Estimating Physical Properties of the Products of an Atmospheric Distillation Column by Support Vector Regression

by

Firat Uzman

A Thesis Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science
in
Industrial Engineering

Koc University

June 2015

Koc University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Firat Uzman

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Metin Turkay, Ph. D. (Advisor)

_____

Pelin Gulsah Canbolat, Ph. D.

_____

Serkan Kincal, Ph. D.

Date: _____

ABSTRACT

Atmospheric distillation column is one of the most important units in an oil refinery where crude oil that is little use as it is, is fractioned into its more valuable constituents. Different fractions of the crude distillation unit are then further processed in downstream conversion and purification units to produce final products such as gasoline, diesel and jet fuel. The physical properties and the quantity of the final products vary depending on the physicochemical properties of the crude oil being processed and the operation parameters of the atmospheric distillation column. The process operators should keep the physical properties of hydrocarbon products in specified limits and operate the crude distillation unit according to instructions from the production planning department for maximizing the profit from operations. The physical properties of different crude distillation unit fractions must be measured by taking a sample from the stream periodically and analyzing these samples in a laboratory with appropriate equipment or by online analyzers that are very expensive to install, operate and maintain. Almost all of the state-of-the art online equipment has a time lag to complete the analysis in real time due to complexity of the analyses. Therefore, the laboratory or on-line measurements become available to decision makers and operator with a time lag. The intermittent nature of the measurements leads to sub-optimal control of the unit or in some cases leads to off-spec products that have considerably less economic value than the products that are within specified limits. As a result, estimation of the physical properties from online plant data and implementation of a soft sensor has a potential benefit in improving the profit. Among different approaches, the support vector regression shows great promise in machine learning due to its ability in generalizing well to unseen test data.

The objective of this study is to fully estimate the physical properties of the hydrocarbon products of an atmospheric distillation column by support vector regression. Linear, Polynomial and Gaussian Radial Basis Function (Gaussian RBF) kernels are tested and the SVR parameters are optimized by embedding *k*-fold cross validation into

a variety of algorithms including genetic algorithm (GA), grid search (GS) and non-linear programming (NLP). The performance of SVR is compared against artificial neural network (ANN) method and robust quality estimator (RQE) already functioning in the ADC. The testing results suggest that SVR with any of the integrated kernels performs well in estimating the property and generalizes well to blind test data. Compared to RQE, the mean testing error of estimation is improved by 31% with SVR from 6.4˚C to 4.4˚C and the standard deviation of estimation error is improved by 23% with SVR from 7.3 ˚C to 5.6 ˚C.

ÖZET

Atmosferik damıtma kolonu bir petrol rafinerisinin en önemli birimlerinden biridir ve bu haliyle kullanımı mümkün olmayan ham petrolü daha değerli bileşenlerine ayırır. Ham petrol damıtma ünitesinin damıttığı bileşenler daha sonra dönüştürme ve saflaştırma ünitelerinde işlenerek benzin, dizel ve jet yakıtı gibi nihai ürünler elde edilir. Nihai ürünlerin fiziksel özellikleri ve miktarları, işlenen ham petrolün fizikokimyasal özelliklerine ve atmosferik damıtma kolonunun operasyon parametrelerine göre değişecektir. Üretimin kârlılığını maksimize etmek için işletme operatörleri, hidrokarbon ürünlerinin fiziksel özelliklerini belirtilen sınırların içinde tutmalı ve üretim planlama bölümünden gelen talimatlara göre ham petrol damıtma ünitesini işletmelidir. Ham petrol damıtma ünitesi ürünlerinin fiziksel özellikleri, periyodik olarak ürün akışından numune alınıp bu numunelerin laboratuvarda uygun cihazlarda analiz edilmesi ile veya kurulumu, işletimi ve bakımı oldukça pahalı olan çevrimiçi analizörler vasıtası ile ölçülmesi gerekmektedir. Analizlerin karmaşıklığından dolayı, hemen hemen tüm son teknoloji ürünü cihazların gerçek zamanlı analizi tamamlamak için bir zaman gecikmesi vardır. Bu nedenle, laboratuvar veya çevrimiçi ölçüm sonuçları karar vericiler ve operatörlere gecikmeli olarak ulaşmaktadır. Ölçümlerin kesikli ve gecikmeli olması, ünitenin optimal noktanın uzağında kontrol edilmesine veya bazı durumlarda ekonomik değeri oldukça düşük olan fiziksel özellik kısıtları dışında ürün üretimine neden olmaktadır. Bunun sonucu olarak, çevrimiçi ünite verilerinden fiziksel özelliklerin tahmin edilmesi ve bir sanal ölçüm uygulamasının ünite karlılığının iyileştirilmesinde önemli bir potansiyeli vardır. Farklı yaklaşımlar arasında destek vektör regresyon, daha önce karşılaşılmayan test verilerini genellemedeki yeteneği ile makine öğrenmede gelecek vadetmektedir.

Bu çalışmanın amacı, bir atmosferik damıtma kolonunun hidrokarbon ürünlerinin fiziksel özelliklerini destek vektör regresyon yöntemi ile tam olarak tahmin etmektir. Doğrusal, polinom ve gauss radyal tabanlı fonksiyon çekirdekleri sınanmış ve destek vektör regresyon parametreleri, genetik algoritma, örgü arama ve doğrusal olmayan

programlama yöntemlerine *k*-kat çapraz doğrulama entegre edilerek optimize edilmiştir. Destek vektör regresyon performansı yapay sinir ağı yöntemi ve zaten atmosferik damıtma kolonunda faaliyette olan tahminci ile karşılaştırılmıştır. Test sonuçlarına göre destek vektör regresyon herhangi entegre çekirdek ile ürün özellikleri tahmini ve gözü kapalı sınamada gayet iyi performans göstermiştir. Destek vektör regresyon ile mevcut tahmincinin ortalama mutlak test hatası %31 oranında iyileştirilerek 6,4˚C'den 4,4˚C'ye düşürülürken, hatadaki standart sapma %23 oranında azaltılarak 7,3˚C'den 5,6˚C'ye düşürülmüştür.

ACKNOWLEDGEMENTS

I would like to express my respect and gratitude to *Prof. Dr. Metin TÜRKAY* for his guidance, encouraging support and patience during the most challenging period of my life, graduate study and thesis period concurrent with my role as an Optimization and Software Superintendent in the Research and Development Department of Turkish Petroleum Refineries Corporation (Tüpraş).

I would like to extend my grateful thanks to *Prof. Dr. Yaman ARKUN* and Tüpraş R&D Department Manager, *Dr. Ahmet Murat YILDIRIM*, who made it possible for me to pursue Master of Science degree in Koç University while working in Tüpraş.

I would like to thank *Sadık ÖDEMİŞ* and *Hilal FİDAN ACAR* for their helpful suggestions and contribution.

I want to thank the members of Optimization and Software team, *Eren Yaşar ÇİÇEK, Elif METE, Nergiz HAYTURAL, Sevi Zeynep HASDEMİR* and *Sinem NALBANT*. Together we accomplished many projects that are now providing decision support to various departments in Tüpraş.

Last, but not least, I would like to thank my wife *Bihter* for her understanding and love. Her support and encouragement was in the end what made this dissertation possible. My mother, *Fatma*, my father *Rifat* and my sister *Nilhan* receive my deepest gratitude and love for their dedication and many years of support that provided the foundation for this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	Inequality matrix of quadratic programing
A_{eq}	Equality matrix of quadratic programing
A_i	First coefficient of heat capacity [J/K]
A_j	Activation function of node *j*
A_j	First coefficient of simplified phase equilibrium coefficient of base component *b* in stage *j*
A_j	First coefficient of stage *j* in tridiagonal matrix *T*
b	Base component number
b	Bias term of SVR
b	Damping factor
b	Inequality vector of quadratic programing
b_{eq}	Equality vector of quadratic programing
B_i	Second coefficient of heat capacity [J/K^2]
B_j	Second coefficient of simplified phase equilibrium coefficient of base component *b* in stage *j*
B_j	Second coefficient of stage *j* in tridiagonal matrix *T*
C	Total number of components
C	Weight factor of cost
C_i	Third coefficient of heat capacity [J/K^3]
c_j	First coefficient of simplified excess enthalpy of stage *j* [Gcal/kmol]
C_j	Third coefficient of stage j in tridiagonal matrix *T*
C_P	Heat capacity of a component [J/K]
d	Degree of polynomial kernel
D	Vector matrix of tridiagonal method
D_i	Fourth coefficient of heat capacity [J/K^4]
d_j	Second coefficient of simplified excess enthalpy of stage *j* [Gcal/kmol.K]

F	Vector for first degree in quadratic programing
$f_{i,j}$	Component molar flow rate of component *i* in feed entering stage *j* [kmol/h]
F_j	Input feed entering stage *j* [kmol/h]
$G_{\max}$	Generation limit of GA
H	Matrix for second degree in quadratic programing
h_j^{ex}	Liquid phase excess enthalpy of stage *j* [Gcal/kmol]
H_j^{ex}	Vapor phase excess enthalpy of stage *j* [Gcal/kmol]
H_j^f	Enthalpy of feed entering stage *j* [Gcal/kmol]
$H_{i,j}^{\text{ideal}}$	Ideal enthalpy of component *i* in stage *j* [Gcal/kmol]
h_j	Enthalpy of liquid phase in stage *j* [Gcal/kmol]
H_j	Enthalpy of vapor phase in stage *j* [Gcal/kmol]
i	Component number
i	Dataset number
$i,\ j$	Node number
J	Jacobian matrix
j	Stage number
k	Iteration number
k	Parameter of *k*-fold cross validation
$K_{b,j}$	Phase equilibrium coefficient of base component *b* in stage *j*
$K_{i,j}$	Phase equilibrium coefficient of component *i* in stage *j*
l	Total number of dataset
L_{j1}^{PI}	Liquid phase return from pump-around of stage *j*1 to stage *j* [kmol/h]
L_j^{PO}	Liquid phase side draw from stage *j* to pump-around [kmol/h]
L_{j1}^{PO}	Liquid phase side-draw from stage *j*1 to pump-around [kmol/h]
$l_{i,j}^S$	Liquid phase component molar flow rate of component *i* in side-draw from stage *j* [kmol/h]
L_j^S	Output liquid side-draw from stage *j* [kmol/h]

L_j^{SS}	Liquid phase side-draw from stage j to side-stripper [kmol/h]
$l_{i,j}$	Liquid phase component molar flow rate of component i in stage j [kmol/h]
L_j	Liquid flowing down from stage j to j+1 [kmol/h]
M	Coefficient for liquid return rate from pump-around in tridiagonal matrix T
M	Dot product matrix of SVR
N	Total number of stages in a distillation column
O_j	Output function of node j
p	Significance level
P	Population size of GA
P_j^f	Pressure of feed entering stage j [bar]
P_C	Critical pressure of a component [bar]
p_i	Best known position of particle i
P_j	Pressure of stage j [bar]
p_{KERNEL}	Parameter of selected kernel
Q_j	Heat duty [Gcal/h]
R	Coefficient for vapor return rate from side-stripper in tridiagonal matrix T
r_C	Cross over ratio of GA
r_p, r_s	Random numbers
RL_j^{PO}	Withdrawal ratio of liquid side-draw from stage j to pump-around
RL_{j1}^{PO}	Withdrawal ratio of liquid side-draw from stage j1 to pump-around
RL_j^S	Withdrawal ratio of liquid side-draw from stage j
RL_j^{SS}	Withdrawal ratio of liquid side-draw from stage j to side-stripper
RV_j^S	Withdrawal ratio of vapor side-draw from stage j
s	Best known swarm position
s	Support vector number
S	Total number of support vectors

$S_{b,j}$	Stripping factor of the base component *b* in stage *j*
s_E	Elite count of GA
T	Tridiagonal matrix
T_j^f	Temperature of feed entering stage *j* [˚C]
T_C	Critical temperature of a component [˚C]
T_j	Temperature of stage *j* [˚C]
T_r	Reference temperature [K]
$v_{i,j}^S$	Vapor phase component molar flow rate of component *i* in side-draw from stage *j* [kmol/h]
V_j^S	Output vapor side-draw from stage *j* [kmol/h]
v_i	Velocity of particle *i*
$v_{i,j}$	Vapor phase component molar flow rate of component *i* in stage *j* [kmol/h]
V_j	Vapor flowing up from stage *j* to *j*-1 [kmol/h]
V_{j2}	Vapor flowing up from stage *j*2 of side-stripper to stage *j* [kmol/h]
w	Combination of support vectors of SVR
$w_{j,i}$	Weight of the interconnection from node *i* to node *j*
x_i	Position of particle *i*
$x_{i,j}$	Liquid phase molar fraction of component *i* in stage *j*
$y_{i,j}$	Vapor phase molar fraction of component *i* in stage *j*
$z_{i,j}$	Molar fraction of component *i* in feed entering stage *j*
$\alpha_i, \alpha_i^*, \eta_i, \eta_i^*$	Lagrange multipliers
$\alpha_{i,j}$	Relative volatility of component *i* in stage *j*
β_i^*	Addition of Lagrange multipliers
β_s^*	Addition of Lagrange multipliers of support vectors
β_i	Difference between Lagrange multipliers
β_s	Difference between Lagrange multipliers of support vectors

ε	Error tolerance value of inside-out algorithm
ε	Width of insensitive region of SVR
γ	Simplified coefficient of RBF kernels
κ	Scalar coefficient of hyperbolic tangent kernel
μ	Parameter of Huber loss function
ω	Acentric factor of a component
ω	Velocity factor in PSO
$\phi_{i,j}^{L}$	Liquid phase fugacity coefficient of component *i* in stage *j*
$\phi_{i,j}^{V}$	Vapor phase fugacity coefficient of component *i* in stage *j*
φ_p	Local best factor in PSO
φ_s	Swarm best factor in PSO
ρ	Weight coefficient of hyperbolic tangent kernel
σ	Coefficient of bias term
σ	Coefficient of RBF kernels
ξ_i, ξ_i^*	Slack variables

ABBREVIATIONS

ADC	Atmospheric Distillation Column
ANN	Artificial Neural Network
CDU	Crude Distillation Unit
CPU	Central Processing Unit
DCS	Distributed Control System
FCC	Fluidized Catalytic Cracking
GA	Genetic Algorithm
GS	Grid Search
KKT	Karush-Kuhn-Tucker
LB	Lower bound
LIBSVM	Library for Support Vector Machines
LPG	Liquefied Petroleum Gas
MATLAB	Matrix Laboratory
MAE	Mean absolute error
ME	Mean error
MSE	Mean squared error
NLP	Non-linear Programming
PCA	Principle Component Analysis
PR	Peng-Robinson
PSO	Particle Swarm Optimizer
RBF	Radial Basis Function
RQE	Robust Quality Estimator
SRK	Soave-Redlich-Kwong
STD	Standard deviation
SVM	Support Vector Machine
SVR	Support Vector Regression
UB	Upper bound

Chapter 1

INTRODUCTION

Crude Distillation Unit (CDU) is the heart of a petroleum refinery where the crude oil is separated into its naturally occurring fractions. Atmospheric Distillation Column (ADC) is the most complex asset in the CDU which operates at atmospheric pressure. The operation parameters of an ADC influences the amount and hydrocarbon properties of its products. Dynamically changing demands and prices for end products forces the planners to frequently update the optimal operating parameters and maximize the amount of a product while keeping the products within their specification limits. This gives rise to the need for online monitoring of the properties.

The online monitoring of temperature, pressure and flowrate within the ADC is possible with measurement devices connected to Distributed Control Systems (DCS). However online monitoring of hydrocarbon properties are only possible by installing online analyzers which are very complex, hard-to-maintain and expensive devices. Therefore hydrocarbon properties are monitored by taking samples periodically from the ADC and analyzing the samples in a laboratory with appropriate equipment which takes up a considerable amount of time.

The lack of online measurement of hydrocarbon properties can be complemented by online estimation methods. An estimation function of temperature and pressure as input variables can be fitted to laboratory analysis data of the hydrocarbon properties. The ADC in İzmit Refinery of Turkish Petroleum Refineries Corporation which is the subject of this study has employed Robust Quality Estimators (RQE) for the online estimation of the critical properties of hydrocarbon products like the 95% boiling point temperature of heavy diesel. This RQE embodies a fixed function of heavy diesel tray temperature and

flash zone temperature and pressure which is then fitted to laboratory data by a gain and a bias.

There have been studies on rigorous modeling of ADCs. This method is highly complex and embodies a comprehensive physical properties library, refers to the laws of thermodynamics and is hard to fit to a real operating unit. Though many of the hydrocarbon properties can be gathered from the simulation results of a rigorous model, the rigorous model still needs periodic maintenance as it sways away from real system in time.

Recent studies have successfully employed Machine Learning in modeling chemical processes in petroleum refining. Artificial Neural Network (ANN) has been a popular choice in this area, but has suffered from over-fitting. ANN like any regression method that embodies Empirical Risk Minimization, minimizes the estimation error without concern for the model complexity. This in turn leads to poor performance in generalization to unseen testing data.

More recent studies have utilized Support Vector Machines (SVM) in regression problems. SVMs, in contrast to ANN embodies Structural Risk Minimization that aims to obtain a flatter and less complex function. By implementing a new loss function, Support Vector Regression (SVR) chooses the flattest path in which error is kept within a predefined width for insensitive region and a predefined cost factor handles outlier data. Moreover, by employing kernel functions, SVR can be trained to generate a nonlinear estimation function. Lastly, the user-defined parameters of SVR can be optimized via cross validation embedded in global optimizers like Genetic Algorithm or simpler solvers like Grid Search, to maximize generalization performance.

This study concentrates on the estimation of physical properties of Heavy Diesel fraction of an ADC by Support Vector Regression. Chapter 2 presents a review of modeling methods used in similar problems. Chapter 3 focuses on the concept of an SVR, the training of an SVR and the optimization of SVR parameters. Chapter 4 explains the

implementation of SVR in estimation of physical properties of the hydrocarbon products of an ADC. Chapter 5 presents and discusses the results and Chapter 6 concludes with a summary.

Chapter 2

LITERATURE SURVEY

2.1 Petroleum Refining

Petroleum has been known to man since ancient times. It is a mixture of hydrocarbon molecules which can be found in the ground, but its use was at limited amounts in its natural form. Until 19th century, it was not considered as a good source for burning since it produced smoke and ash. The petroleum industry boomed when it was discovered that useful products can be distilled by boiling the crude oil. First reliable sources of oil were drilled and first petroleum refineries were built to distillate crude oil in the second half of the 19th century [1].

In current times, petroleum refineries process crude oil into useful and valuable products like naphtha, gasoline, diesel, jet fuel and asphalt [1, 2]. They have vital role in the supply network of petroleum. They operate in energy-intense industries where continuous investments at great costs must be pursued for sustainability in a challenging global competition.

Crude oil contains millions of different hydrocarbon molecules with different molecular weights, boiling points, structures and lengths, and other non-hydrocarbon molecules like sulfur and nitrogen. The varying physical and chemical properties of hydrocarbon molecules make them useful in a wide range of different applications. The crude oil should go through separation, conversion and purification processes to obtain various useful products within specification limits.

In petroleum refining, the crude oil is first separated into its fractions in the crude distillation unit, which is the focus of this study. Then each fraction is further processed in conversion units by changing the size of the molecules in cracking units or changing the structure of the molecules in reforming units. The converted streams are then treated in purification and separation processes to further improve product quality. Finally similar products from varying units are blended to form the final products strictly within strict specification limits. Every refinery has a similar but unique setup and Figure 2.1 depicts the flow diagram of a typical petroleum refinery.

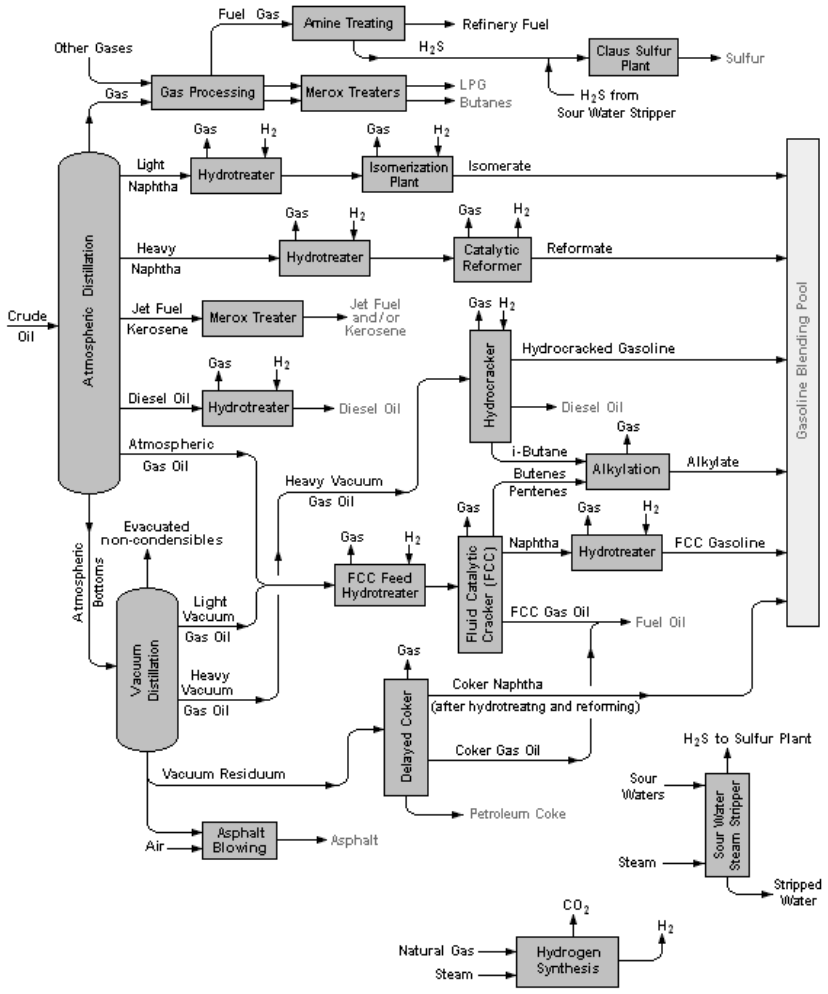

Figure 2.1: Flow diagram of a typical petroleum refinery

2.2 Crude Distillation Unit

Petroleum refining begins with the Crude Distillation Unit (CDU) which separates the crude oil into streams with different boiling ranges which are processed further in conversion or purification processes. The CDU often contains more than one distillation column and the configuration of the CDU differs from one refinery to another. The atmospheric distillation column is always the heart of this configuration, which operates around atmospheric pressure.

