Probability Based Approaches to Process Data Modeling and Rectification

by

Lloyd Patrick Murphy Johnston

MSCEP, Massachusetts Institute of Technology
June, 1992
B.Sc.Eng.(Hons.), Queen's University
June, 1990

Submitted to the Department of Chemical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 1996

© Massachusetts Institute of Technology 1996. All rights reserved.

Author

Department of Chemical Engineering
May 8, 1996

Certified by

Mark A. Kramer
Thesis Supervisor

Accepted by

Robert E. Cohen
Chairman, Committee for Graduate Students
Probability Based Approaches to Process Data Modeling and Rectification
by
Lloyd Patrick Murphy Johnston

Submitted to the Department of Chemical Engineering on May 8, 1996 in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Keywords: data rectification, data reconciliation, gross error detection, nonparametric models, expectation maximization, state estimation, probability density function, probabilistic inference, Bayes theorem, maximum likelihood, elliptical basis functions, systems engineering, probability theory, chemical engineering.

Abstract

Chemical plants measure a large number of variables to obtain a picture of the state of the plant, and use this picture to make operating decisions. All measurements contain an error, either a "small" error due to sensor noise, or a "large" error (gross error) due to sensor miscalibration or failure. It is preferable to base operating decisions on the true state of the plant and not on the error containing measurements. Data rectification is the process of removing the measurement errors, to obtain an estimate of the true plant state. This research applies a first principles approach to data rectification, and incorporates all knowledge of the process that is available from the plant operating history and any known plant models.

The proposed probabilistic framework for data rectification is maximization of the probability of the true plant states given the measurements. That is, a set of true plant states that are most likely, given the measurements, is sought. Bayes theorem is applied to this statement, and the result is the Maximum Likelihood Rectification (MLR) objective function. The MLR objective function is the product of the probability of the true plant state and the probability of the adjustment made to the measurement to reach the true plant state. Robust sensor models, that allow for multiple modes of operation including normal, miscalibrated, and failed, are developed and incorporated into the MLR framework.

The probability distribution of the true plant states is estimated from a set of noisy and corrupted measurement data using the developed Recursive State Density Estimation algorithm. The RSDE uses the principles of Expectation Maximization (EM), to link probability density estimation and data rectification, to estimate the probability distribution of the true plant states. Elliptical Basis Function (EBF) networks are developed as a nonparametric probability density function estimation technique. The EBF is capable of fitting any arbitrary probability distribution in the limit of infinite data. In the realistic case of finite data sets, the EBF is demonstrated to
give good approximations to a number of distributions, including approximating
distributions for chemical process data. Advantages of EBF's over parametric and other
nonparametric probability density function estimation techniques is demonstrated on a
number of examples.

The developed MLR and RSDE methods are able to incorporate any form of
probability density function, and any type of sensor model. For the particular case of a
robust sensor model and an EBF estimate of the probability distribution of the true plant
states, which was found to apply to a number of chemical engineering example
problems, the MLR objective function is complex and time consuming to solve using
existing nonlinear programming methods. An efficient solution algorithm, which
transforms the complex objective function to a surrogate quadratic objective function by
the introduction of two sets of indicator variables, is developed. The EM algorithm is
used to iteratively estimate the indicator variable values and solve the simpler surrogate
objective function, until a solution is obtained. This solution technique was found to be
approximately 20 times faster than conventional nonlinear programming techniques.

The MLR, RSDE, and solution algorithms were applied to a number of chemical
engineering examples, including a cyclohexane plant, simulated using AspenPlus, that
had 30 measurements. The RSDE was able to effectively estimate the probability
distribution of the true plant states for all of the examples, even when part of the data set
was corrupted by gross errors. The MLR framework was also able to effectively
estimate the true plant states, and remove both noise and gross errors from the
measurements in one-step, in the examples examined. In all cases the MLR produced
more accurate rectified states than conventional data reconciliation techniques. The
MLR also is able to solve a new class of problem, one in which no model constraints
are known.

Thesis Supervisor: Professor Mark A. Kramer
Acknowledgments

There are many people here at MIT who have contributed to my life over the past years. First and foremost I thank my advisor Mark Kramer for his dedication to this work. I cannot express my appreciation for all the time he has given up to see this thesis through to completion, especially after he departed from MIT. The thesis, and the ideas within, would not have come about without his keen insight, broad range of knowledge, and enthusiastic interest. He is also responsible for trying to teach me (perhaps fruitlessly), where the “,” on the keyboard is located (inside joke).

My thesis committee, Professor Paul Barton, Professor Greg McRae, and Professor George Stephanopoulos, added insight and helped bring the present form of this work about. I also thank my research group members, Dr. Carlos Rojas-Guzman, Dr. Jonathan Tan, and particularly Dr. Mike Thompson, for their stimulating discussions and ideas regarding this work. My officemate Fred von Gottberg was always there to listen and help out on problems, or was there just to take my mind of the work for awhile, and for that I thank him.

Outside of the office, there are many people who have made my stay at MIT very enjoyable. I thank Melissa for being there over the last three years to enjoy the good times, and for putting up with me during the bad (stressed out) times. My good friends and roommates, Dr. Colin Wolden, Dr. Rick Batycky, and Andrena Batycky, made life away from MIT very enjoyable, if not a little blurry on Friday mornings (Plough anyone?).

My involvement in MIT athletics has also been a great diversion from the stresses of research. I particularly thank Joe Quinn, coach of the MIT hockey team which I played on for four years. Joe is more than a coach, he is a friend, and his outlook on life and caring attitude has helped me get through a few of the rough times I experienced here at MIT.

Last, but certainly not least, I thank all of the MIT staff that have made MIT a wonderful place to work. Janet Fischer and Elaine Auñiero daily brightened my day on my visits to their office, and were always extremely helpful in sorting out any administrative mess that I found myself in. It will be strange not to make the daily walk down the hall. Carol Philips was always extremely helpful, and Arline Benford and Linda Mousseau in headquarters were always there to help out and to make me laugh. Bob Morrow brightened (or blinded) my day with his colorful outfits and witty remarks both on and off the golf course.

There are numerous other colleagues and friends that have made life interesting and exciting, and I thank all of you for that (especially the weekly golf crowd Todd, Gabe and Mike).
Table of Contents

List of Figures ................................................................. 11
List of Tables ................................................................. 13

Chapter 1: Introduction .................................................. 15
  1.1 Motivating Issues .................................................. 16
  1.1.1 Plant Safety .................................................. 16
  1.1.2 Efficient Plant Operation .................................. 17
  1.2 Data Rectification and Data Reconciliation ................. 17
  1.3 Outline of Approach ............................................. 19
  1.4 Research Objectives ............................................ 21
    1.4.1 Data Rectification Framework Capturing .............. 21
    All Process Knowledge
    1.4.2 Probability Density Function Estimation of ......... 21
    Plant States
    1.4.3 Non Parametric Data Density Estimation .............. 22
    1.4.4 Model Selection ........................................ 22
    1.4.5 Sensor Models ........................................... 22
    1.4.6 Gross Error Identification .............................. 23
    1.4.7 Efficient Solution of Posed Problem ................. 23
  1.5 Thesis Overview .................................................. 23
Chapter 5: Estimating State Probability Distributions

From Noisy and Corrupted Data ........................................ 118

5.1 Introduction ....................................................................... 119
5.2 Expectation Maximization .................................................. 120
5.3 Recursive State Density Estimation ................................. 121
5.4 Incorporating Linear Constraints ......................................... 124
5.5 Performance Measures ..................................................... 126
   5.5.1 Log Probability (J value) ............................................. 126
   5.5.2 Log Likelihood .......................................................... 127
   5.5.3 Mean Squared Error ................................................... 128
5.6 Examples ........................................................................... 129
   5.6.1 One Dimensional Gaussian Distribution ..................... 130
   5.6.2 Linearly Constrained Flow Network ......................... 134
   5.6.3 Heat Exchanger Network .......................................... 142
5.7 Conclusions ................................................................. 147

Chapter 6: An Efficient Algorithm for Maximum Likelihood Data Rectification ........................................... 149

6.1 Introduction ....................................................................... 150
6.2 Surrogate Objective Function ............................................ 155
6.3 Solution Using Expectation Maximization ....................... 158
6.4 Initial Conditions .................................................. 163
6.5 Finding the Global Optimum ...................................... 164
6.7 Examples .................................................................. 169
  6.7.1 Heat Exchanger Example ...................................... 170
  6.7.2 Flow Network Example ........................................ 174
6.8 Algorithm Efficiency ................................................ 176
6.9 Conclusions ............................................................. 180

Chapter 7: Case Study - Cyclohexane Process Plant ............ 181
  7.1 Introduction .......................................................... 181
  7.2 Process Description ................................................ 182
  7.3 Process Simulation .................................................. 184
  7.4 Process Operating Conditions .................................... 184
  7.5 Calibration of P(x) ................................................... 185
  7.6 MLR Performance ................................................... 187
  7.7 Identification of Sensor Miscalibration ....................... 198
  7.8 Conclusions .......................................................... 198

Chapter 8: Conclusions and Future Directions ..................... 200
  8.1 Conclusions .......................................................... 200
  8.2 Future Directions ................................................... 202
    8.2.1 Dynamic System Data Rectification ....................... 202
    8.2.2 Probabilistic Process Modeling ............................ 203
    8.2.3 Multivariate Statistical Process Control .................. 204
    8.2.4 Fault Detection and Fault Diagnosis ...................... 204
List of Figures

Figure 3.1  conceptual fit of Parzen windows versus EBF estimator .................. 43
Figure 3.2  mesh plots for EBF, Parzen windows, and Gaussian estimators for case 1 .......... 61
Figure 3.3  mesh plots for EBF, Parzen windows, and Gaussian estimators for case 2 .......... 62
Figure 3.4  mesh plots for EBF, Parzen windows, and Gaussian estimators for case 3 .......... 63
Figure 3.5  typical training data for 2 dimensional classification problem ................. 67
Figure 3.6  heat exchanger network diagram .................................................... 72

Figure 4.1  state subspace for mixing T showing historical data distribution ............... 84
Figure 4.2  schematic representation of P(x) versus P(δ) tradeoff in MLR .................... 92
Figure 4.3  influence curve for Lorentzian distribution of errors .......................... 96
Figure 4.4  influence curve for ordinary and robust regression using bivariate Gaussian errors .......... 99
Figure 4.5  influence curve for MLR and conventional reconciliation - example 3 ........... 101
Figure 4.6  influence curve for MLR and conventional reconciliation - example 1 ........... 103
Figure 4.7  flow network diagram ................................................................. 106
Figure 4.8  heat exchanger network diagram ..................................................... 109
Figure 4.9  plot of rectified, measured, and true values for TA4 ................................ 112
<table>
<thead>
<tr>
<th>Figure 5.1</th>
<th>Recursive State Density Estimation algorithm</th>
<th>............</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5.2</td>
<td>flow network example</td>
<td>............</td>
<td>135</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>log likelihood at each RSDE iteration for flow network with P(x) in 2 dimensions</td>
<td>............</td>
<td>138</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>MSE at each RSDE iteration for flow network</td>
<td>............</td>
<td>139</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>plot of rectified, measured, and true values for TA4, for training data with 10% gross errors</td>
<td>............</td>
<td>146</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>heat exchanger network</td>
<td>............</td>
<td>143</td>
</tr>
<tr>
<td>Figure 6.1</td>
<td>schematic diagram for investigation of objective function shape</td>
<td>............</td>
<td>165</td>
</tr>
<tr>
<td>Figure 6.2</td>
<td>graphic outline of solution algorithm</td>
<td>............</td>
<td>167</td>
</tr>
<tr>
<td>Figure 6.3</td>
<td>heat exchanger network diagram</td>
<td>............</td>
<td>171</td>
</tr>
<tr>
<td>Figure 6.4</td>
<td>flow network diagram</td>
<td>............</td>
<td>175</td>
</tr>
<tr>
<td>Figure 7.1</td>
<td>cyclohexane plant diagram and sensor location</td>
<td>............</td>
<td>183</td>
</tr>
<tr>
<td>Figure 7.2</td>
<td>RSDE log likelihood results for cyclohexane plant</td>
<td>............</td>
<td>188</td>
</tr>
<tr>
<td>Figure 7.3</td>
<td>RSDE MSE results for cyclohexane plant</td>
<td>............</td>
<td>189</td>
</tr>
<tr>
<td>Figure 7.4</td>
<td>plot of rectified, measured, and true values for product flow rate</td>
<td>............</td>
<td>192</td>
</tr>
<tr>
<td>Figure 7.5</td>
<td>plot of rectified, measured, and true values for reactor pressure</td>
<td>............</td>
<td>193</td>
</tr>
<tr>
<td>Figure 7.6</td>
<td>plot of rectified, measured, and true values for reactor temperature</td>
<td>............</td>
<td>194</td>
</tr>
<tr>
<td>Figure 7.7</td>
<td>plot of rectified, measured, and true values for separator temperature</td>
<td>............</td>
<td>195</td>
</tr>
</tbody>
</table>
Figure 7.8  plot of rectified, measured, and true values for distillation column feed tray temperature ........................................ 196

Figure 7.9  identification of a sensor bias by MLR ........................................ 199

List of Tables

Table 3.1  cross validation log probabilities for the 3 test cases ........................................ 60

Table 3.2  performance of EBF on 2 dimensional classification problem ........................................ 70

Table 3.3  input distributions for heat exchanger network example ........................................ 73

Table 3.4  fault classification tables for identifying a bias ........................................ 76

Table 4.1  MSE results for flow network data rectification ........................................ 108

Table 4.2  MSE results for heat exchanger network data rectification ........................................ 111

Table 4.3  MSE results for flow network data rectification, with $P(x)$ from data partially corrupted with gross errors ........................................ 114

Table 4.4  effect of b and p parameters on MSE results for noise data ........................................ 115

Table 4.5  effect of b and p parameters on MSE results for gross error data ........................................ 116

13
Table 5.1  log probability values for RSDE, x data, and y data derived probability distribution functions  .......... 137

Table 5.2  operating conditions for heat exchanger network  .......... 144

Table 6.1  heat exchanger network operating conditions  .......... 172

Table 6.2  MSE results for the flow network  .......... 177

Table 6.3  comparative algorithm speed results  .......... 179

Table 7.1  cyclohexane plant operating conditions  .......... 186

Table 7.2  comparative MSE results for all measurements for cyclohexane plant data with noise and individual gross errors in 25 sensors  .......... 190

Table 7.3  comparative MSE results calculated only on flow measurements for cyclohexane plant data with noise and individual gross errors in all flow sensors  .......... 197
Chapter 1: Introduction

Chemical processing plants measure many variables to ensure plant safety, control product quality, and minimize operating costs. All measurements include an error, either sensor noise which occurs under normal operation or gross errors due to sensor miscalibration or failure. Operating decisions based on the measurements may be bad decisions because they are based on poor information due to the presence of measurement errors. This can lead to safety incidents, production of off-specification product resulting in financial loss, or higher operating costs. The problem of errors in measurements is well known in the process industry and many methods have been developed over the last 30 years to remove these errors. In previous formulations of the data rectification problem many restrictive assumptions about the process have been made, based on the belief that the problem could only be solved under these restrictive assumptions. Operating chemical plants often violated these assumptions and many auxiliary techniques were then developed to pre-process the data to try and make it conform to the assumptions that allowed for a solution. This thesis takes a first-principles look the data rectification and develops a framework where the very restrictive assumptions made in previous schemes are not required, leading to higher performance and enabling the solution of a new class of rectification problem.
1.1 Motivating Issues

The general motivation for removing errors from measured data, is to provide better estimate of the true plant state on which to base operating decisions.

1.1.1 Plant Safety

Safety operating procedures are designed for chemical processing facilities to minimize the risk of a catastrophic event occurring, by outlining the appropriate steps to take given the situation. The current situation is derived from the picture of the plant that is painted by the measurements that are being made. When sensor errors exist this is an inaccurate picture and incorrect steps may be taken due to the incorrect information the decisions are based on. In some cases the sensor error cause a step to be taken that makes the situation more hazardous.

For example, a thermocouple on a reactor, in which an exothermic reaction was occurring, failed by sticking at a low value. Thus the operator decided not to increase the cooling water flow to the reactor even though the actual temperature in the reactor was steadily increasing. The reaction ran away causing the reactor to explode (Kletz, 1994). In this case if the true temperature of in the reactor had been accurately estimated the operator would have known to increase the cooling water flow and the explosion would have been avoided.

