Operation for Algebraic Equation Manipulation](image)

Off-spring1: $a \cdot x^2 + b$

Off-spring2: $x + a \cdot b$
The places from which the crossover fragments were removed are identified with dash line.

### 2.2.3.2.2 Mutation

Mutation is another important feature of genetic programming. Two types of mutations are possible. In the first kind, a function can only replace a function or a terminal can only replace a terminal. In the second kind, an entire sub-tree can replace another sub-tree. Fig 2.6a & b explains the concept of mutation:

![Mutation Diagram](image)

**Figure 2.6 Example of Mutation Operation, (a) Type I & (b) Type II**
2.2.3.3 The Fitness Measure

The most difficult and most important concept of genetic programming is the fitness function. The fitness function determines how well a program is able to solve the problem. Each individual in a population is assigned a fitness value as a result of its interaction with the environment. Fitness is the driving force of Darwinian natural selection and genetic algorithms.

Fitness cases provide a basis for evaluating a particular program.

The raw fitness of any computer program is the sum, over the fitness cases, of the squares of the distances (taken over all the fitness cases) between the point in the solution space (which is real-valued) returned by the individual program for a given set of arguments and the correct point in the solution space. In particular, the raw fitness \( r(i,t) \) of an individual computer program \( i \) in the population of size \( M \) at any generation \( t \) is

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

(2.2)

where \( S(i,j) \) is the value returned by program \( i \) 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.

Each raw fitness value is then adjusted (scaled) to produce an adjusted fitness measure \( a(i,t) \). The adjusted fitness value is

\[
a(i,t) = \frac{1}{(1 + r(i,t))}
\]

(2.3)

where \( r(i,t) \) is the raw fitness for individual \( i \) 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

\[
n(i,t) = \frac{a(i,t)}{\sum_{j=1}^{M} a(j,t)}
\]  

(2.4)

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.

In a genetic search, each member of a population needs to be evaluated and assigned a fitness value. Obviously, the minimization problem is applied in the whole thesis.

In this research, the raw fitness value is used.

2.2.3.4 Selection Strategy

There are many different selection methods based on fitness. The most popular is fitness-proportionate selection, which is the method used throughout this research. 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:
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. When two bulls fight over the right to mate with a given cow, tournament selection is occurring.

### 2.2.4 Applications of Intelligent System in Chemical Engineering

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

- Process design: designing processes, selecting and estimating physical and thermodynamic properties [Friese 1998], developing flowsheets,
specifying equipment, ensuring process safety, assessing process flexibility, and estimating cost.

- Fault diagnosis: process troubleshooting, i.e., determining the origins of process problems and recommending solutions [Frank 1997, 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].


- 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.

The goal of process design is to configure an industrial chemical process that achieves technical and economical objectives. Intelligent system can be applied to process design in a number of areas. In general, they can be used to [Quantrille et al. 1991]:

28
Select thermodynamic models and estimate physical properties: select the best thermodynamic model(s) for the problem [Fredenslund 1980, Banares-Alcantara et al. 1985, Gani 1989]. If no explicit models are available, the system could select a method and estimate the value of the physical property.

Develop flowsheets: utilize engineering knowledge to synthesize the process [Bamicki 1990, Csukas 1998].

Select methods: use process data with engineering heuristics to recommend the optimal processing method for the task at hand [Bamicki 1990].

Automate the design process: link process synthesis with method selection and rigorous CAD programs [Stephanopoulos 1987].

Select materials: determine the appropriate materials of construction based on process and environmental conditions [King 1986].

The foundation of accurate chemical process design and simulation is a correct estimate of physical and thermodynamic properties. Volumetric Equation of State (EOS) and Heat Capacity Data ($C_p$) are the minimum amount of information needed to evaluate thermodynamic properties of real substance accompanying a change of state, i.e. compute the changes in substance’s internal energy, enthalpy, and entropy for any change of state. 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.

2.3 An Overview of Optimization Methods and Objective Functions

Optimization techniques are used to find a set of design variables that can in some
way be defined as optimal. In parameter estimation, the optimum parameter value will be
searched by a selected optimization method, to minimize or maximize a well-defined
objective function that is dependent on parameter.

The widely used objective function in parameter regression is least square
estimation.

For \( Y = f(x, b) \), which is a relation that is nonlinear with respect to the
parameters. The sum of squared residuals is:

\[
\Phi = \varepsilon' \varepsilon = (Y^* - Y)'(Y^* - Y)
\]

(2.6)

where \( Y^* \) is vector of experimental observations of the dependent variable.

Gauss 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 \( b \):

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

(2.7)

where the Taylor series has been truncated after the second term. Eq. (2.7) is linear in \( \Delta b \).
Therefore, the problem has been transformed from finding $b$ to that of finding the correction to $b$, that is $\Delta b$, which must be added to an estimate of $b$ to minimize the sum of squared residuals.

$$
J = \begin{bmatrix}
\frac{\partial Y_1}{\partial b_1} & \cdots & \frac{\partial Y_1}{\partial b_k} \\
\vdots & \ddots & \vdots \\
\frac{\partial Y_n}{\partial b_1} & \cdots & \frac{\partial Y_n}{\partial b_k}
\end{bmatrix},
$$

(2.8)

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

(2.9)

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

(2.10)

where $m$ is the iteration counter.