The crude oil is first preheated by a group of heat exchanger where the distilled products or pump-around streams that need to be cooled down exchange heat with the yet cold crude oil. Afterwards the crude oil first goes through desalters to remove inorganic salts that cause corrosion, then goes through a heat exchanger group and finally arrives at the furnace. The furnace may be preceded by a preflash column to remove light-end products beforehand. This reduces the operating cost of the furnace. The furnace heats the crude oil up to around 400˚C which is then fed to the bottom tray of the atmospheric distillation column.

The bottom of the atmospheric distillation column is heated by a stripping steam flow which vaporizes the molecules with lower boiling points to recover higher valued distillates like diesel in the upper trays. There are intermediate outlets on the column that draw out fractions with different boiling point ranges and these fractions like kerosene and diesel are called sidecuts. These sidecuts are fed into side strippers which also have stripping steam. The vaporized molecules are sent back to the atmospheric distillation column. There are also pump-arounds that draw out a stream to be cooled down and pumped back to the atmospheric distillation at a higher tray. At the top is the condenser and the reflux drum where the uncondensed vapor is outlet, the water in the mixture is removed, a controlled ratio of the condensed liquid is pumped back to the atmospheric distillation column and the rest is outlet as naphtha.

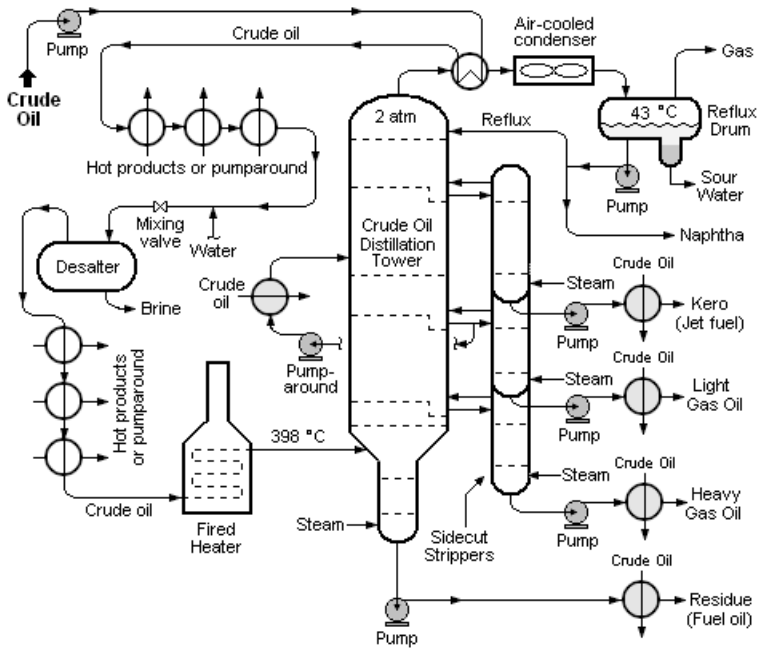

Figure 2.2: Flow diagram of a typical crude distillation unit

The atmospheric bottom or atmospheric residue is fed to vacuum distillation column for further separation. The naphtha output may also be further separated into purer components, thus concluding the crude distillation unit. The conversion units superseding the CDU like Fluidized Catalytic Cracker (FCC), Hydrocracker or Delayed Coker may also need separation columns similar to the ones in the CDU since the feeds to these units are converted into a range of higher valued distillates that need separation.

2.3 Modeling Approaches

As it is with the crude oil, the prices and the demand for each product change dynamically leading to frequent updates in planning instructions. Since Crude Distillation Units are complex and energy intense units, safe and optimum operation of the unit has a great significance in oil refining. This leads to a need in proper monitoring of the

operation for process engineers and a proper estimation of the operation dynamics for planning engineers. Both objectives prove to be troublesome as the physical properties can be measured by taking a sample from the stream periodically and then analyzing these sample in a laboratory with appropriate equipment or by online analyzers that are very expensive, complex and hard-to-maintain sets of equipment. The periodicity in measurement leads to sub-optimal control of the unit or in some cases to off-spec products. This gives rise to the need for estimating the plant dynamics and physical properties of the product streams.

Different methods like rigorous distillation column models that depend on the laws of thermodynamics, and statistical models that depend on online plant data have been incorporated to estimate the behavior of a distillation column.

2.3.1 Rigorous Models

First principal mechanistic models have been developed for simulating a steady state, multicomponent and multistage distillation column. A rigorous method by Naphtali and Sandholm [3] has been further improved by Boston and Sullivan [4], Hofeling and Seader [5], Russell [6], Kumar [7] and many others. Many academicians have further studied the use of rigorous methods in optimization of distillation columns [8-11].

Any rigorous model has mass and energy balance equations, need a comprehensive physical properties library and are very complex. One needs a proper knowledge of the method to be able to fit a model to an existing distillation column and the model usually deviates from the real data in time. The model is as good as the physical properties estimation library selected for predicting properties like phase equilibrium coefficients and enthalpies.

2.3.1.1 The Mesh Equations

A column is modeled as a multi-stage multi-component system and each stage is modeled modularly as shown in Figure 2.4 where the stages of the column are numbered starting with the top tray or condenser drum if there is one ending with the bottom tray or the reboiler drum. If there are any side-strippers, the stages of these are numbered after the bottom tray of the column to a total of *N* stages as shown in Figure 2.3. For any stage *j*, there may be an input feed stream F_j, an output liquid side-draw L_j^s, an output vapor side-draw V_j^s and an input or output heat duty Q_j. Also liquid flowing down from stage *j* to stage *j* + 1 is L_j and vapor flowing up from stage *j* to stage *j* – 1 is V_j. The temperature of stage *j* is T_j and the pressure is P_j. Component *i* in stage *j* has liquid phase molar fraction as $x_{i,j}$ and vapor phase molar fraction as $y_{i,j}$. Enthalpy of vapor phase in stage *j* is H_j, while enthalpy of liquid phase in stage *j* is h_j. Lastly, any feed stream entering stage *j* has temperature T_j^F, pressure P_j^F and enthalpy H_j^F.

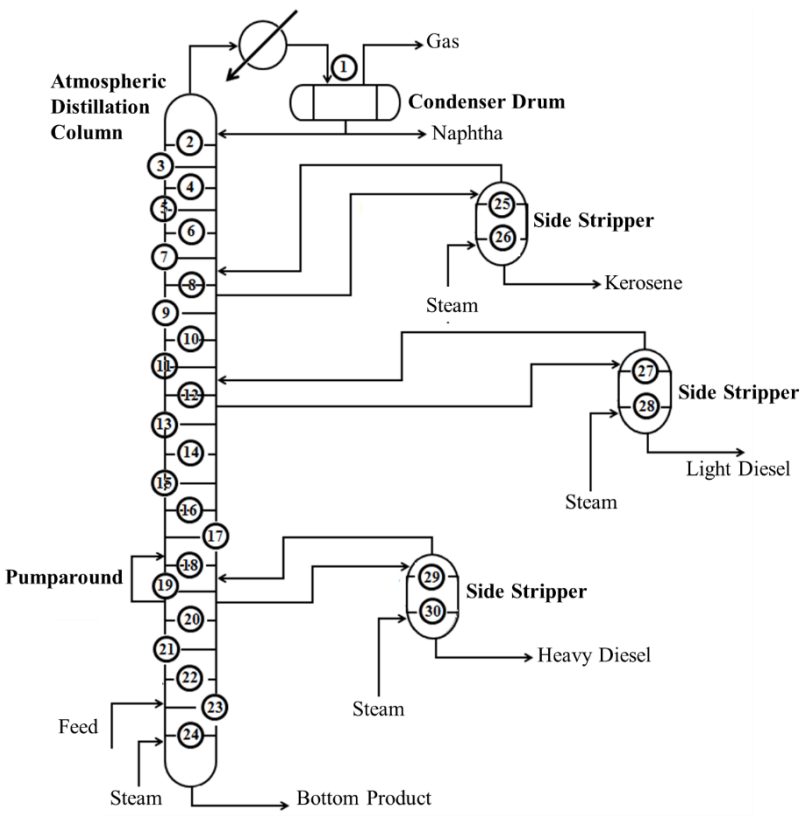

Figure 2.3: Stage numbering of a typical atmospheric distillation column

A complex atmospheric distillation column most usually has pump-arounds and side-strippers. The input and output streams from or to these special assets should be tied together. Liquid phase side-draw from stage *j* to pump-around is L_j^{PO} and liquid phase return from pump-around is L_{j1}^{PI} where *j*1 is the stage pump-around draws the stream from. L_{j1}^{PI} is equal to L_{j1}^{PO}. Liquid phase side-draw from stage *j* to side-stripper is L_j^{SS}. As it is with the main column, the top stage *j* of a side-stripper does not have liquid flowing down from stage $j - 1$, but from the corresponding L_j^{SS}. Similarly there is no vapor flowing up from stage $j + 1$ to the bottom stage *j* of a side-stripper. The vapor flowing up from the top stage *j*2 of a side-stripper is connected to the corresponding stage of the main

column as V_{j2}. The inputs and outputs of a stage can be generalized as shown in Figure 2.4.

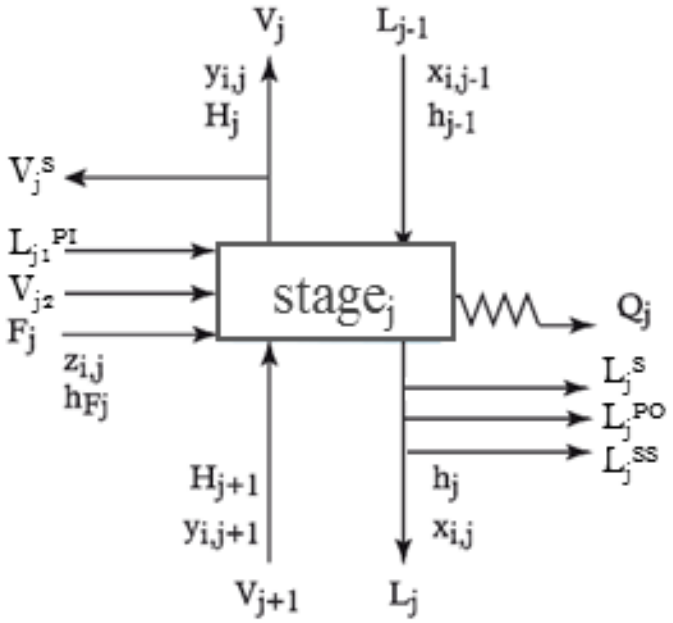

Figure 2.4: A stage of a distillation column

Then the component molar rates are computed as below where *i* is for component and there are a total of *C* components:

$$\begin{aligned} l_{i,j} &= L_j x_{i,j}, \\ v_{i,j} &= V_j y_{i,j}, \\ f_{i,j} &= F_j z_{i,j}, \\ l^s_{i,j} &= L^s_j x_{i,j}, \\ v^s_{i,j} &= V^s_j y_{i,j}. \end{aligned} \tag{1}$$

The equations required to define a stage are mass balance, energy balance and phase equilibrium equations. The mass balance for any component *i* in stage *j* is:

$$\begin{aligned} & L_{j-1}x_{i,j-1} - (L_j + L_j^S + L_j^{SS} + L_j^{PO})x_{i,j} \\ & + V_{j+1}y_{i,j+1} - (V_j + V_j^S)y_{i,j} \\ & + L_{j1}^{PI}x_{i,j1} + V_{j2}y_{i,j2} + F_j z_{i,j} = 0. \end{aligned} \tag{2}$$

The component molar fractions in stage *j* should add up to 1 (one):

$$\begin{aligned} \sum_i x_{i,j} &= 1, \\ \sum_i y_{i,j} &= 1. \end{aligned} \tag{3}$$

The energy balance equation for stage *j* is:

$$\begin{aligned} & L_{j-1}h_{j-1} - (L_j + L_j^S + L_j^{SS} + L_j^{PO})h_j + V_{j+1}H_{j+1} \\ & -(V_j + V_j^S)H_j + L_{j1}^{PI}h_{j1} + V_{j2}H_{j2} + F_j H_j^f + Q_j = 0. \end{aligned} \tag{4}$$

The phase equilibrium equation of a component *i* in stage *j* is:

$$y_{i,j} = K_{i,j}x_{i,j}. \tag{5}$$

2.3.1.2 Phase Equilibrium Coefficient and Enthalpy

The phase equilibrium coefficients $K_{i,j}$, liquid phase enthalpies h_j, and vapor phase enthalpies H_j should be predicted by the physical properties estimation library as functions of temperature, pressure and component fractions:

$$K_{i,j} = f(T_j, P_j, x_{i,j}, y_{i,j}), \tag{6}$$

$$H_j = f(T_j, P_j, y_{i,j}), \tag{7}$$

$$h_j = f(T_j, P_j, Y_{i,j}). \tag{8}$$

The enthalpies cannot be calculated with the assumption that the components act like an ideal gas, since the molecules attract and repulse each other, and have finite volume. Equations of state proposed by many academicians [12-14] can be used to solve for the phase equilibrium coefficients and excess enthalpies. Peng-Robinson equation of state (PR) [12] and Soave modification of Redlich-Kwong (SRK) [14] are popularly used for predicting these properties of hydrocarbons.

The crude oil has over 1 million different molecules in its mixture where some are non-hydrocarbon molecules. Light hydrocarbons with as much as 6 or 7 carbon atoms may be defined as pure components, but the rest of the molecules are grouped by true boiling point ranges as pseudo-components. To solve for phase equilibrium coefficients and enthalpies, one needs the properties of these components like critical temperature T_C, critical pressure P_C, acentric factor ω.

Properties for pseudo-components like molecular weight, critical temperature, critical pressure, acentric factor, heat of formation, ideal gas enthalpy can be predicted by empirical correlations which are documented in the American Petroleum Institute's Petroleum Data Book [15].

For heat capacity C_p calculations of pure components, one needs the coefficients of the estimation function:

$$C_P = A_i + B_i T + C_i T^2 + D_i T^3. \tag{9}$$

Then the ideal enthalpy of a pure component can be calculated by

$$H_{i,j}^{ideal} = \int C_P dT_j = A_i T_j + \frac{B_i}{2} T_j^{\,2} + \frac{C_i}{3} T_j^{\,3} + \frac{D_i}{4} T_j^{\,4}. \tag{10}$$

The vapor phase excess enthalpies H_j^{ex} and liquid phase excess enthalpies h_j^{ex} can be calculated by equations of state to give total enthalpy of stage *j* as:

$$H_j = \sum_i H_{i,j}^{ideal} + H_j^{ex}, \tag{11}$$

$$h_j = \sum_i H_{i,j}^{ideal} + h_j^{ex}. \tag{12}$$

The phase equilibrium coefficients $K_{i,j}$ are calculated from the vapor phase and liquid phase fugacity coefficients $\phi_{i,j}^{V}$ and $\phi_{i,j}^{L}$ respectively as:

$$K_{i,j} = \frac{\phi_{i,j}^{L}}{\phi_{i,j}^{V}}. \tag{13}$$

The equation of state of PR [12] or SRK [14] can iteratively solve for the fugacity coefficients and excess enthalpies for a stage at once for vapor phase and once for liquid phase, but at high computational cost. More information on equation of state can be found in the book Multistage Separation Processes by Khoury [16].

2.3.1.3 Degree of Freedom

To simulate any problem, one needs the degree of freedom equal to zero. To accomplish this, the pressure value of stages are fixed with a fixed pressure value for the top stage and a fixed pressure drop value towards the bottom stage. Still one needs to define more equations against any heat duty or side-draws. These equations are called performance specifications and are expressed as:

$$G(w) = 0. \tag{14}$$

An example for a performance specification can be a fixed value for the top tray temperature against the condenser heat duty Q_1 and expressed as $G(w) = T_1 - 40^{o}C = 0$.

2.3.1.4 Inside-out Algorithm

The commonly used method in many chemical process simulators is the inside-out algorithm by Russell [6], which is explained in [16]. This method simplifies the phase equilibrium coefficient and enthalpy calculations which are computationally expensive, by linearizing the function in terms of the stage temperature. The inner loop solves the model according to the performance specifications given by the user with the simplified functions. Then the outer loop updates the phase equilibrium coefficients, and enthalpy calculations, and checks if the update is within the specified limits. The algorithm terminates if the update is within the defined limits. If it is not, the functions are linearized again for another set of inner loop iterations. This method requires little information for generating initial estimates and has lower computational cost.

The inside-out algorithm also incorporates special iteration variables that increase convergence ability. First the mass balance is written in terms of component molar flow rates instead of component molar fractions by combining equations (1) and (2):

$$\begin{aligned} & l_{i,j-1} - (l_{i,j} + l^{S}_{i,j} + l^{SS}_{i,j} + l^{PO}_{i,j}) \\ & + v_{i,j+1} - (\mathrm{v}_{i,j} + v^{S}_{i,j}) \\ & + l^{PI}_{i,j1} + v_{i,j2} + f_{i,j} = 0. \end{aligned} \tag{15}$$

Now the withdrawal ratios are used as variables instead of molar rates for side-draws:

$$\begin{aligned} & RL^{S}_{j} = L^{S}_{j} / L_{j}, \\ & RL^{SS}_{j} = L^{SS}_{j} / L_{j}, \\ & RL^{PO}_{j} = L^{PO}_{j} / L_{j}, \\ & RV^{S}_{j} = V^{S}_{j} / V_{j}, \\ & RL^{PO}_{j1} = L^{PO}_{j1} / L_{j1} = L^{PI}_{j1} / L_{j1}. \end{aligned} \tag{16}$$

Then the mass balance can be written as:

$$\begin{aligned} & l_{i,j-1} - \left(1 + RL_j^S + RL_j^{SS} + RL_j^{PO}\right) l_{i,j} \\ & + v_{i,j+1} - \left(1 + RV_j^S\right) v_{i,j} \\ & + RL_{j1}^{PO} l_{i,j1} + v_{i,j2} + f_{i,j} = 0. \end{aligned} \tag{17}$$

Next the vapor phase component molar rates $v_{i,j}$ are written in terms of liquid phase component molar rates $l_{i,j}$ by:

$$v_{i,j} = K_{i,j} \left(V_j / L_j\right) l_{i,j}. \tag{18}$$

Then the relative volatility $\alpha_{i,j}$ is defined as:

$$\alpha_{i,j} = K_{i,j} / K_{b,j}, \tag{19}$$

where $K_{b,j}$ is the phase equilibrium coefficient of the base component *b*. Base component is selected by the user. A pure component with highest true boiling point or a component with phase equilibrium coefficient value closest to 1 (one) can be selected to have the mean of relative volatility close to 1 (one).

Lastly the special iteration variable, the stripping factor of the base component is defined as:

$$S_{b,j} = K_{b,j} V_j / L_j. \tag{20}$$

By combining equations (18), (19) and (20) the vapor phase component molar rates can be written as:

$$v_{i,j} = \alpha_{i,j} S_{\mathrm{b},j} l_{i,j}. \tag{21}$$

Finally the mass balance equation can be written as:

$$\begin{aligned} & l_{i,j-1} - \left(1 + RL_j^S + RL_j^{SS} + RL_j^{PO} + (1 + RV_j^S)\alpha_{i,j} S_{b,j}\right) l_{i,j} \\ & + \alpha_{i,j+1} S_{b,j+1} l_{i,j+1} + RL_{j1}^{PO} l_{i,j1} + \alpha_{i,j2} S_{b,j2} l_{i,j2} + f_{i,j} = 0. \end{aligned} \tag{22}$$

Energy balance equation can be written in terms of withdrawal factors as:

$$\begin{aligned} &L_{j-1}h_{j-1} - (1 + RL_j^S + RL_j^{SS} + RL_j^{PO})L_j h_j + V_{j+1}H_{j+1} \\ &-(1 + RV_j^S)V_j H_j + RL_{j1}^{PO} L_{j1} h_{j1} + V_{j2} H_{j2} + F_j H_j^f + Q_j = 0. \end{aligned} \tag{23}$$

2.3.1.5 Initialization

All variables need initial values before the inside-out algorithm begins. The temperature, pressure, flow rate and molar fractions of any feed is predefined and fixed. Any side-draw or withdrawal rate needs estimates from the user. The temperatures can be initialized as an interpolation of an estimate for top stage and an estimate for bottom stage. The liquid and vapor phase molar fractions can be initialized as feed stream, flashed at the average temperature inside the column. Inner liquid and vapor flow rates can be initialized based on estimated side-draws and withdrawal rates and by assuming constant inner flows. The heat duties also need estimates. Any side-draw, withdrawal rate or heat duty that does not exist for a stage should be fixed to 0 (zero).