The accident at Three Mile Island in 1979 was also an example of poor operating decisions being made because of faulty measurements. There were several indications that the level in the primary water circuit were low but two instruments erroneously indicated a high level. The operators believed the high readings and ignored the other indicators that implied that the level was low. The safety procedures were initiated for a situation where the water level was high, but because the actual
problem was a low water level the situation deteriorated (Kletz, 1994). Thus not only is it important to have a data rectification system that can estimate the true states of the plant but it is also important to incorporate all information sources and appropriately weight the evidence from each source.

1.1.2 Efficient Plant Operation

The plant measurements are used to make many operating decisions that affect product quality and operating costs. When these decisions are based on erroneous data product quality can decrease and operating costs can increase. Sanders (1995) studied key chemical processes at Monsanto Chemical Company to determine the primary cause of upsets. Upsets were defined as excursions that were severe enough to result in restriction or downgrading of the product. It was found that nearly two-thirds of the process upsets could be traced to sensor errors. Many of these upsets were caused by fairly large errors due to various types of sensor failures, but errors as small as 1.5% in a flow signal caused a process upset. Often the sensor fault itself does not cause the process upset, but it is the operating decisions based on the error containing measurements that can lead to a process upset. Thus it is important to develop systems that remove both large and small errors from measurement data so that operating decisions can be made on a more accurate estimate of the true state of the plant.

1.2 Data Rectification and Data Reconciliation

The goal of data rectification is to estimate the true states of the system from error-containing measurements. This thesis takes a first principles approach by posing the logical question: “What set of estimated plant states will maximize the probability of the states given the measurements?” That is, we maximize the probability of what
we desire to know, the states, given what we observe, the measurements. Bayes theorem is then applied to this statement to obtain:

\[
\max_x P(x|y) = \max_x P(y|x)P(x) / P(y) = \max_x P(y|x)P(x)
\]

where \(x\) represents the states, \(y\) represents the measurements and the final equality results because \(P(y)\) is only a normalizing constant and does not depend on \(x\). We call the problem posed in equation 1.1 the Maximum Likelihood Rectification (MLR) problem. The first term in the MLR is \(P(y|x)\), which is the probability of the measurements given the states. This is the probability distribution of the difference between the true plant states and the measurement, and thus is equal to the probability distribution of the measurement error. The second term in the MLR is \(P(x)\), which is the probability of the true plant states.

The problem of noisy and corrupted measurements in process data has long been known and many approaches to removing these errors have been made over the last 30 years. These methods are reviewed in Chapter 2. These previous approaches to data rectification fall into the category of data reconciliation and result from equation 1.1 by making two main assumptions. The first assumption, that has been explicitly made in previous approaches, is that the measurement error is zero-centered Gaussian distribution. The second assumption, that has not been clearly stated prior to this work, is that the probability of the plant states are represented only by a set of analytical equations, and all states that satisfy the constraints are equally likely. Thus the measurements are reconciled with the prescribed plant model, leading to the term data reconciliation. However, reconciliation does not always lead to the most likely state
estimate. Under these two assumptions, equation 1.1 leads to a constrained least squares problem. The analytical solution to this problem when the constraints are linear has become the standard, although not applicable to cases where the two restrictive assumptions are not met. Numerous auxiliary gross error detection techniques have been developed to pre-process the data so that the Gaussian noise assumption is met.

Data reconciliation, described above, adjusts measurements to satisfy a prescribed process model. This is a special case of data rectification which we define more broadly according to the literal meaning of rectification, which is to make right. Only under the specific assumptions outlined in this section does data reconciliation lead to the most correct states given the measurements. The numerous auxiliary techniques developed to pre-process the data to meet the Gaussian noise assumption have been used because a solution that was fast enough to apply on-line was required, and the analytical solution to the standard problem allowed this. The data rectification method developed in this thesis does not make the two restrictive assumptions made in previous approaches, and in doing so, incorporates more information about the process, resulting in better data rectification. A solution method which gives an efficient computational solution to the posed problem is also developed.

1.3 Outline of Approach

One of the major contributions of this thesis is the development of methods to avoid the restrictive assumption that all plant states that satisfy the prescribed model constraints are equally likely. There are two problems with this assumption. First it assumes that all the available process knowledge is contained in the model of the plant. This completely ignores the wealth of information that is contained in the operating history of the plant. In many cases only partial process models are available, even
though there is much more structure in the process data, the models describing these correlations are too complex to derive. An example of this would be a distillation column where the tray temperatures are certainly correlated, but it may be very difficult to write a reliable model. There are also statistical relationships between variables that cannot be described in a deterministic model. A properly constructed probability distribution of the true plant states would capture all the physical and statistical relationships between variables. This information is captured in the probability distribution of the plant states $P(x)$, then exploited in the MLR (equation 1.1) to achieve an improved estimate of the plant states. Similarly, all of the states that satisfy the prescribed model constraints are not equally likely. Process plants tend to operate in preferred operating regimes, and the distribution of the plant states, in the subspace of the constraints, can also be exploited to improve the data rectification. In the general case where a partial model of the plant exists, a distribution of states that satisfy the constraints is obtained. In this distribution, physical and statistical correlation between variables that do not appear in the model constraints are also captured. In this manner all of the available information about the process is being used to rectify the data, and improved state estimates over previous approaches is obtained.

This thesis also removes the requirement for auxiliary gross-error detection schemes by avoiding the assumption that the measurement error is described by a zero-mean Gaussian distribution. A sensor can be in one of a number of modes of operation. The most common mode is a normal mode, where the sensor is operating as designed and the sensor error is described by random Gaussian noise. The sensor could also be in a number of failed modes that include miscalibration, failure to a fixed value, and a random output failure. A distribution for the normal mode and each of the failure modes can be constructed, and the compound error distribution is derived by an
appropriately weighted sum of these distributions. In this manner the assumption of Gaussian sensor noise is removed and the data rectification is performed in one step. By simply looking at the difference between the measurements and the rectified values, sensors with likely gross-errors can be identified for repair.

1.4 Research Objectives

1.4.1 Data Rectification Framework Capturing All Process Knowledge

The primary source of prior knowledge used by previous methods of data reconciliation (with exception of neural networks) have been in the form of physical models of the process. In many chemical engineering applications the models of the process are too complicated, or based on too many simplifying assumptions, to be useful in data rectification, or the models are simply non-existent. One of the primary goals of this thesis is to formulate the rectification problem so that prior knowledge of the process not in the form of analytical models is explicitly placed in the rectification objective function. At the same time, the rectification framework developed should be able to incorporate prior knowledge of the process in the form of physical models. The main source of knowledge about the process outside the physical models is historical data from the plant that empirically shows relationships among the process variables.

1.4.2 Probability Density Function Estimation of the Plant States

As shown in Chapter 4 of this thesis, the knowledge contained in the plant operating history can be effectively captured in a probability density function of the plant states. Unfortunately the true states of the plant are never known, and all that exists is a historical database of plant measurements that include corruptions due to sensor noise and gross errors. Thus an aim of this thesis is to develop a method for
estimating the probability density function of the plant states from the corrupted measurements that can be applied to any functional form of probability distribution.

1.4.3 Non-Parametric Data Density Estimation

Probability density functions can either be parametric, with an assumed fixed model form (e.g. Gaussian), or non-parametric with a flexible model form. Non-parametric distributions adapt to the data set, and many have been shown to approximate any functional form of distribution under certain conditions. Using a parametric density estimator by assuming a model form puts unnecessary restrictions on the density function. Thus an objective of this thesis is to develop a non-parametric density estimation method that is capable of fitting a wide variety of distributions.

1.4.4 Model Selection

A methodology is also required to choose which type estimator is the best estimator for a particular data set. The ranking criterion should be based on the performance of the estimator on data that was not used in the calibration of the estimator. Data is limited, and thus a methodology must be selected to estimate the estimators performance on future data using all of the available data.

1.4.5 Sensor Models

In order to complete the generalized rectification framework, sensor models must also be developed. It is important to capture different types of sensor behavior, such as normal operation and operation in a failed mode, with models so that adjustments to the measurements are made to correct errors that are likely. A sensor can fail in number of ways, including miscalibration, failure to a fixed value, failure by an additive bias, and a change in the sensor parameters. Defining different modes of sensor operation allows for a variety of different types of sensor errors to be considered
in the rectification scheme. This will lead to a general solution of the rectification problem for any type of sensor error, which is a goal of this thesis.

1.4.6 Gross Error Identification

After data rectification has been performed, it is important to be able to identify which sensors contain gross errors so that the sensor can be repaired or replaced. An objective of this thesis is to present a method for assessing the likelihood that a given sensor has failed.

1.4.7 Efficient Solution of Posed Problem

Once the framework for the data rectification has been set, and the objective function decided upon, then a numerical solution is required. The numerical technique must solve the problem fast enough so that the rectification scheme is applicable to real-time on-line measurements. Some of the primary issues are the existence of multiple solutions, ranking of local solutions, location of a global optimum, the number of variables involved in the optimization, and the speed of the numerical technique. Solution of industrial rectification problems is the ultimate objective, and thus developing numerical techniques for the solution of the rectification problem is an important goal of this thesis.

1.5 Thesis Overview

The four main pillars that the thesis is built on are: 1) Maximum Likelihood Rectification (MLR), 2) Recursive State Density Estimation (RSDE), 3) Elliptical Basis Function (EBF) probability density estimation, and 4) an efficient algorithm for the solution of the MLR problem. The four pillars are interrelated and depend on each other. The EBF, which is a non-parametric probability density function estimator, is first developed in Chapter 3. The need for a non-parametric density estimator in data
rectification is seen in Chapter 4 where the proposed MLR method for data rectification is developed. The MLR casts the data rectification problem in a probabilistic framework and maximizes the probability distribution of the states given the measurements. Applying Bayes theorem to this results in the MLR objective function (equation 1.1) which is the product of the probability of the rectified state, based on the prior probability of the plant states (P(x)), and the probability of the adjustment made to the measurement to achieve the rectified state, based on the probability distribution of the sensor error (P(y|x)). Chapter 5 focuses on the estimation of P(x) from a set of corrupted measurement data by developing the RSDE algorithm. Recursive State Density Estimation combines data rectification and data density estimation, through the principles of Expectation Maximization (EM), to arrive at the most likely distribution of the plant states given the measurements. Chapter 6 uses a different application of the EM algorithm to develop an efficient algorithm for the solution of the MLR problem. Chapter 7 is devoted to a case study of a cyclohexane production process. Background and review of relevant data rectification literature is found in Chapter 2. Prior work on other areas explored in the thesis, such as density estimation and Expectation Maximization, are found in the relevant chapters.
Chapter 2:  
Background and Literature Review

All chemical processing plants make measurements of a wide variety of process variables both on and off-line. These measurements are used to control product quality, monitor plant safety, and to maximize profit through operation decision making. All measurements are subject to error, and they are only an approximation of the true state of the world. The presence of these errors can adversely affect product quality, plant safety and profit maximization. Because of the value of knowing the true state of the plant, many efforts are made to remove the errors from the measured data. These errors are divided into two categories. The first and most common category is random noise in the sensor signal. All process sensors have some amount of noise associated with their signal. The noise on the sensors are usually modeled as independent, zero mean Gaussian random variables, with standard deviations much less than the normal span of the instruments. The other category of errors are gross errors, which are caused by non-random events, such as a malfunctioning sensor. Both random noise and gross errors corrupt measurements, which in turn give operators an inaccurate picture of the true state of the plant. This incorrect picture can lead to operation decisions that adversely
affect product quality, cause non-economic operation of the plant, or affect the safety of the process.

Process data is the foundation upon which all control and evaluation of process performance is based. The integration of modern chemical plants cause the ramifications of process operating decisions to be far reaching and difficult to foresee. Inaccurate process data can lead to poor decisions that can adversely affect many parts of the plant. In addition, many process control and optimization activities rely on small improvements in process performance, and errors in process measurements can easily exceed, or mask actual changes in process performance. The problems associated with random and gross errors are as old as the processing industry itself. Before automatic data acquisition techniques, measurements were normally carried out by plant operators and engineers. These people used their knowledge of, and experience with, the process to assess the accuracy and consistency of the collected data. With the advances in computer technology and automatic control of chemical plants, much more data is collected which needs to be checked and processed. There is too much data for it to be done by humans, and thus great effort has been made to develop a rational and systematic basis for checking process data.

The term data rectification literally means to make data right. As will be shown in the following sections most previous data rectification schemes are really data reconciliation, which is the adjustment of the data to fit a prescribed modeled, that is reconciling the data to the model. Data reconciliation is a special case of the general data rectification problem, and is the area that has received the most research attention.

The data reconciliation literature for steady state systems is divided into two main categories, linear and nonlinear systems. The steady state linear problem has received the most attention in the literature, and the most common type of real system
that fits this criteria are mass balance networks. Some work has been done on the nonlinear steady state problem, but it is not nearly as extensive as the linear steady state literature. In many cases the techniques, specifically general optimization based approaches, developed for nonlinear systems, are applicable to their linear counterparts. Using these techniques for the simpler linear problem has the disadvantage of taking more computation time than the linear techniques developed that exploit the linear characteristics of the system. Neural networks have also been applied to the data rectification problem with some success.

2.1 Linear Data Reconciliation

The most common reconciliation method for a steady-state linear system is the minimum weighted least squares approach first suggested by Kuehn and Davidson (1961). The system of interest is modeled by linear steady state equations, most often mass balances, and then the measurements are adjusted so that the model equations are satisfied. The sum of the square of the adjustments, each term weighted by the inverse of its measurement noise variance, is used as the objective function in the minimization so that the smallest adjustments that satisfy the model equations are found. Mathematically the problem is:

$$\min (y-x)^{T}Q^{-1}(y-x) \quad \text{s.t. } Ax=0 \quad (2.1)$$

where $y$ is the vector of measurements, $x$ is the vector of adjusted values, $Q$ is the covariance matrix of the measurement noise, and $A$ is the matrix of linear constraints. An analytical solution of this minimization exists, which makes calculation of the rectified values very fast, and suitable for on-line application.
The usual assumption is that measurement errors are normally distributed. When this assumption is true (which is most of the time), the weighted least squares estimation is the same as the maximum likelihood and minimum variance estimation, and gives rise to unbiased estimates (Mah, 1987). A complication arises in the linear reconciliation problem when not all of the variables that appear in the model are measured. Crowe et al (1983) use a projection matrix to solve this problem. The constraint matrix $A$ is partitioned into two constraint matrices, $A_1$ for the measured variables and $A_2$ for the unmeasured variables (denoted by $u$). Thus the constraints on the minimization become:

$$A_1 x + A_2 u = c \tag{2.2}$$

where $c$ is a constant vector that represents any measurements (if any) that are known exactly. This equation is then pre-multiplied by the projection matrix $P$, that has the property $PA_2=0$. This then transforms the problem with unmeasured variables to one with measured variables only and the same least squares solution can be used.

The weighted least squares method of data reconciliation is the primary method of reconciling data for a linear system. If there are gross errors in the data then the weighted least squares method gives biased results, as the gross error measurement is assumed to be a true value with a small amount of additive noise. Thus much of the literature on linear data reconciliation and rectification has focused on gross error detection and removal. Ripps (1965) realized that gross errors would invalidate the weighted least squares procedure giving poor reconciled values and thus should be removed before the data reconciliation was performed. Ripps results showed that when a gross error was present in one measurement that the error was distributed throughout
the other measurements. Thus "medium" sized adjustments were made to all measurements, as opposed to a large adjustment in the biased measurement and small adjustments in the other measurements. Ripps laid the ground work for the serial elimination procedure developed by Nogita (1972). The procedure calculates a test statistic based on the adjustments and eliminates measurements one at a time, each time calculating the test statistic for the reduced set of measurements. The set of measurements that pass the no gross error test, comparison of the test statistic with an appropriate standard value (e.g. normal variate score), are deemed not to contain gross errors. The measurements that are not in the set (whose presence had previously caused failure of the test) are said to contain gross errors. The procedure is most effective for a single gross error in the measurement set, as with multiple gross errors the possible combinations of measurements to delete becomes very large quickly.