In Gauss method, the drawback is the fact that the incremental changes, namely the $\Delta b$ s as described previously, can be estimated very poorly due to computation of the partial derivative matrix $(J'J)^{-1}$ when it’s 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 $\Delta b$ s, and then the procedure will move in the wrong direction. Then, 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 convergibility is greater for a close-to-linear model than a far-from-linear one. Since this objective function is a quadratic one in nature, so, Gauss method is susceptible.
The idea for steepest descent method is that the gradient of a scalar function is a vector that gives the direction of the greatest decrease of the function at any point to reach a lower function value. Therefore, in this method, the initial vector of parameter estimated is corrected in the direction of the negative gradient of $\Phi$

$$\Delta b = -K \left( \frac{\partial \Phi}{\partial b} \right)$$  \hspace{1cm} (2.11)

where $\Phi$ is the sum of squared residuals, and $K$ is a suitable constant factor and $\Delta b$ 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:

$$b^{m+1} = b^m + \Delta b$$ \hspace{1cm} (2.12)

where $m$ is the iteration counter.

Then, $\Delta b$ can be calculated from:

$$\Delta b = 2KJ'(Y' - Y)$$ \hspace{1cm} (2.13)

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 method.

The steepest descent method has the advantage that guarantees moving 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 an interpolation technique between the Gauss and the steepest descent methods. This interpolation is achieved by adding the diagonal matrix $(\lambda I)$ to the matrix $(J'J)$ in the function of $\Delta b$ in Gauss method above:

$$\Delta b = (J'J + \lambda I)^{-1} J'(Y^* - Y) \quad (2.14)$$

The value of $\lambda$ 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 $\Delta b$ equation above, if $\lambda$ is small, then, Marquardt method approaches the Gauss method; when $\lambda$ is very large, 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 $\lambda$, an indicator of compromising between Gauss 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].

However, all of these methods are local methods that providing global minimum only for convex cases.

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 $\mathbf{e}$ for an experiment is a random vector following a probability density function of specified form $p_i(\mathbf{e},b)$, where $b$ is the unknown vector of parameters. The experimental outcome of a $\mathbf{e}_i$ vector is regarded as a random sample out of the distribution defined by $p_i$. The combination of all $p_i$ for all $\mathbf{e}_i$ results in the Likelihood Function:

$$L(b;\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3,...)$$

which, for correct specification of the joint probability density function for all $\mathbf{e}$’s and known true $b$ values, represents the probability density of getting just that set of $\mathbf{e}_i$ vectors obtained experimentally.

In the parameter estimation situation, $b$ is not known. So in Maximum Likelihood Estimation, those unknown values are searched by 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.

Marquardt optimization method with LS as the objective function is well developed in Matlab code, which is used throughout this research.

2.4 Test of Model Adequacy

After fitting data with one or more models, you should evaluate the goodness of fit. A visual examination of the fitted curve displayed should be the first step. Beyond that, some measures on goodness of fit are provided [Englezos et al. 2001]:

35
- Residuals

- Goodness of fit statistics

The measures can be grouped into two types, graphical and numerical. The residuals and prediction bounds are graphical measures, while the goodness of fit statistics and confidence bounds are numerical measures.

2.4.1 Residuals

All of the information on lack of fit is contained in the residuals. The residuals from a fitted model are defined as the differences between the response data and the fit to the response data at each predicted value.

\[ \text{Residual}_i = \text{data}_i - \text{fit}_i \]

Mathematically, the residual for a specific predicted value is the difference between the response value \( y \) and the predicted response value, \( \hat{y} \).

\[ r = y - \hat{y} \quad (2.15) \]

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.
2.4.2 Goodness of Fit Statistics

There is a plethora of model adequacy tests that can be employed to decide whether the assumed mathematical model is indeed adequate. Generally speaking, these tests are based on the comparison of the experimental error variance for the model to that obtained experimentally or through other means.

For single response model, two cases have to be considered. One is that we know precisely the value of the experimental error variance and the other when we have an estimate of it.

Since our data is from the IUPAC CO$_2$ data, we may assume that we know the experimental error variance precisely and introduce noise ourself to test the model in ceratin instances.

To test whether the model is adequate, we simply need to test the hypothesis

$H_0$: $\sigma^2_{mdl} = \sigma^2_e$

$H_1$: $\sigma^2_{mdl} > \sigma^2_e$

at any desirable level of significance, i.e. $\alpha = 0.05$. Here, $\sigma^2_{mdl}$ is the error variance of the model equations that is estimated to be $\sigma^2_e$. Since $\sigma^2_e$ is known exactly (i.e., there is no uncertainty in its value, it’s a given number) the above hypothesis test is done through a $\chi^2$-test.
Namely, if 
\[ \chi^2_{data} > \chi^2_{\nu=(N_m-p),1-\alpha} \Rightarrow \text{Reject } H_0, \] 
where 
\[ \chi^2_{data} = (N_m - p) \frac{\hat{\sigma}^2}{\sigma^2_e} = \frac{\sum (y - \hat{y})^2}{\sigma^2_e} \] 
(2.16)

and \( \chi^2_{\nu=(N_m-p),1-\alpha} \) is obtained from the tables of the \( \chi^2 \)-distribution with degrees of freedom \( \nu = (N_m - p) \). \( N_m \) is the total number of measurements, \( p \) is the number of parameters.
CHAPTER THREE

A HYBRID SYSTEM FOR STRUCTURAL AND PARAMETRIC OPTIMIZATION

This chapter has four sections. In the first section, the structure of hybrid system is given. The data used throughout this research is given in the second section. The regression strategy is given in the third section. The fourth section describes the features of MATLAB genetic search toolbox, and the setup of algorithm controlling parameters.

3.1 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 method to solve the function identification problems is to combine a symbolic computing method (Genetic Programming) and a numeric computing method (the Levenberg-Marquardt nonlinear regression algorithm). The two-layer structure is shown in Fig. 3.1. The Marquardt nonlinear regression is embedded in the genetic programming as an inner layer. The machine function identification system searches the space of function models, dynamically creates new generation of function models using genetic programming, and optimizes the coefficients of the function models using the Levenberg-Marquardt nonlinear regression algorithm,
and tries to make the function models “best” fit the given sample data points. The complete procedure ends with a statistical analysis which is used to test model’s adequacy.