2.3.1.6 The Outer Loop

In the first iteration, the outer loop begins with the fixed values for pressure in each stage P_j and roughly initialized values for temperatures T_j, liquid flow rates L_j, vapor flow rates V_j, liquid phase component molar fractions $x_{i,j}$, vapor phase component molar fractions $y_{i,j}$, withdrawal rates RL_j^S, RL_j^{SS}, RL_j^{PO}, RV_j^S and heat duties Q_j for each stage. In the next iterations, the outer loop continues with these variables updated by the inner loop.

Firstly, the phase equilibrium coefficients $K_{i,j}$ and excess enthalpies for liquid phase h_j^{ex} and vapor phase H_j^{ex} are solved for as stated in Section 2.3.1.2. If this is the first iteration, the outer loop continues by simplifying the *K*-values and excess enthalpies for inner loop calculations. If this is an intermediate iteration, the updated *K*-values and excess enthalpies are compared with their previous values from the inner loop and the algorithm is accepted as converged if the error is lower than a predefined tolerance value ε that has a default value of 10^{-6}.

If the problem did not converge, the algorithm carries on by fixing the relative volatility value $\alpha_{i,j}$ by equation (19), and computing the stripping factor $S_{b,j}$ by equation (20). Simplified functions for *K*-values of the base component $K_{b,j}$ and excess enthalpies are generated by perturbing the stage temperatures T_j, meaning the equation of state is called for each stage twice with $\pm\Delta T$ at a default value of 1^oC.

The simplified function for *K*-values is:

$$K_{b,j} = \exp(A_j - B_j / T_j), \tag{24}$$

where coefficients A_j and B_j can be easily calculated from:

$$\begin{aligned} B_j &= \frac{\ln(K_{b,j-\Delta T} / K_{b,j+\Delta T})}{\dfrac{1}{T_j + \Delta T} - \dfrac{1}{T_j - \Delta T}}, \\ A_j &= \ln(K_{b,j-\Delta T}) + \frac{B_j}{T_j - \Delta T}. \end{aligned} \tag{25}$$

The simplified functions for excess enthalpies are:

$$\begin{aligned} h_j^{ex} &= c_j + d_j(T_j - T_r), \\ H_j^{ex} &= e_j + f_j(T_j - T_r), \end{aligned} \tag{26}$$

where T_r is the reference temperature and is equal to 298.15 K. The ideal enthalpies do not need simplified functions since they are already functions of the temperature.

2.3.1.7 The Inner Loop

The inner loop begins by solving for the liquid phase component molar rates $l_{i,j}$ by the tridiagonal matrix method. The recurrence relation in equation (22) can be formed into a system of equations as $Tx = D$ where T is the tridiagonal matrix [17]. The vector of variables x can be solved by taking the inverse of the tridiagonal matrix [18-23] and solving the system of equations $x = T^{-1}D$ [24].

The tridiagonal matrix has the form:

$$T = \begin{bmatrix} B_1 & C_1 & & & & & \\ A_2 & B_2 & C_2 & & & R & \\ & A_3 & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & M & & & A_{N-1} & B_{N-1} & C_{N-1} \\ & & & & & A_N & B_N \end{bmatrix}, \tag{27}$$

where

$$\begin{aligned} A_j &= -1, \\ B_j &= 1 + RL_j^S + RL_j^{SS} + RL_j^{PO} + (1 + RV_j^S)\alpha_{i,j}S_{b,j}, \\ C_j &= \alpha_{i,j+1}S_{b,j+1}, \end{aligned} \tag{28}$$

and M and R, scattered in the tridiagonal matrix are for liquid return rate from the pump-around RL_{j1}^{PO} and vapor return rate from the side-stripper $\alpha_{i,j2}S_{b,j2}$.

The vector matrix D has the form:

$$D = \begin{bmatrix} f_{1,i} \\ \vdots \\ f_{j,i} \\ \vdots \\ f_{N,i} \end{bmatrix}. \tag{29}$$

The system of equations is solved for each component *i* to get *x* as vector containing liquid phase component molar rates $l_{i,j}$. Then vapor phase component molar rates $v_{i,j}$ are calculated from equation (21). Total liquid and vapor phase inner flow rates L_j, V_j and liquid and vapor phase component molar fractions $x_{i,j}$, $y_{i,j}$ are calculated as:

$$L_j = \sum_i l_{i,j}, \tag{30}$$

$$V_j = \sum_i v_{i,j}, \tag{31}$$

$$x_{i,j} = l_{i,j} / L_j \ , \tag{32}$$

$$y_{i,j} = v_{i,j} / V_j \ . \tag{33}$$

The next step is to update *K*-values of base component $K_{b,j}$ and stage temperatures T_j by:

$$K_{b,j} = \frac{1}{\sum_i \alpha_{i,j} x_{i,j}} \ , \tag{34}$$

$$T_j = \frac{B_j}{A_j - \ln K_{b,j}} \ . \tag{35}$$

The final step of the inner loop solves for the independent variables which are the stripping factor $S_{b,j}$, the withdrawal rates RL_j^S, RL_j^{SS}, RL_j^{PO}, RV_j^S and the heat duties Q_j using the energy balance equations (23) and performance specification equations (14).

The withdrawal rates and heat duties that do not exist in the column are fixed to 0 (zero) and are not included in the set of independent variables. One should remember that if there are *N* stages and a total of *n* withdrawal rates and heat duties in a system, then there are *N* energy equations and *n* performance specification equations.

Since this step uses the simplified functions for *K*-values and excess enthalpies, and the first iteration begun with roughly initialized variables, continuing from this point on will degenerate the solution [25]. As a result, only in the first iteration, the inner loop ends after calculating equation (35) and the algorithm returns back to the beginning of the outer loop with the updated values of total liquid and vapor phase inner flow rates L_j, V_j, liquid and vapor phase component molar fractions $x_{i,j}$, $y_{i,j}$ and stage temperatures T_j. In intermediate iterations, the inner loop continues with final step.

In the final step, Newton method is used where the residuals of energy balance and performance specification equations are minimized by perturbing each of the independent variables and recalculating the residuals starting from the beginning of the inner loop to get partial derivatives of the equations. The Jacobian matrix is formed from the partial derivatives as:

$$J = \begin{bmatrix} \frac{\partial E_1}{\partial S_{b,1}} & \frac{\partial E_1}{\partial S_{b,2}} & \cdots & \frac{\partial E_1}{\partial S_{b,N}} & \frac{\partial E_1}{\partial RL_1^S} & \cdots & \frac{\partial E_1}{\partial Q_N} \\ \frac{\partial E_2}{\partial S_{b,1}} & \frac{\partial E_2}{\partial S_{b,2}} & \cdots & \frac{\partial E_1}{\partial S_{b,N}} & \frac{\partial E_1}{\partial RL_1^S} & \cdots & \frac{\partial E_2}{\partial Q_N} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial E_N}{\partial S_{b,1}} & \frac{\partial E_N}{\partial S_{b,2}} & \cdots & \frac{\partial E_N}{\partial S_{b,N}} & \frac{\partial E_N}{\partial RL_1^S} & \cdots & \frac{\partial E_N}{\partial Q_N} \\ \frac{\partial G_1}{\partial S_{b,1}} & \frac{\partial G_1}{\partial S_{b,2}} & \cdots & \frac{\partial G_1}{\partial S_{b,N}} & \frac{\partial G_1}{\partial RL_1^S} & \cdots & \frac{\partial G_1}{\partial Q_N} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial G_n}{\partial S_{b,1}} & \frac{\partial G_n}{\partial S_{b,2}} & \cdots & \frac{\partial G_n}{\partial S_{b,N}} & \frac{\partial G_n}{\partial RL_1^S} & \cdots & \frac{\partial G_n}{\partial Q_N} \end{bmatrix}. \quad (36)$$

Then the independent variables are updated by the use of the inverse of the Jacobian matrix as:

$$\begin{bmatrix} S_{b,1} \\ S_{b,2} \\ \vdots \\ S_{b,N} \\ RL_1^S \\ \vdots \\ Q_N \end{bmatrix}_{k+1} = \begin{bmatrix} S_{b,1} \\ S_{b,2} \\ \vdots \\ S_{b,N} \\ RL_1^S \\ \vdots \\ Q_N \end{bmatrix}_k - bJ^{-1} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \\ G_1 \\ \vdots \\ G_n \end{bmatrix}_k , \quad (37)$$

where *k* is the iteration number and *b* is the damping factor for the correction and a value of 0.7 is recommended for most cases [25] for faster convergence. One may also optimize the value of *b* at each inner loop iteration which may be computationally costly or halve the value if the new residuals are larger and continue halving until the new residuals are smaller.

The inner loop is iterated until the residuals reach a tolerance value or an iteration limit is reached. When the inner loop ends, the algorithm returns back to the beginning of the outer loop.

2.3.1.8 Discussion

A rigorous model for simulation of a distillation column gives detailed output of what is going on in the system, once the distillation column is properly defined. When the simulation is converged, the 95% distillation point temperature of heavy diesel can be easily calculated from liquid phase molar fraction $x_{i,j}$ from the bottom stage of the heavy diesel side-stripper and boiling point temperatures of the components. However the proper definition of the real system needs an expert in this subject.

This section gives a moderate detail on rigorously modeling a distillation column. However one needs additional calculations or libraries for fully describing an atmospheric distillation column. The rigorous model should include the ability to define a third phase that mostly consists of water at the condenser drum or a total condenser that is not at equilibrium state or have further estimation methods for predicting hydrocarbon properties from the simulation results.

A rigorous model may prove to be computationally costly, hard to model and need maintenance which makes statistical models more attractive.

2.3.2 Statistical Models

Statistical models are widely used for prediction of the relationship between a dependent variable and one or more independent variables. There are many methods for generating a statistical model and the most common among them is the least squares regression method which may give misrepresentative results with outliers and noise in the training data and may not generalize well to unseen testing data.

Recent studies employ Machine Learning and Artificial Neural Network in Machine Learning has been widely adopted in solving regression problems. The concept and its use in chemical engineering is discussed in the next subsection 2.3.2.1.

The main subject of this study, Support Vector Machines in Machine Learning have been used for classification purposes like text recognition and recent studies have adapted the concept in regression problems. The background and its use in chemical engineering is discussed in subsection 2.3.2.2.

2.3.2.1 Artificial Neural Network

Artificial neural networks (ANN) have been a popular choice for simulation of chemical processes [26-28] and predicting of physical properties [29, 30]. Since ANN is a general purpose statistical modeling method, it has been applied to the simulation of hydrocracking units [31], desulphurization units [32], delayed coker units [33], fluid catalytic cracking units [34] and crude distillation units [35-38]. ANN requires the regression of a model for each dependent variable to be estimated unlike a single rigorous model. López and Mahecha [35] uses rigorous models to generate data to use in regression of ANN models rather than using plant data that has outliers and noises. The purpose here is to make the estimation faster and more robust. Liau [36], Motlaghi [37] and Ochoa-Estopier [38] further optimizes the operations of a crude distillation columns which was modeled with ANN from plant data.

Artificial Neural Networks are formed from a group of neurons interconnected to create a network that mimics the function of a brain. Input variables are connected to input nodes and these input nodes are connected to intermediate nodes and then to output nodes as depicted in Figure 2.5. There may be more than one layer of intermediate nodes, which are called hidden layers. Each connection has a weight and each node has an activation function and an output function.

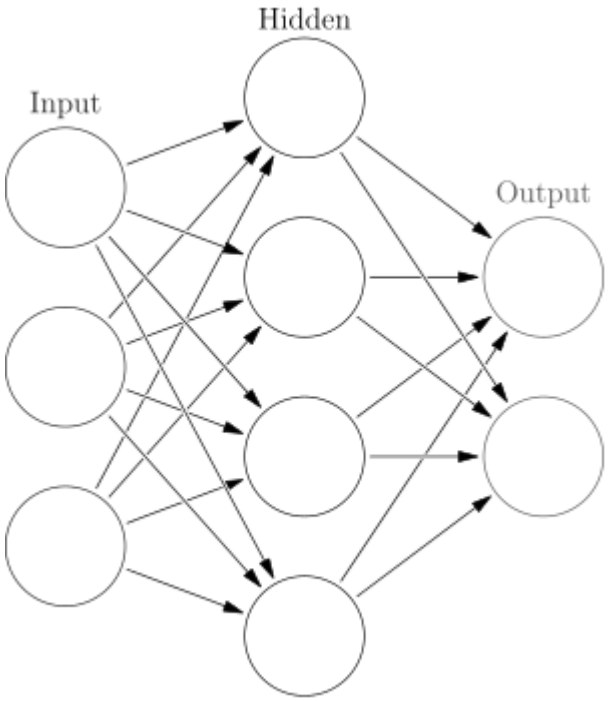

Figure 2.5: Typical structure of an Artificial Neural Network

An activation function takes the inputs and multiplies them by their interconnection weights as:

$$A_j = \sum_i x_i w_{j,i}, \tag{38}$$

where A_j is the calculated value of the activation function in node *j*, x_i is the incoming outputs from other nodes *i* and $w_{j,i}$ is the weight of the interconnection from node *i* to node *j*.

The output from node *j* may be equal to A_j, making the node a linear one, but this defeats the purpose of a neural network and has limitations. For this reason, ANN nodes also employ an output function in terms of its activation function. The most common output function is the sigmoidal function [39]:

$$O_j = \frac{1}{1+e^{A_j}}, \tag{39}$$

where O_j is the final value of the output from node *j*. The sigmoidal function has a plot as in Figure 2.6 and is close to 0 (zero) for large negative numbers and 1 (one) for large positive numbers and has a smooth transition for values of A_j close to 0 (zero).

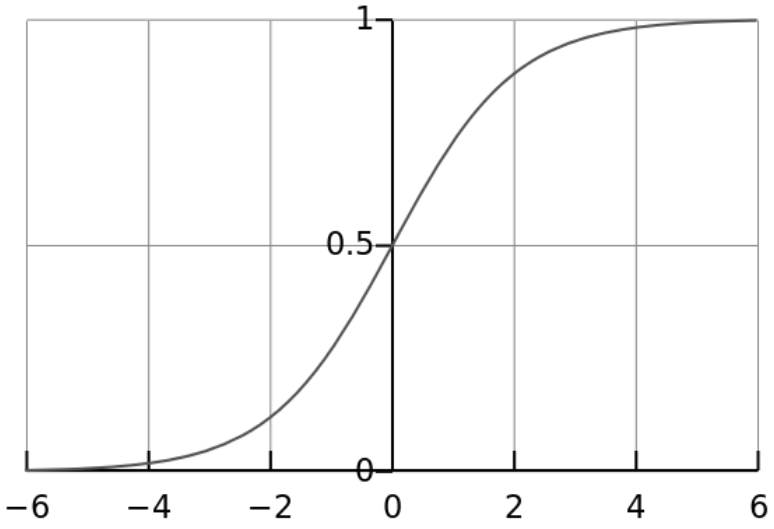

Figure 2.6: Plot of a sigmoidal function

A function similar to the sigmoidal function, the hyperbolic tangent function has been used by [38] which has an output value closer to -1 (negative one) for large negative numbers instead of 0 (zero). This function has the form:

$$\mathrm{O}_j = \tanh\ A_j = \frac{\sinh\ A_j}{\cosh\ A_j} = \frac{e^{A_j} - e^{-A_j}}{e^{A_j} + e^{-A_j}} = \frac{1+e^{-2A_j}}{1-e^{-2A_j}}. \tag{40}$$

Sigmoidal, hyperbolic tangent or similar functions are used for the nodes in the hidden layers, while a linear function is used for the nodes in the output layer.

The structure of the network and the activation function are predetermined usually by trial and error for simplicity. Then the Artificial Neural Network is trained to decrease

the error between the training data and the output of the Artificial Neural Network. This is done by adjusting the weights of the interconnections to minimize the sum of the squares of the errors. The Backpropagation Algorithm proposed by Rumelhart and McClelland [40] is used for the training where the errors are propagated backwards to find the effect of a weight in previous layers.

Since the Artificial Neural Networks minimize the estimation error, this approach usually leads to over-fitting and poor generalization of the system meaning poor estimation of unseen testing data.

2.3.2.2 Support Vector Regression

The concept of Support Vector Machines (SVM) has been established by Vladimir N. Vapnik and Alexey Y. Chervonenkis [41, 42] in 1963 for solving classification problems. Then nonlinear classification has been made possible by the introduction of kernel functions, proposed by Boser, Guyon and Vapnik [43] in 1992 which has been suggested by Aizerman [44] back in 1964. The problem of outliers have been finally solved by soft margins which was proposed by Cortes and Vapnik [45] in 1995. Then SVM was updated to solve regression problems with the introduction of a loss function, proposed by Vapnik, Drucker, Burges, Kaufman and Smola [46] in 1997.

Interest in SVM has increased over the last 15 years due to its distinctive features and performance in solving regression and classification problems. A detailed technical report on SVM for its use in classification and regression problems has been published by Gunn [47] in 1998. Detailed tutorials on Support Vector Regression (SVR) were published by Smola and Schölkopf [48, 49] in 1998 and 2004. A general purpose library has been distributed by Chang and Lin [50] as LIBSVM that can be integrated to many programming languages and data mining software.

SVM has also been used in modeling of chemical processes like polymerization [51], desulfurization [52], polymer extrusion [53] and isomerization [54]. Yao and Chu [55] employed SVM in modeling and optimization of an atmospheric distillation column. Yan, Shao and Wang [56] developed a soft sensor for estimating the freezing point of light diesel in a distillation column. Lahiri and Ghanta [57] employed SVR in estimating critical velocity of solid liquid slurry flow.

Yao and Chu [55] applied SVR for a real atmospheric distillation column and optimized the SVR parameters using particle swarm optimizer with linearly decreased inertia weight (LinWPSO), a version of a global optimizer proposed by Shi and Eberhart [58]. Lahiri and Ghanta [57] optimized the SVR parameters using genetic algorithm (GA). Yao and Chu [55] modeled the atmospheric distillation column in a commercial chemical process simulation tool, Aspen Plus and used data generated from case studies. Since the training data contained simulation results, there was neither noise nor outlier. Therefore the SVR parameter ε to be explained below was fixed to 0 (zero) which defeated the purpose of creating scarcity in the set of support vectors.

The difference between SVM and the other regression methods is that SVM minimizes the structural risk as opposed to other regression methods that employed empirical risk minimization. Empirical risk minimization only focuses on minimizing the error of estimation which may end up with a model that describes the noise or outliers, and will have poor performance in predicting unseen test data, whereas structural risk minimization focuses on training a function that is as flat as possible.

In the regression case, SVR has two main parameters ε and C that affect the balance between flatness versus error of estimation. Parameter ε describes the insensitive region in which the error of estimation has no cost, and parameter C describes the weight of error outside the insensitive region. When kernel functions are used, one or more extra parameters may be introduced. A large value for parameter C will make the algorithm put all training error within the insensitive region which will cause the SVR to memorize the training data and generalize poorly to unseen test data. A small value for parameter ε

will assume that there is no noise in the data and again will cause the SVR to overfit. In contrast a small value for parameter C or a large value for parameter ε will cause the SVR to underfit. Applying cross validation method in optimizing these parameters will ensure the regression of a function that has great performance in prediction of unseen data.

The next chapter gives comprehensive details on the concept of Support Vector Regression, how SVR is trained and how the parameters of SVR can be optimized.

Chapter 3

SUPPORT VECTOR REGRESSION

3.1 The Concept

Suppose we have a set of data $\{(x_1, y_1), (x_2, y_2) ..., (x_l, y_l)\}$, $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$, where x_i can be a scalar or a vector containing the input values and y_i is the corresponding output value. In this study, the data set will contain the control parameters of a distillation column as x, and one of the hydrocarbon properties to be estimated as y. In Support Vector Regression, we want to estimate y with a function $f(x)$ that has least deviation from the data, and at the same time is as flat as possible and has lowest model complexity.

The estimation function has the form,

$$f(x) = \langle w, x \rangle + b, \quad w \in \mathbb{R}^n, \quad b \in \mathbb{R}. \tag{41}$$

where w is a combination of the support vectors that have been selected among the data set, $\langle ., . \rangle$ demotes the dot product, and b is the bias term. The use of the dot product leads to a linear estimation function, but it is possible to use kernel functions that construct a mapping into a high dimensional feature space which enables the linear regression of a non-linear estimation function. Kernel functions will be discussed later in Section 3.6. We will continue with the linear function in this section for simplicity.

A special loss function has been introduced to Support Vector Regression similar to, but different than widely known loss functions like Quadratic, Laplace or Huber

functions. This loss function is insensitive to errors less than an error tolerance of ε which enables sparseness at the set of support vectors [45]. This new loss function is named ε-insensitive function and will be further discussed together with the other mentioned loss functions in Section 3.7.