Essentially all gross error detection schemes for linear steady state systems follow the same pattern. A test statistic is calculated then compared to the appropriate score for that test statistic at a certain level of significance (e.g. comparison to a $\chi^2$ variable). If the test is passed then no gross error exists, if it fails then a gross error is present. One of the early gross error detection schemes to follow this general procedure was presented by Madron et al (1977) and applied to a fermentor. Almasy and Sztano (1975) classified an error to be regular (zero mean Gaussian variable) or an extreme error by means of a Chi-squared test, and depending on the classification different reconciliation methods were used to estimate the true value. A heuristic method developed by Vaclavek as well as other early gross error detection schemes are reviewed by Hlavacek (1977).

Gross error detection schemes fall into two categories, ones that allow for direct identification of which measurement contains the gross error, and ones that require
further identification schemes such as serial elimination procedure. Three of the most common gross error detection tests are the global, nodal and measurement tests (Mah, 1987). The global test and nodal test do not require that reconciliation of the data be done first, but look at the constraint residuals based on the measurement data (i.e. how well the measurements satisfy the constraints) (Tamhane and Mah, 1985). The global test combines all of the constraint residuals into a $\chi^2$ variable which is then compared to an appropriate value from tables, to determine if a gross error exists. The nodal test examines each node (i.e. each constraint) equation separately. The constraint residual is a normal variate which is compared to an appropriate value from tables to determine if a gross error exists. Both the nodal and global tests indicate if a gross error exists, but identification of which measurement has the gross error requires further investigation. Most often serial elimination is used. On the other hand, the measurement test has the advantage of directly identifying which measurement contains the gross error, but has the minor disadvantage of requiring data reconciliation to be performed first. The measurement test (Mah and Tamhane 1982, Tamhane and Mah 1985) is based on classical Neyman-Pearson hypothesis testing, and looks at the size of the adjustments made in the reconciliation to decide if a gross error exists. Iordache et al (1985) investigated the performance of the measurement test, and not surprisingly found that the performance depended both on the absolute size of the gross error, and on the ratio of the size of the gross error to the size of the noise standard deviation for the measurement.

Mah and Tamhane (1982) showed that the measurement test has maximum power, in the sense that the test has a greater probability of finding a single gross error in a particular measurement than would any other test based on a linear combination of the measurements. This test was developed into the maximum power test for
constraints by Crowe (1989). This test looks for constraints that are violated and identifies units whose constraints are not satisfied by the measurements. This type of test is designed not to look for sensor failures, but for leaks and other failures in a particular unit. The test eliminated balances from the constraint matrix until the test was passed by the remaining units. This procedure continually combined and reduced the balance equations. If the test concluded that an error in one or more of the constraints existed, and the identified constraint was one of the reduced subset, there may not be a direct relationship to a particular processing unit. This problem was solved by Crowe (1991), and the new method applied the maximum power test to the original constraints.

There are many other gross error detection tests, some bringing new ideas to the field and others expanding on previously established tests. Narasimhan and Mah (1987) used the well known statistical technique of generalized likelihood ratio to develop a general framework for identifying gross errors (other than sensor faults), that can be modeled. Serth and Heenan (1986) developed two gross error detection schemes, the screened combinatorial method and the modified iterative measurement test. They were developed to specifically deal with mass balance problems where their is a wide range of flow rates (two orders of magnitude), and are somewhat ad hoc in nature. Rosenberg and Mah (1987) recognized that sometimes data reconciliation schemes gave results that were physically impossible, such as negative concentrations, and modified the measurement test to incorporate bound violation. The two algorithms they developed are the extended measurement test and the dynamic measurement test. The extended measurement test uses a modified serial elimination scheme, and the dynamic measurement test uses the same modified serial elimination scheme but the candidate set of potential gross error measurements can be augmented as the procedure continues.
Romagnoli and Stephanopoulos (1980) used structural analysis of the balance equations, along with statistical testing, and sequential analysis of the identified candidate set to detect gross errors. Tamhane (1988) used a Bayesian approach to incorporate past information about sensor failure rates into the gross error detection process. Mah et al (1976) used directed graphs and graph theoretic results to break down and solve problems with missing measurements. Rollins and Davis (1992) developed the unbiased estimation technique to deal with biased process measurements, which has the added feature of reporting confidence limits on the reconciled values.

In all of these reconciliation and gross error detection schemes, two assumptions have been made, first that the model equations can readily be put in the linear constraint form and second, that the noise covariance matrix of the data is known. It is not always easy to put the model equations into the appropriate linear constraint form. Narasimhan and Mah (1989) developed a method for taking generalized steady state models, including unmeasured and indirectly measured variables, and put it into a form containing measured variables only. The basis of the technique is partitioning of the constraint matrix and pre-multiplication of the partitions to remove the unmeasured variables and then manipulations to get the indirectly measured variables in terms of the measured variables. It is generally assumed that the noise covariance matrix of the measurements is known, but in practice this is not usually true and it often must be estimated. One way to estimate it is to hold the plant and steady state for an extended period of time, take many measurements, and do the covariance calculation. It is virtually impossible to hold a plant precisely at steady state, and the noise covariance matrix calculated will be corrupted by deviations from steady state. Keller and Zasadzinski (1992) proposed a method for calculating the noise covariance matrix that does not require the plant to be at steady state, but instead relies on the mass balance,
and calculates the noise covariance matrix based on the residuals of the mass balance. This method implicitly makes the assumption that the plant is at a quasi-steady-state, that is although the state may change over time, at any given instant in time the steady-state mass balances hold. This assumption is good for pipeline networks but not for chemical plants whose processing vessels have dynamics that invalidate the quasi-steady-state assumption.

2.2 Nonlinear Data Reconciliation

Much less work has been done on nonlinear data reconciliation than on linear data reconciliation. The most common approach to dealing with nonlinear constraints is to linearize them and use one of the linear techniques (Mah, 1987). The primary problem with this is that the linearization is an approximation, and thus greater errors may be introduced by forcing the measurements to conform to the linearized model than originally existed in the data. Knepper and Gorman (1980) estimated measured and unmeasured variables using a Gauss-Newton type iterative algorithm for nonlinear constraints that were approximated by Taylor series expansions. Crowe (1986) used matrix projections to handle reconciliation for bi-linear constraints and linearized nonlinear constraints. Efforts have also been made for solving general nonlinear reconciliation problems, one of the most successful methods being the errors-in-variables (EIV) technique. Britt and Luecke (1973) applied the EIV method for data reconciliation and parameter estimation for nonlinear systems. The EIV technique was originally conceived to improve parameter estimation from experimental data, but the application to data rectification follows directly.

The EIV method is extremely useful when the form of the process model is known, but the parameters of the model are unknown. The model equations are in the
form of measured dependent variables (column vector $y$) being a function of the measured independent variables (column vector $x$) and the set of unknown parameters ($\theta$). Thus the estimated value of the measured variables given the dependent variables and the parameters is

$$y_{est} = f(x, \theta)$$  \hspace{1cm} (2.3)

Normal parameter estimation techniques allow for errors in the dependent variables, but assume that there are no errors in the independent variables. The measurements are used to compute the values of the parameters that are most likely (maximum likelihood estimation) by minimizing the model residuals (Bard, 1974)

$$\min_{\theta} e_y^T Q^{-1} e_y$$  \hspace{1cm} (2.4)

where

$$e_y = y - f(x, \theta)$$  \hspace{1cm} (2.5)

and $Q$ is the noise covariance matrix of the dependent variables. For almost all systems there are errors in the measured independent variables as well as the dependent variables. The EIV method allows for this by forming a column vector of error terms consisting of the errors in the measured variables and the model residuals,

$$e = \begin{pmatrix} e_x \\ e_y \end{pmatrix}$$  \hspace{1cm} (2.6)

where

$$e_x = x - x_{est}$$  \hspace{1cm} (2.7)
The parameters and reconciled x values are then estimated by solving the optimization problem (Bard, 1974)

\[
\min_{x_{\text{est}}, \theta} e^T V^{-1} e^T
\]  

(2.8)

Here the weighting matrix V has \(E(e_x e_x^T)\) in the upper left block, \(E(e_y e_y^T)\) in the lower right block, \(E(e_x e_y^T)\) and \((E(e_x e_y^T))^T\) in the upper right and lower left blocks respectively. The reconciled y values are then computed by using the calculated parameters and reconciled x values in the model equations (Eq. 2.3).

Much attention has been focused on the solution of the problem posed in Eq. 2.8. Deming (1943) solved the problem based on the Lagrange method, with linearization of the model at the experimental measurements. Similar approaches have been developed by Peneloe et al (1976), Sutton and MacGregor (1977) and Reily and Patino-Leal (1981). Britt and Luecke (1973) based their solution on the Gauss-Newton approach, and Powell and Macdonald (1972) used a Newton Raphson approach. Other investigators have focused on decomposing the problem, and solving a set of sub problems in a reduced space (Bard, 1974; Anderson et al, 1978; Kim et al, 1990; Schwetlick and Tiller, 1985, 1989; Boggs et al, 1987). The performance of EIV solution techniques has been evaluated by Ricker (1984). Almost all of the EIV solution methods experience convergence problems in certain situations. The problems can be overcome by using general nonlinear programming (NLP), but the computational cost can be prohibitively expensive, even for simple models. Thus focus in data rectification has been on solution techniques that are "quick" enough to be applied in real time. Tjoa and Biegler (1992) developed a decoupled sequential quadratic programming (DSQP) method to solve the EIV problem. The solutions
obtained were close to the NLP solutions, but the DQSP solutions were found in a fraction of the time.

Reconciliation methods that reconcile data to fully specified nonlinear models, have also been developed. Rollins and Roelfs (1992) extended the unbiased estimation technique (UBET) of Rollins and Davis (1992) to bilinear constraints, specifically heat balances. In one proposed solution the flows are reconciled using the mass balances and the UBET, then the reconciled values of the flows are assumed to be correct, and the temperatures are reconciled using the heat balances and the UBET. Another proposed solution was to linearize the bilinear heat balances with first order Taylor series and apply the UBET. Neither method proved to be superior over the other in all cases. Tjoa and Biegler (1991) took a consolidated approach to data reconciliation and gross error detection, which takes care of gross errors in the reconciliation step and leads to unbiased estimates. This is accomplished by using a bivariate objective function. Instead of strictly minimizing the adjustments, as in the weighted least squares approaches, a probability distribution of the adjustments is formed, and it is maximized subject to the constraints. That is the most probable adjustments that satisfy the constraints are used to find the reconciled values. The probability distribution of the adjustment contains two parts, a "narrow" Gaussian distribution centered at zero, representing normal process noise, and a "broad" Gaussian distribution, also centered at zero, that makes large adjustments somewhat probable. The two Gaussian distributions are weighted by the probability of not having a gross error in a sensor, and the probability of a gross error in a sensor, respectively. There are no restrictions on the constraints the objective function is optimized subject to, and the method works equally well for linear and nonlinear constraints. The primary drawback is the computation time in solving the optimization problem. The distribution (normal noise or gross error)
that the final adjustment is most likely to have come from is used to identify possible gross errors in the measurement. Tamhane and Mah (1985) pointed out that much work is required in the area of gross error detection for systems with nonlinear constraints, and that still remains true.

2.3 Neural Network Data Reconciliation

The previously reviewed methods have all approached the data reconciliation problem assuming that an adequate model of the process is available for use in the reconciliation procedure. Often this is not the case, and the methods in this category try to solve the reconciliation problem by using neural networks to learn about the process from historical data, and then use that knowledge to reconcile future data.

Dobrzeniecki and Lidsky (1990) developed sensor reasoning and abstraction with neural networks (SRANN), to identify gross errors based on signal patterns over time from sensors. The SRANN is a fully connected feedforward network that is trained with examples of normal and abnormal (gross error present) wave forms for a sensor. The SRANN then classifies the sensor wave form it is presented with on-line, as either normal or abnormal, and notifies the operator of potentially faulty sensors. Karjala and Himmelblau (1992) reported limited success using recurrent Elman neural networks (Elman, 1990) for rectification of time series of data. In an Elman net, the output of the hidden layer nodes at the current time step are fed back to the input layer, to what are called context nodes, and then used in the feedforward propagation of the network in the next time step. This structure is supposed to have the affect of retaining more knowledge about the process, but much more work on this is required before definite conclusions can be drawn.
Kramer (1991, 1992) developed autoassociative neural networks (AANN) for data rectification. The AANN is a fully connected feedforward neural network with three hidden layers. The first layer is the mapping layer, that maps the input space to a lower dimensional bottleneck space, represented by the middle bottleneck layer. The third hidden layer, is the demapping layer that maps the bottleneck space to the output space. The network is trained to learn the identity function, input equals output. Although this sounds trivial, the compression of the data through the bottleneck layer, the true underlying dimensionality of the process, allows for the removal of noise from the data. Any neural networks performance greatly degrades outside the region of the data it was trained on and thus AANN do not perform well in the presence of gross errors. By artificially corrupting measurements in the training set, and thus increasing the size of the training set, mapping from sensors with gross errors to there correct values can be learned. These so called robust autoassociative neural networks (RAANN) performed, well in identifying and removing gross errors from corrupted data.

These neural network approaches to data rectification do not require process models, but often partial models are available and their inclusion would enhance the rectification process. For example, for a complex processing unit a neural network approach could be used to rectify the data but the rectified values may not precisely satisfy the known mass balance around the unit. Incorporation of this type of partial modeling knowledge would greatly increase the applicability of these neural network approaches.
Chapter 3
Probability Density Estimation Using Elliptical Basis Functions

Elliptical Basis Function (EBF) networks are introduced as a new non-parametric method of estimating probability density functions for process data. Unlike Parzen window density estimators that use identical hyper-spherical basis functions, the EBF method uses elliptical basis functions adapted to the local character of the data. This technique overcomes the "spikiness" problem associated with Parzen windows, where in high dimension, Parzen windows can fail to produce smooth probability density estimates. The EBF estimator is shown to produce valid density functions that converge to the underlying distribution of the data in the limit of an infinite number of training examples. A technique based on statistical cross-validation is introduced for evaluating different density estimators. The criterion is a measure of how well the density estimator estimates the density of data not used in the training. The EBF density estimation method and the evaluation technique are demonstrated using several examples of fault diagnosis.
3.1 Introduction

Statistical characterization of a process is required for many process monitoring and control tasks. Although many statistics can be calculated, one of the most useful statistics is the joint probability density function (PDF) of the process variables. Once the PDF is obtained it is possible to perform a variety of monitoring and control tasks. The PDF can be used to describe the normal operating regime of a process, or the parametric region capable of producing product within acceptable quality specifications, and to evaluate the probability that current process conditions will produce product within specifications. If this probability is judged to be too low, then adjustments can be made to the process until the conditions are such that “good” product is highly likely. In fault diagnosis, PDFs can be used to estimate the relative probability of classes of events, given the current information about the process. Using separate PDF estimators for each class, Bayes theorem can determine the class with the highest probability given the current conditions (Duda and Hart, 1973). In sensor validation and gross error detection, a low joint PDF of the current measurements indicates that there is a gross error in one or more of the sensors (assuming that the plant is in a normal operating state). There is also the potential of applying the PDF to data rectification as shown in chapter 4.

Because of the wide applicability of PDFs in process monitoring and control tasks, accurate estimation of PDFs from process data is an important task. There are currently several techniques, both parametric and non-parametric, for estimating PDFs from data. The parametric techniques assume a model of the density function and then regress any required parameters. The major shortcoming of parametric approaches is that often the model form is unknown. A common assumption is that the data approximates normal behavior and a multivariate Gaussian (MVG) model is used to
approximate the density function. This method is not adequate for approximating
density functions that are multi-modal, or density functions in which the process
variables are related by non-linear correlations. When there is prior knowledge that the
underlying distribution is of a certain type (e.g. Gaussian) then the appropriate
parametric technique can be used, but for most processes the underlying distribution of
the data is not known and most likely does not follow a particular class of density
function. In these cases non-parametric techniques are the methods of choice. Non-
parametric techniques avoid prior assumptions of the form of the PDF, and thus
maintain a broader applicability than parametric methods.