Outer layer:

![Diagram of data processing flow](image)

Inner layer:

![Diagram of parameter regression](image)

**Figure 3.1  Hybrid System Structure for Structural and Parametric Optimization**

### 3.2 Data and Data Preparation

For real gas, Heat Capacity $C_p$ is a function of $T$ and $P$. By given volumetric equation-of-state, $C_p$ for real gas can be calculated by Eq. (3.1)

$$C_p (P,T) = C_{p}^{id} (T) - T \int_{P=0,T}^{P,T} \left( \frac{\partial^2 V}{\partial T^2} \right) dP$$  \hspace{1cm} (3.1)

where $C_{p}^{id}$ is ideal gas heat capacity that is the function of $T$ only.

Since this study is to explore the application of Genetic programming in developing ideal gas heat capacity model. The data is from IUPAC, a set of well modeled...
data, so, we assume that, the data is free of outliers and missing value, and the data set is ready for modeling without the pre-process step.

IUPAC model is as follows:

\[ C^{\text{II}}_p / R = \sum_{n=0}^{3} \gamma_n \tau^n \]  
(3.2)

where, \( \tau = 304.2K / T \)

\[ \gamma_0 = 0.769441246 \times 10^1 \]

\[ \gamma_1 = -0.249610766 \times 10^0 \]

\[ \gamma_2 = -0.254000397 \times 10^2 \]

\[ \gamma_3 = 0.651102201 \times 10^2 \]

\[ \gamma_4 = -0.820863624 \times 10^2 \]

\[ \gamma_5 = 0.574148450 \times 10^2 \]

\[ \gamma_6 = -0.212184243 \times 10^2 \]

\[ \gamma_7 = 0.323362153 \times 10^1 \]

### 3.3 Regression Strategy

The evolution of the model is an automatic process. The individual modules 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. A bad initial guesses for parameter value may also cause the failure to converge.
To ensure convergence of parameter regression robustly and more precisely, an automated re-start operation and two stop criteria are implemented, which enforce the algorithm repeat the regression process before it stops. The automated re-start operation is that the parameter value got at the current iteration will be automatically assigned as the initial guesses for the next iteration. Two stop criteria are: the relative change of sum of the least square is less than a setup value, or the number of iterations exceeds a pre-defined number. The flowchart is shown as Fig 3.2:

Figure 3.2 Flowchart for Marquardt Regression

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.
### 3.4 Implementation with MATLAB Genetic Search Toolbox

The Genetic Search Toolbox provides an integrated environment for performing all aspects of a genetic search, including a collection of operations found to be the most useful in implementing genetic search methods in common applications.

A schematic representation of the genetic search process is shown in Fig 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.

![Figure 3.3 Schematic Diagram of the Genetic Search Methodology](image)

The genetic search toolbox can implement the genetic search process in the following manners. One or two new off-springs are created with crossover or mutation each generation. In the second, many new off-springs are created each generation; one generation may represent almost complete population turnover (some members may be retained unmodified through the "reproduction" loop). This second approach is one that is
adopted in much published literature. Since sequential computations are used in digital computers, however, the second approach is wasteful. Thus, in this thesis, the term "generation" will refer to a single iteration of creating one or two new off-spring, as shown in Fig 3.4.

**Figure 3.4 Flowchart of Genetic Search**
3.4.1 Terminal Set and Function Set

At the heart of every genetic search is the population. The Genetic Search Toolbox can handle any number of separate populations. The populations can contain three types of chromosomes: algebraic, space-separated, and binary strings. Algebraic chromosomes are algebraic expressions, which will be applied in this problem. The chromosomes can contain MATLAB functions or other user-defined functions written in the MATLAB environment; interpretation of the chromosome is problem-specific and is up to the user.

The genetic programming module starts with a set of primitive functions like +, -, *, /, %, exp, square root (sqrt), log… a terminal set consisting of temperature, pressure etc. It performs symbolic regression on the experimental data to extract functional representation of the data.

Based on the pre-knowledge on heat capacity $C_p$, we believe that some composition of the functions and terminals supplied here can yield a solution to the problem.

3.4.2 Parameters for Controlling Runs

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. Other minor numerical parameters include the probability of crossover, reproduction and mutation.

The population size is 500. The maximum number of generations, $G$, to be run is 51 (i.e., an initial random population, called generation 0, and 50 additional generations).
Crossover is performed on 80% of the population. That is, if the population size is 500, then 200 pairs of individuals from each generation are selected (with reselection allowed) from the population with a probability equal to their normalized adjusted fitness. In addition, fitness proportionate reproduction is performed on 20% of the population on each generation. That is, 100 individuals from each generation are selected (with reselection allowed) from the population with a probability equal to their normalized adjusted fitness. Note that the parents remain in the population and can often repeatedly participate in other operations during the current generation. Several minor parameters are used to control the computer implementation of the algorithm as described in Koza [1992].

The toolbox provides several genetic operations with which to create new members of the population, algebraic chromosomes can undergo crossover or mutation. The crossover operations produce two off-spring chromosomes, and the mutation produce one. The new chromosomes are automatically added to the population.