The first attempt at Support Vector Regression was with a hard constraint where the goal was to estimate y with a function $f(x)$ that has at most ε deviation from the training data and at the same time has lowest model complexity and is as flat as possible. This means any error lower than ε is neglected, while any error more than ε is not tolerated. For flatness and lower model complexity, one should pursue a small combination of support vectors, w. To do this we can minimize the function $\|w\|^2$ which is the dot product, $\langle w, w \rangle$. The error of estimation will have a hard constraint and the optimization problem for regression can be written as:

$$\begin{aligned} &\min \quad \frac{1}{2}\|w\|^2 \\ &s.t. \begin{cases} y_i - f(x_i) \leq \varepsilon \\ f(x_i) - y_i \leq \varepsilon. \end{cases} \end{aligned} \tag{42}$$

This optimization problem will estimate all input and output pairs within a tolerance of ε. However this problem may not always have a feasible solution or we may want to tolerate some degree of error. Thus the soft constrained loss function of Bennett and Mangasarian [59] was adapted into support vector machines by Vapnik [45]. Slack variables, ξ_i and ξ_i^* and a scalar constant C was introduced to the optimization problem. Slack variables are for the positive or the negative errors of estimation outside of the insensitive region. These slack variables need to be minimized according to the user-defined positive weight factor C, which determines the tradeoff between the flatness of the estimation function and the error tolerance. The resulting optimization problem for regression can be written as:

$$\min \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\left(\xi_i + \xi_i^*\right),$$
$$s.t.\begin{cases} y_i - f(x_i) \le \varepsilon + \xi_i, \\ f(x_i) - y_i \le \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \ge 0. \end{cases} \tag{43}$$

Some kernels may introduce one or more additional parameters to be defined beforehand, but other than that the generalization performance of the estimation function is greatly dependent on the parameters C and ε, and will be discussed further in Section 3.9.

A case for training a linear estimation function is shown in Figure 3.1 [49], where the loss is $|\xi|$ if the estimation error is $\varepsilon + \xi$, and the loss is 0 (zero) if the training data is in the grey region so that the estimation error is less than ε.

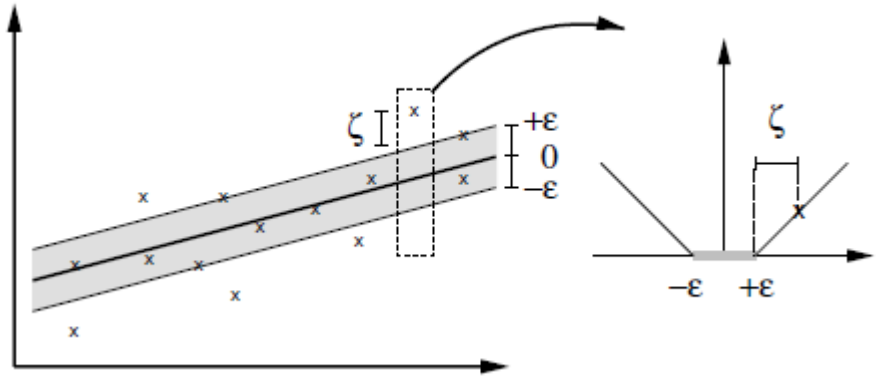

Figure 3.1: Soft constraint loss function

The optimization problem (43) for support vector regression can be easily solved in its dual form. The dual problem can be formed by the method of Lagrange multipliers which is a standard dualization method and explained by Fletcher [60].

3.2 The Dual Problem

First of all the primal form is converted into a Lagrange function where the constraints of the primal problem are added to the objective function by the introduction of Lagrange multipliers for each of the constraints. The Lagrange function can be written as:

$$\begin{aligned} f_L \quad &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\left(\xi_i + \xi_i^*\right) \\ &-\sum_{i=1}^{l}\alpha_i\left(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b\right) \\ &-\sum_{i=1}^{l}\alpha_i^*\left(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b\right) \\ &-\sum_{i=1}^{l}\left(\eta_i\xi_i\right) \\ &-\sum_{i=1}^{l}\left(\eta_i^*\xi_i^*\right). \end{aligned} \tag{44}$$

The Lagrange multipliers are $\eta_i,\ \eta_i^*,\ \alpha_i,\ \alpha_i^*$ and they are nonnegative:

$$\eta_i,\ \eta_i^*,\ \alpha_i,\ \alpha_i^* \geq 0. \tag{45}$$

The Lagrange function has a saddle point where the partial derivatives of the Lagrange function with respect to the primal variables $w,\ b,\ \xi_i,\ \xi_i^*$, are equal to 0 (zero) at the optimal point:

$$\frac{\partial L}{\partial w} \quad = \quad 0 \quad = \quad w - \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right)x_i, \tag{46}$$

$$\frac{\partial L}{\partial b} \quad = \quad 0 \quad = \quad \sum_{i=1}^{l}\left(\alpha_i - \alpha_i^*\right), \tag{47}$$

$$\frac{\partial L}{\partial \xi_i} \quad = \quad 0 \quad = \quad C - \alpha_i - \eta_i, \tag{48}$$

$$\frac{\partial L}{\partial \xi_i^*} = 0 = C - \alpha_i^* - \eta_i^*. \quad (49)$$

The optimality conditions above helps us simplify the dual problem to be in terms of dual variables which are the Lagrange multipliers. In this case, the equations (48) and (49) can be used to eliminate the Lagrange multipliers η_i, η_i^* by rearranging the equations as:

$$\begin{aligned} \eta_i &= C - \alpha_i, \\ \eta_i^* &= C - \alpha_i^*. \end{aligned} \quad (50)$$

Since the Lagrange multipliers have to be nonnegative, the Lagrange multipliers α_i, α_i^* cannot be larger than C to comply with equalities (50). Equation (47) becomes a constraint for the dual problem. We also see that the equation (46) can be rearranged to get the term w as a combination of the input set of the training data:

$$w = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^* \right) x_i. \quad (51)$$

By substituting equation (51) into equation (41), we can rewrite the estimation function as:

$$f(x) = \sum_{i=1}^{l} \left(\left(\alpha_i - \alpha_i^* \right) \langle x_i, x \rangle \right) + b. \quad (52)$$

By substituting equations (50), (51) into equation (44) with constraints deduced from equations (47) and (50), the primal variables w, b, ξ_i, ξ_i^* are eliminated and we get the dual problem as:

$$\begin{aligned} \max\ & -\frac{1}{2}\sum_{i=1}^{l}\left(\sum_{j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,\, x_j\right\rangle\right) \\ & -\varepsilon\sum_{i=1}^{l}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{l} y_i\left(\alpha_i-\alpha_i^*\right), \\ st.\ & \begin{cases} \sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)=0, \\ 0\le\alpha_i,\,\alpha_i^*\le C. \end{cases} \end{aligned} \tag{53}$$

Now that the optimization problem does not include w, support vector regression problems especially with nonlinear feature space transformations can be solved independently of the dimension of the feature space with quadratic programming.

3.3 Training with Quadratic Programming

Quadratic programming minimizes a quadratic objective function with decision variables subject to linear equality and inequality constraints.

$$\begin{aligned} & \min\ \frac{1}{2}x^T Hx+f^T x, \\ & s.t.\begin{cases} Ax\le b, \\ A_{eq}x=b_{eq}, \\ lb\le x\le ub. \end{cases} \end{aligned} \tag{54}$$

The dual problem is transformed into a minimization problem:

$$\begin{aligned} \min\ & \frac{1}{2}\sum_{i=1}^{l}\left(\sum_{j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)\left\langle x_i,\, x_j\right\rangle\right) \\ & +\varepsilon\sum_{i=1}^{l}\left(\alpha_i+\alpha_i^*\right)-\sum_{i=1}^{l} y_i\left(\alpha_i-\alpha_i^*\right), \\ st.\ & \begin{cases} \sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)=0, \\ 0\le\alpha_i,\,\alpha_i^*\le C. \end{cases} \end{aligned} \tag{55}$$

We need to solve for the optimum values for the Lagrange multipliers, α_i and α_i^*. This study uses $\beta_i = \alpha_i - \alpha_i^*$, and $\beta_i^* = \alpha_i + \alpha_i^*$ as decision variables in the quadratic programming to create sparseness in the *H* matrix for faster convergence. The size of decision variable vector is 2*l*, two times the number of training samples. The problem is reformulated as,

$$\begin{aligned} &\min \ \frac{1}{2}\sum_{i=1}^{l}\left(\sum_{j=1}^{l}\beta_i\beta_j\left\langle x_i, x_j\right\rangle\right)+\varepsilon\sum_{i=1}^{l}\beta_i^*-\sum_{i=1}^{l}y_i\beta_i, \\ &st.\begin{cases}\sum_{i=1}^{l}\beta_i = 0, \\ 0 \le (\beta_i + \beta_i^*),\ (\beta_i^* - \beta_i) \le 2C, \\ -C \le \beta_i \le C, \\ 0 \le \beta_i^* \le 2C.\end{cases} \end{aligned} \tag{56}$$

First we need to calculate the dot product matrix *M* which is of the size $l \ x \ l$ as below for linear functions,

$$M_{l \text{ x } l} = \begin{bmatrix} \left\langle x_1, x_1\right\rangle & \left\langle x_1, x_2\right\rangle & \cdots & \left\langle x_1, x_l\right\rangle \\ \left\langle x_2, x_1\right\rangle & \left\langle x_2, x_2\right\rangle & \cdots & \left\langle x_2, x_l\right\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \left\langle x_l, x_1\right\rangle & \left\langle x_l, x_2\right\rangle & \cdots & \left\langle x_l, x_l\right\rangle \end{bmatrix}. \tag{57}$$

Then the *H* matrix of the quadratic programming is written as,

$$H_{2l \text{ x } 2l} = \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}, \tag{58}$$

and the *f* matrix is

$$f_{2l \text{ x } 1} = \begin{bmatrix} -y_1 \\ \vdots \\ -y_l \\ \varepsilon \\ \vdots \\ \varepsilon \end{bmatrix}. \tag{59}$$

Then the matrixes A, A_{eq} and vectors b, b_{eq} are formed according to the constraints in equation (56).

3.4 The Support Vectors

The Karush-Kuhn-Tucker (KKT) conditions or complementary slackness conditions state that at the optimum point, the product between dual variables and primal constraints should be equal to zero:

$$\begin{aligned} \alpha_i \left(\varepsilon + \xi_i - y_i + \left\langle w, x_i \right\rangle + b \right) &= 0, \\ \alpha_i^* \left(\varepsilon + \xi_i^* + y_i - \left\langle w, x_i \right\rangle - b \right) &= 0. \end{aligned} \tag{60}$$

Also the product of primal variables and dual constraints should be equal to zero:

$$\begin{aligned} \xi_i \left(C - \alpha_i \right) &= 0, \\ \xi_i^* \left(C - \alpha_i^* \right) &= 0. \end{aligned} \tag{61}$$

From the equation (60), we can see that $\alpha_i \alpha_i^* = 0$, meaning at least one of α_i or α_i^* should be equal to zero. From the equation (61), training data with $\alpha_i = C$ or $\alpha_i^* = C$ can have positive primal slack variables ξ_i or ξ_i^*, meaning the training data can be outside of the ε-insensitive zone if and only if α_i or α_i^* is equal to C.

Remember that in equation (51), the term w is expanded into a combination of the input vectors. From the equation (60), when the estimation error, $|f(x_i)-y_i|$ is lower than ε, both Lagrange multipliers α_i and α_i^* should be equal to zero, since the slack variables have to be nonnegative. Therefore $(\alpha_i-\alpha_i^*)$ vanishes in equation (51) which creates scarcity in the expansion of w. As a conclusion, the training data with estimation error, $|f(x_i)-y_i|\geq\varepsilon$ have one of the Lagrange multipliers with a nonzero value which makes them Support Vectors. From now on the support vectors are x_s for $s=1, 2, \ldots, S \leq l$, and have the coefficient $\beta_s=(\alpha_s-\alpha_s^*)$. The estimation function is rewritten as:

$$f(x)=\sum_{s=1}^{S}\left(\beta_s\langle x_s, x\rangle\right)+b. \tag{62}$$

Although we expect one or both of the Lagrange multipliers equal to zero at optimal point, quadratic programming solves them to be very small numbers close to zero, so support vectors are selected by the relation:

$$(\alpha_i-\alpha_i^*)>10^{-6}C. \tag{63}$$

3.5 The Bias Term

We will find the bias from the average of support vectors with estimation error equal to ε. These support vectors are x_{ss} and are identified from x_s with $0<|\beta_s|<C$. To filter out very small numbers the relation is updated to:

$$10^{-6}C<|\beta_s|<(1-10^{-6})C. \tag{64}$$

Finally the bias term is calculated by:

$$b = \frac{1}{SS} \sum_{ss=1}^{SS} \left(y_{ss} - \varepsilon\sigma - \beta_{ss} \sum_{s=1}^{S} \langle x_{ss}, x_s \rangle \right), \tag{65}$$

where σ is equal to 1 if $\beta_{ss} > 0$ or -1 otherwise.

3.6 Kernel Functions

Kernel functions are used to construct a mapping into a high dimensional feature space which enables the linear regression of a non-linear function, meaning that mapping is preprocessed and SVR is repurposed to find the flattest function in this feature space.

The input data x_i is mapped into $\phi(x_i)$, but we do not need to define $\phi(x_i)$ explicitly since SVR only needs the dot product in the regression procedure and in the final estimation function. Hence we may define and use the kernel function as:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle. \tag{66}$$

In this case, we may rewrite the SVR problem in equation (55) as:

$$\begin{aligned} &\min \quad \frac{1}{2} \sum_{i=1}^{l} \left(\sum_{j=1}^{l} \beta_i \beta_j K(x_i, x_j) \right) + \varepsilon \sum_{i=1}^{l} \beta_i^* - \sum_{i=1}^{l} y_i \beta_i, \\ &st. \begin{cases} \sum_{i=1}^{l} \beta_i = 0, \\ 0 \le (\beta_i + \beta_i^*), (\beta_i^* - \beta_i) \le 2C, \\ -C \le \beta_i \le C, \\ 0 \le \beta_i^* \le 2C. \end{cases} \end{aligned} \tag{67}$$

The equation (51) for term w is expanded to:

$$w = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^* \right) \phi(x_i). \tag{68}$$

The matrix M in quadratic programming is rewritten as:

$$M_{l\,x\,l} = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_l) \\ K(x_2, x_1) & K(x_2, x_2) & \cdots & K(x_2, x_l) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_l, x_1) & K(x_l, x_2) & \cdots & K(x_l, x_l) \end{bmatrix}. \tag{69}$$

Finally the estimation function in equation (62) is rewritten as:

$$f(x) = \sum_{s=1}^{S} \left(\beta_s K(x_s, x) \right) + b, \tag{70}$$

where the equation (65) for bias term is revised to:

$$b = \frac{1}{SS} \sum_{ss=1}^{SS} \left(y_{ss} - \varepsilon\sigma - \beta_{ss} \sum_{s=1}^{S} K(x_{ss}, x_s) \right). \tag{71}$$

It should be noted that the mapping into high dimensional feature space leads to the curse of dimensionality, that is, as dimensionality increases, the volume of space increases that the training data becomes sparse in this high dimensional space. To overcome the curse of dimensionality, a large training set with a good data distribution should be provided to SVR [47].

There are various kinds of kernels that can be applied to SVR and combinations of these kernels are possible. Nonlinear kernels may have parameters that are predefined by the user like the main parameters of the SVR, ε and C.

3.6.1 Linear Kernel

This is actually the dot product of the vectors and no mapping is applied. Since the SVR will be coded with a kernel function, linear kernel will be among choices.

$$K(x_i, x_j) = \left\langle x_i, x_j \right\rangle. \tag{72}$$

3.6.2 Polynomial Kernel

Polynomial kernel is among popular choices for nonlinear modelling and is formulated as:

$$K(x_i, x_j) = \left(\langle x_i, x_j \rangle + 1 \right)^d , \tag{73}$$

where $d \in \mathbb{N}$ is the kernel parameter for the degree of the polynomial. The addition of 1 (one) prevents the Hessian from becoming zero.

3.6.3 Gaussian Radial Basis Function Kernel

Gaussian radial basis function (RBF), which uses the square of the Euclidean distance between the two vectors, is also widely used in SVM for its flexibility. The Gaussian RBF kernel is defined as:

$$K(x_i, x_j) = \exp\left(- \frac{\| x_i - x_j \|^2}{2\sigma^2} \right). \tag{74}$$

This function can be simplified to:

$$K(x_i, x_j) = \exp\left(-\gamma \| x_i - x_j \|^2 \right), \tag{75}$$

where $\gamma = 1/2\sigma^2$, $\gamma \in \mathbb{R}$, $\gamma > 0$ is the kernel parameter.

3.6.4 Exponential Radial Basis Function Kernel

Exponential RBF, which employs the first degree Euclidean distance between the two vectors, generates a piecewise linear solution and is defined as:

$$K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|\right), \tag{76}$$

where $\gamma = 1/2\sigma^2$, $\gamma \in \mathbb{R}, \gamma > 0$ is the kernel parameter.

3.6.5 Other Kernels

There are other types of kernels that have limited output range and are more suitable for classification purposes. These kernels and their descriptions are stated in Table 3.1.

Table 3.1: Other types of kernels not discussed in this paper

Kernel	Description
Hyperbolic Tangent Function	Defined on interval [-1, 1]. Very popular in ANN applications, but not applicable in this study.
Sigmoidal Function	Defined on interval [0, 1]. Very popular in ANN applications, but not applicable in this study.
Fourier Series	Defined on interval [-π/2, π/2]. Poor performance in regularization and not applicable in this study.
Splines	Defined on interval [0, 1). Very complex kernel and not applicable in this study.
B-splines	Defined on interval [-1, 1]. Not applicable in this study.

3.6.6 Sum of Kernels

More complex kernels can be formed by summation of two or more kernels as:

$$K(x_i, x_j) = \sum_k K_k(x_i, x_j). \tag{77}$$

3.6.7 Product of Kernels

Similarly kernels can be formed by taking products of two or more kernels as:

$$K(x_i, x_j) = \prod_k K_k(x_i, x_j). \tag{78}$$

3.7 Loss Functions

This section studies four types of loss function which penalizes the distance between the estimation function and the training data. Figure 3.2 illustrates these four functions. Any of the loss function mentioned in this section can be integrated into SVR. However the loss function with best generalization performance and that creates sparseness in the set of support vectors has been employed in this study.

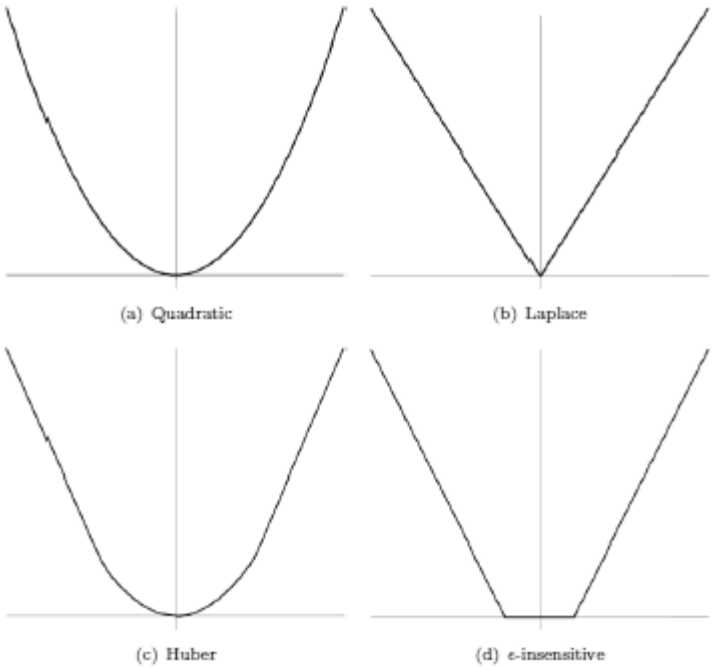

Figure 3.2: Loss Functions

3.7.1 Quadratic Loss Function

Quadratic loss function in Figure 3.2(a) is actually the squared error of estimation and is commonly used for regression. The quadratic loss function is sensitive to outliers, leading to a sway from the mean by heavy-tailed distributions.

3.7.2 Laplace Loss Function

On the other hand, the Laplace loss function in Figure 3.2(b) is the absolute value of the estimation error, which is less sensitive to outliers. However Laplace loss function cannot be differentiated at error equal to zero.

3.7.3 Huber Loss Function

Huber loss function in Figure 3.2(c) is a combination of the Quadratic and Laplace loss functions. The function is quadratic for small error values and linear for large error values. The functions and their slopes are equal at the intersections of these two functions. This property combines the sensitivity of the Quadratic loss function and the robustness of the Laplace loss function. The Huber Loss Function has the form:

$$L\left(f(x)-y\right)=\begin{cases}\frac{1}{2}\left(f(x)-y\right)^2 & \text{for } \left|f(x)-y\right|<\mu,\\ \mu\left|f(x)-y\right|-\frac{\mu^2}{2} & \text{otherwise.}\end{cases} \tag{79}$$

There exist smooth approximations of Huber loss function which have single functions that mimic quadratic function for small values and linear function for larger values, however these are not discussed in this study.

3.7.4 ε-insensitive Loss Function

The mentioned three loss functions will show no sparseness if used in support vector regression, since any error even small creates a cost. The proposer of support vector machine method, Vapnik [46] has modified Huber loss function to ε-insensitive loss function in Figure 3.2(d) which has an insensitive region at small error values, leading to a sparseness at the set of support vectors. With a large training set, sparseness greatly reduces computational time. This function has parameter ε which controls the width of the insensitive region. When this parameter is equal to zero, we have the Laplace loss function. This study focuses on ε-insensitive loss function as it is mainly used by support vector machines.