One of the most common non-parametric techniques for density estimation is
Parzen Windows (PW) (Parzen, 1962). In this estimator, an identical basis (window)
function is centered at every point in the data set. The density estimate is found by
summing up the density estimates from the individual basis density functions. There
are three problems with this approach. First, there are as many basis functions as data
points, leading to a high computational load. Second, individual basis functions of the
Parzen window estimator are unable to capture the local distribution characteristics of
the data because each basis function is identical, leading to the requirement for many
basis functions to converge to the true local density. It may be much more efficient to
use fewer basis functions that are adapted to the local character of the data. Third the
use of a large number of identical basis functions in Parzen windows can lead to
"spiky" (as opposed to smooth) density estimates (Duda and Hart, 1973). The primary
reasons for this are because the identical window size is inappropriate when the data is
more dense in some areas than others, and because the hyper-spherical shape of the
units are inappropriate when the data is correlated. With a finite data set, the
distribution of the data may require large window sizes to completely cover the data in
one area of the space while in another area of the same space the large window size may have the effect of smoothing over correlations in the underlying density function. On the other hand, using small window sizes to capture the detailed shape of the underlying distribution in one area of the space may have the effect of producing a "spiky" estimate in another area of the space where the training data is more sparse (see Figure 3.1). The proposed Elliptical Basis Function (EBF) estimator is also non-parametric, but because it is capable of capturing the local characteristics of the data, it is less prone to produce "spiky" or poorly fit density functions. The EBF estimator overcomes these problems by allowing for differently sized and shaped basis functions. In areas where the density function is rapidly changing, many small, differently shaped basis functions are placed to capture these variations, and in areas of relative smoothness fewer larger units are placed to reflect the characteristics of that region of the data space (see Figure 3.1). The proposed EBF estimator is shown later in this chapter to be a valid data density function and to converge to the underlying data density distribution in the limit of infinite data.

Kavuri and Venkatasubramanian (1993), Renals and Rohwer (1989), Holcomb and Morari (1991), and Poggio and Girosi (1990) have reported on the use of networks of elliptical basis functions for functional approximation and classification, but not for estimation of probability density functions. Kavuri and Venkatasubramanian solved classification problems by using elliptical units that are oriented with their axes parallel to the axes of the input space (i.e. units had diagonal covariance matrices). Renals and Rohwer used elliptical units with diagonal covariance matrices to solve static speech classification problems. The simplification of a diagonal covariance matrix reduces the computational load but is inefficient, in the sense that it takes many units to capture class regions with large correlations between variables. Poggio and Girosi used regularization techniques to arrive at generalized radial basis function (GRBF) networks
Figure 3.1: Conceptual view of Parzen windows versus EBF fit to data. a) Small Parzen window size captures the variation in the lower left corner, but leaves gaps in the top right corner. The density estimate in the top right portion of the graph would be "spiky". b) Larger Parzen window size eliminates spikiness problem in top right portion but would give an overly smooth estimate in lower left corner missing the correlations among the data. c) EBF covering the data set in a smooth manner, all correlation among the variables is captured and there are no discontinuities.
that use elliptical units with non-diagonal covariance matrices, but have as many units as data points. Based on the high computational load that would result from the large number of units used, Poggio and Giorosi advocated a heuristic approach involving fewer basis function units than data points. The GRBF method was directed at solving functional approximation and classification problems. Holcomb and Morari also allowed for arbitrary orientation of their elliptical units but used a penalty function method that requires some tuning. They also failed to define a technique for selecting the number of units to use and the penalty function to ensure elliptical shapes that conform well to the data.

This chapter not only introduces a new technique for density estimation but it also proposes a methodology for selecting the best data density estimator for a given application. The well-known technique of cross-validation (Weiss and Kulikowski, 1991) is used in conjunction with the "log probability" utility function, to rank the predictive power of each estimator. This methodology can be used to compare estimators of the same basic type (e.g. two Parzen Window estimators with different window widths), or estimators of different types (e.g. EBF versus MVG). The estimator with the highest log probability is chosen as the optimal estimator for the particular application.

This chapter first describes current parametric and non-parametric density estimation techniques and then introduces the proposed EBF technique. The selection of the best technique using cross validation and the log probability is then shown. This process is demonstrated with three examples of density estimation.
3.2 Density Estimators

3.2.1 Multivariate Gaussian

The multivariate Gaussian (MVG) is a standard functional form for parametric density estimation. The assumption underlying the use of the MVG function is that the data are samples from a multivariate Gaussian distribution. The mean $\mathbf{m}$ and the covariance $\mathbf{Q}$ of the data are calculated in the standard way, and the resulting density function is (Ross, 1987):

$$ \rho(x) = \frac{1}{\sqrt{(2\pi)^n \det(Q)}} \exp\left(-\frac{(x-m)Q^{-1}(x-m)^T}{2}\right) $$  \hspace{1cm} (3.1)

3.2.2 Parzen Windows

The Parzen window technique is described by Parzen (Parzen 1962). Each point in the training set is designated as a unit center and an identical basis function is constructed at each center. The functional form of the estimator is

$$ \rho(y) = \frac{1}{K} \sum_{k=1}^{K} \psi(y-x_k) $$  \hspace{1cm} (3.2)

where $\psi$ is a basis function that satisfies the following criteria to ensure $\rho(y)$ is a valid density function:

$$ \psi(u) \geq 0 $$  \hspace{1cm} (3.3)
\[
\int_{-\infty}^{\infty} \psi(u) du = 1 \quad (3.4)
\]

By satisfying the following conditions,

\[
\sup_u \psi(u) < \infty \quad (3.5)
\]

\[
\lim_{u \to \infty} \psi(u) \prod_{i=1}^{n} u_i = 0 \quad (3.6)
\]

\[
\lim_{K \to \infty} \sigma^m(K) = 0 \quad (3.7)
\]

\[
\lim_{K \to \infty} K \sigma^m(K) = \infty \quad (3.8)
\]

the Parzen window estimate will converge to the true distribution of the data in the limit of infinite data (Duda and Hart, 1973). Equations 3.5 and 3.6 ensure \( \psi \) is bounded in the vertical and horizontal directions respectively, and are satisfied by most density functions that would be considered for use as basis functions. Equations 3.7 and 3.8 impose conditions on the choice of the window width as a function of the number of data points, \( \sigma(K) \), to ensure that the window size shrinks to zero but at a slower rate than the growth of the number of data points. Parzen investigated many different forms of basis functions (\( \psi \)) that satisfy equations 3.3 to 3.8, but the one most often used is spherically symmetrical MVG, equation 3.1 with diagonal covariance \( \mathbf{Q} = \sigma^2 \mathbf{I} \). Because the training set fixes the unit centers, the method is fully described by the single parameter \( \sigma \). The optimal choice of \( \sigma \) is described later in this chapter.
3.3 Elliptical Basis Functions

The proposed EBF density estimator has fewer basis functions than Parzen window estimators and the basis functions are non-spherical and not identical for each unit. This allows for capture of the local character of the data. The technique has two parameters, the number of basis functions $H$, and the amount of unit overlap $P$. The form of each basis function is similar to that of a multivariate Gaussian, excluding a constant multiplicative factor:

$$a_h(x) = \exp(-(x-m_h)Q_h^{-1}(x-m_h)^T)$$  

(3.9)

$a_h$ is to be interpreted as the "activation" of each basis function. There are two values that must be found for each unit, the unit center $m_h$, and the unit shape matrix $Q_h^{-1}$. The first two steps in finding these values are essentially identical to the training procedure for radial basis function networks proposed by Moody and Darken (1989), and involve setting unit centers and hyperspherical unit widths. In the first step, $H$ unit centers $(m_h)$ are assigned using the k-means clustering algorithm (MacQueen, 1967). Then basis functions are placed at each center. Initially these basis functions are given hyperspherical shapes. Thus each of the $H$ centers has associated with it a single width parameter $\sigma_h$. The unit widths are found using the following $P$-nearest neighbor heuristic. The width of any unit $(\sigma_h)$ is taken as the root mean square distance to the $P$ nearest centers:

$$\sigma_h = \sqrt{\frac{\sum_{p=1}^{P} \|m_h - m_p\|^2}{P}}$$  

(3.10)
The next step is to adapt the units to the local character of the data. The covariance of the training points “captured” by each unit is taken to get the $Q_h$ matrix for each basis function. The inverse of this matrix defines the unit's shape. To determine which training points are “captured” by a particular unit the following weight function is used.

$$w_h(x_k) = \exp(-\|x_k - m_h\|^2 / \sigma_h^2) \quad (3.11)$$

The weight for point $x_k$ in unit $h$ is $w_h(x_k)$. The weighting function is used instead of a hard or crisp boundary because such hard boundaries imply a threshold value that would have to be set arbitrarily. Using the continuous membership function, the points that are near the unit center have a significant influence on the shape of the unit, while points far away from the center have a small effect on the shape. Using all the data in the shape calculation, even though some of it is weighted very little, helps orient all of the basis functions in a smooth manner. Smoothness would not necessarily be achieved if hard boundaries around each unit were used to define the data that belongs to and does not belong to a particular unit.

The shape matrix $Q_h$ is calculated as a weighted covariance of the data points. The membership function used in calculating the weighted covariance matrix for unit $h$ is the activation of each training point, $w_h(x_k)$. To calculate the weighted covariance matrix for a particular unit $h$, the weighted mean is calculated for each unit and each dimension of the input space,
\[ w_{m_{h,j}} = \frac{\sum_{k=1}^{K} x_{k,j} a_h(x_k)}{\sum_{k=1}^{K} a_h(x_k)} \]  

(3.12)

where \( j \) goes from 1 to \( n \) (thus forming a vector \( w_{m_{h}} \)), \( x_{k} \) is the \( k \)th training example, and \( x_{k,j} \) is the \((k, j)\)th element in the \( X \) matrix, which is made up of the \( K \) row vectors \( x_{k} \).

The weighted mean vector is then subtracted from each of the training examples to form the centered data matrix \( C_{h} \),

\[
C_{h} = \begin{bmatrix}
    x_{1} - w_{m_{h}} \\
    x_{2} - w_{m_{h}} \\
    \vdots \\
    x_{K} - w_{m_{h}}
\end{bmatrix}
\]  

(3.13)

The weighted covariance matrix for a particular unit \( h \) is \( C_{h}^T \text{diag}(w_{h}) C_{h} / (K-1) \). This gives the desired local shape of the unit, but in order to get the overlap specified from the \( P \) heuristic, the covariance matrix for each unit \( h \) is re-scaled. The Mahalanobis distances from unit \( h \) to all other unit centers are calculated using \( C_{h}^T \text{diag}(w_{h}) C_{h} / (K-1) \) as the covariance matrix for unit \( h \). Then the mean square Mahalanobis distance (MSM) of the \( P \) nearest centers is used as the scaling factor:

\[
\text{MSM}_{h} = \sum_{p=1}^{P} \frac{(m_{h} - m_{p}) (C_{h}^T \text{diag}(w_{h}) C_{h} / (K-1))^{-1} (m_{h} - m_{p})}{P}
\]  

(3.14)

Introducing these scale factors to assure the desired overlap, the final form for the unit shape factors is:

\[
Q_{h} = \left( \frac{C_{h}^T \text{diag}(w_{h}) C_{h}}{K-1} \right)^{*} \text{MSM}_{h}
\]  

(3.15)

49
After the data in the input space has been covered by elliptical basis functions, the next task is to combine the basis functions to calculate the overall PDF. To do this, we first calculate a local density in the region covered by unit \( h \). The local density in any defined space is estimated by (Duda and Hart, 1973):

\[
\rho_h = \frac{n_h}{n_t V_h}
\]  

(3.16)

where \( n_h \) is the number of data points in the defined space, \( n_t \) is the total number of points, and \( V_h \) is the volume of the defined space. If the defined space has a hard boundary then it is easy to count every point inside the boundary once to get the number of points in the defined space. EBF estimators employ overlapping continuous basis functions that do not have hard boundaries. Thus each term in equation 3.16 requires re-definition consistent with the idea of a region whose boundaries are not crisply defined. We will state these definitions without justification, and then show that they lead to a valid density estimator. To obtain a count of the number of points associated with a particular unit \( h \) we use the activation of a point with respect to that unit as a weighting function. The total number of points associated with unit \( h \) is given by:

\[
n_h = \sum_{k=1}^{K} a_h(x_k)
\]  

(3.17)

This defines an effective number of points that are contained within unit \( h \). For the mutually-exclusive hard boundary case, the total number of points in the total space is equal to the sum of the number of points in each of the defined spaces. The number of points in a particular unit for the continuous case is defined as an effective number of
points. Thus, the total number of points in the space must also be defined in terms of an effective number of points. This is accomplished by summing the effective number of points from each of the individual units:

\[ n_t = \sum_{h=1}^{H} n_h \]  

(3.18)

The volume of a unit is taken as the integral of the activation function:

\[ V_h = \int_{-\infty}^{\infty} a_h(x) dx = \sqrt{\pi^n} \det(Q_h) \]  

(3.19)

Equations 3.16 to 3.19 give a local density estimate for basis function \( h \),

\[ \rho_h = \frac{\sum_{k=1}^{K} a_h(x_k)}{\left( \sum_{h=1}^{H} \sum_{k=1}^{K} a_h(x_k) \right)^{\frac{1}{2}} \sqrt{\pi^n} \det(Q_h)} \]  

(3.20)

In Parzen windows, the density estimate is made by an equally weighted average of the contributions of each of the basis functions. The Parzen window density estimate is:

\[ \rho(x) = \frac{1}{H} \sum_{h=1}^{H} \psi_h(x) \]  

(3.21)
The term $1/H$ can be considered the portion of the total probability mass given to each unit. In Parzen windows, the probability mass is equally divided among units. In the EBF estimator, units may represent different portions of the probability mass, since units are not identical. Intuitively, the probability mass associated with unit $h$ is given by the ratio of the effective number of points captured by the unit to the total effective number of points, $n_h/n_t$. This leads to the analogous form for non-identical units,

$$\rho(x) = \sum_{h=1}^{H} \left( \frac{n_h}{n_t} \right) \psi_h(x)$$  \hspace{1cm} (3.22)

By definition $\psi$ must be a valid density function that integrates to 1. In the case of the EBF $\psi$ is given by:

$$\psi_h(x) = \frac{a_h(x)}{V_h} = \frac{a_h(x)}{\int_{-\infty}^{x} a_h(x) \, dx}$$  \hspace{1cm} (3.23)

Using this definition of $\psi$ in equation 3.22 leads to the density estimate for the EBF estimator:

$$\rho(x) = \sum_{h=1}^{H} \left( \frac{n_h}{n_t} \right) \frac{a_h(x)}{V_h} = \sum_{h=1}^{H} \left( \frac{n_h}{n_t} \right) a_h(x) = \sum_{h=1}^{H} \rho_h a_h(x)$$  \hspace{1cm} (3.24)

To summarize, to construct an EBF estimator, first the number of units $H$, and the overlap parameter P are chosen. Optimal choice of these values is shown later in this chapter. The unit centers $m_h$, and the unit shapes $Q_h$ are then found using k-means clustering and equation 3.15 respectively. The density estimate for each unit, $\rho_h$, is then
calculated from equations 3.16 to 3.19. The density value for new points is obtained by using equations 3.24.

3.4 Properties of EBF Estimators

The definition of a probability density function requires that it be non-negative at all points and that it integrates to unity over the entire space. It is obvious from equation 3.24, and the definitions of \( a_h \) (equation 3.9) and \( \rho_h \) (equation 3.16), that the EBF density estimate is always non-negative. The integral of the EBF estimator comes from integrating equation 3.24

\[
\int \rho(x) dx = \int \sum_{h=1}^{H} a_h(x) \rho_h dx
\]

\[
= \int \sum_{h=1}^{H} \exp(-\left(x-m_h\right)Q^{-1}_{h} \left(x-m_h\right)^T) \frac{n_h/n_t}{V_h} dx
\]

\[
= \sum_{h=1}^{H} \left( \frac{n_h/n_t}{V_h} \int \exp(-\left(x-m_h\right)Q^{-1}_{h} \left(x-m_h\right)^T) dx \right)
\]

\[
= \sum_{h=1}^{H} \frac{n_h/n_t}{V_h} V_h
\]

\[
= 1
\]

and satisfies this criterion. Equality between 25c and 25d results from the definition of the volume of unit being the integral of its activation function over the entire space.