3.4.3 Selection Strategy

In a genetic search, each member of a population needs to be evaluated and assigned a fitness value. In the toolbox environment, this fitness can be any function that can be calculated within MATLAB. It can be nonlinear, discontinuous, or non-smooth. The fitness can be either maximized or minimized during a search. Obviously, the minimization problem is applied in the whole thesis. At the end of the each GP run, various population fitness statistics (such as the mean and variance of the fitness) is computed.
Members of a population can be selected for examination or for genetic operations through a variety of criteria. In general, members with the highest or lowest fitness can be selected. Members can be selected randomly. Members can be selected in a randomized fashion with the probability proportional or inversely proportional to fitness. They can be selected with probability dependent on their fitness ranks (the rank of the chromosome if all the chromosomes in the population are numbered in order of fitness). They can also be selected using tournament selection, in which a group of chromosomes is chosen at random from the population, and the best or worst chromosomes from that group are then selected deterministically for further genetic operations. In this problem, random selection based on fitness is used.

3.4.4 Decimation Strategy

In order to manage the size of the population in a genetic search, it is prudent to make some provision for deleting members of the population. In addition, it may be desirable to delete chromosomes to keep the better chromosomes in the population. The decimation strategy used here is to set a max population size, if the population size is bigger than this limit, then, the chromosomes with worst fitness will be deleted from the population.

3.4.5 Result Designation and Termination Criterion

The single individual with the best value of fitness over all the generations (the so-called best so-far individual) is designated as the result of a run. Each run is terminated after running G = 51 generations.
CHAPTER FOUR
RESULTS AND DISCUSSIONS

There are two sections in this chapter. Ten experiments are run with different forms of input variables, number of data points, standard errors added to data set, and fitness functions. In the first section, input and result are summarized for each case (Tables 4.1-4.10), including model fit and the residuals (Figures 4.1-4.30). In the second section, the results will be compared and discussed based on the scaling, ability to cope with situations with noise, parsimony and local vs. global optimization for parameter.

4.1 Results

In case I, 39 data points are generated from the original IUPAC model, the data stand error is 0.

Table 4.1 Input and Result Summary for Case I

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: +, -, *, /, sqrt, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: T (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, temperature range: 210 to 970 K, step length: 20 K, 39 points in total</td>
</tr>
</tbody>
</table>
Table 4.1 Input and Result Summary for Case I (Continued)

<table>
<thead>
<tr>
<th>Testing Data: IUPAC model generated data, temperature range 970 to 1100 K, step length: 10 K, 14 points in total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression Conversion Criteria: 1e-8</td>
</tr>
<tr>
<td>Result:</td>
</tr>
<tr>
<td>Evolutionary Model: $C_p = v_1 \cdot T + v_2 \cdot \exp\left(\frac{v_3}{T}\right) - v_4$</td>
</tr>
<tr>
<td>Parameters: $v_1 = 0.0018$, $v_2 = 38.4872$, $v_3 = -433.6387$, $v_4 = -27.5793$</td>
</tr>
</tbody>
</table>

Figure 4.1 Heat Capacity $C_p$ Model Fit for Case I
Figure 4.2 Residual of Model Testing as a Function of Temperature for Case I

Figure 4.3 Residual of Model Prediction as a Function of Temperature for Case I
In case II, the exponential function and logarithmic function are removed from function set.

**Table 4.2 Input and Result Summary for Case II**

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: +, -, *, /</td>
</tr>
<tr>
<td>Terminal Set: T (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, temperature range: 210 to 970 K, step length: 20 K, 39 points in total</td>
</tr>
<tr>
<td>Testing Data: IUPAC model generated data, temperature range 970 to 1100 K, step length: 10 K, 14 points in total</td>
</tr>
<tr>
<td>Regression Conversion Criteria: 1e-8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary Model:</td>
</tr>
<tr>
<td>[ C_{r} = v_{1} + \frac{v_{2}}{T_{r}} + \frac{v_{3}}{T_{r}^{2}} + \frac{v_{4}}{T_{r}^{3}} + \frac{v_{5}}{T_{r}^{4}} + \frac{v_{6}}{T_{r}^{5}} + \frac{v_{7}}{T_{r}^{6}} + \frac{v_{8}}{T_{r}^{7}} = \sum_{n=1}^{8} v_{n} \frac{1}{T_{r}^{n-1}} ]</td>
</tr>
<tr>
<td>Parameters:</td>
</tr>
<tr>
<td>( v_{1} = 65.4051 )</td>
</tr>
<tr>
<td>( v_{2} = -1.1516e+004 )</td>
</tr>
<tr>
<td>( v_{3} = -1.1954e+006 )</td>
</tr>
<tr>
<td>( v_{4} = 1.8328e+009 )</td>
</tr>
<tr>
<td>( v_{5} = -7.0292e+011 )</td>
</tr>
<tr>
<td>( v_{6} = 1.4956e+014 )</td>
</tr>
<tr>
<td>( v_{7} = -1.6814e+016 )</td>
</tr>
<tr>
<td>( v_{8} = 7.7948e+017 )</td>
</tr>
</tbody>
</table>
Figure 4.4 Heat Capacity $C_p$ Model Fit for Case II

Figure 4.5 Residual of Model Testing as a Function of Temperature for Case II
In case III, the complete function set and the original IUPAC data are used. The standard error of data is zero. The temperature $T$ is scaled to reduced temperature $T_r = (T/T_c)$, where $T_c$ is the critical temperature of CO$_2$, a constant of 304.2 K.