3.8 Data Normalization

The rate of convergence in training the support vector model is greatly affected by the eigenvalues of the Hessian. We can improve this rate by decreasing the difference between the smallest and largest eigenvalue [47]. Therefore this study strongly recommends and applies normalization for input data to improve the rate of convergence. Some kernel functions can only be defined in restricted intervals which already require data normalization. For unrestricted kernels, this study scales data between the interval $[-1, 1]$.

Isotropic or anisotropic scaling can be applied for data normalization. With anisotropic scaling, each input variable is separately scaled between the lower bound and the upper bound. With isotropic scaling, all of the input variables are shrunk into the bounds by the same scaling factor. Figure 3.3 illustrates the scaling of two data ranges by these methods. Isotropic scaling method may be adopted when all input variables are of

the same unit. This study applies anisotropic scaling, because the input variables contain temperature and pressure readings.

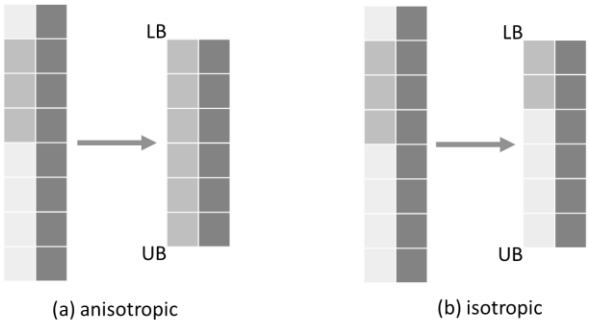

Figure 3.3: Scaling Methods

3.9 Optimizing SVR Parameters

The ε-insensitive loss function, proposed by Vapnik, Drucker, Burges, Kaufman and Smola [46] introduced the user-defined parameter ε that describes the width of the insensitive range. The value of the parameter ε directly affects the number of support vectors in the estimation function. A small value will increase model complexity and cause the SVR to select more support vectors, memorize the training data, and have poor generalization performance while a larger value will lead to fewer support vectors and a flatter estimation function, but then fail to estimate the effect of an input variable on the output variable.

Soft constrained loss function was incorporated into SVM by Cortes and Vapnik [45]. It introduces the SVR parameter C which handles the problem of outliers by creating a cost to estimation errors larger than ε. A large value will increase the model complexity and cause the SVR to tolerate less to deviations larger than ε while a small value will lead to a flatter estimation function.

Kernel functions were incorporated into SVM by Boser, Guyon and Vapnik [43] and some kernels may have user-defined parameters as explained in Section 3.6.

The user-defined parameters of SVR can be optimized by cross validation methods integrated into global optimization tools where a cross validation is applied at each evaluation called by the optimization solver.

3.9.1 Cross Validation

Cross validation methods are used for partitioning the data set into training and testing sets. The training set is used in training the estimation function while testing set is used for monitoring the performance of generalization of the estimation function. There are many kinds of cross validation methods and this study focuses on *k*-fold cross validation method, because it is a non-exhaustive method with reasonable computational cost.

k-fold cross validation method shuffles and partitions the data set into *k* equally sized subsets. One of the subsets is selected as the testing set and the rest of the subsets are used as training set. The cross validation process is repeated *k* times thus naming the method *k*-fold, where each subset is used exactly once for testing. The squared error from each iteration is averaged to give a single mean squared error (MSE). The parameter *k* is user-defined and a high value for *k* will increase the performance of generalization while also increasing computational cost.

3.9.2 Optimization Solver

Previous studies show that optimization tools like Genetic Algorithm [57] or Particle Swarm Optimizer [55] are used for optimizing SVR parameters in modeling chemical processes. Grid search or constrained nonlinear programming may also be used for this purpose.

3.9.2.1 Genetic Algorithm

Genetic Algorithm (GA) is a heuristic optimization method, belonging to the class of evolutionary algorithms. This method mimics the process of natural selection where a population of candidate solutions evolve through generations to a better solution [61].

The initial population is usually generated randomly at a user-defined size and at each generation, a predefined ratio of the population survives to see the next generation, also called the elites. Now the elites will breed via crossover and mutation at predefined ratios to complete the size of the population to form the next generation. In crossover process, elites exchange their variable values to create their children. In mutation process, the variable values of the children mutate into new values. The iteration is looped until the problem does not improve or a pre-defined generation limit is reached. Table 3.2 gives the critical parameters of GA that are needed to be tuned for a better performance. The ratios for elite selection, crossover and mutation should be tuned so that the algorithm can evolve to a better solution but without losing good solutions. Population size and generation limit should also be tuned, so that a moderate amount of evaluation is conducted at an acceptable computational cost.

Table 3.2: Tuned parameters of Genetic Algorithm

Parameter	Description
P	Population size
$G_{\max}$	Generation limit
s_E	Elite count
r_C	Cross-over ratio

3.9.2.2 Particle Swarm Optimizer

Particle Swarm Optimizer (PSO) is another member of the evolutionary algorithms class. This method has a population of particles which move in the search space by simple mathematical calculations according to their position in the space and velocity [58]. Velocity of each particle is updated at each iteration by its distance from its local best known position and also by best known positions of other particles. This updating function helps the algorithm mimic a pack of organisms moving together towards the best solution.

The particles are initialized at uniformly distributed positions x_i inside the search space. Their velocities v_i are initialized at a value uniformly distributed between negative and positive values of the size of the search space.

For each iteration the velocity of the particle is updated by:

$$v_i \leftarrow \omega v_i + \varphi_p r_p (p_i - x_i) + \varphi_s r_s (s - x_i), \tag{80}$$

where p_i is the best known position of particle *i* and s is the best known position of the swarm. r_p and r_s are random numbers on the interval $(0, 1)$ and ω, φ_p and φ_s are user-defined parameters of the PSO.

Then the position of each particle is updated by adding its velocity. Lastly, the best known position of the swarm s and of each particle p_i is updated if the condition holds.

The PSO converges when all particles converge to a point or alternatively if the generation limit is reached. As expected, the performance of PSO depends on the selection of user-defined parameters.

Table 3.3: Tuned parameters of PSO

Parameter	Description
P	Particle count
$G_{\max}$	Generation Limit
ω	Velocity factor
φ_p	Local best factor
φ_s	Swarm best factor

3.9.2.3 Grid Search Algorithm

Grid Search (GS) method is another tool popularly used for optimizing SVR parameters [62]. This method applies a mesh on the search space and re-meshes around the best known solutions until the best solution improves under a given tolerance or the mesh size is under a given tolerance.

The performance of the grid search methods highly depends on the resolution of the mesh and search method. The grid search method may miss a good solution due to low resolution of the mesh. However a high resolution will have high computational cost since each edge of the mesh should be evaluated.

3.9.2.4 Constrained Nonlinear Programming

Nonlinear Programming (NLP) starts at an initial point and moves to a better solution at each iteration by perturbing the variables. NLP is applied to convex problems since the algorithm stops at the first locally optimal solution it encounters. For non-convex problems, a good initial point is required.

Chapter 4

ESTIMATING HYDROCARBON PROPERTIES

4.1 The Data

The atmospheric distillation column (ADC) subject to this study is an actual operating unit in İzmit Refinery of Turkish Petroleum Refineries Corporation. The ADC together with its pump-arounds, side-strippers and condenser drum is depicted in Figure 4.1.

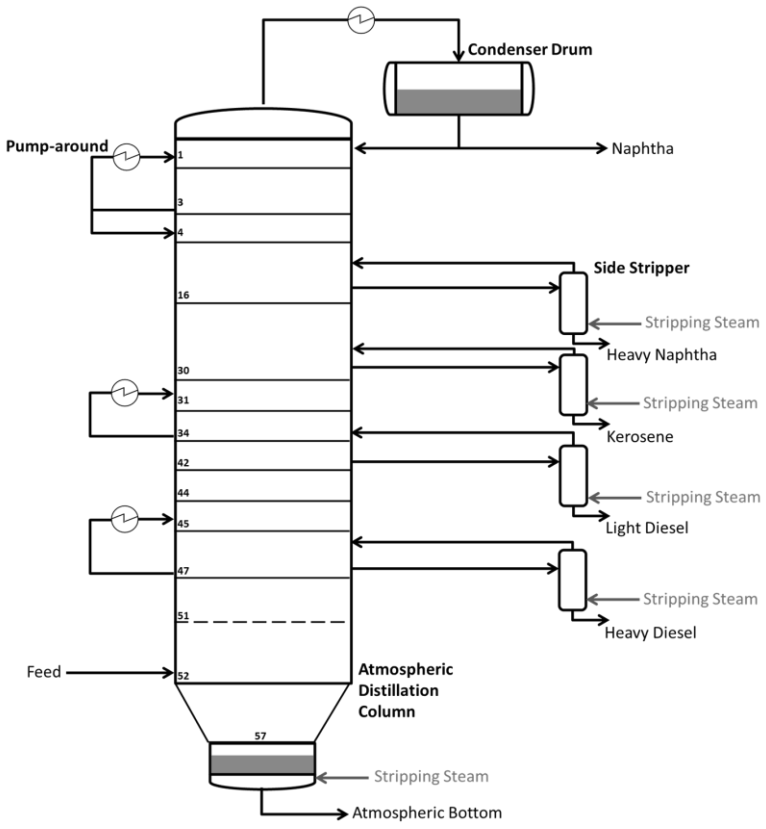

Figure 4.1: Atmospheric Distillation Column

The crude distillation unit as a total was designed for processing crude oil from Kirkuk, Iraq, but currently processes different kinds of crude oils. A pre-flash column separates LPG and Light Naphtha and sends the remaining stream to the ADC. Feed to ADC is heated up to around 350˚C before entering the flash zone. Then the ADS separates the feed into Naphtha, Heavy Naphtha, Kerosene, Light and Heavy Diesel and Atmospheric Bottom.

There are critical hydrocarbon properties for the products of the atmospheric distillation column as given in Table 4.1 and these properties are optimized daily by the Planning Department according to changing prices and demand. The plant operators receive the optimal values and monitor these properties together with online plant measurements like temperature, pressure and flowrates. There are no online analyzers for the measurement of the critical hydrocarbon properties, but instead a sample from each of the streams is collected three times a day and is analyzed at the laboratory. There are also robust quality estimators (RQE) which are nonlinear regression functions employed for online prediction of the hydrocarbon properties.

Table 4.1: Critical Hydrocarbon Properties

Product	Property
Light Naphtha	Distillation curve 5% point
Light Naphtha	Distillation curve 95% point
Heavy Naphtha	Flash point
Kerosene	Flash point
Kerosene	Distillation curve 95% point
Light Diesel	Distillation curve 95% point
Heavy Diesel	Distillation curve 95% point

The distillation curve in Figure 4.2 gives a plot of temperature versus fraction of the liquid boiled at that temperature. The most critical properties are the temperature at which 5% and 95% of the sample boils. The distillation column cannot perfectly separate the products, so the lighter end of the heavy product can be seen in the lighter product and

via versa. The transitioning boiling point temperature between two products is called the cut point. The cut point can be controlled by manipulating the flowrate of a side-drawn product. If the flowrate of the lighter product is increased, more of the heavier product moves up the tray of the light product, leading to an increase in the tray temperature and the 95% point temperature of the light product. As a result the production rate of a product can be maximized within specification limits for the 5% and 95% point temperatures.

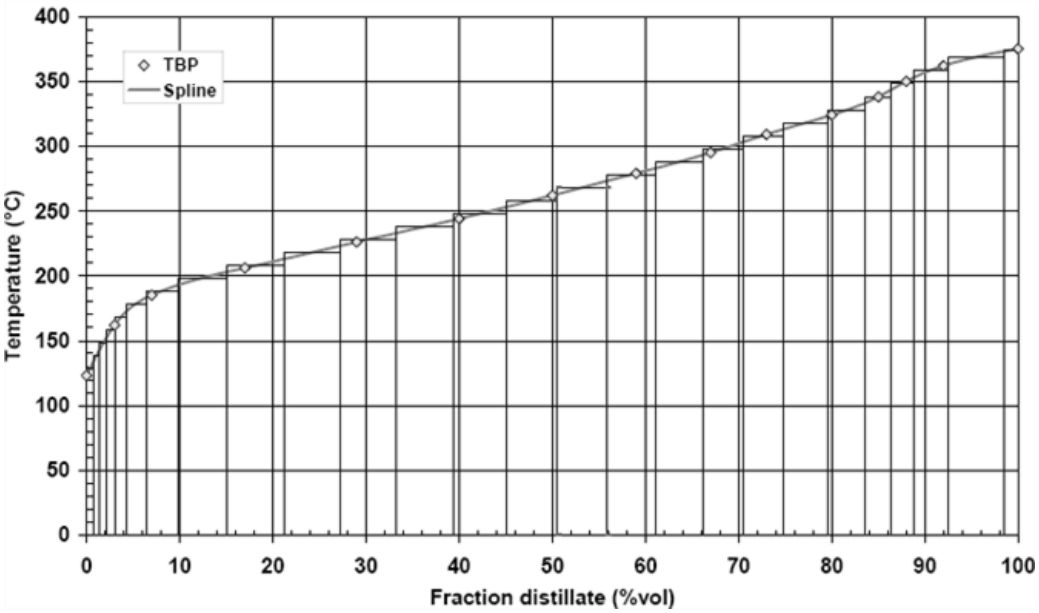

Figure 4.2: Distillation Curve of Diesel

This study focuses on estimation of the 95% point temperature of heavy diesel since this property should be at maximum specification value or valuable fraction of heavy diesel is lost to atmospheric bottom. This property can be estimated by the flash zone temperature and pressure and the heavy diesel tray temperature. The variables used in SVR and RQE is given in Table 4.2.

Table 4.2: Variables used for regression

Type	Variable
output	Heavy diesel 95% point
input	Flash zone temperature
input	Flash zone pressure
input	Heavy diesel tray temperature

Online plant data as well as laboratory analysis results can be reached from process historical database which has years of data stored. The laboratory analysis data was gathered as raw data since it is stored three times a day, while online plant data was gathered as the average of thirty minutes of raw data for smoothening noise. A sample is gathered and analyzed only when the ADC is in stable operation condition and the corresponding online measurements with thirty minutes period ending with the timestamp of the sample were gathered.

4.2 The Procedure

The algorithm for normalizing and splitting data, SVR training and SVR parameter optimization was coded in MATLAB®. Quadratic Programming tool of MATLAB® was employed in SVR training and Cross Validation, Genetic Algorithm, Pattern Search and Constrained Nonlinear Programming tools of MATLAB® were employed in SVR parameter optimization.

We have 381 sample sets dating from the beginning of the year 2015 until the middle of May, 2015. The input data is normalized between -1 and 1 by anisotropic scaling. Then the dataset is split into training and testing sets with a 10% testing ratio. The sizes of training and testing sets are 343 and 38 respectively. This is accomplished by first partitioning the dataset into segments of 10% ratio and then by splitting each segment

into training and testing sets of 90% and 10% ratios respectively. The sizes of the segments and their training and testing sub-segments may differ by only one sample set.

The kernel is selected for the SVR. This study employed Linear, Polynomial and Gaussian Radial Basis Function kernels. The initial values, upper and lower bounds for the main SVR parameters and the kernel parameters are supplied to the solvers for SVR parameter optimization purposes and are given in Table 4.3. It should be noted that initial values are only supplied to Grid Search and Nonlinear Programming.

Table 4.3: Defined bounds for SVR parameters

Parameter	Initial Value	Lower Bound	Upper Bound
Cost of error, C	1E-03	1E-03	1000
Width of insensitive region, ε	1E-03	1E-03	10
Polynomial degree, d	1	1	10
Radial basis function gain, γ	1E-06	1E-06	1000

Then the solver is selected for the optimization of SVR parameters. This study has employed Genetic Algorithm, Grid Search and Nonlinear Programming. The solver parameters are given in Tables 4.4, 4.5 and 4.6 respectively.

Table 4.4: Defined parameters for Genetic Algorithm

Parameter	Value
Population size, P	20
Generation limit, $G_{\max}$	20
Elite count, s_E	4
Cross-over ratio, r_C	0.6
Creation function	uniform
Selection function	tournament
Tournament count	4
Cross-over function	scattered
Mutation function	adaptive feasible

Table 4.5: Defined parameters for Grid Search

Parameter	Value
Iteration limit	300
Mesh contraction	0.5
Mesh expansion	2
Initial mesh size	1

Table 4.6: Defined parameters for Nonlinear Programming

Parameter	Value
Iteration limit	300
Algorithm	interior point

The cross validation embedded inside the optimization solvers uses *k*-fold partitioning and the partition count is set to 5 as given in Table 4.7.

Table 4.7: Defined parameters for Cross Validation

Parameter	Value
Partitioning method	*k*-fold
Partition count	5

Only training set is supplied to parameter optimization and cross validation further partitions the training set into 5 subsets with each subset treated once as a training set inside the optimization solver.

When optimization ends, the total training set is trained for the last time with the optimized SVR parameters and then the testing data is used for monitoring the performance of SVR.

4.3 Evaluation

First of all, the rate of convergence of SVR training with decision variables β_i, β_i^* and α_i, α_i^* were compared.

For evaluation, SVR with Linear, Polynomial and Gaussian RBF kernel optimized by Genetic Algorithm, Grid Search and Non-Linear Programming were compared against RQE and ANN models. The bias of the RQE was trained with the training set. ANN was trained with the training set and with the parameters given in Table 4.8. The mean absolute errors, standard deviations and maximum absolute errors of training, testing and combined sets are presented in the next section together with the optimized parameters, the number of support vectors and the CPU time for SVR. The mean absolute errors, standard deviations and maximum absolute errors are given for the RQE and ANN for comparison in the next section.

Table 4.8: Defined parameters for ANN

Parameter	Value
Hidden layer count	2
Hidden layer size	5, 5
Training method	levenberg-marquardt
Hidden layer function	Hyperbolic tangent function
Output layer function	Linear

"Student's *t*-test" was conducted to evaluate if any method among RQE, ANN and SVR is significantly outperforming other methods. Paired-sample *t*-test method determines if two sets of data are significantly different from each other. The method examines the difference between the two sets of data, their means and variances, and outputs *p*-value, which is the significance level in the interval [0, 1]. If the *p*-value is close to 0 (zero), the two sets of data are similar. The *t*-test was performed between the testing set of laboratory data and the estimation of evaluated methods.

Then case studies were conducted on SVR with Gaussian RBF kernel to examine the effects of SVR and Gaussian RBF kernel parameters. A further study was conducted on best case study to examine if the SVR estimation error correlates with the selected input variables.

Chapter 5

RESULTS AND DISCUSSION

5.1 Optimization Results

The use of $\beta_i = \alpha_i - \alpha_i^*$, and $\beta_i^* = \alpha_i + \alpha_i^*$ as decision variables in QP greatly improved the rate of convergence. The CPU time of SVR training with α_i and α_i^* as decision variables is approximately 1.9 seconds while the CPU time of SVR training with the simplified variables was reduced by 80% to a range between 350 and 400 milliseconds even though the problem size is the same. This improvement is achieved by halving the number of decision variables in the second degree part of the cost function, which creates a sparseness in the *H* matrix.

The estimation performance of RQE and ANN are given in Table 5.1 together with a comparison with the best of SVR. The estimation of RQE and ANN is compared with the laboratory analysis data in Figures 5.1 and 5.2 respectively.

Table 5.1: Comparison of RQE and ANN against SVR

		RQE	ANN	Best	
Mean Absolute Error	**Train**	5.58	4.41	4.35	NLP-Poly.
	Test	6.35	4.45	4.41	GA-Gaus.
	All	5.65	4.42	4.36	NLP-Poly.
Standard Deviation	**Train**	7.25	5.65	5.61	GS-Gaus.
	Test	7.74	5.68	5.63	GS-Linear
	All	7.29	5.65	5.61	GS-Gaus.
Max Absolute Error	**Train**	25.28	20.66	20.78	GS-Gaus.
	Test	18.52	12.68	12.81	NLP-Poly.
	All	25.28	20.66	20.78	GS-Gaus.