The most essential feature of a density estimator is that it converges to the true underlying distribution of the data. The proof we give here for the convergence of the EBF estimator to the underlying distribution function, follows directly from the proof of Parzen window estimators converging to the underlying density function, in the limit of
infinite data, first given by Parzen (1962) and then by Duda and Hart (1973). In Parzen windows, letting the number of data points go to infinity also implies that the number of basis functions goes to infinity. In EBFs there are fewer basis functions than the number of data points. For convergence, we must allow the number of basis functions, \( H \), to go to infinity but at a slower rate than the number of data points \( K \). We also let \( P \) tend to infinity but at a slower rate than \( H \). That is,

\[
K \to \infty, \ H \to \infty, \ P \to \infty, \ H \ll K, \text{ and } P \ll H
\]  

(3.26)

To prove convergence, consider a differential area of the input space in the case where we have infinite data. By proving that the EBF estimator converges to the underlying distribution in an arbitrary differential area, it can be directly concluded that the estimator will converge to the underlying distribution over the entire space. In this differential element there are \( K' \) data points and \( H' \) units, that satisfy the \( K \) and \( H \) relationship in 3.26. If the differential element is small enough then the data will be uniformly distributed over the element. The k-means clustering algorithm that sets the unit centers will distribute the centers uniformly over the element because the data is uniform (\( H' \) centers are chosen randomly from the \( K' \) data points in the first step of the k-means algorithm, as the k-means algorithm starts moving the unit centers, the center is pulled equally in all directions because of the uniform distribution of the data in the differential element, resulting in no net movement of the center).

The shape matrix of the EBF becomes diagonal in the infinite data limit because there is no correlation among the input variables in the differential element. Furthermore, each diagonal element of the covariance matrix is the same, because the
data is distributed uniformly in all directions. This means that the EBF units are hyperspherical.

Each unit has the same activation function and therefore the sum of the activations in Eq. (17), \( n_h \), will be the same for each unit in the differential element. Thus in the limit of infinite data the ratio of \( n_h / n_t \) becomes

\[
\frac{n_h}{n_t} = \frac{\sum_{h=1}^{H} n_h}{H n_h} = \frac{1}{H} \tag{3.27}
\]

Using equation 3.9 and equation 3.16 to write equation 3.24 the result is

\[
\rho(y) = \frac{1}{n_t} \sum_{h=1}^{H} \frac{n_h}{V_h} \exp(- (y - m_h) Q_h^{-1} (y - m_h)^T) \tag{3.28}
\]

applying the simplifications that have been shown, in the limit of infinite data for a differential element in the input space we have

\[
\rho(y) = \frac{1}{H} \sum_{h=1}^{H} \frac{1}{V_h} \exp(- (y - m_h) (\sigma^2 I)^{-1} (y - m_h)^T) \tag{3.29}
\]

which is exactly a Parzen window estimator for the unit centers \( m_h \). In the differential area the data is uniformly distributed and the centers are also uniformly distributed, thus converging to the distribution of the unit centers is the same as converging to the distribution of the data points. This means that in the differential area, the EBF density estimator will converge to the underlying distribution. The same result holds for any differential element in the space. Since the EBF estimator converges to the underlying
distribution function in every differential element, then it can be concluded that it converges to the underlying distribution in the entire space, in the limit of infinite data.

3.5 Model Selection and Optimum Model Structure

When one (as always) has a finite data set, an optimal estimator with particular finite \( H \) and \( P \) will exist. The choice of \( H \) and \( P \) or choice of the Parzen window width should be based on a measure of future performance. The purpose of these density estimators is to predict the density of a new, previously unseen data point. The measure of how well a particular estimator approximates a density function is the product of the probabilities of sample points independently and randomly drawn from the true underlying distribution (Traven, 1991). For a test set drawn from the true underlying distribution of size \( n_{\text{test}} \), the overall fit measure is given by:

\[
J' = \prod_{i=1}^{n_{\text{test}}} \rho(x_i)
\]

(3.30)

For convenience of calculation, usually the log of equation 3.30 is used,

\[
J = \log(J') = \sum_{i=1}^{n_{\text{test}}} \log(\rho(x_i))
\]

(3.31)

The parameters of a particular method that maximizes the log probability are the ones most suited to the particular data density estimation task.

The difficulty in using the log probability is that an independent test set of data is required. Data is limited in most applications and it is desirable calculate the estimator using all available data. The well-known technique of cross-validation (Weiss
and Kulikowski, 1991) can be used to fully utilize the data set while still providing test sets which were not used to calculate the estimator. The s-fold cross validation procedure, where s is the number of test sets, is as follows:

1) Divide the training data into s equally-sized (or almost equally-sized if s does not divide evenly into the number of training points) randomly chosen sets.

2) Fix parameters for the density estimation technique (e.g. H and P)

3) For i = 1 to s, repeat
   a) Withhold set i
   b) Train estimator on all training data except set i
   c) Use equation 3.31 to calculate log probability of set i

4) Sum up the values of J calculated for each subset in step 3. This is the total log probability for the parameters chosen in step 2.

This procedure can be repeated for various estimator parameters. The parameters that maximize the log probability are the best parameters for the density estimation task at hand. Once the parameters have been decided upon then the entire training data set is used to train an estimator with those optimal parameters. Determination of the log probability allows for optimization of the parameters of Parzen window and EBF techniques, it also is a valid criterion for selecting one method of density estimation over another. After parameter optimization, the method that has the highest log probability is the most applicable method for approximating the underlying density function of the data. Cross validation can also be used for estimators whose parameters are found deterministically (such as MVG) to find a comparable log probability. This
can then be compared to other log probabilities to select the best method. For the MVG method, step 2 of the cross validation algorithm can be omitted.

For the purposes of validating the cross validation method, if the underlying distribution is known, we can normalize the log probability calculated by cross validation by the number of training points, and compare it to the normalized log probability of a new test set drawn from the parent distribution. If the cross validation estimate is valid, these two numbers should be close to one another. This is demonstrated in the next section.

3.6 Examples

Three abstract examples of using the EBF technique for estimating probability density functions were investigated. Case 1 is a two-dimensional Gaussian distribution, centered at the origin with an identity covariance matrix, \( N(0, I) \). Case 2 is a two-dimensional bimodal distribution. Case 2 data has a 0.55 probability of being drawn from a \( N(0, I) \) distribution and a 0.45 chance that the data is drawn from a Gaussian distribution centered at \((4, 2.5)\) with a covariance matrix of \( \text{diag}(1.5, 0.2) \). Case 3 data is seven-dimensional. It has the same distribution as case 2 in the first two dimensions, and has five other dimensions, each of which has an independent Gaussian distribution of data centered at the origin with a variance of 0.07.

Although all of the examples have diagonal covariance, they are equivalent to more general problems. In the bimodal case, translations, rotations, and linear scaling can be used to move one of the distributions to the origin and make it spherical. The distributions can then be rotated about the origin until the other distribution has its axes aligned with the principal axes, having a diagonal covariance matrix (Van Ness, 1980).
Therefore, this problem represents two Gaussian distributions with arbitrary centers and arbitrary covariance matrices.

For each case, a training set of 100 points was generated, as well as a separate test set of 1000 points. The training set was used in the cross validation methodology to choose optimal parameters and estimate log probabilities for multivariate Gaussian, Parzen windows, and EBF density estimators. Density estimates for the 1000 point test sets were then made for each of the estimators and a log probability calculated. Table 3.1 shows that the log probabilities estimated by the cross validation procedure are good estimates of the actual performance of the estimator. The test set log probability is within 12% of the cross validation log probability, for all estimators except the Parzen window estimator of case 3 data. This estimate is an order of magnitude different than the test set log probability. The order of magnitude difference in the log probability however is only a 25% absolute difference. The absolute (non-logged) cross validation and test set probabilities differ by 3% to 37% for the cases studied.

Figures 3.2 through 3.4 show mesh plots of the actual density function and the estimates generated by the three methods under consideration. In each plot, the two horizontal axes are the two independent variables and the vertical axis is the density value. For case 3 the two dimensions of interest are shown and the other five dimensions are held at their nominal value of 0. The $J'$ value appearing is the normalized absolute probability (not logged) from the 1000 point test set.

The best predictor in case 1 was the multivariate Gaussian estimator, as expected. The Parzen window and EBF estimators give adequate estimates, but are poorer than the multivariate Gaussian. For the bimodal (case 2) data the multivariate Gaussian is unable to capture the bimodal feature and thus is a poor estimate of the true density function. The Parzen window estimator captures the bimodal nature of the data
Table 3.1: Normalized specific log probabilities from 6-fold cross validation procedure (V) and from the 1000 point test sets. Multivariate Gaussian (MVG), Parzen Windows (PW), and Elliptical Basis Functions (EBF).

<table>
<thead>
<tr>
<th>Case</th>
<th></th>
<th>MVG</th>
<th>PW</th>
<th>EBF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV test</td>
<td>-1.214</td>
<td>-1.249</td>
<td>-1.257</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>-1.253</td>
<td>-1.261</td>
<td>-1.278</td>
</tr>
<tr>
<td></td>
<td>CV test</td>
<td>-1.639</td>
<td>-1.326</td>
<td>-1.299</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>-1.604</td>
<td>-1.464</td>
<td>-1.397</td>
</tr>
<tr>
<td></td>
<td>CV test</td>
<td>1.116</td>
<td>-0.1468</td>
<td>1.111</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0.995</td>
<td>-0.0162</td>
<td>1.051</td>
</tr>
</tbody>
</table>
Figure 3.2: Mesh plots for a) actual density function, b) multivariate Gaussian estimator, c) Parzen Window estimator, and d) elliptical basis function estimator for case 1. The Parzen Window width was 0.5419, and the EBF had H=15 units with an overlap of P=1. The Parzen window width and EBF parameters were the optimal parameters for the data set for this case. The $J'$ values shown are the product of the densities for the 1000 point test case, normalized by the number of test cases. The largest value, indicating the best estimator, is shown in bold.
Figure 3.3: Mesh plots for a) actual density function, b) multivariate Gaussian estimator, c) Parzen Window estimator, and d) elliptical basis function estimator for case 2. The Parzen Window width was 0.3420, and the EBF had H=12 units with an overlap of P=1. The Parzen window width and EBF parameters were the optimal parameters for the data set for this case. The $J$ values shown are the product of the densities for the 1000 point test case, normalized by the number of test cases. The largest value, indicating the best estimator, is shown in **bold**.
Figure 3.4: Mesh plots for a) actual density function, b) multivariate Gaussian estimator, c) Parzen Window estimator, and d) elliptical basis function estimator for case 3. The Parzen Window width was 0.1503, and the EBF had H=8 units with an overlap of P=1. The Parzen window width and EBF parameters were the optimal parameters for the data set for this case. The $J'$ values shown are the product of the densities for the 1000 point test case, normalized by the number of test cases. The largest value, indicating the best estimator, is shown in bold.
but produces a density estimate that is not smooth enough. The optimal value of \( \sigma = 0.342 \) was obtained by optimizing the predictive capability of the estimator.

Increasing sigma, which would have the affect of smoothing the density estimate, would not improve the estimator’s predictive ability. The best estimate of the case 2 data comes from the EBF estimator. The bimodal nature of the data is captured and the density estimate is relatively smooth. The case 3 density estimates are similar to the case 2 estimates. The multivariate Gaussian is unable to capture the bimodal nature of the data in the two dimensions shown. In the other five dimensions, which are Gaussian, the multivariate Gaussian fits these very well. The resulting probability score is \( J' = 9.89 \). The added dimensionality shows the shortcoming of Parzen window estimators. There is only one unit width and the close packing of the points in the five dimensions not shown cause small unit sizes (\( \sigma = 0.1503 \), resulting in a very spiky density estimate and a correspondingly low \( J' \) value of 0.963. The EBF density estimate captures the bimodal nature of the data in a smooth fashion in the two dimensions shown. The variable width of the elliptical units in each dimension allows for small widths in the five remaining dimensions. This eliminates the spikiness problem exhibited by Parzen windows.

Using cross validation to decide on the best method of density estimation for a given set of data has been shown to work on three sets of test data. The elliptical basis function estimation has also been shown to overcome some of the problems of more conventional density estimation techniques such as multivariate Gaussian and Parzen window estimators.
3.7 Classification Examples

Elliptical basis function estimators can be used to make maximum likelihood decisions about the classification of a data point. This ability is useful in fault detection and fault diagnosis. To illustrate the application of EBF estimators, the problem first presented by Kramer and Leonard (1990) is used. This problem is a simplified version of many static fault diagnosis problems. The measured states are represented by vector $Y$, and the process is at a nominal steady state $Y_0$. Malfunctions are represented by changes in fault parameters $q$. These parameters are the extent of the fault, and are zero at the normal operating condition. The measurement $Y$ is given by

$$Y = Y_0 + f(q) + v$$

(3.32)

where $v$ represents measurement noise. In this example the extent of the fault has a linear effect on $Y$:

$$Y = Y_0 + \alpha q + v.$$  

(3.33)

Each column of the matrix $\alpha$ represents the directional effect of one of the fault parameters on the measurement vector. Fault categories are then defined by inequalities on the fault parameters:

$$C_k: g_k(q) > 0, \quad k=1, \ldots, nc$$

(3.34)
This formulation of the failure model is quite general and conceptually fits the Hoskins and Himmelblau (1988) model of three CSTRs in series, the reactor example of Watanabe et al (1989), and other classification problems.

In the work of Kramer and Leonard (1990) and Holcomb and Morari (1991), a two-dimensional form of the problem posed in equation 3.33, was studied. In both of these works, cases are limited to only one element of q being non zero at a time. The noise v is taken to be zero mean Gaussian, and the fault direction matrix α is taken as:

\[ \alpha = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \]

Normal (C₀):

\[ |q_1| < 0.05, \quad |q_2| < 0.05 \]

Fault 1 (C₁):

\[ |q_1| \geq 0.05 \]

Fault 2 (C₂):

\[ |q_2| \geq 0.05 \]

\[ v_1, v_2 = N(0, 0.015) \]

Fault 1 is when \( Y_1 \) and \( Y_2 \) both fail in the same direction, and fault 2 is when \( Y_1 \) and \( Y_2 \) fail in opposite directions. This results in the "X" shape, with the normal class at the intersection of the two fault classes, shown in figure 3.5. Training and test data were generated by sampling the fault extent \( q_i \) from a normal distribution, \( N(0, 0.25) \). If the absolute value of the fault extent was less than 0.05 then the point was assigned to the normal class.

Training sets consisting of 30 examples per class were generated, as well as test sets of 1000 examples per class. Elliptical basis function density estimators and Parzen window density estimators were fit to the data from each class. The EBF H and P parameters and the optimal window width for the Parzen window estimator were decided upon using the cross validation technique previously described in this chapter.
Figure 3.5: Typical set of training data for two dimensional classification problem.
For each type of estimator, the class that the point had the highest density in was assigned as the class for that point. This is a maximum likelihood decision, assuming equal class priors. To make a statistically meaningful comparison between methods, the comparison was repeated 500 times and average results were calculated. Table 3.2 shows the EBF and Parzen window results for 500 independently chosen pairs of training data and test data. The mean correct identification rate is given as well as the standard deviation of correct identification for each class. The overall score is the total percentage of correctly identified cases. The table shows that the EBF slightly out performs the Parzen window estimator, 93.2% versus 91.6%. It is important to note that the EBF had about one third the number of basis functions of the Parzen windows. The optimal EBF had from 9 to 11 basis functions and the Parzen window estimator had in each case 30 basis functions. It is also not surprising that the Parzen window estimator does well as the data in this problem have about the same variance in each direction and the data is quite evenly spread over the entire range of the sensors.

The classification problem formulated in equation 3.33 is quite general and applies to any number of dimensions. Kramer and Leonard (1990) present a 10 dimension classification problem that has an $\alpha$ matrix of:
The same testing procedure for the two dimension problem was used for the ten
dimension problem, although only 100 train-and-test set pairs were used to find the
average performance, as opposed to the 500 train-and-test set pairs used for the two
dimension problem. For the ten dimension problem, 90 examples per class were used
for training. The overall score for the EBF was 89.4% correctly identified classes, with
a standard deviation of 2.9%. The Parzen window estimator performed almost as well,
with an overall score of 85.3%, and a standard deviation of 3.7%. In this problem, the
Parzen window estimator had about six times as many basis functions as the EBF. The
Table 3.2: Performance of EBF and Parzen window (PW) density estimators on the two dimensional classification problem. Results are for 500 independently chosen sets of 30 training points per class, and 1000 test points per class. Results given in percent.