Table 4.3 Input and Result Summary for Case III

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: $+$, $-$, $\ast$, $/$, sqrt, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.065, 39 points in total</td>
</tr>
</tbody>
</table>
Testing Data: IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total
Regression Conversion Criteria: 1e-8

Result:

Evolutionary Model:

\[ C_p = \frac{v_1}{T_r} + \frac{v_2}{T_r^2} + \frac{v_3}{T_r^3} + \frac{v_4}{T_r^4} + \frac{v_5}{T_r^5} + \frac{v_6}{T_r^6} + \frac{v_7}{T_r^7} = \sum_{n=1}^{8} v_n \frac{1}{T_r^{n-1}} \]

Parameters:

\[ v_1 = 69.5934 \quad v_2 = -59.8593 \quad v_3 = 30.1129 \quad v_4 = 9.9382 \quad v_5 = -13.8119 \]

\[ v_6 = -5.3273 \quad v_7 = 9.4657 \quad v_8 = -2.6979 \]

Figure 4.7 Heat Capacity \( C_p \) Model Fit for Case III
Figure 4.8 Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case III

Figure 4.9 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case III
In case IV, the complete function set and reduced temperature $T_r$ are used, normal distributed and randomly generated error with standard deviation of 0.05 is added to the original IUPAC data.

Table 4.4 Input and Result Summary for Case IV

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: $+, -, *, /, \sqrt{}, \text{power}, \log, \exp$</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter $v_1$- $v_8$ (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.065, 39 points in total, standard error 0.05</td>
</tr>
</tbody>
</table>

Table 4.4 Input and Result Summary for Case IV (Continued)

| Testing Data: IUPAC model generated data, temperature range 970 to 1100 K, step length: 10 K, 14 points in total |
| Regression Conversion Criteria: 1e-8 |

Result:

- Evolutionary Model:
  \[ C_p = v_1 + \frac{v_2}{T_r} + \frac{v_3}{T_r^2} + \frac{v_4}{T_r^3} + \frac{v_5}{T_r^4} + \frac{v_6}{T_r^5} + \frac{v_7}{T_r^6} + \frac{v_8}{T_r^7} = \sum_{n=1}^{8} v_n \frac{1}{T_r^{n-1}} \]

- Parameters:
  \[ v_1 = 69.4460 \quad v_2 = -58.7758 \quad v_3 = 27.7712 \quad v_4 = 10.9631 \quad v_5 = -12.4803 \quad v_6 = -5.7555 \quad v_7 = 8.3850 \quad v_8 = -2.1575 \]

- Model Adequacy Testing:
  \[ \chi_{mdl} = 28.7235, \chi_{exp} = 44.9853 \]

\[ \therefore \chi_{mdl} < \chi_{exp}, \therefore \text{the model is adequate.} \]
Figure 4.10 Heat Capacity $C_p$ Model Fit for Case IV
Figure 4.11 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case IV

Figure 4.12 Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case IV

In case V, the complete function set and reduced temperature $T_r (T/T_c)$ are used, where $T$ is from 210 K to 970 K, and $T_c$ is the critical temperature of CO$_2$, a constant of 304.2 K. An error with standard deviation of 0.25 is applied to the original IUPAC data.

Table 4.5 Input and Result Summary for Case V

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: $+$, $-$, $*$, $/$, $\sqrt{}$, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
</tbody>
</table>
Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.065, 39 points in total, standard error 0.25
Testing Data: IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total
Regression Conversion Criteria: 1e-8

Result:

Evolutionary Model:

\[ C_p = \frac{v_1}{T_r} + \frac{v_2}{T_r^2} + \frac{v_3}{T_r^3} + \frac{v_4}{T_r^4} + \frac{v_5}{T_r^5} + \frac{v_6}{T_r^6} + \frac{v_7}{T_r^7} + \frac{1}{\sum_{n=1}^{8} v_n T_r^{n-1}} \]

Parameters:

\[ v_1 = 68.4209 \quad v_2 = -54.7837 \quad v_3 = 26.1280 \quad v_4 = 7.1609 \quad v_5 = -13.5442 \]
\[ v_6 = -3.0722 \quad v_7 = 11.3679 \quad v_8 = -4.4025 \]

Model Adequacy Testing:

\[ \chi_{mdl} = 24.7487, \chi_{exp} = 44.9853, \]
\[ \therefore \chi_{mdl} < \chi_{exp}, \therefore \text{the model is adequate.} \]

Figure 4.13 Heat Capacity $C_p$ Model Fit for Case V
Figure 4.14 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case V

Figure 4.15 Residual of Model Prediction as a Function of Reduced Temperature $T_r$, for Case V
In case VI, the complete function set, original data and reduced Temperature $T_r(T/T_c)$ are used, where $T$ is 210:1:970 K, $T_c$ is the critical point of CO$_2$, a constant of 304.2 K.

Table 4.6 Input and Result Summary for Case VI

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: +, -, *, /, sqrt, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter $v_1$-$v_8$ (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.0033, 761 points in total</td>
</tr>
<tr>
<td>Testing Data: IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total</td>
</tr>
<tr>
<td>Regression Conversion Criteria: 1e-8</td>
</tr>
</tbody>
</table>

Table 4.6 Input and Result Summary for Case VI (Continued)

<table>
<thead>
<tr>
<th>Result:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary Model:</td>
</tr>
<tr>
<td>$C_p = v_1 + \frac{v_2}{T_r} + \frac{v_3}{T_r^2} + \frac{v_4}{T_r^3} + \frac{v_5}{T_r^4} + \frac{v_6}{T_r^5} + \frac{v_7}{T_r^6} + \frac{v_8}{T_r^7} = \sum_{n=1}^{8} \frac{v_n}{T_r^n}$</td>
</tr>
<tr>
<td>Parameters:</td>
</tr>
<tr>
<td>$v_1 = 68.4742 \quad v_2 = -58.3417 \quad v_3 = 28.6743 \quad v_4 = 11.1852 \quad v_5 = -14.2043 \quad v_6 = -6.0584 \quad v_7 = 10.1387 \quad v_8 = -2.8261$</td>
</tr>
</tbody>
</table>
Figure 4.16 Heat Capacity $C_p$ Model Fit for Case VI
Figure 4.17 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VI

Figure 4.18 Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case VI
In case VII, the complete function set and reduced temperature \( T_r \) (\( T/T_c \)) are used, where \( T \) is \( 210:1:970 \) K. A normal distributed, random generated standard error 0.05 is added to original data.