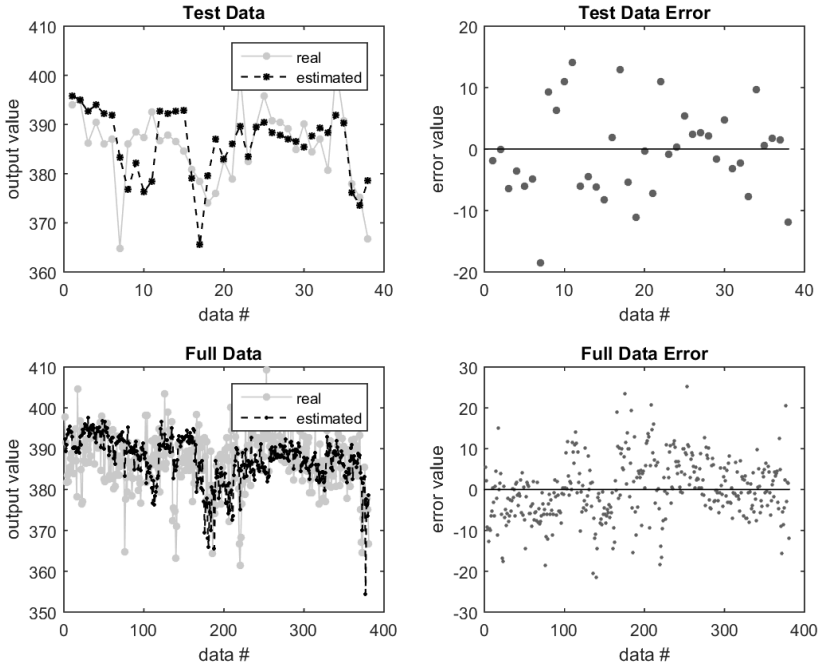

Figure 5.1: Plot of RQE estimation vs. laboratory analysis data

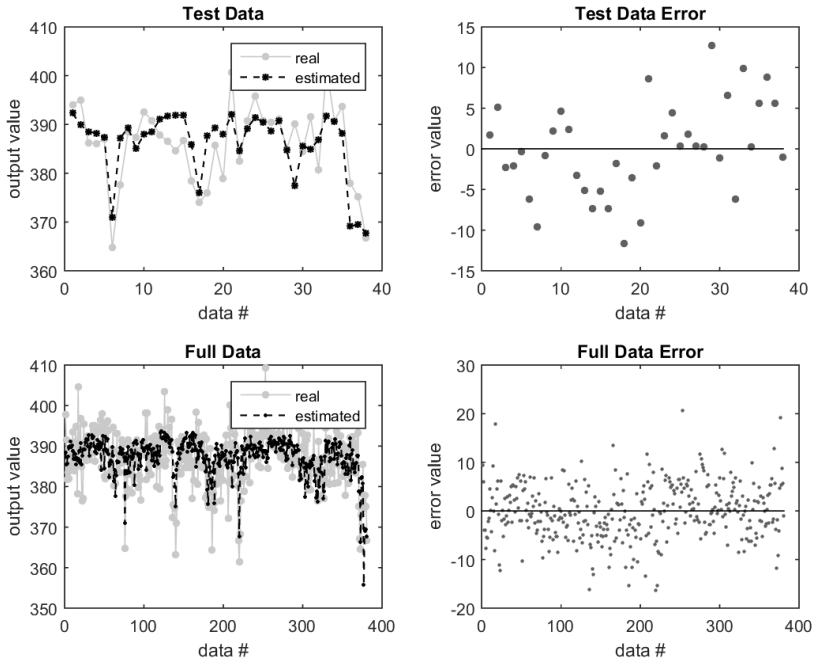

Figure 5.2: Plot of ANN estimation vs. laboratory analysis data

Genetic Algorithm optimization results are given in Table 5.2 together with a comparison against RQE and the best. The table also includes the optimized SVR parameters, the number of support vectors and the CPU time. SVR estimations with Linear, Polynomial and Gaussian RBF kernels against laboratory analysis data are given in Figures 5.3, 5.4 and 5.5 respectively.

Table 5.2: Results for SVR optimized with Genetic Algorithm

		Linear	Polynomial	Gaussian	RQE	Best	
Mean Absolute Error	**Train**	4.52	4.53	4.39	5.58	4.35	NLP-Poly.
	Test	4.44	4.48	**4.41**	6.35	**4.41**	**GA-Gaus.**
	All	4.52	4.53	4.40	5.65	4.36	NLP-Poly.
Standard Deviation	**Train**	5.81	5.79	5.63	7.25	5.61	GS-Gaus.
	Test	5.68	5.68	5.72	7.74	5.63	GS-Linear
	All	5.79	5.78	5.63	7.29	5.61	GS-Gaus.
Max Absolute Error	**Train**	21.52	21.11	20.78	25.28	20.78	GS-Gaus.
	Test	13.27	12.99	13.22	18.52	12.81	NLP-Poly.
	All	21.52	21.11	20.78	25.28	20.78	GS-Gaus.
SVR Parameters	C	970.72	356.46	3.98	-	-	-
	ε	2.43	6.72	1.00	-	-	-
	p_{KERNEL}	-	1.20	0.75	-	-	-
# of Support Vectors		229 / 67%	343 / 100%	299 / 87%	-	-	-
CPU Time (sec)		1056.6	487.6	577.1	-	-	-

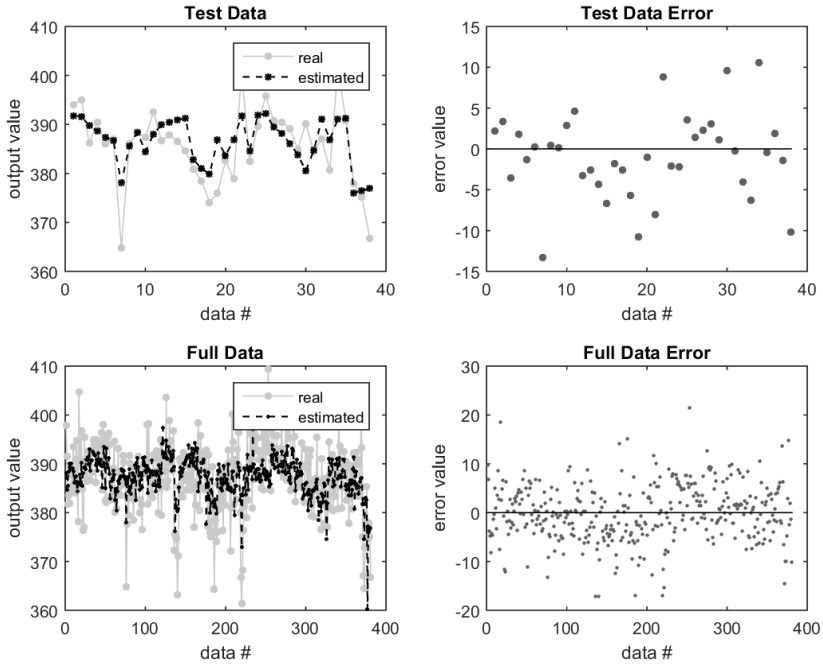

Figure 5.3: Plot of SVR with Genetic Algorithm and Linear kernel

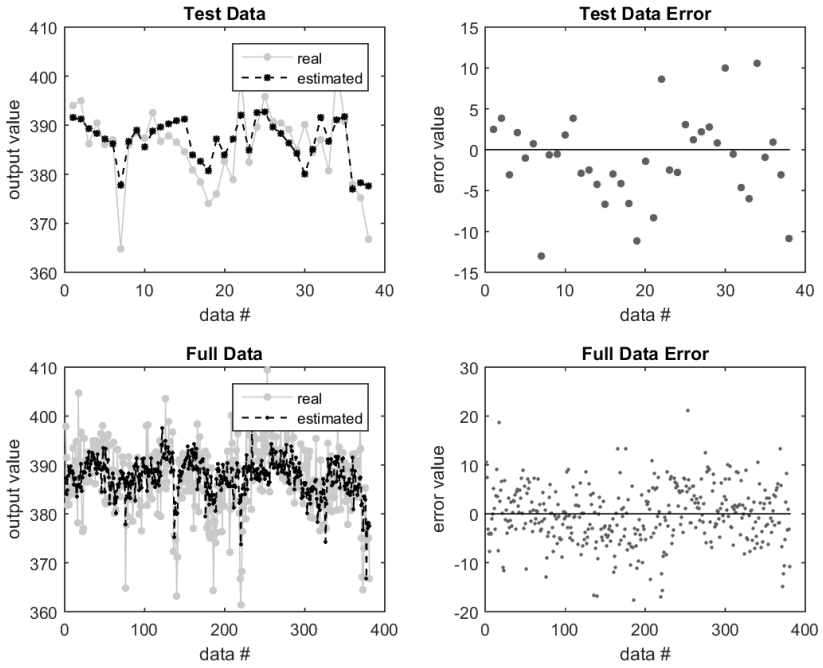

Figure 5.4: Plot of SVR with Genetic Algorithm and Polynomial kernel

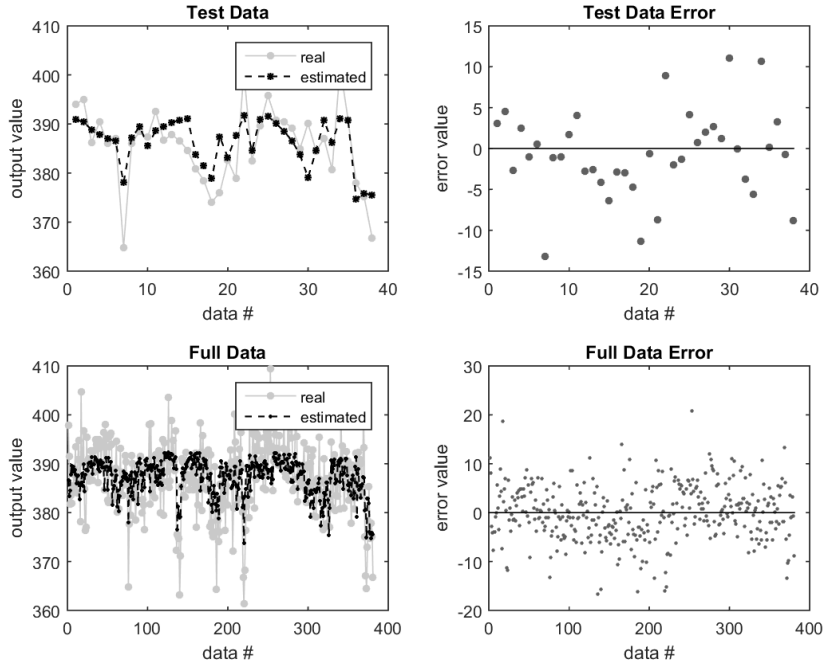

Figure 5.5: Plot of SVR with Genetic Algorithm and Gaussian RBF kernel

The results for optimization of SVR parameters by Grid Search method are given in Table 5.3 and plots for the estimation with Linear, Polynomial and Gaussian RBF kernels versus laboratory analysis data are given in Figures 5.6, 5.7 and 5.8 respectively.

Table 5.3: Results for SVR optimized with Grid Search

		Linear	Polynomial	Gaussian	RQE	Best	
Mean Absolute Error	**Train**	4.54	4.55	4.42	5.58	4.35	NLP-Poly.
	Test	4.44	4.52	4.44	6.35	4.41	GA-Gaus.
	All	4.53	4.55	4.42	5.65	4.36	NLP-Poly.
Standard Deviation	**Train**	5.80	5.81	**5.61**	7.25	**5.61**	**GS-Gaus.**
	Test	**5.63**	5.75	5.66	7.74	**5.63**	**GS-Linear**
	All	5.78	5.80	**5.61**	7.29	**5.61**	**GS-Gaus.**
Max Absolute Error	**Train**	21.03	20.94	**20.78**	25.28	**20.78**	**GS-Gaus.**
	Test	13.46	12.98	13.31	18.52	12.81	NLP-Poly.
	All	21.03	20.94	**20.78**	25.28	**20.78**	**GS-Gaus.**
SVR Parameters	C	152.81	2.69	13.50	-	-	-
	ε	6.07	7.24	5.38	-	-	-
	p_{KERNEL}	-	1.40	0.50	-	-	-
# of Support Vectors		92 / 27%	70 / 20%	120 / 34%	-	-	-
CPU Time (sec)		1134.0	1274.2	1865.9	-	-	-

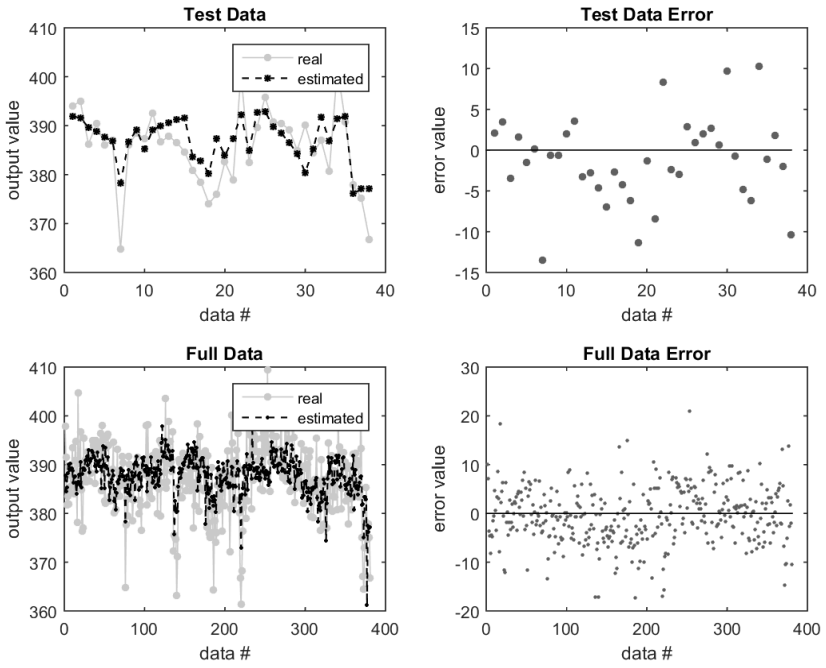

Figure 5.6: Plot of SVR with Grid Search and Linear kernel

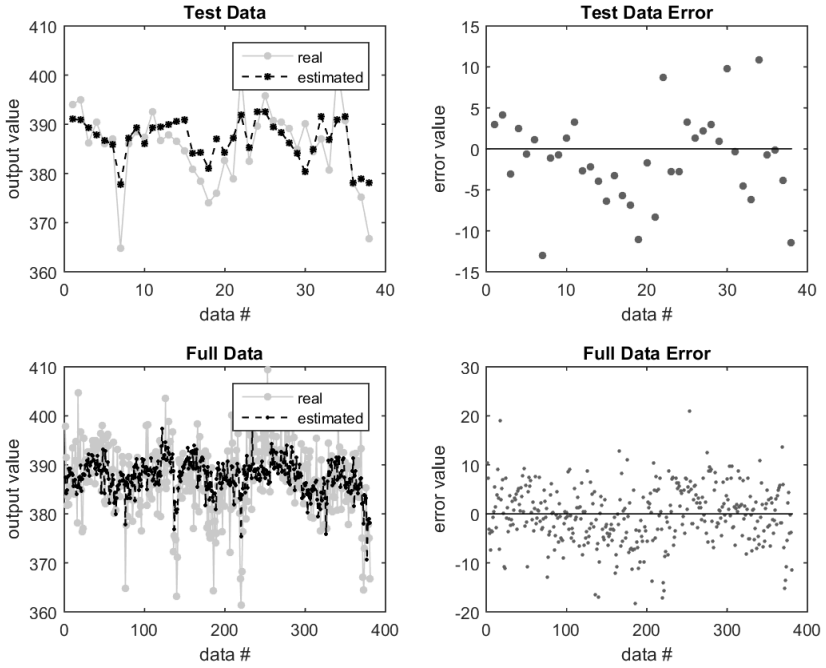

Figure 5.7: Plot of SVR with Grid Search and Polynomial kernel

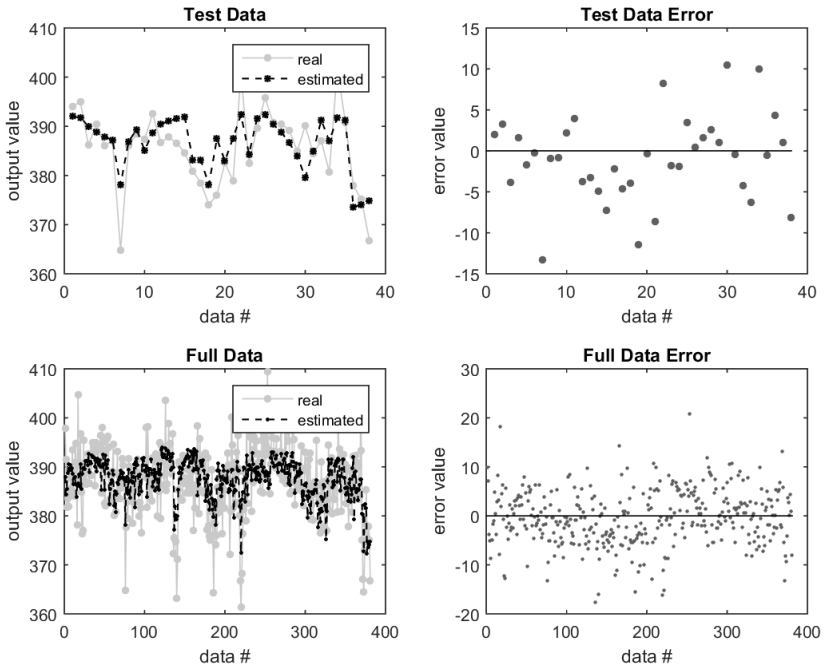

Figure 5.8: Plot of SVR with Grid Search and Gaussian RBF kernel

Finally the results for optimization of SVR parameters by Nonlinear Programming are given in Table 5.4 and plots for the estimation with Linear, Polynomial and Gaussian RBF kernels versus laboratory analysis data are given in Figures 5.9, 5.10 and 5.11 respectively.

Table 5.4: Results for SVR optimized with Nonlinear Programming

		Linear	Polynomial	Gaussian	RQE	Best	
Mean Absolute Error	**Train**	4.57	**4.35**	4.45	5.58	4.35	NLP-Poly.
	Test	4.57	4.45	4.46	6.35	4.41	GA-Gaus.
	All	4.57	**4.36**	4.45	5.65	4.36	NLP-Poly.
Standard Deviation	**Train**	5.84	5.65	5.67	7.25	5.61	GS-Gaus.
	Test	5.88	5.66	5.70	7.74	5.63	GS-Linear
	All	5.84	5.64	5.67	7.29	5.61	GS-Gaus.
Max Absolute Error	**Train**	21.16	20.80	20.92	25.28	20.78	GS-Gaus.
	Test	14.69	**12.81**	13.83	18.52	**12.81**	**NLP-Poly.**
	All	21.16	20.80	20.92	25.28	20.78	GS-Gaus.
SVR Parameters	C	5.73	1.07	8.17	-	-	-
	ε	7.18	0.23	5.66	-	-	-
	p_{KERNEL}	-	3.78	0.40	-	-	-
# of Support Vectors		71 / 20%	338 / 98%	112 / 32%	-	-	-
CPU Time (sec)		305.1	646.5	310.6	-	-	-

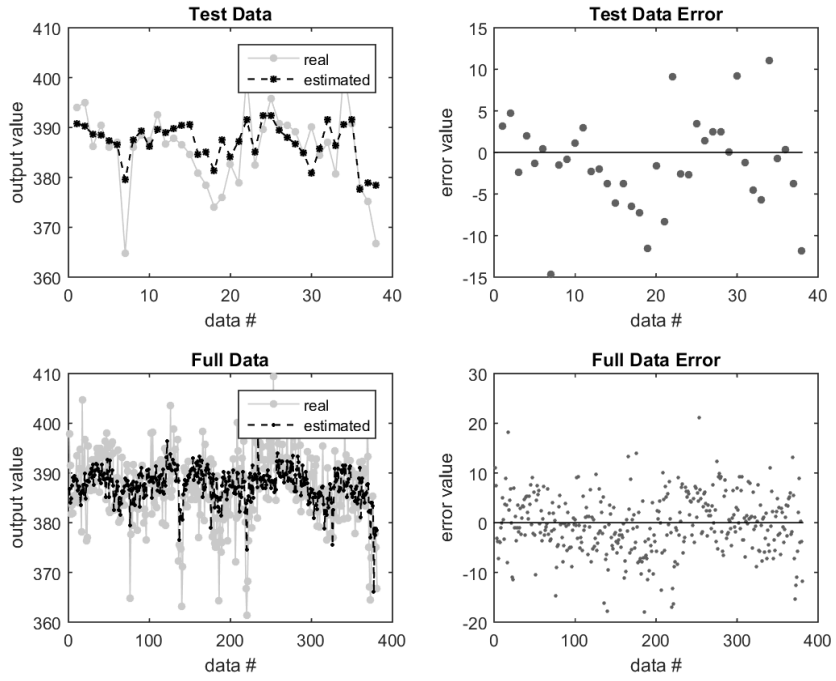

Figure 5.9: Plot of SVR with Nonlinear Programming and Linear kernel

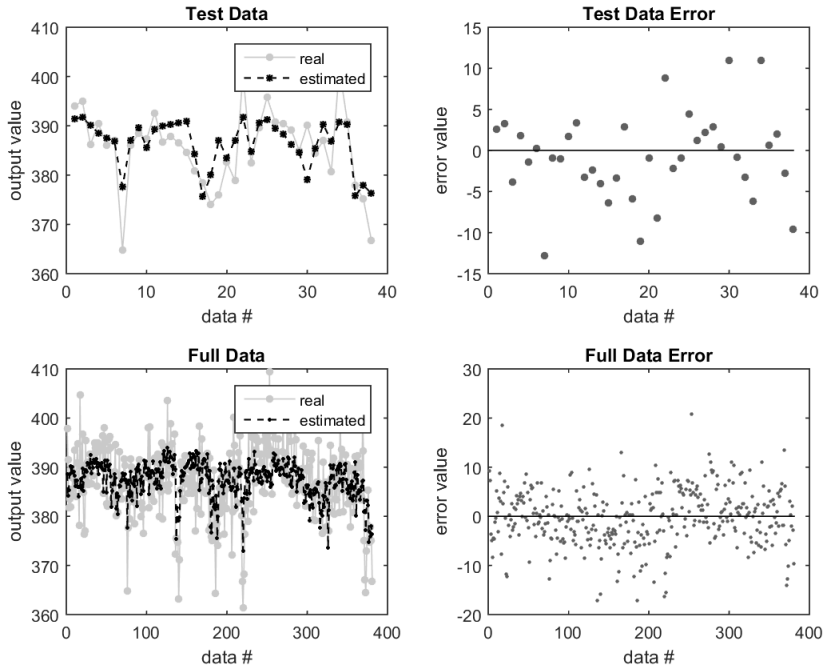

Figure 5.10: Plot of SVR with Nonlinear Programming and Polynomial kernel

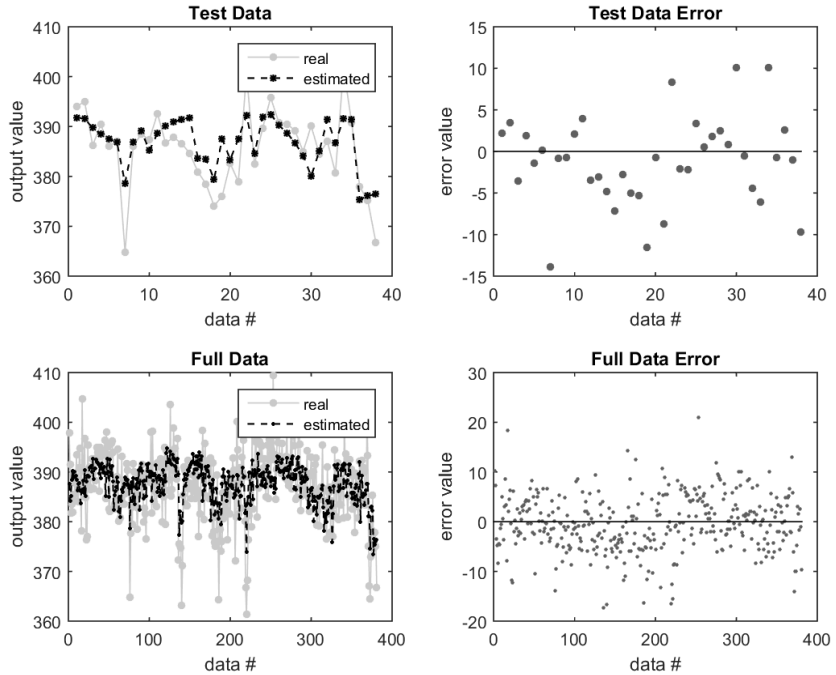

Figure 5.11: Plot of SVR with Nonlinear Programming and Gaussian RBF kernel

Mean absolute errors (MAE), mean errors (ME), standard deviations (STD) and *p*-values from "Student's *t*-test" for testing errors of SVR, RQE and ANN methods are given in Table 5.5.