**EBF**

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Fault 1</th>
<th>Fault 2</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>94.1</td>
<td>93.1</td>
<td>92.4</td>
<td>93.2</td>
</tr>
<tr>
<td>standard deviation</td>
<td>2.9</td>
<td>4.7</td>
<td>4.3</td>
<td>2.2</td>
</tr>
</tbody>
</table>

**PW**

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Fault 1</th>
<th>Fault 2</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>95.3</td>
<td>90.2</td>
<td>89.2</td>
<td>91.6</td>
</tr>
<tr>
<td>standard deviation</td>
<td>2.4</td>
<td>5.2</td>
<td>4.6</td>
<td>2.8</td>
</tr>
</tbody>
</table>
Parzen window estimator had 90 basis functions and the EBF estimator had from 13 to 16 basis functions.

The heat exchanger network (HEN) used by Tjoa and Biegler (1991) with modifications was used as a process example of fault classification using the density estimation technique. The HEN consists of 4 input streams and 4 heat exchangers. Fluid A is heated by fluid B in two of the exchangers and by fluids C and D in the other exchangers (see Figure 3.6). A controller manipulates the flow of stream D to maintain the temperature of the outlet stream at 615°C. Although a controller is used, only steady state measurements were used. Streams A and B were assumed to be process streams that operate at low, medium, and high flowrates, with Gaussian process noise about each operating point. The input flows and temperatures were drawn from distributions described in table 3.2. The flow split between FA3 and FA6 was specified as FA3 = 0.42*FA2.

The heat transfer for each heat exchanger was calculated using

\[ q = UA \frac{[(T_{hi} - T_{co}) - (T_{ho} - T_{ci})]}{\ln \left( \frac{T_{hi} - T_{co}}{T_{ho} - T_{ci}} \right)} \] (3.35)

where \( q \) is the heat transferred, \( U \) is the overall heat transfer coefficient, \( A \) is the heat exchanger area, \( T \) is temperature, and subscript \( h \) refers to the hot stream, \( c \) to the cold stream, \( i \) to the inlet, and \( o \) to the outlet. A heat capacity of unity for all fluids was used. The value of \( UA \) for each exchanger were calculated from Tjoa and Beigler's results and were 400, 265, 492, and 1158, for exchangers 1 to 4 respectively. Equation 3.35 for each exchanger was used in conjunction with the steady state mass and energy balances to calculate all flows and temperatures given the input conditions. The flowrate of stream D was set by the controller such that temperature TA8 is maintained.
Figure 3.6  Heat exchanger network example of sensor fault identification. In all exchangers the A stream is the cold stream.
Table 3.3: Input distributions of flows and temperatures for the heat exchanger network.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC1</td>
<td>N(240,2)</td>
</tr>
<tr>
<td>TA1</td>
<td>N(450,1)</td>
</tr>
<tr>
<td>TB1</td>
<td>N(620,1)</td>
</tr>
<tr>
<td>TC1</td>
<td>N(670,1)</td>
</tr>
<tr>
<td>TD1</td>
<td>N(690,1)</td>
</tr>
<tr>
<td>TA8</td>
<td>N(615,1)</td>
</tr>
<tr>
<td>FA3/FA2</td>
<td>0.42</td>
</tr>
<tr>
<td>Probability</td>
<td>0.3</td>
</tr>
<tr>
<td>FA1</td>
<td>N(800,2)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(200,2)</td>
</tr>
<tr>
<td>Probability</td>
<td>0.4</td>
</tr>
<tr>
<td>FA1</td>
<td>N(900,2)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(250,2)</td>
</tr>
<tr>
<td>Probability</td>
<td>0.3</td>
</tr>
<tr>
<td>FA1</td>
<td>N(990,2)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(320,2)</td>
</tr>
</tbody>
</table>
at 615 C. Sensor noise was added to each of the measurements using the same method of Tjoa and Biegler. All temperatures measurements had zero-centered Gaussian noise with a standard deviation of 0.75 C. All flow measurements had zero-centered Gaussian noise with a standard deviation equal to 2% of the flow.

Before doing density estimation on the data set, the dimensionality was reduced from 13 to 2 using principal component analysis. It was found that 95% of the variation in the data set could be captured by 2 principal components. Density estimation on the 2 dimensional data was done using a 200 point data set with multivariate Gaussian, Parzen window, and elliptical basis function estimators. Six-fold cross-validation was used to find the predicted log probability and a 1000 point separate test set was used to find an application log probability. The EBF estimator with 22 units and overlap of 1 was the best at capturing the underlying distribution of the data, having a predicted J’ value of 0.0471. This was larger than both the Parzen window (optimal σ = 0.987) value of 0.0241 and the multivariate Gaussian value of 0.0173. The test set J’ values were 0.0467, 0.0276, and 0.0172 for EBF, Parzen window, and multivariate Gaussian estimators respectively.

Data sets were calculated for positive and negative biases of 2 and 4 degrees in the outlet temperature sensor TA8, using the same method and inputs previously described. EBF, Parzen window and multivariate Gaussian density estimators were fit to these data sets after principal component analysis was used to reduce the dimensionality to two. Separate test sets of 1000 points each were also calculated for the positive and negative biases of 2 and 4 degrees in TA8. The bias in TA8 effects the flow of stream D selected by the controller.

The following procedure was used for the EBF, Parzen window, and multivariate Gaussian estimators. The density of each point in the test set was
calculated using the estimators for positive and negative biases of 2 and 4 degrees in TA8, and using the normal state. The point was assigned membership in the class of the estimator with the highest density. The results are shown in classification matrices shown in table 3.4. The training and test points both contained zero-centered Gaussian noise with a standard deviation of 0.75 C. The class boundaries used in table 3.4 are 2 C (only 2.67 standard deviations), which leads to some misclassification of the faults. Despite this, the EBF estimator was able to identify the correct class over 90% of the time, in all but one case. The diagonal sum of the confusion matrices represents the total number of correctly identified cases. The EBF estimator had the highest diagonal sum of 465.3 out of a possible total of 500. The Parzen window estimator was second best, having a diagonal sum of 441.7, and the multivariate Gaussian was the poorest having a diagonal sum of 309.1. The relative performance of the estimators in the class identification problem is the same as the estimators’ relative performance in the density estimation task, as measured by the predicted log probability. This confirms the probability as a figure of merit for predicting performance of density estimators. The EBF outperformed the Parzen window estimator even though the Parzen window estimator used 200 basis functions compared to the EBF estimator’s 22. Because the EBF estimator has fewer basis functions calculations with the EBF are an order of magnitude faster than the Parzen window estimator.

3.8 Conclusions

A new technique of estimating data density functions has been introduced. The elliptical basis function density estimator has been shown to overcome the restrictions of parametric density estimators, while reducing the number of units and “spikiness” problem of Parzen Window estimators. A methodology for selecting the best density
Table 3.4: Fault classification tables for sensor bias classification in a heat exchanger network for a) EBF, b) multivariate Gaussian, and c) Parzen window density estimators.

### a) EBF

<table>
<thead>
<tr>
<th>BIAS IDENTIFIED</th>
<th>BIAS PRESENT</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-4</td>
<td>-2</td>
<td>0</td>
<td>+2</td>
<td>+4</td>
</tr>
<tr>
<td>-4</td>
<td>96.2</td>
<td>1.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>3.7</td>
<td>94.4</td>
<td>2.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>4.1</td>
<td>93.8</td>
<td>2.6</td>
<td>0</td>
</tr>
<tr>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>3.5</td>
<td>95.9</td>
<td>15.0</td>
</tr>
<tr>
<td>+4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.5</td>
<td>85.0</td>
</tr>
</tbody>
</table>

### b) Multivariate Gaussian

<table>
<thead>
<tr>
<th>BIAS IDENTIFIED</th>
<th>BIAS PRESENT</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-4</td>
<td>-2</td>
<td>0</td>
<td>+2</td>
<td>+4</td>
</tr>
<tr>
<td>-4</td>
<td>65.5</td>
<td>17.6</td>
<td>4.2</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>32.2</td>
<td>47.4</td>
<td>10.0</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2.2</td>
<td>32.4</td>
<td>56.1</td>
<td>11.8</td>
<td>1.0</td>
</tr>
<tr>
<td>+2</td>
<td>0.1</td>
<td>2.5</td>
<td>28.2</td>
<td>60.6</td>
<td>19.5</td>
</tr>
<tr>
<td>+4</td>
<td>0</td>
<td>0.1</td>
<td>1.5</td>
<td>26.8</td>
<td>79.5</td>
</tr>
</tbody>
</table>

### c) Parzen Window Density Estimators

<table>
<thead>
<tr>
<th>BIAS IDENTIFIED</th>
<th>BIAS PRESENT</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-4</td>
<td>-2</td>
<td>0</td>
<td>+2</td>
<td>+4</td>
</tr>
<tr>
<td>-4</td>
<td>91.0</td>
<td>6.7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-2</td>
<td>9.0</td>
<td>79.2</td>
<td>3.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>14.1</td>
<td>92.2</td>
<td>5.7</td>
<td>0</td>
</tr>
<tr>
<td>+2</td>
<td>0</td>
<td>0</td>
<td>4.7</td>
<td>86.2</td>
<td>6.9</td>
</tr>
<tr>
<td>+4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.1</td>
<td>93.1</td>
</tr>
</tbody>
</table>

76
estimator for a particular application, using cross validation and the log probability, has also been presented. The cross validation methodology and the EBF estimator was demonstrated on three density estimation test problems. The results indicate that both the EBF estimator and the cross validation methodology are good techniques for data density estimation and model discrimination for multivariate correlated data. The EBF estimator was also shown to be the best estimator at fault classification for two abstract fault classification problems previously studied in the literature, as well as being the best classifier for detecting sensor biases in a heat exchanger network. The EBFs outperformed the Parzen window estimator in all classification examples, even though the EBF method used up to an order of magnitude fewer basis functions. The lower computational load of the EBF estimator makes it more practical to use in on-line where speed of calculation may be critical.
Chapter 4:  
Maximum Likelihood Data Rectification: 
Steady State Systems

A maximum likelihood rectification (MLR) technique that poses the data 
rectification problem in a probabilistic framework and maximizes the probability of the 
estimated plant states given the measurements is developed in this chapter. The 
proposed approach does not divide the sensors into “normal” and “gross error” classes 
but instead uses all of the data in the rectification, each sensor being appropriately 
weighted according to the laws of probability. In this manner, the conventional 
assumption of no sensor bias is avoided, and both random errors (noise) and systematic 
errors (gross errors) are removed simultaneously. A novel technique utilizing historical 
plant data to determine a prior probability distribution of the plant states is also 
introduced in this chapter, and fully developed in Chapter 5. This type of historical 
plant information, which contains the physical relationships between the variables 
(mass balances, energy balances, thermodynamic constraints), as well as statistical 
correlations between the variables, has been ignored in prior data rectification schemes. 
The proposed approach can use the historical plant information to solve a new class of 
data rectification problems in which there are no known model constraints. The MLR 
method is demonstrated on data from a simulated flow network and a simulated heat 
exchanger network. The MLR technique provides considerably improved performance 
over existing data reconciliation schemes in these examples.
4.1 Introduction

All chemical processing plants make measurements of a wide variety of process variables, both on and off-line. These measurements are used to make decisions that affect product quality, plant safety, and plant profitability. Inaccurate process data can lead to poor decisions that can adversely affect many plant functions. Measurement instruments are subject to random errors (noise), and non-random errors, such as sensor bias or sensor failure (gross errors). The objective of data rectification is to obtain an estimate of the true state of the plant by removing both the random and non-random errors from the data set. The problem of data rectification is posed in this work as a maximum likelihood problem, where the probability of the estimated plant state is maximized given the measurement set.

Although the terms are often used indiscriminately, data rectification and data reconciliation are distinct. Data rectification means literally "to make data right", while data reconciliation refers to adjusting data to conform to prior constraints (Kramer and Mah, 1993). Most previous work in the area of data rectification addresses data reconciliation, and this work is reviewed in Chapter 2. Since making the data conform to constraints will not in itself necessarily correct noisy or biased data, one can view data reconciliation as a special case, or a subset, of the broader problem of data rectification. The approach developed in this chapter sharpens the distinction between rectification and reconciliation using the framework of maximum likelihood estimation, and shows that, under certain assumptions of the prior knowledge, the rectification problem reduces to the classical reconciliation problem.
4.2 Approach

The proposed approach poses the data rectification problem in a probabilistic framework and uses the laws of probability and maximum likelihood estimation to determine the most likely rectified state. The probability distribution of the process states \( (x) \), given the process measurements \( (y) \) is maximized to find the rectified states. The rectification problem is posed as:

\[
\max_x P(x|y)
\]

By Bayes' theorem, the probability of the states given the measurements can be written in terms of the probability of the measurements given the states, the prior probability density of the states, and the probability density of the measurements:

\[
\max_x P(x|y) = \max_x P(y|x)P(x)/P(y)
\]

The denominator term \( P(y) \), independent of \( x \), acts as a normalizing constant (for any given set of measurements) and does not have to be considered further.

The first term of the numerator of equation 4.2 represents the probability density of the measurements given the states, which is the distribution of the measurement error. Instead of assuming this is given by a normal distribution, we obtain this probability distribution by representing each sensor and its operating modes explicitly. Given an exhaustive set of mutually exclusive modes of sensor operation \( m_j \), a measurement model for sensor \( i \) can be expressed by the weighted sum over its modes:

\[
P(y_i|x_i) = \sum_j P(y_i|x_i,m_j) P(m_j)
\]
Modes that could be considered include but are not limited to: normal, biased, and failed. Summing the probability distributions of several modes broadens the basic noise distribution and increases the probability of large errors, relative to the normal mode alone. If sensor errors are independent, the product of this probability over all sensors yields the first term of the rectification objective function:

$$P(y|x) = \prod_{i} P(y_i|x_i)$$  \hspace{1cm} (4.4)

The other term in equation 4.2, $P(x)$, is a probability density function (PDF) representing the prior joint distribution of the states. This distribution gives the relative merit of each possible solution to the rectification problem in the absence of measurement data. All previously developed reconciliation schemes carry a critical but implicit assumption about $P(x)$, specifically that $P(x)$ is a binary variable: that $P(x) = 1$ if and only if $x$ satisfies the model equations (equality and inequality constraints), and otherwise $P(x) = 0$ (if $x$ fails to satisfy the model constraints). Under this assumption, the $P(x)$ term is converted to a set of constraints, and equation 4.2 is converted to the following constrained optimization:

$$\max_x P(y|x)$$
$$s.t. \ h(x) = 0$$
$$\quad g(x) \leq 0$$  \hspace{1cm} (4.5)

Tjoa and Biegler (1991) modeled $P(y|x)$ as a bivariate Gaussian and solved equation 4.5 using sequential quadratic programming. However, all states that satisfy the constraints $g(x) \leq 0$, $h(x) = 0$ are not necessarily equally likely, because there may be
preferred operating regions within the reduced space described by the constraints. Approximating $P(x)$ by a binary variable may introduce unnecessary error into the rectification calculation.

If, in addition to the binary assumption on $P(x)$, sensor errors are assumed to follow a normal distribution, taking the logarithm of the objective function in equation 4.5 results in the conventional weighted least squares formulation of the steady-state rectification problem:

$$
\min_x (y-x)^T Q^{-1} (y-x)
\text{s.t. } h(x) = 0 \quad g(x) \leq 0
$$

(4.6)

For linear equality constraints $Ax = 0$ and in the absence of inequality constraints, this problem has a well-known, closed-form solution (Mah, 1987):

$$
\hat{x} = (I - Q A^T (AQAT)^{-1} A)y
$$

(4.7)

Therefore it can be seen that the two key assumptions needed to derive the typical least squares formulation from the maximum likelihood problem are: (1) the distribution of sensor errors are zero-mean and Gaussian; and (2) the prior probability distribution of the states can be reduced to a set of constraints. To overcome limitations of the first assumption, we model the probability distribution of measurement errors by equation 4.3, which considers multiple modes of sensor operation. This results in a model of $P(y|x)$ that is typically not normal, even when the random noise contribution is normal. In prior work, the effect of the second assumption has not been clearly explored. To relax the assumption about $P(x)$, we model the probability distribution $P(x)$ using a
combination of constraints and a probability distribution of the states in the subspace of the constraints.