**Table 4.7 Input and Result Summary for Case VII**

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function Set:</strong>  +, -, *, /, sqrt, power, log, exp</td>
</tr>
<tr>
<td><strong>Terminal Set:</strong>  reduced Temperature ( T_r ) (independent variable), parameter ( v_1-v_8 ) (eight parameters maximum)</td>
</tr>
<tr>
<td><strong>Maximum Population Size:</strong> 500</td>
</tr>
<tr>
<td><strong>Maximum Number of Generations:</strong> 51</td>
</tr>
<tr>
<td><strong>Probability of Crossover:</strong> 80%</td>
</tr>
<tr>
<td><strong>Probability of Mutation:</strong> 20%</td>
</tr>
<tr>
<td><strong>Method of Decimation:</strong> maximum population size</td>
</tr>
<tr>
<td><strong>Selection Rule:</strong> randomly selection, fitness based</td>
</tr>
<tr>
<td><strong>Training Data:</strong> IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.0033, 761 points in total, standard error 0.05</td>
</tr>
<tr>
<td><strong>Testing Data:</strong> IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total</td>
</tr>
<tr>
<td><strong>Regression Conversion Criteria:</strong> 1e-8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Evolutionary Model:</strong></td>
</tr>
<tr>
<td>[ C_p = v_1 + \frac{v_2}{T_r} + \frac{v_3}{T_r^2} + \frac{v_4}{T_r^3} + \frac{v_5}{T_r^4} + \frac{v_6}{T_r^5} + \frac{v_7}{T_r^6} + \frac{v_8}{T_r^7} = \sum_{n=1}^{8} v_n \frac{1}{T_r^{n-1}} ]</td>
</tr>
</tbody>
</table>
| **Parameters:**  
| \( v_1 = 69.5180 \)  
| \( v_2 = -59.3949 \)  
| \( v_3 = 28.9846 \)  
| \( v_4 = 11.2025 \)  
| \( v_5 = -14.1001 \)  
| \( v_6 = -6.0755 \)  
| \( v_7 = 10.1414 \)  
| \( v_8 = -2.8663 \) |
| **Model Adequacy Testing:**  
| \( \chi_{mdl} = 792.7254, \chi_{exp} = 817.9488 \)  
| \( \therefore \chi_{mdl} < \chi_{exp}, \therefore \) the model is adequate. |
Figure 4.19 Heat Capacity $C_p$ Model Fit for Case VII

Figure 4.20 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VII
In case VIII, the complete function set and reduced temperature $T_r (T/T_c)$ are used, where $T$ is $210:1:970$ K, $T_c$ is the critical point of CO$_2$, a constant of 304.2 K. A normal distributed, randomly generated error with standard deviation of 0.25 is applied to the original data.

**Table 4.8 Input and Result Summary for VIII**

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: +, -, *, /, sqrt, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
</tbody>
</table>
Selection Rule: randomly selection, fitness based
Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.0033, 761 points in total, standard error 0.25
Testing Data: IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total
Regression Conversion Criteria: 1e-8

Result:

Evolutionary Model:

\[ C_p = \frac{v_1}{T_r} + \frac{v_2}{T_r^2} + \frac{v_3}{T_r^3} + \frac{v_4}{T_r^4} + \frac{v_5}{T_r^5} + \frac{v_6}{T_r^6} + \frac{v_7}{T_r^7} = \sum_{n=1}^{8} v_n \frac{1}{T_r^{n-1}} \]

Parameters:

\[ v_1 = 69.8432 \quad v_2 = -60.5539 \quad v_3 = 29.9869 \quad v_4 = 11.3731 \quad v_5 = -14.4051 \]
\[ v_6 = -6.2118 \quad v_7 = 10.1692 \quad v_8 = -2.8058 \]

Model Adequacy Testing:

\[ \chi_{mdl} = 791.2623, \chi_{exp} = 817.9488, \]
\[ \because \chi_{mdl} < \chi_{exp}, \because \text{the model is adequate.} \]

![Figure 4.22 Heat Capacity C_p Model Fit for Case VIII](image)
Figure 4.23 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case VIII

Figure 4.24 Residual of Model Prediction as a Function of Reduced Temperature $T_r$ for Case VIII
In case IX, the complete function set and reduced temperature $T_r (T/T_c)$ are used, where $T$ is from 210 K to 970 K. A normal distributed, randomly generated error with standard deviation of 0.05 is added to the original data. Model complexity is also added to the fitness function.

Table 4.9 Input and Result Summary for Case IX

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: +, -, *, /, sqrt, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter v1-v8 (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
<tr>
<td>Training Data: IUPAC model generated data, reduced temperature range:</td>
</tr>
<tr>
<td>0.690 to 3.189, step length: 0.065, 39 points in total, standard error is 0.05</td>
</tr>
<tr>
<td>Testing Data: IUPAC model generated data, reduced temperature range:</td>
</tr>
<tr>
<td>3.189 to 3.616, step length: 0.033, 14 points in total</td>
</tr>
<tr>
<td>Regression Conversion Criteria: 1e-8</td>
</tr>
</tbody>
</table>

| Result:                                                              |
| Evolutionary Model:                                                  |
| $C_p = v_1 \cdot T + v_2 \cdot \exp\left(\frac{v_3}{T}\right) - v_4$ |
| Parameters:                                                         |
| $v_1 = 0.8444$  $v_2 = 37.1949$  $v_3 = -1.3694$  $v_4 = -27.0985$   |
| Model Adequacy Testing:                                              |
| $\chi^2_{mdl} = 37.8272$, $\chi^2_{exp} = 49.8018$                |
| $\therefore \chi^2_{mdl} < \chi^2_{exp}$, $\therefore$ the model is adequate. |
Figure 4.25 Heat Capacity $C_p$ Model Fit for Case IX

Figure 4.26 Residual of Model Testing as a Function of Reduced Temperature $T_r$ for Case IX
In case X, the reduced temperature $T_r (T/T_c)$ is used, where $T$ is 210:1:970 K. Randomly generated error, normally distributed with standard deviation of 0.05, is applied to the original IUPAC data. The model complexity is added to fitness function.