Table 5.5: *t*-test comparison of SVR, RQE and ANN methods

			MAE	ME	STD	*p*-value
SVR	GA	Linear	4.44	-0.61	5.68	0.51
		Polynomial	4.48	-0.93	5.68	0.32
		Gaussian	4.41	-0.43	5.72	0.65
	GS	Linear	4.44	-1.00	5.63	0.28
		Polynomial	4.52	-0.99	5.75	0.29
		Gaussian	4.44	-0.84	5.66	0.32
	NLP	Linear	4.57	-1.06	5.88	0.27
		Polynomial	4.45	-0.45	5.66	0.63
		Gaussian	4.46	-0.96	5.70	0.31
RQE			6.35	-0.12	7.74	0.93
ANN			4.45	-0.09	5.68	0.72

5.2 Discussion of Optimization Results

The testing errors are lower than the training errors due to a moderate number of outliers within the training data. The results show that different optimization solvers have found different local optimal points with similar mean absolute errors and standard deviations for the same kernels, meaning that there are many similar local optima. Grid Search method has triumphed in most of the criteria. *t*-test results of SVR optimized by Grid Search have low *p*-values, meaning the estimation results are not significantly different than actual testing set. SVR with Linear kernel optimized by Grid Search has best performance in testing while SVR with Gaussian RBF kernel optimized with Grid Search has very close values in testing and best results in training and overall results. Although SVR with Gaussian kernel optimized by Genetic Algorithm has best

performance in mean absolute error of testing set, the optimization results from Genetic Algorithm show that local optima have high rate of support vectors. Unlike Genetic Algorithm, Grid Search and Nonlinear Programming solvers found similar parameters for SVR with Gaussian kernel but with a smaller set of support vectors than GA. Grid Search method has highest computational cost while Nonlinear Programming has lowest computational cost. The Nonlinear Programming method has low *p*-values and good performance with low computational cost, but Grid Search is more robust to initialization.

It should be noted that any solution surpassed the performance of an RQE which was also trained and tested with the same datasets. ANN has also shown good performance, but each training attempt resulted in a significantly different model unlike SVR. The overall results show that Grid Search should be considered for SVR parameter optimization despite of its high computational cost, and Gaussian RBF kernel is recommended for SVR training problems if a linear model is not strictly needed.

5.3 Case Study Results and Discussion

The effect of the SVR parameters on the estimation performance is discussed in this section. Gaussian Kernel was employed in the case studies for its flexibility. The main SVR parameters were evaluated for their effect on the memorization of training set and generalization performance while the Gaussian kernel parameter was evaluated for the flexibility of the estimation function.

Case studies 1, 2 and 3 were conducted to see the effect of the Gaussian kernel parameter. The main parameters are selected to force the SVR to memorize the training set and have a poor performance in generalization of the testing set. Table 5.6 depicts the results of the mentioned case studies. Case 1 shows that a low value for γ has reduced the flexibility of the mapping and has reduced memorization. As γ is increased throughout cases 2 and 3, the flexibility of the mapping increases together with the number of support

vectors and memorization. A high value for γ minimized the mean error in training set and maximized the mean error in testing data. That is why Case 3 as depicted in Figure 5.12 is the best in training error, but suffered in testing error.

Table 5.6: Case Studies 1, 2 and 3 with γ=1, 10 and 100

		Case-1	Case-2	Case-3	Best	
Mean Absolute Error	**Train**	4.04	2.39	**0.07**	**0.07**	**Case-3**
	Test	4.49	5.43	7.23	4.46	Case-9
	All	4.08	2.69	**0.78**	**0.78**	**Case-3**
Standard Deviation	**Train**	5.81	4.30	**0.88**	**0.88**	**Case-3**
	Test	5.62	7.31	9.29	5.70	Case-9
	All	5.79	4.68	**3.06**	**3.06**	**Case-3**
Max Absolute Error	**Train**	19.87	18.08	**15.02**	**15.02**	**Case-3**
	Test	15.65	22.16	21.14	13.88	Case-9
	All	**19.87**	22.16	21.14	**19.87**	**Case-1**
SVR Parameters	C	1000	1000	1000	-	-
	ε	0	0	0	-	-
	γ	1	10	100	-	-
% of Support Vectors		100%	100%	100%	-	-

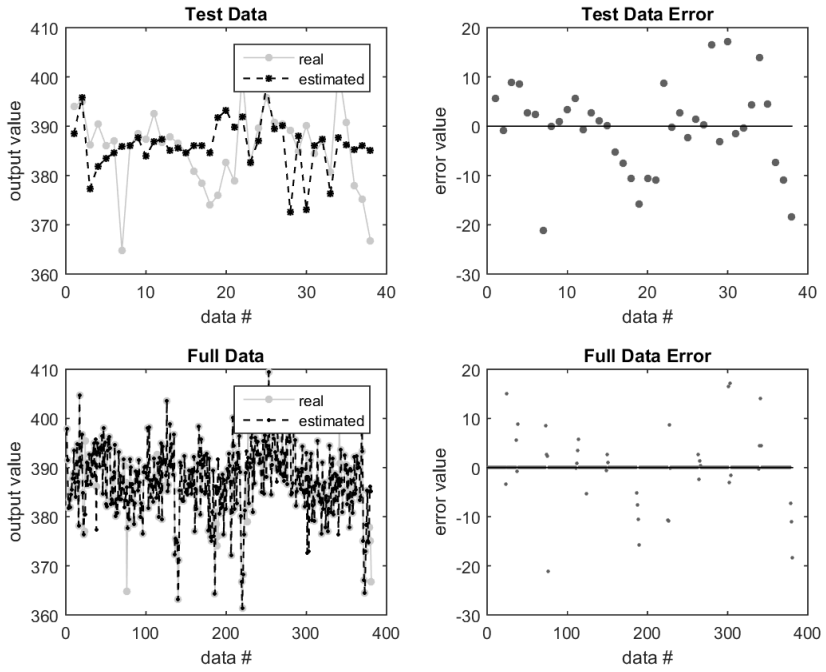

Figure 5.12: Case Study 3 with C=1000, ε=0, γ=100

Case studies 2, 4, 5 and 6 were conducted to see the effect of the SVR parameter ε. Gaussian kernel parameter γ is fixed to 10 for intermediate rate of flexibility. The cost parameter C is fixed to 1000 for minimizing any training error greater than ε. Table 5.7 depicts the results of the mentioned case studies. Case 2 shows that lowest value for ε has lower training error, but a higher testing error. As ε is increased throughout cases 4, 5 and 6, insensitive region expands leading to a flatter estimation function. Case 6 which has the highest value for ε shows that training error has increased since we have a flatter estimation function. This leads to a decrease in memorization and an increase in generalization performance which resulted with a decrease in the testing error.

Table 5.7: Case Studies 2, 4, 5 and 6 with ε=0, 1, 3 and 5

		Case-2	Case-4	Case-5	Case-6	Best	
Mean Absolute Error	**Train**	2.39	2.74	3.36	4.05	0.07	Case-3
	Test	5.43	5.21	5.51	5.87	4.46	Case-9
	All	2.69	2.99	3.58	4.23	0.78	Case-3
Standard Deviation	**Train**	4.30	4.22	4.35	4.79	0.88	Case-3
	Test	7.31	7.01	7.31	7.73	5.70	Case-9
	All	4.68	4.57	4.72	5.15	3.06	Case-3
Max Absolute Error	**Train**	18.08	18.73	20.27	19.33	15.02	Case-3
	Test	22.16	22.32	23.59	22.69	13.88	Case-9
	All	22.16	22.32	23.59	22.69	19.87	Case-1
SVR Parameters	C	1000	1000	1000	1000	-	-
	ε	0	1	3	5	-	-
	γ	10	10	10	10	-	-
% of Support Vectors		100%	90%	67%	47%	-	-

Figures 5.13, 5.14 and 5.15 are for cases 4, 5 and 6 respectively. The full data error plots on the right bottom corner of the figures clearly show that the training errors are packed within the insensitive region.

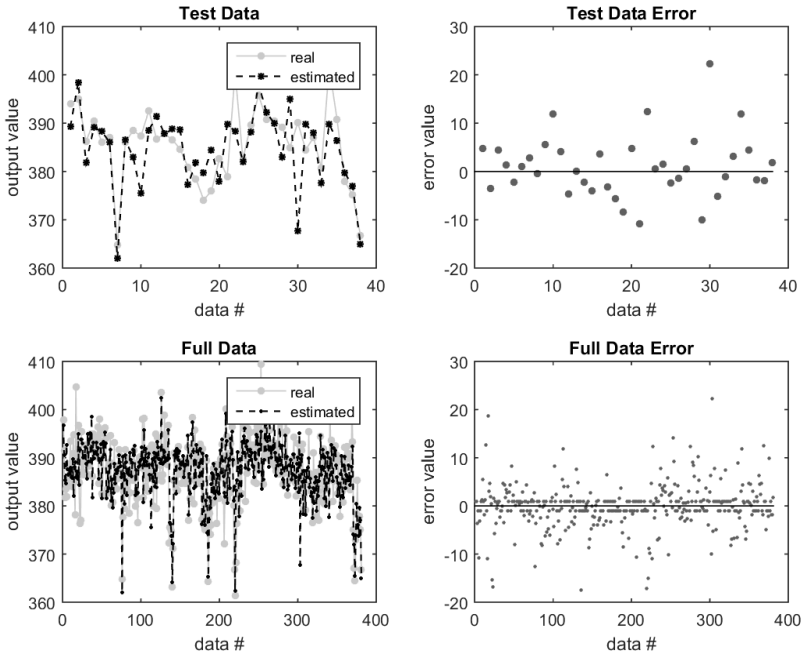

Figure 5.13: Case Study 4 with C=1000, ε=1, γ=10

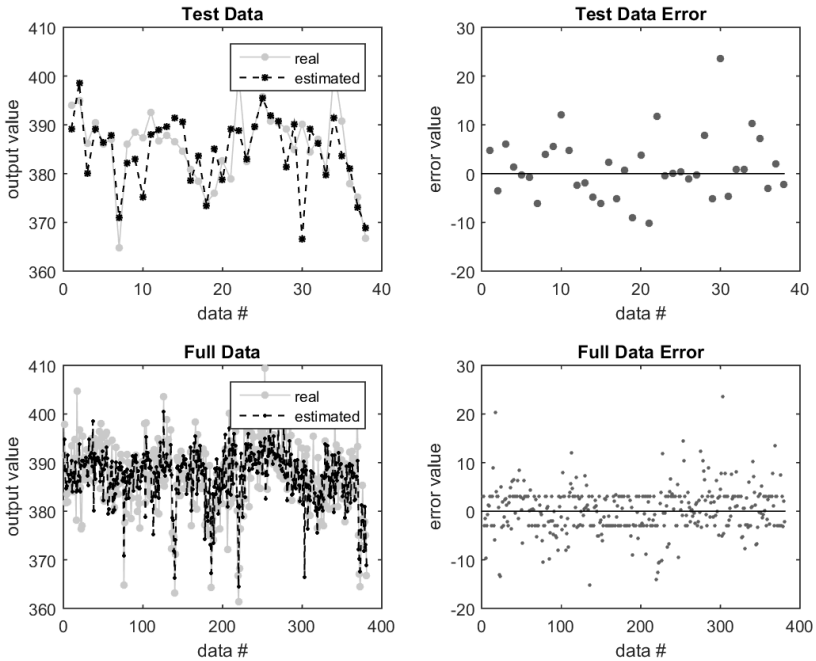

Figure 5.14: Case Study 5 with C=1000, ε=3, γ=10

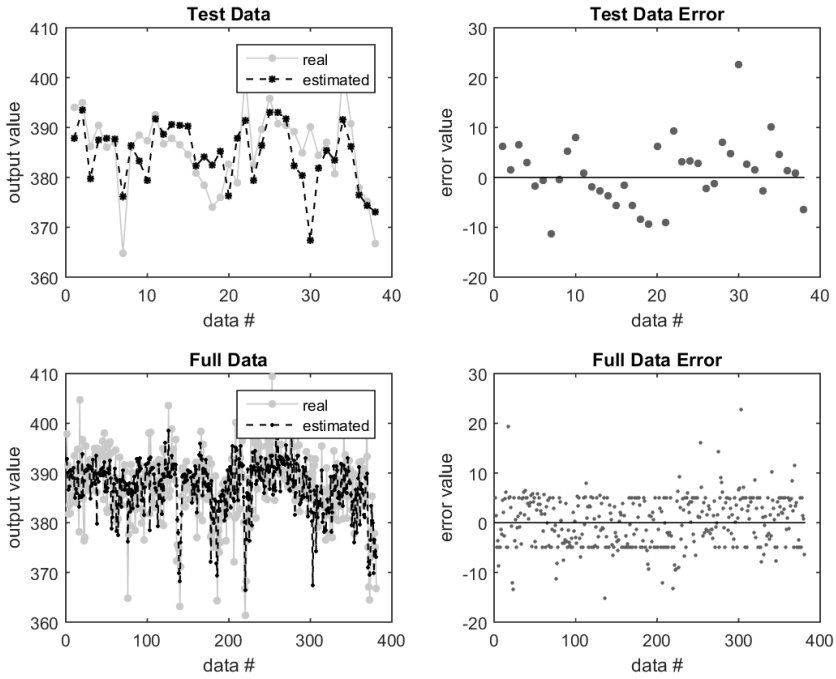

Figure 5.15: Case Study 6 with C=1000, ε=5, γ=10

Finally case studies 6, 7, 8 and 9 were conducted to see the effect of the cost factor C. The other main parameter ε is fixed to 5 and kernel parameter γ is fixed to 10 except for the last case where the parameter is fixed to 1 (one). Table 5.8 depicts the results of the mentioned case studies. As the cost factor is decreased throughout cases 7, 8 and 9, the training error higher than ε is better tolerated. The SVR estimation function is flatter and has a better generalization performance with a low value for the cost parameter C and Gaussian RBF kernel parameter γ in case study 9. The full data error plots on the right bottom corner in Figure 5.16 for case study 9 clearly show that the training errors are not forced to stay within the insensitive region anymore.

Table 5.8: Case Studies 6, 7, 8 and 9 with C=1000, 500, 100 and 10

		Case-6	Case-7	Case-8	Case-9	Best	
Mean Absolute Error	**Train**	4.05	3.99	3.99	4.38	0.07	Case-3
	Test	5.87	5.51	4.89	**4.46**	**4.46**	**Case-9**
	All	4.23	4.15	4.08	4.39	0.78	Case-3
Standard Deviation	**Train**	4.79	4.79	4.85	5.56	0.88	Case-3
	Test	7.73	7.31	6.65	**5.70**	**5.70**	**Case-9**
	All	5.14	5.09	5.05	5.57	3.06	Case-3
Max Absolute Error	**Train**	19.33	19.66	19.63	20.40	15.02	Case-3
	Test	22.69	22.59	22.62	**13.88**	**13.88**	**Case-9**
	All	22.69	22.59	22.62	20.40	19.87	Case-1
SVR Parameters	C	1000	500	100	10	-	-
	ε	5	5	5	5	-	-
	γ	10	10	10	1	-	-
% of Support Vectors		47%	44%	43%	38%	-	-

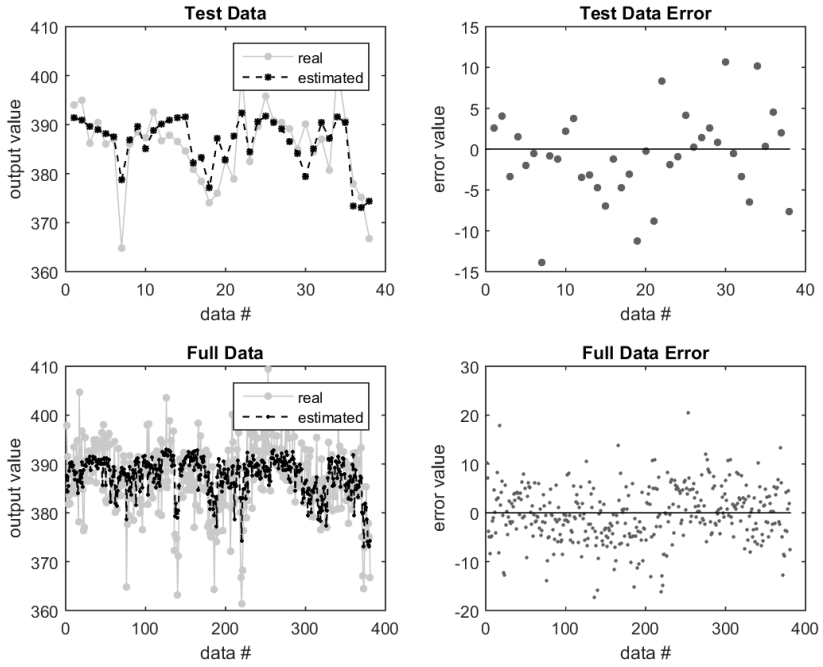

Figure 5.16: Case Study 9 with C=10, ε=5, γ=1

A further study was conducted on case study 9 to examine if the SVR estimation error correlates with the selected input variables. Table 5.9 gives the correlation values of input variables versus SVR estimation error. Figures 5.17, 5.18 and 5.19 are for comparison of SVR estimation error versus the input variables, flash zone temperature, flash zone pressure and heavy diesel tray temperature respectively. The figures clearly show that there is no correlation between the input variables and the error of estimation.