To illustrate the affect of assuming \( P(x) \) is a binary variable, consider a mixing tee where flow 1 and flow 2 are mixed together, forming flow 3. The mass balance constraint for this problem is \( F_1 + F_2 = F_3 \). This defines a plane in the measurement space, as shown in figure 4.1. The probability of points off the plane is zero. However, not all of the points on the plane are equally likely, and some may be infeasible (e.g. negative flows if using positive displacement pumps). If a history of the actual flows were plotted in the measurement space, there would be regions on the plane in which the actual flows would tend to lie, schematically represented by the probability contours in figure 4.1. The separate regions correspond to different modes of plant production. The different likelihood of states in the space of constraints (in this case the plane) is additional information that can be exploited to obtain a better estimate of the true plant state.

Taking into account the probability function of the sensor error modes, given by equations 4.3 and 4.4, and the model of \( P(x) \), the Maximum Likelihood Rectification (MLR) method solves the following problem to obtain the most likely rectified states, given the measurements:

\[
\begin{align*}
\max_x P(x) P(y|x) \\
\text{s.t. } h(x) &= 0 \\
g(x) &\leq 0
\end{align*}
\]  

In the case where there is no prior knowledge of \( P(x) \), equation 4.8 reduces to equation 4.5, where only the probability of the adjustments to the measurements \( P(y|x) \) is maximized subject to the constraints. In this chapter the solution of the more general
Figure 4.1: The rectification subspace for a mixing tee where $F_1 + F_2 = F_3$. Historical data from the process falls in the shaded regions, implying that not all points on the rectification surface are equally likely.
problem in equation 4.8 is developed. Solution of equation 4.8 results directly in the plant states, without the requirement for a separate gross error detection step.

One problem immediately encountered in solving equation 4.8 is the modeling the prior distribution, \( P(x) \). We assume that this distribution is not known in advance, but must be calibrated from historical data. It cannot be derived directly from measurements because the measurements contain errors, which corrupt the distribution of \( P(x) \). Our technique uses Recursive State Density Estimation to find the distribution of \( P(x) \) indirectly from a calibration measurement set, as described in Chapter 5. Furthermore, the formulation takes into account the statistical tendencies of the process that are represented by the term \( P(x) \). This information, ignored in other rectification methods, can be exploited to obtain improved estimates of the plant state, as the following simple example shows:

*Example 1.* Two identical thermocouples measure the temperature of a stream. The sensors normally agree within a degree or two, but the current readings are 20 C and 50 C, respectively. Traditional gross error detection techniques, given the constraint \( T_1 = T_2 \), can detect that a sensor failure has occurred, but cannot determine which sensor is at fault. Therefore, data cannot be rectified with traditional techniques. However, historical data from the plant has shown that the temperature of the stream follows a Gaussian distribution with a mean of 50 C and a standard deviation of 10 C. Knowing this, an operator would rationally conclude that it is most likely that the 20 C measurement is erroneous, and the actual temperature of the stream is 50 C. Assume that both sensors have Gaussian noise with a
standard deviation of 1°C when working normally, and the probability of either sensor failing is 2% in which case the sensor reports a random value in the range of 0 to 100°C (i.e. is modeled as a uniform distribution between 0 and 100). Then the problem can be formally modeled as (with \(y_i\) representing the \(i\)th temperature measurement and \(x_i\) representing the true temperature state):

\[
P(y_i|x_i, \text{normal}_i) = N(x_i, 1) \text{ evaluated at } y_i
\]

\[
\equiv P_N(y_i - x_i, 1)
\]

\[
P(y_i|x_i, \text{failed}_i) = 0 \text{ if } y_i < 0 \text{ or } y_i > 100
\]

\[
= 0.01 \text{ if } 0 \leq y_i \leq 100
\]

\[
P(\text{normal}_i) = 0.98, \ P(\text{failed}_i) = 0.02
\]

Thus,

\[
P(y_i|x_i) = 0.98P_N(y_i - x_i, 1) + 0.02*0.01
\]

and,

\[
P(x_i) = N(50, 10) \text{ evaluated at } x_i
\]

\[
\equiv P_N(50 - x_i, 10)
\]

The MLR formulation is:

\[
\max_{x_1, x_2} \ P_N(50 - x, 10)*[(.98*P_N(50 - x_1, 1)+0.0002)*(.98*P_N(20 - x_2, 1)+0.0002)]
\]

\[
\text{s.t. } \ x_1 = x_2 \equiv x
\]

same as:

\[
\max_x \ .9604*P_N(50 - x, 10)*P_N(50 - x, 1)*P_N(20 - x, 1)
\]

\[
+ 1.96e-4*P_N(50 - x, 1)*P_N(50 - x, 10)
\]

86
\[ + 1.96e-4 \cdot P_N(20 - x, 1) \cdot P_N(50 - x, 10) \]

For \( x = 20 \) C, all three terms in the above equation are very small, the largest being the third term which is \( 4.2e-92 \). For \( x = 50 \) C, the first and last terms are vanishingly small (<\( 1e-197 \)), and the second term is relatively large, \( 3.1e-6 \). Thus the solution to the maximization will be very close to \( x_1 = x_2 = x = 50 \) C.

From a conceptual point of view, the data rectification in Example 1 can be explained using Occam’s razor. Three scenarios are possible: 1) the true temperature is near 50 C, sensor 1 has not failed and sensor 2 has failed with approximately a -30 C bias, 2) the true temperature is near 20 C, sensor 2 has not failed and sensor 1 has failed with approximately a +30 C bias, and 3) the true temperature is neither near 50 or 20 C, and both sensors have failed with a large bias (compared to the measurement noise standard deviation of 1 C). Scenario one has two likely events, that the true temperature is 50 C and that sensor 1 has not failed, and one unlikely event, that sensor 2 has failed. Scenario 2 has two unlikely events, that the true temperature is 20 C and that sensor 1 has failed, and one likely event, that sensor 2 has not failed. Scenario three has at least two unlikely events, that sensor 1 and sensor 2 have both failed, and another event that is less likely than the true temperature being 50 C. According to Occam’s razor, and to the laws of probability, scenario one is the more likely than scenario two or three because it requires only one unlikely event to explain the observations, whereas the other two scenarios require at least two unlikely events.
This simple example shows how information on the operation of the plant can be exploited to improve the data rectification process, and in this case solve a problem not solvable by traditional methods.

It is important that the distribution of $P(x)$ include a full range of plant behaviors. For plants that operate at a variety of steady states, each of the steady states needs to be represented in the distribution of $P(x)$, appropriately weighted by the prior probability of each plant state. Assuming the plant has a number of mutually-exclusive and exhaustive operating modes, the distribution $P(x)$ is given by:

$$P(x) = \sum_{j} P(x|o_j)P(o_j)$$

(4.9)

where the $o_j$'s are the different plant operating modes. Such modes could be operation at different rates of production, operation for different product purities, or processing of different feed compositions as in a refinery. If $P(x)$ is calculated from historical plant data, then this data set should be a representative set of the plant states, and not just a single operating point. Should a novel operating state not accounted for in the distribution of $P(x)$ occur, the MLR technique will make erroneous adjustments to the data because $P(x)$ will be underestimated at the measurement point. The resulting rectified state will contain erroneously large adjustments necessary to increase the $P(x)$ contribution to the MLR objective function, and thus $P(y|x)$ will be small. The result will be a MLR objective function value that is outside the norm, and this small value can be used as a flag to indicate that a novel operating point has been reached and the rectified values should not be trusted. The threshold value could be easily determined by plotting a histogram of objective function values over a set of calibration cases, and determining the 95% (or similar) point of the distribution.
4.3 Sensor Model

The $P\{y | x\}$ term in equation 4.8 is the model of the measurement error. For a particular measurement $y_i$ the measurement can be modeled by:

$$y_i = x_i + \delta_i$$ \hspace{1cm} (4.10)

where $\delta_i$ is the error. The usual case is that the error in the sensor is an additive random value, but multiplicative errors, where the size of the error is dependent on the size of the signal, are also possible (Liptak and Venczel, 1982). Equation 4.10 can be used to model multiplicative errors by allowing $\delta_i$ to be a function of $x_i$. For the case where $\delta_i$ is independent of $x_i$, then $P\{y_i | x_i\} = P\{\delta_i\}$, which from equation 4.3 is given by:

$$P\{\delta_i\} = \sum_j P\{\delta_i | m_j\} P\{m_j\}$$ \hspace{1cm} (4.11)

When a sensor is in its normal mode of operation the error term $\delta_i$ represents the sensor "noise", and is usually modeled as a zero mean Gaussian distribution. The magnitude of the noise term is reported by sensor manufacturers and contains the effects of conformity, hysteresis, dead band, and repeatability errors (Liptak and Venczel, 1982). Because the noise term is a culmination of a number of different errors when the sensor is operating normally, a Gaussian distribution is appropriate for the noise contribution to equation 4.11, as the central limit theorem states that the cumulative error will approach a Gaussian distribution as the number of error sources increase. Additional failed modes can also be characterized, such as failure to a fixed value (modeled as a delta function), and a failure to a random value (modeled as a uniform distribution). However, the exact distributions of other failure modes may be
difficult to characterize. If, over a period of time, smaller gross errors are more likely than larger gross errors, the distribution of gross errors might be approximated by a Gaussian distribution whose standard deviation is greater than the standard deviation of the normal noise. Under this assumption, the total distribution of the error term $\delta_i$ can be reasonably modeled by a bivariate Gaussian:

$$P(\delta_i) = (1 - p_i) \frac{1}{\sqrt{2\pi} \sigma_i} \exp\left(-\frac{(x_i - y_i)^2}{2\sigma_i^2}\right) + p_i \frac{1}{\sqrt{2\pi} b_i \sigma_i} \exp\left(-\frac{(x_i - y_i)^2}{2b_i^2\sigma_i^2}\right) \quad (4.12)$$

In this expression, $p_i$ is the probability of a gross error in sensor $i$, $\sigma_i$ is the standard deviation of the normal noise band in sensor $i$, and $b_i$ is the ratio of the standard deviation of the gross error distribution of sensor $i$ to the standard deviation of the normal noise band for sensor $i$. Jeffreys (1932) was the first to apply this form of error distribution to regression of experimental data, although applications were limited due to lack of computation power. Fariss and Law (1979) presented a numerical technique for regression using the bivariate Gaussian. Tjoa and Biegler (1991) applied equation 4.12 for data rectification. As mentioned previously, the MLR approach reduces to Tjoa and Biegler's approach if equation 4.12 is used to model the sensor error, and the $P(x)$ is modeled simply by constraints (binary assumption).

If one wishes to assume a minimal amount about the distribution of the gross errors, then the second term in equation 4.12 (the broad Gaussian), can be replaced with a uniform distribution. This formulation is analogous to traditional gross error detection methods in that no assumption (other than limits) is made on the distribution of the gross error. In traditional gross error detection schemes (e.g. global, nodal, and measurement test), no information about how gross errors are distributed is required. Modeling the gross error contribution to equation 4.11 using a uniform distribution
corresponds to no prior knowledge about the distribution of gross errors, according to the principle of maximum entropy (Levine and Tribus, 1979). The impact of different gross error distributions on the solution to the MLR problem is explored in section 4.6.

Regardless of the choice of the individual sensor error distributions, by making the assumption that the sensor biases are independent of each other, the joint distribution of the biases is the product of the individual biases:

\[ P(\delta) = \prod_i P(\delta_i) \quad (4.13) \]

This reduces the MLR objective function in equation 4.8 to:

\[
\begin{align*}
\max_x P(x) P(\delta) \\
\text{s.t. } h(x) &= 0 \\
g(x) &\leq 0
\end{align*}
\quad (4.14)
\]

The MLR objective function for additive errors has two terms. \( P(x) \) is the probability of the rectified state, and \( P(\delta) \) is the probability of the adjustments made to the measurements. Thus there is a tradeoff between likely rectified states and likely adjustments. The optimization seeks a solution that not only ends at a probable rectified state, but also is a probable adjustment to the measurement set. This is conceptually shown in figure 4.2. The "+" is the measurement point, and the concentric circles about it are equal probability contours \( P(\delta) \), the highest probability being at the measurement point itself. The elliptical probability contours conceptually represent the prior distribution of the plant states, \( P(x) \) in the \( x_1-x_2 \) state space. At the measurement point, although \( P(\delta) \) is high, the \( P(x) \) contribution to the objective function is low, resulting in a low objective function value. The MLR process adjusts the measurement
Figure 4.2: Schematic representation of MLR problem formulation showing tradeoff between likely adjustments ($P\{\delta\}$) and likely rectified states ($P\{x\}$).
to increase the contribution of $P(x)$, by moving the rectified state to increase the $P(x)$ value. As larger adjustments are made to the measurement, the $P(\delta)$ contribution to the objective function decreases while the $P(x)$ contribution increases, thus the objective function goes through a maximum. This point is the most likely rectified state given the measurement and information in $P(x)$.

4.4 Data Rectification and Robust Regression

In this section, we explore the analogy between maximum likelihood rectification and robust regression, a technique used in parameter estimation when the data contains outliers. In steady-state data rectification, the measurements $y$ represent a single point in the measurement space and the goal is to find the point $x$ that lies on a predefined surface (the surface described by the model constraints), that is most likely given $y$. In regression, there are a large number of points in the measurement space, and the goal is to find a set of parameters ($\hat{\beta}$) that yield the most likely description of the surface (the parameterized model), given the measurements and the model form. Rectification fits a point to a surface and regression fits a surface to a set of points. The maximum likelihood regression problem is posed as:

$$
\max_{\hat{\beta}} \rho(Y, X, \hat{\beta})
$$

s.t. $f(X, \hat{\beta}) = 0$

$$g(\hat{\beta}) \geq 0$$

(4.15)

where $\rho$ is the objective function to be maximized, and $Y$ and $X$ are matrices consisting of the observed individual row vectors $x$ and $y$. The data rectification problem (assuming $P(x)$ is modeled by a set of constraints) is posed as:
\[
\begin{align*}
\max_{\hat{x}} & \quad \rho(y, \hat{x}) \\
\text{s.t.} & \quad f(\hat{x}, \beta) = 0 \\
& \quad g(\hat{x}, \beta) \geq 0
\end{align*}
\]

(4.16)

where the model parameters ($\beta$) are given.

In ordinary regression or data rectification, weighted least squares is the objective function. For clarity of presentation the regression and rectification problems are posed here assuming that the noise covariance matrices are diagonal, and the regression problem is posed such that the predicted $y$ is a scalar and not a vector. These not a necessary assumptions but they greatly simplifies the notation. For both problems the weighted least squares objective function is:

\[
\rho(z) = \sum_{i=1}^{n} \frac{1}{2} z_i^2
\]

(4.17)

where for regression,

\[
z_i = \frac{y_i - f(x_i, \hat{\beta})}{\sigma_i}
\]

(4.18)

and for rectification,

\[
z_i = \frac{y_i - \hat{x}_i}{\sigma_i}
\]

(4.19)

For the regression case the $z_i$'s are summed over the number of data points, and in the rectification case the $z_i$'s are summed over the dimension of the measurement space.

The effect of outliers on the solution of regression problems has been extensively studied in the statistics literature. Hampel (1974) uses the concept of an influence curve to study ordinary and robust regression. The influence of data point $i$ is defined as the derivative of the objective function with respect to the error term $z_i$:  

94
The influence curve defines the region in which a measurement has an effect on the estimation of the parameters. For weighted least squares regression, from equation 4.17, the influence curve is \( I_i = z_i \). Thus the influence of an error on the estimation of the parameters is unbounded (Cook and Weisberg, 1982). This means that as the error in the \( i \)th measurement grows it exerts an increasing influence on the estimated parameters, even though at some level of error it becomes likely that the data point is an outlier. Regressing a data set containing outliers with least squares will yield poorly fit models because the outliers dominate the other data.