**Table 4.10 Input and Result Summary for Case X**

<table>
<thead>
<tr>
<th>Input:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Set: $+$, $-$, $*$, $/$, $\sqrt{\cdot}$, power, log, exp</td>
</tr>
<tr>
<td>Terminal Set: reduced Temperature $T_r$ (independent variable), parameter $v_1$-$v_8$ (eight parameters maximum)</td>
</tr>
<tr>
<td>Maximum Population Size: 500</td>
</tr>
<tr>
<td>Maximum Number of Generations: 51</td>
</tr>
<tr>
<td>Probability of Crossover: 80%</td>
</tr>
<tr>
<td>Probability of Mutation: 20%</td>
</tr>
<tr>
<td>Method of Decimation: maximum population size</td>
</tr>
<tr>
<td>Selection Rule: randomly selection, fitness based</td>
</tr>
</tbody>
</table>
Table 4.10 Input and Result Summary for Case X (Continued)

| Training Data: IUPAC model generated data, reduced temperature range: 0.690 to 3.189, step length: 0.0033, 761 points in total, standard error:0.05 |
| Testing Data: IUPAC model generated data, reduced temperature range: 3.189 to 3.616, step length: 0.033, 14 points in total |
| Regression Conversion Criteria: 1e-8 |

**Result:**

**Evolutionary Model:**

\[ C_p = v_1 \cdot T + v_2 \cdot \exp\left(\frac{v_3}{T}\right) - v_4 \]

**Parameters:**

\[ v_1 = 0.6911 \quad v_2 = 37.8267 \quad v_3 = -1.4132 \quad v_4 = -27.4908 \]

**Model Adequacy Testing:**

\[ \chi_{mdl} = 779.0629, \chi_{exp} = 822.1182 \]

\[ \therefore \chi_{mdl} < \chi_{exp}, \therefore \text{the model is adequate.} \]

Figure 4.28 Heat Capacity \( C_p \) Model Fit for Case X
Figure 4.29 Residual of Model Testing as a Function of Reduced Temperature \( T_r \) for Case X

Figure 4.30 Residual of Model Prediction as a Function of Reduced Temperature \( T_r \) for Case X
4.2 Discussions

In Case I, a simpler model which contains an exponential term is obtained. The structure of the new model is different from the given IUPAC model which is a polynomial equation with eight parameters. In Fig 4.1, it is shown that the exponential model fits the data well. In Case II, the exponential function is removed from the function set, and the same polynomial equation as IUPAC model is obtained, however, the parameters are in the form of:

\[
\gamma_n \cdot (304.2)^n
\]  

(4.1)

where \(\gamma_n\) represents the eight parameters in IUPAC model, \(n\): 0-7.

With increasing power of \(n\), the value of parameters in Case II increases significantly. In this case, scaling problem may arise, which causes the matrix of partial derivatives with respect to the eight parameters become near singular. In this case, the parameters cannot be estimated accurately. When we compare the residuals plots of Case I and Case II (Fig 4.2 and Fig 4.5), it can be seen that the simpler exponential model, with the magnitude of 1E-2, has significantly reduced residuals than the polynomial model. Therefore, the solution with the GP algorithm will evolve towards the exponential structure. In Case III, the reduced temperature \(T/T_c\) is used as the independent variable. By properly scaling the input variable, the same IUPAC model structure is obtained. Meanwhile, the residuals are reduced further (Fig. 4.8). The group of Case I, II & III shows that, proper scaling or transformation of input variables may increase the accuracy of regression significantly, which help GP algorithm evolve towards the correct model.
structure. The residual plots of Case I, II & III, i.e. Fig 4.2, Fig 4.5 and Fig 4.8, which are used to test the model adequacy, exhibit a pattern. The same is true for Case VI, in which the number of data points is increased to 761 points while other inputs are kept same as those in Case III. As mentioned earlier, if the model fit to the data is correct, the residuals should be randomly distributed. A systematic pattern is a sign that the model probably has a bias in representing data. But, other than model inadequacy, different types of errors, such as random measurement errors, systematic measurement errors and computation related errors may affect the distribution of residuals, and make the residuals display a systematic pattern. The residuals which are about 0.01% are well within the tolerable computation and experimental error limits. We believe that the bias is due to difference between the IUPAC model and the proposed models.

In this research, the IUPAC data is used, which is a well modeled, smoothly fitted data set with no error. In an effort to see whether the approach could be extended to instances with experimental error, error was introduced to the data by us in cases IV and V.

Case IV and V have errors with different standard deviation, 0.05 and 0.25 respectively, added to the original IUPAC data. The same IUPAC model structure is obtained and the approach was successfully validated. For these cases, one should point out that, the systematic pattern maybe buried in the residuals plots Fig 4.11 and Fig 4.14. This is due to relative magnitude of the noise introduced. The goodness of fit is also tested by $\chi^2$-test and further demonstrate the adequacy of model obtained by GP.
In this group of tests, with different standard errors are added to the IUPAC data, GP can search in its domain of candidate models in an efficient way, and finally, the evolution process moves to the target model successfully. It shows that GP is a robust and efficient method that can handle noise in the data.