Table 5.9: Correlation between input variables and SVR estimation error

Input Variable	Correlation
Flash zone temperature	0.0723
Flash zone pressure	0.0637
Heavy diesel tray temperature	0.0085

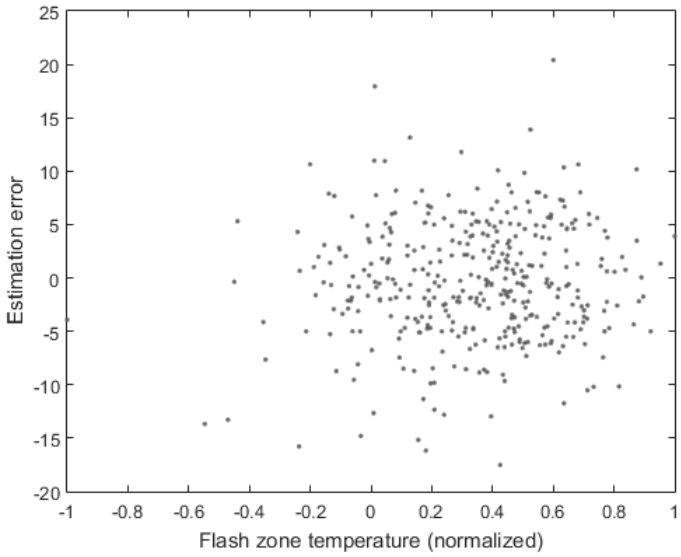

Figure 5.17: Plot of normalized flash zone temperature vs. SVR estimation error

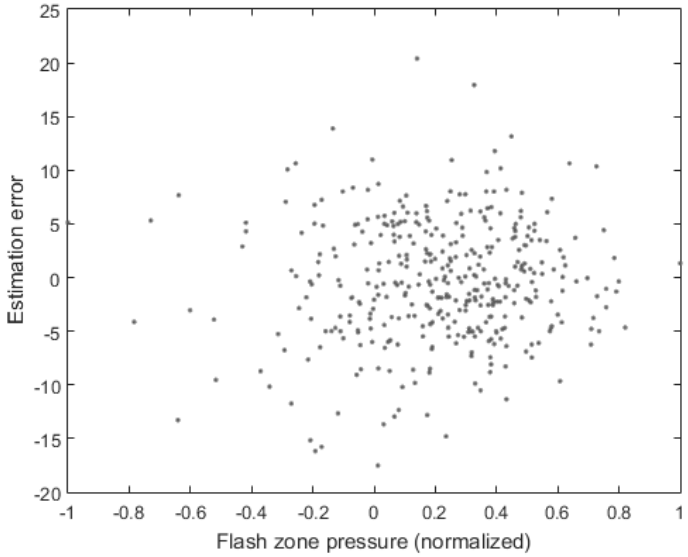

Figure 5.18: Plot of normalized flash zone pressure vs. SVR estimation error

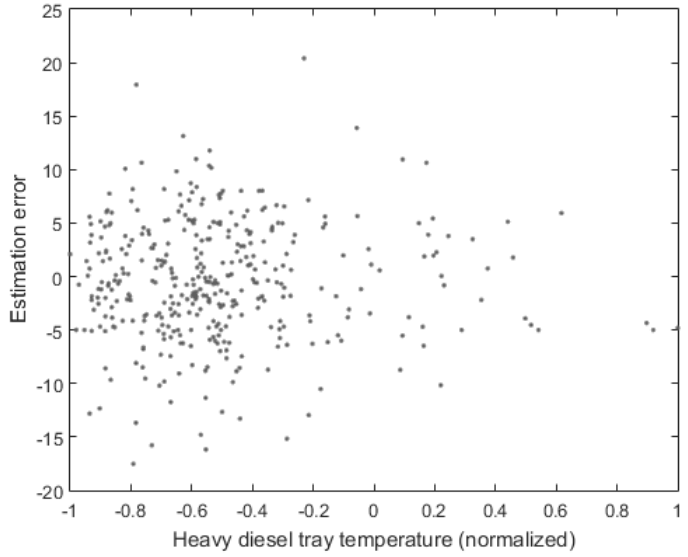

Figure 5.19: Plot of normalized h. diesel tray temperature vs. SVR estimation error

Histograms of the estimation errors of RQE and SVR in case study 9 are compared in Figure 5.20. The figure clearly shows that the standard deviation is reduced and the estimation errors are closer to 0 (zero). The standard deviations of RQE and SVR are 7.29 and 5.57 respectively, which support the histogram.

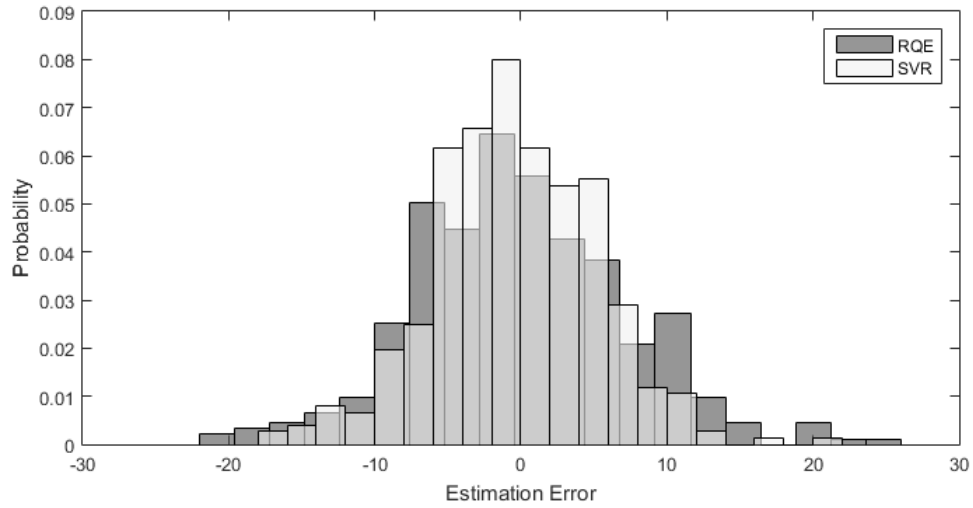

Figure 5.20: Histogram of estimation errors of RQE and SVR

Histograms of case studies 2, 4, 5 and 6 were compared with the histogram of case study 9 as shown in Figure 5.21. The case studies 2, 4, 5 and 6 have high cost factor resulting in forcing the estimation error to be equal to ε since the VSR training aims to fit the flattest function. The insensitive region width parameters of these case studies are equal to 0, 1, 3 and 5 respectively and the estimation errors have highest probability distributions at these values. Case study 9 has ε equal to 5, but has a lower cost factor value, which leads to a flatter function with almost uniformly distributed estimation error inside the insensitive region.

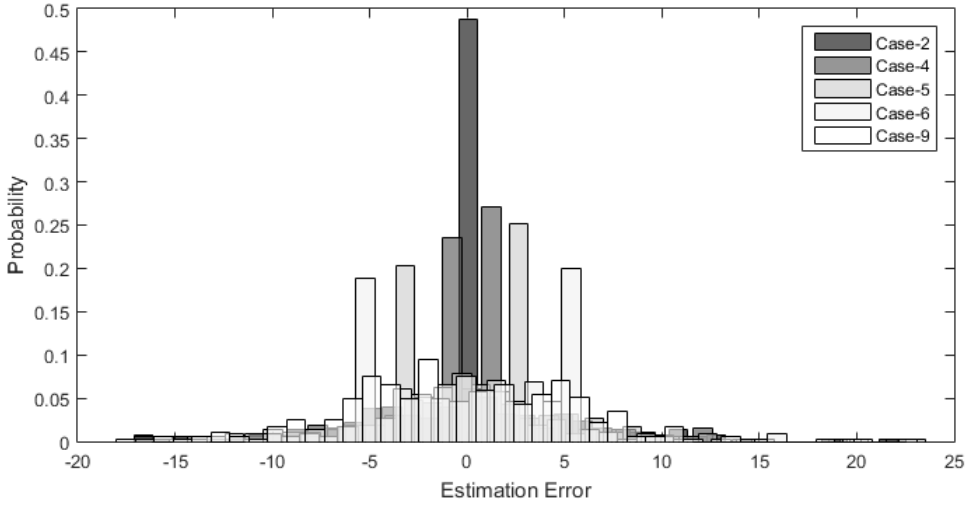

Figure 5.21: Histogram of Case Studies 2, 4, 5, 6 and 9

Chapter 6

CONCLUSION

In this study, Support Vector Regression (SVR) was employed for estimating 95% Boiling Point Temperature property of Heavy Diesel product of a real operating Atmospheric Distillation Column (ADC) in İzmit Refinery of Turkish Petroleum Refineries Corporation. Linear, Polynomial and Gaussian Radial Basis Function (Gaussian RBF) kernels were tested and the SVR parameters were optimized by embedding *k*-fold cross validation in an optimizer like Genetic Algorithm (GA), Grid Search (GS) or Non-linear Programming (NLP). The performance of SVR was compared against Robust Quality Estimator (RQE) already functioning in the ADC.

The SVR parameters were optimized in a time period ranging from 5 minutes to 30 minutes, but once the parameters are optimized, the SVR can be trained with the optimum parameters in approximately 350 milliseconds. As a result the SVR model can be updated with incoming laboratory analysis data within a second while the SVR parameters can be updated periodically in monthly intervals.

The rate of convergence of SVR training is reduced by 80% approximately from 1.9 seconds to 350 milliseconds by incorporating simplified decision variables instead of using the Lagrange multipliers as decision variables. This improvement is achieved by creating sparseness in the second degree part of the cost function.

The testing results show that SVR with any of the integrated kernels performed well in estimating the property and generalized well to unseen testing data. The mean absolute testing error of RQE is improved by 31% with SVR from 6.4˚C to 4.4˚C. The standard deviation of RQE estimation error is improved by 23% with SVR from 7.3 ˚C to 5.6 ˚C. The maximum absolute testing error is reduced by 31% from 18.5˚C to 12.8˚C. Grid Search method is clearly most robust among the tested optimizers but with high

computational cost. SVR with Gaussian kernel optimized by Grid Search has shown top performance in nearly all criteria.

Moreover, a case study was conducted to monitor the effects of the SVR parameters on the generalization performance. It is shown that when tolerance on estimation error was reduced by shrinking the insensitive region or by increasing the cost factor, the SVR memorized the training data and performed poorly in estimating testing data. Also the Gaussian RBF kernel parameter was studied for its effect on the flexibility of the kernel. A kernel parameter with a higher value than optimum value has increased the flexibility of the kernel which enabled the SVR to better memorize the training data and decreased the generalization performance.

Yet a recent subject in Machine Learning, modified versions of SVRs can be implemented for improving the SVR parameter optimization process. One of the promising versions is the *v*-Support Vector Regression where the width of the insensitive region is optimized within the training process by fixing the fraction of support vectors in a training set. The fraction will be a real number between 0 and 1 and can be optimized in the SVR parameter optimization step.

Finally, there may be other known column dynamics correlating with the output parameter. The online measurements incorporated in the RQE and in this study were selected by experts of the process, but a principle component analysis (PCA) can be conducted to see the effects of other column measurements.

BIBLIOGRAPHY

1. Gary, J.H., G.E. Handwerk, and M.J. Kaiser, *Petroleum refining: technology and economics*. 2010: CRC press.
2. Leffler, W.L., *Petroleum refining for the nontechnical person.* 1985.
3. Naphtali, L.M. and D.P. Sandholm, *Multicomponent separation calculations by linearization.* AIChE Journal, 1971. **17**(1): p. 148-153.
4. Boston, J.F. and S.L. Sullivan, *A new class of solution methods for multicomponent, multistage separation processes.* The Canadian Journal of Chemical Engineering, 1974. **52**(1): p. 52-63.
5. Hofeling, B.S. and J.D. Seader, *A modified Naphtali-Sandholm method for general systems of interlinked, multistaged separators.* AIChE Journal, 1978. **24**(6): p. 1131-1134.
6. Russell, R.A., *A flexible and reliable method solves single-tower and crude distillation-column problems.* Chemical Engineering, 1983. **90**(21): p. 52-59.
7. Kumar, V., et al., *A crude distillation unit model suitable for online applications.* Fuel Processing Technology, 2001. **73**(1): p. 1-21.
8. Basak, K., et al., *On-Line Optimization of a Crude Distillation Unit with Constraints on Product Properties.* Industrial & Engineering Chemistry Research, 2002. **41**(6): p. 1557-1568.
9. More, R.K., et al., *Optimization of crude distillation system using aspen plus: Effect of binary feed selection on grass-root design.* chemical engineering research and design, 2010. **88**(2): p. 121-134.
10. Inamdar, S.V., S.K. Gupta, and D.N. Saraf, *Multi-objective Optimization of an Industrial Crude Distillation Unit Using the Elitist Non-Dominated Sorting Genetic Algorithm.* Chemical Engineering Research and Design, 2004. **82**(5): p. 611-623.
11. Seo, J.W., M. Oh, and T.H. Lee, *Design Optimization of a Crude Oil Distillation Process.* Chemical Engineering & Technology, 2000. **23**(2): p. 157-164.
12. Peng, D.-Y. and D.B. Robinson, *A new two-constant equation of state.* Industrial & Engineering Chemistry Fundamentals, 1976. **15**(1): p. 59-64.
13. Redlich, O. and J. Kwong, *On the thermodynamics of solutions. V. An equation of state. Fugacities of gaseous solutions.* Chemical Reviews, 1949. **44**(1): p. 233-244.
14. Soave, G., *Equilibrium constants from a modified Redlich-Kwong equation of state.* Chemical Engineering Science, 1972. **27**(6): p. 1197-1203.
15. API, *Basic Petroleum Data Book*. Vol. 28. 2008: American Petroleum Institute.
16. Khoury, F.M., *Multistage Separation Processes, Fourth Edition*. 2014: Taylor & Francis.

17. El-Mikkawy, M.E.A., *On the inverse of a general tridiagonal matrix.* Applied Mathematics and Computation, 2004. **150**(3): p. 669-679.
18. da Fonseca, C.M., *On the eigenvalues of some tridiagonal matrices.* Journal of Computational and Applied Mathematics, 2007. **200**(1): p. 283-286.
19. Usmani, R.A., *Inversion of a tridiagonal jacobi matrix.* Linear Algebra and its Applications, 1994. **212–213**(0): p. 413-414.
20. Hu, G.Y. and R.F.O. Connell, *Analytical inversion of symmetric tridiagonal matrices.* Journal of Physics A: Mathematical and General, 1996. **29**(7): p. 1511.
21. Huang, Y. and W.F. McColl, *Analytical inversion of general tridiagonal matrices.* Journal of Physics A: Mathematical and General, 1997. **30**(22): p. 7919.
22. Mallik, R.K., *The inverse of a tridiagonal matrix.* Linear Algebra and its Applications, 2001. **325**(1–3): p. 109-139.
23. Kılıç, E., *Explicit formula for the inverse of a tridiagonal matrix by backward continued fractions.* Applied Mathematics and Computation, 2008. **197**(1): p. 345-357.
24. Golub, G.H.V.L., Charles F., *Matrix Computations*. 3rd ed. 1996, 2715 North Charles Street, Baltimore, Maryland 21218-4319: Johns Hopkins University Press.
25. Chun-shan, L., et al., *Simulation of Multi-component Multi-stage Separation Process--An Improved Algorithm and Application.* The Chinese Journal of Process Engineering, 2006. **6**(2): p. 247-254.
26. Himmelblau, D.M., *Applications of artificial neural networks in chemical engineering.* Korean Journal of Chemical Engineering, 2000. **17**(4): p. 373-392.
27. Nascimento, C.A.O., R. Giudici, and R. Guardani, *Neural network based approach for optimization of industrial chemical processes.* Computers & Chemical Engineering, 2000. **24**(9–10): p. 2303-2314.
28. Palmer, K. and M. Realff, *Metamodeling Approach to Optimization of Steady-State Flowsheet Simulations: Model Generation.* Chemical Engineering Research and Design, 2002. **80**(7): p. 760-772.
29. Osman, E.-S.A. and M.A. Aggour, *Artificial neural network model for accurate prediction of pressure drop in horizontal and near-horizontal-multiphase flow.* Petroleum Science and Technology, 2002. **20**(1-2): p. 1-15.
30. Shirvany, Y., G. Zahedi, and M. Bashiri, *Estimation of sour natural gas water content.* Journal of Petroleum Science and Engineering, 2010. **73**(1–2): p. 156-160.
31. Alhajree, I., et al., *Modeling and optimization of an industrial hydrocracker plant.* Journal of Petroleum Science and Engineering, 2011. **78**(3–4): p. 627-636.
32. Bellos, G.D., et al., *Modelling of the performance of industrial HDS reactors using a hybrid neural network approach.* Chemical Engineering and Processing: Process Intensification, 2005. **44**(5): p. 505-515.
33. Zahedi, G., A. Lohi, and Z. Karami, *A neural network approach for identification and modeling of delayed coking plant.* International Journal of Chemical Reactor Engineering, 2009. **7**(1).

34. Zhao, W., D. Chen, and S. Hu, *Optimizing operating conditions based on ANN and modified GAs.* Computers & Chemical Engineering, 2000. **24**(1): p. 61-65.
35. López, D.-C., et al., *Optimization model of a system of crude oil distillation units with heat integration and metamodeling.* CT&F-Ciencia, Tecnología y Futuro, 2009. **3**(5): p. 159-173.
36. Liau, L.C.-K., T.C.-K. Yang, and M.-T. Tsai, *Expert system of a crude oil distillation unit for process optimization using neural networks.* Expert Systems with Applications, 2004. **26**(2): p. 247-255.
37. Motlaghi, S., F. Jalali, and M.N. Ahmadabadi, *An expert system design for a crude oil distillation column with the neural networks model and the process optimization using genetic algorithm framework.* Expert Systems with Applications, 2008. **35**(4): p. 1540-1545.
38. Ochoa-Estopier, L.M., M. Jobson, and R. Smith, *Operational optimization of crude oil distillation systems using artificial neural networks.* Computers & Chemical Engineering, 2013. **59**: p. 178-185.
39. Gershenson, C., *Artificial neural networks for beginners.* arXiv preprint cs/0308031, 2003.
40. Rumelhart, D.E., J.L. McClelland, and P.R. Group, *Parallel distributed processing*. Vol. 1. 1988: IEEE.
41. Vapnik, V. and A. Lerner, *Generalized portrait method for pattern recognition.* Automation and Remote Control, 1963. **24**(6): p. 774-780.
42. Vapnik, V.N. and A.Y. Chervonenkis. *On the uniform convergence of relative frequencies of events to their probabilities*. in *Soviet Math. Dokl*. 1968.
43. Boser, B.E., I.M. Guyon, and V.N. Vapnik, *A training algorithm for optimal margin classifiers*, in *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, ACM: Pittsburgh, Pennsylvania, USA. p. 144-152.
44. Aizerman, A., E.M. Braverman, and L. Rozoner, *Theoretical foundations of the potential function method in pattern recognition learning.* Automation and remote control, 1964. **25**: p. 821-837.
45. Cortes, C. and V. Vapnik, *Support-vector networks.* Machine Learning, 1995. **20**(3): p. 273-297.
46. Drucker, H., et al., *Support vector regression machines.* Advances in neural information processing systems, 1997. **9**: p. 155-161.
47. Gunn, S.R., *Support vector machines for classification and regression.* ISIS technical report, 1998. **14**.
48. Smola, A.J. and B. Schölkopf, *A tutorial on support vector regression*, in *NeuroCOLT2 Technical Report Series*. 1998.
49. Smola, A. and B. Schölkopf, *A tutorial on support vector regression.* Statistics and Computing, 2004. **14**(3): p. 199-222.
50. Chang, C.-C. and C.-J. Lin, *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. **2**(3): p. 27.

51. Lee, D.E., et al., *Weighted Support Vector Machine for Quality Estimation in the Polymerization Process.* Industrial & Engineering Chemistry Research, 2005. **44**(7): p. 2101-2105.
52. Shokri, S., et al., *Soft sensor design for hydrodesulfurization process using support vector regression based on WT and PCA.* Journal of Central South University, 2015. **22**(2): p. 511-521.
53. Chitralekha, S.B. and S.L. Shah, *Application of support vector regression for developing soft sensors for nonlinear processes.* The Canadian Journal of Chemical Engineering, 2010. **88**(5): p. 696-709.
54. Li, L., H. Su, and J. Chu, *Modeling of Isomerization of C8 Aromatics by Online Least Squares Support Vector Machine.* Chinese Journal of Chemical Engineering, 2009. **17**(3): p. 437-444.
55. Yao, H. and J. Chu, *Operational optimization of a simulated atmospheric distillation column using support vector regression models and information analysis.* Chemical Engineering Research and Design, 2012. **90**(12): p. 2247-2261.
56. Yan, W., H. Shao, and X. Wang, *Soft sensing modeling based on support vector machine and Bayesian model selection.* Computers & Chemical Engineering, 2004. **28**(8): p. 1489-1498.
57. Lahiri, S.K. and K.C. Ghanta, *Hybrid Support Vector Regression and Genetic Algorithm Technique-A Novel Approach in Process Modeling.* Chemical Product and Process Modeling, 2009. **4**(1).
58. Shi, Y. and R. Eberhart. *A modified particle swarm optimizer*. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998. IEEE.
59. Bennett, K.P. and O.L. Mangasarian, *Robust linear programming discrimination of two linearly inseparable sets.* Optimization methods and software, 1992. **1**(1): p. 23-34.
60. Fletcher, R. and E.S. de la Maza, *Nonlinear programming and nonsmooth optimization by successive linear programming.* Mathematical Programming, 1989. **43**(1-3): p. 235-256.
61. Melanie, M., *An introduction to genetic algorithms.* Cambridge, Massachusetts London, England, Fifth printing, 1999. **3**.
62. Lin, C.J., C.W. Hsu, and C.C. Chang, *A practical guide to support vector classification.* National Taiwan U., 2003.

VITA

Fırat Uzman graduated from Üsküdar American Academy in 2002. Then he obtained BS degree in Mechatronic Engineering at Sabanci University in 2006. He worked as an Automation Engineer at Kale Altınay Robotics and Automation, a Design Engineer in Mekano Technic and an R&D Project Engineer in TEMSA Global before joining the R&D Department of Turkish Petroleum Refineries Corporation (Tüpraş) as a Modeling, Control and Optimization Engineer in 2010. He lead or supported many projects regarding simulation, control or optimization of refinery units and developed decision support tools for utility production and consumption management and for inter-refinery hydrocarbon logistics management in cooperation with universities like Boğaziçi University, Middle East Technical University, İstanbul Technical University and Koç University. While working as the Superintended of the Optimization and Software Team in R&D Department of Tüpraş, he joined Koç University to pursue MSc. degree in Industrial Engineering. He is currently a Contract Management Coordinator at Tüpraş.