In this research, the effect of number of data points on GP evolution process is also tested. In group of Case VI, VII and VIII, the number of data points are increased to 761 points, instead of 39 points in the group of Case III through V, with the corresponding error in the data. The same IUPAC model is obtained in these three cases. The residuals display a random distribution, if the noise and/or error are added to the original IUPAC data (See Fig 4.17, Fig 4.20 and Fig 23). The adequacy of model is proved by $\chi^2$-test. Compared with Case III through V, it shows that there is no obvious improvement on regression accuracy and the structure of model. However, increasing the data points will overload the matrix computation, then, slowdown the evolution speed of GP.

In my GP problem, it’s important to keep the tree structure, i.e. the model structure, simple for thermodynamics computation and analysis in later work. Meanwhile, the complex model may bring over fitting problem. Generally speaking, the simpler the solution is, the higher the probability that it can be generalized. So, embedding parsimony concept into the measure of GP fitness may fulfill these purposes. Parsimony can be included as a factor in fitness function. For example, the standard fitness function is the sum of least squares. Then, the modified fitness could be standard fitness function plus
the structural complexity measure which is the total number of function points and terminal points. Simpler model will have the better (i.e. smaller in this case) value of fitness.

In Case IX and X, parsimony is considered and added to the fitness function. The size of sampling data is 39 points and 761 points respectively. The exponential model obtained is same as the one obtained in Case I. This result shows GP’s competency on discovering the model structure, even searching beyond the scope of traditional polynomial structure. This exponential model structure has a new look and could be an alternative to polynomial model.

Some drawbacks in traditional modeling and regression also exist in this developed methodology, such as extrapolation problem and local minimum problem for parameter regression. Throughout these ten experiments, all residual plots for model prediction display a trend where the deviation of the model from data increases with extrapolation.

It’s interesting to see that, although the same model structure is obtained, but the parameter values regressed by Marquardt method are different from those provided by IUPAC. For example, the comparison between the parameters obtained in Case III and IUPAC parameters are different as can be seen in Table 4.11:

<table>
<thead>
<tr>
<th></th>
<th>IUPAC</th>
<th>Case III</th>
<th></th>
<th>IUPAC</th>
<th>Case III</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63.9713</td>
<td>69.5934</td>
<td>4</td>
<td>-682.4660</td>
<td>-13.8119</td>
</tr>
<tr>
<td>1</td>
<td>-2.0753</td>
<td>-59.8593</td>
<td>5</td>
<td>477.3470</td>
<td>-5.3273</td>
</tr>
<tr>
<td>2</td>
<td>-211.1759</td>
<td>30.1129</td>
<td>6</td>
<td>-176.4100</td>
<td>9.4657</td>
</tr>
<tr>
<td>3</td>
<td>541.3264</td>
<td>9.9382</td>
<td>7</td>
<td>26.8843</td>
<td>-2.6979</td>
</tr>
</tbody>
</table>
When Table 4.11 is examined, one can conclude that the solutions reached are local as opposed to global parameter value. Furthermore, one should have a robust algorithm that converges all the time, since parameter optimization is integral part of each fitness function evaluation. Therefore, the performance of the approach will improve significantly when a global optimizer is used instead.
CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

In this chapter, the conclusions will be summarized in the first section, and recommendations on future work will be given in the second section.

5.1 Conclusions

To explore the application of evolutionary algorithm in modeling, this research proposes a hybrid evolutionary modeling algorithm to build thermodynamic model automatically, i.e., the application of genetic programming to the development of $C_p$ model has been considered. The main idea of the algorithm is to embed Maquardt nonlinear regression into genetic programming where GP is employed to optimize the structure of a model, while Maquardt 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.

Using the data originally generated by IUPAC model, ten experiments were ran with different sets of input variable, errors added to the original data, varying number of data points and different fitness functions.
The results revealed that in each case the hybrid system is able to find the new exponential model that closely approximated the data, even finding the same function model structure that generated the data.

From the results, we also see that the properly scaled input variables may improve the accuracy of the regression, and a robust regression method ensure that we get an accurate model.

The robustness and effectiveness of the algorithm was tested on data by adding normally distributed error with different standard deviations to IUPAC data. The results show that the \( C_p \) model built by using the hybrid system can generate satisfactory fitting values.

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

The preliminary results presented in this research indicate the potential of GP for developing thermo physical model. In conclusion, genetic programming is a robust and efficient paradigm for discovering model structure using the expressiveness of symbolic representation.

5.2 Recommendations

As stated before, \( C_p \) model was a single dependent variable and a single independent variable. This is one of the simplest model structures possible. Applying GP algorithm to more complicated problems should be studied further. For example, development of a pressure-explicit Equation of State, which is a model with one explicit
dependent variable (Pressure) but many implicit dependent variables (fugacity, enthalpy, entropy, density) and few independent variables, will be studied next.

In this research, Marquardt nonlinear regression method is used to get the parameter values for a given model structure. This method results in local as opposed to global solutions. The GP as well as other evolutionary methods would 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.

Applying parsimony principle may bring a simpler model structure to overcome over-fitting problem, but at a cost of losing accuracy of the model. In order to keep a good balance between simpler structures and accuracy of a model, Zhang [2000], using Bayesian method and a concept from information theory, developed a theoretical foundation and relation for searching the optimum point between simpler structure and accuracy. However, how to formulate the practical problem into the formula developed by Zhang is a new challenge brought by this method. Although, up to date, no other paper gives a deep study on this issue, it is still an interesting topic worthy of study further.
REFERENCES