The concept of robust regression was developed to negate or limit the effect of outliers on the regressed parameters. Robust regression techniques use probability distribution of errors that have longer tails, thus making large adjustments in some measurements more probable. One such distribution used for robust estimation is the Lorentzian distribution (Huber, 1981)

\[
\rho(z) = \sum_{i=1}^{n} \frac{1}{1 + \frac{1}{2} z_i^2}
\] (4.21)

Figure 4.3 shows the influence curve for the Lorentzian objective function. The influence of an error in a measurement increases as the error grows from zero, but then decreases and eventually becomes zero once the error becomes large. This is a desirable form of the influence curve because severely outlying points are discounted, rather than emphasized as in least squares. The formulation in equation 4.3 using multiple modes of sensor operation achieves a similar broadening of the tails of the
Figure 4.3: Influence curve for Lorentzian distribution of errors.
distribution while appealing to more fundamental considerations to motivate the form of
the distribution (Kramer and Mah, 1993). Thus the distribution of errors given in
equation 4.3 could be interpreted as a robust regression probability function, as the
following example illustrates.

Example 2. This example is a simple linear regression between one
dependent variable (y) and one independent variable (x) with an
intercept at the origin,

\[ y = \beta x \]

The data set contains 20 points drawn from a uniform distribution
between \( x = 0 \) and \( x = 1 \). The \( y \) points were generated by adding
Gaussian noise

\[ y = x + \varepsilon, \quad \varepsilon = N(0, 0.1) \]

The \( \hat{\beta} \) parameter was regressed using ordinary least squares
regression (equation 4.17) and using the robust bivariate sensor
model. For the robust regression, the objective function is

\[ \rho(y, x, \hat{\beta}) = \prod_{i=1}^{20} \left[ (1-p) \frac{1}{\sqrt{2\pi} \sigma} \exp\left(-\frac{(y_i - \hat{\beta} x_i)^2}{2\sigma^2}\right) + p \frac{1}{\sqrt{2\pi} b \sigma} \exp\left(-\frac{(y_i - \hat{\beta} x_i)^2}{2b^2 \sigma^2}\right) \right] \]

For this case \( \sigma = 0.1, p = 0.05, \) and \( b = 20. \)
The estimated $\hat{\beta}$ for both the ordinary and robust regression were calculated as a function of the error in one data point (at $x=0.341$), by adding errors ranging from $-5\sigma$ to $+5\sigma$ to the corresponding $y$ value, and minimizing $\rho$. The results are shown in Figure 4.4. The compound objective function clearly has the desired effect of discounting outliers, in a manner qualitatively equivalent to the Lorentzian distribution. Errors greater than $\pm 4\sigma$ effectively have no influence on the estimated slope. Errors smaller than $\pm 4\sigma$ are not considered outliers and thus do have an influence on the solution.

For data rectification it is desirable to discount the measurements that contain gross errors, in a manner analogous to outliers in robust regression. The sensor mode model (equation 4.3) again gives the same desirable shape of influence curve, as in robust regression, for rectification problems.

**Example 3.** Consider a process stream whose temperature is measured by three thermocouples. The model equations are $T_1 = T_2 = T_3$. Each measurement of the true temperature $T$ is given by:

$$T_i = T + \delta_i$$

where $\delta_i$ is the error in the $i^{th}$ measurement. For simplicity assume the true stream temperature varies uniformly between 50 and 150 C. Thus $P(T)$ is a constant, and the MLR objective function reduces to:
Figure 4.4: Influence curve for weighted least squares (dashed line) and robust regression using a bivariate Gaussian error distribution (solid line).
\[
\max_{\hat{T}} \prod_{i=1}^{3} P(T_i \mid \hat{T}) \\
\text{s.t. } 50 \leq \hat{T} \leq 150
\]

Note that \( P(T_i \mid \hat{T}) \) is analogous to \( P(y_i \mid x_i) \) where all of the \( x_i \)'s are equal because of the model constraints. The compound objective function in equation 4.12 is used with \( p = 0.05, b = 20, \) and \( \sigma = 1. \) The nominal measurement point was \( T_1 = 100, T_2 = 102, T_3 = 100. \) The error in \( T_3 \) was systematically varied from \(-10\) to \(+10\) C and the rectified temperature for each of the error levels is plotted against the error as a solid line in figure 4.5. Again the influence curve shows that large errors in \( T_3 \) have no effect on the rectified temperature.

We can also compare robust rectification to traditional rectification with auxiliary gross error detection and removal. The dashed line in figure 4.5 shows the influence curve for weighted least squares rectification using the global test at a 0.05 level of significance for gross error detection, and serial elimination for gross error identification. Qualitatively, the effect of gross error detection and removal are the same as robust rectification, for this example. The discontinuities in the shape of the influence curve are due to incorrect identification of a gross error by the serial elimination procedure, for certain levels of error in \( T_3. \)

The foregoing example shows that the sensor mode model in equation 4.3 gives the desired influence curve given a fully specified model and no prior distribution of the
Figure 4.5: Influence curve for MLR (solid line) and traditional linear reconciliation with gross error detection and removal using the global test and serial elimination (dashed line), for Example 3.
underlying process states. We will now show that the MLR objective function gives a similar influence curve when there is a known prior distribution of the states, but the lack of constraints precludes the use of conventional gross error approaches.

Example 4. Consider again the problem in Example 1, where a stream is measured by two thermocouples, and it is known from historical data that the true temperature is distributed normally with a mean of 50 C and a standard deviation \( \sigma_T = 10 \) C. The MLR objective function for this data rectification problem is

\[
\max_T \frac{1}{\sigma_T \sqrt{2\pi}} \exp \left( -\frac{(T-50)^2}{2\sigma_T^2} \right) \prod_{i=1}^2 \left( 1-p \right) \frac{1}{\sigma_Y \sqrt{2\pi}} \exp \left( -\frac{(T-T_i)^2}{2\sigma_Y^2} \right) + \frac{1}{b \sigma_Y \sqrt{2\pi}} \exp \left( -\frac{(T-T_i)^2}{2b^2 \sigma_Y^2} \right)
\]

where the measurement errors have been modeled using a bivariate Gaussian (equation 4.12) with \( p = 0.05 \), \( b = 20 \), and \( \sigma = 1 \) C. With \( T_1 \) fixed at 50 C, \( T_2 \) was varied from 25 to 75 C and the rectified temperature was plotted as a function of the error in \( T_2 \) as a solid line in figure 4.6. An influence curve which gives little or no weight to measurements with large errors is again observed. The dashed line in figure 4.6 is the rectified temperature using weighted least squares rectification. With only two measurements, gross error tests can be done to signal that a gross error has occurred, but no conventional procedure can identify which sensor has the gross error. Tjoa and Biegler's approach of maximizing the robust objective function (equation 4.12) subject to the \( T_1 = T_2 \) constraint, without using the distribution of \( P(T) \) (i.e. assuming \( P(T) \) is uniform and thus equals a
Figure 4.6: Influence curve for MLR (solid line) and traditional linear reconciliation (dashed line), for Example 1.
constant), also yields the unbounded influence curve shown by the
dashed line in figure 4.6. The prior distribution of the temperature is
required to make correct inferences about which sensor has failed and
automatically remove it from the rectification process by giving it zero
weight in the solution.

4.5 Estimating $P(x)$

The preceding example demonstrates the potential importance of modeling the
prior state distribution. $P(x)$ captures the physical constraints of the process, such as
mass balances, energy balances, thermodynamic relationships, and transport
relationships, as well as the statistical tendencies in operation of the plant (e.g. the
distribution over time of set points). The Recursive State Density Estimation (RSDE)
algorithm for estimating $P(x)$ from historical plant measurement data is fully developed
and its performance assessed in Chapter 5.

4.6 Process Examples

In this section, two chemical engineering examples are presented to demonstrate
data rectification by the MLR technique. The first is a flow system from Mah (1987),
and the second example is a heat exchanger network that was taken, with some
modification, from Tjoa and Biegler (1991). The performance criterion used to gauge
the efficacy of the rectification is the mean squared error:

$$MSE = \frac{1}{N*K} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{(x_{n,k} - \hat{x}_{n,k})^2}{\sigma_n^2}$$  \hspace{1cm} (4.22)

104
where \( N \) the dimension of the measurement vector, \( K \) is the number of measurement vectors used to calculate the MSE, \( \hat{x} \) is the rectified state, \( x \) is the true plant state, and \( \sigma_n \) is the standard deviation of the Gaussian noise on sensor \( n \). The MSE is structured such that if there are no sensor failures (i.e. the measurements only contain random Gaussian noise with standard deviations \( \sigma_n \)) then the MSE = 1.0 for \( \hat{x} = y \) (exact equality to 1.0 occurs only in the limit of infinite data). If there are gross errors, the MSE can be greater than 1.0. If the rectification is perfect (i.e. the true state of the plant was found for every measurement vector) then the MSE = 0.0. Thus the goal is to drive the MSE towards zero.

All of the sensors, in both examples, are assumed to have a bivariate Gaussian distribution (equation 4.12). For all sensors \( p_i = 0.05 \), and \( b_i = 20 \).

Figure 4.7 shows the flow network from Mah (1987). The total flow of each stream is measured. Data were generated by sampling \( F_1 \) from \( U(15,40) \), calculating the remaining flows, and generating the \( y \) values by adding random, zero-centered, Gaussian noise with standard deviations 1.0, 4.0, 4.0, 3.0, and 1.0 respectively, for the five sensors. A training data set of 100 points was generated to estimate \( P(x) \) using the RSDE algorithm in Chapter 5, and a separate 1000-point test set was also generated for the performance studies. A multivariate Gaussian distribution was used for \( P(x) \), which, in spite of the mismatch from the actual distribution, was found to be the best estimator for this data set using cross-validation (see Chapter 3). Another 1000-point data set was also created, and in this set a failure of the \( F_3 \) sensor was simulated by adding a bias sampled from \( N(0, 20) \).

Although this example involves simple linear balance equations, the example is used to demonstrate how the number of known constraints affects the data rectification process. There are three independent mass balances that can be written for this system.
Figure 4.7: Flow network used in data rectification.
By assuming that only a subset of these balances are known, we can demonstrate how different degrees of prior process knowledge affect the rectification. In general, the more relationships that are known, the better the data rectification will be. Table 4.1 shows the MSE results for the MLR method as well as the traditional linear weighted least squares method as a function of the number of constraints used. In the case of simulated failures in F3, the global test and serial elimination were used detect and remove gross errors before reconciling the data. As the number of constraints increases the MSE is driven lower for both the linear reconciliation and the MLR method. In all cases the MLR method outperforms the linear method. In the case of the failure of F3 the linear reconciliation technique, including serial elimination, is unable to satisfactorily remove the errors from the measurements (MSE > 1.0). However, the results show that even in the complete absence of known process constraints, the data is rectified reasonably well by the MLR method. The MSE is reduced from 1.02 to 0.431 in the case of no gross error, and from 19.93 to 0.485 for simulated failure in F3. The capture of the physical constraints between the process variables in P(x) allows the MLR method to rectify the data in the absence of analytical constraints, and the added information about the probability distribution of the rectified states causes the MLR method to outperform the linear reconciliation even when constraints are used.

Figure 4.8 shows a heat exchanger network that was modified from Tjoa and Biegler (1991). Process stream A is heated by process stream B in two exchangers, and with utility stream C in one exchanger. The flow of stream D is adjusted so that the outlet temperature of stream A, TA8, will be 61.5°C. The plant operates at low, medium, and high capacity levels where process streams A and B are at low, medium, and high flow rates. The temperatures of all streams are at constant values, with some process noise. The operating conditions are summarized in figure 4.8. Using these
Table 4.1: MSE results for rectification of flow example data, $P(x)$ calibrated from normal data only.

<table>
<thead>
<tr>
<th># Constraints</th>
<th>Linear</th>
<th>MLR</th>
<th>Linear*</th>
<th>MLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>---</td>
<td>0.431</td>
<td>---</td>
<td>0.485</td>
</tr>
<tr>
<td>1</td>
<td>0.793</td>
<td>0.392</td>
<td>---</td>
<td>0.426</td>
</tr>
<tr>
<td>2</td>
<td>0.597</td>
<td>0.337</td>
<td>3.51</td>
<td>0.362</td>
</tr>
<tr>
<td>3</td>
<td>0.408</td>
<td>0.263</td>
<td>1.56</td>
<td>0.276</td>
</tr>
</tbody>
</table>

* with serial elimination
Figure 4.8: Heat exchanger network example.

Operating Conditions

\[ \begin{align*}
\text{FC1} &= N(240,5) \\
\text{TA1} &= N(450,5) \\
\text{TB1} &= N(620,5) \\
\text{TC1} &= N(670,5) \\
\text{TD1} &= N(690,5) \\
\text{FA3}/\text{FA2} &= 0.42
\end{align*} \]

Probability = 0.3:
\[ \begin{align*}
\text{FA1} &= N(800,5) \\
\text{FB1} &= N(200,5)
\end{align*} \]

Probability = 0.4:
\[ \begin{align*}
\text{FA1} &= N(900,5) \\
\text{FB1} &= N(250,5)
\end{align*} \]

Probability = 0.3:
\[ \begin{align*}
\text{FA1} &= N(990,5) \\
\text{FB1} &= N(320,5)
\end{align*} \]
operating conditions, a training set of 120 points was generated, from which \( P(x) \) was estimated using the RSDE technique. The noise added to the training and test sets had a standard deviation of 0.75 for the temperature measurements, and standard deviations of 4.0, 2.0, 3.0 and 3.0 for flows FA1, FB1, FC1, and FD1 respectively. In this case the density estimator that best fit the data was an elliptical basis function estimator with five units and overlap of 1 (see Johnston and Kramer, 1994). Three separate test sets were created for performance testing. One set had only Gaussian noise in the sensors. Another set simulated a failure in sensor TA4, by adding a bias from a \( N(0, 10) \) distribution, and the final test set simulated a failure in both TA4 and TB1 by adding a biases drawn from a \( N(0, 10) \) distribution.

The results are shown in Table 4.2. The MLR was done without any constraints because the instrumentation on the network does not allow for any mass or energy balances to be written. Again the MLR shows its power in estimating the state of the plant, without any prior constraints. The MSE was reduced from 1.0 to 0.53, from 10.31 to .724, and from 20.62 to .864, for the three data sets. Even when two sensors had failed the MLR was able to find the rectified values and lower the MSE below the random noise level of 1.0. Figure 4.9 shows that the MLR effectively rejects the gross errors present in the sensor. The results show that even in a 13-dimensional problem the MLR is capable of finding the rectified values without any prior process model. The addition of constraints would improve the performance, as shown in the previous example.

In the foregoing examples, the \( P(x) \) was generated from data that contained no sensor failures. In general there could be some sensor failures in the data set used to calibrate \( P(x) \). The RSDE technique is capable of estimating \( P(x) \) accurately even when some sensor failures are present in the calibration data. The data rectification for
Table 4.2: MSE results for rectification of heat exchanger network data by MLR technique.

<table>
<thead>
<tr>
<th></th>
<th>No Failures</th>
<th>Failure of TA4</th>
<th>Failure of TA4 and TB1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unrectified</td>
<td>1.016</td>
<td>10.31</td>
<td>20.62</td>
</tr>
<tr>
<td>Rectified</td>
<td>0.527</td>
<td>0.724</td>
<td>0.864</td>
</tr>
</tbody>
</table>
Figure 4.9: Plot of the measurement (+) and the final rectified value (o) of temperature TA4 against the true value of temperature TA4 for a data set with gross errors in TA4. Perfectly rectified values lie on the rectified value equals true value line shown.
the flow network shown in figure 4.7, was also performed when 10% of the 100-point training set contained $F_3$ measurements from a failed sensor. The results for the MLR rectification are shown in Table 4.3, and compared to the case where there were no sensor failures in the calibration data used to find $P(x)$. The RSDE methodology removed the sensor failures in the calibration data, and the $P(x)$ that was finally arrived at was essentially the same as the $P(x)$ found from the data set without the sensor failures. The performance of the MLR was essentially unchanged. The performance of the RSDE methodology is further examined in Chapter 5.

The effect of different gross error distributions on the MLR solution was also investigated. The MLR of data from the flow network were calculated for different values of $b$ and $p$, the parameters of the bivariate Gaussian sensor model (equation 4.12). Separate 1000 point test sets, simulated the same way as the data used for the results in Table 4.1, were used and the mean square error for the different combinations of $b$ and $p$ are shown in Tables 4.4 and 4.5. The MLR was done without analytic constraints. The solution of the MLR is not sensitive to the choice of the probability of a gross error ($p$) or to the width of the gross error distribution ($b$) for the ranges investigated.

4.7 Conclusions

A new method for data rectification for steady state systems, based on maximum likelihood estimation, has been introduced and demonstrated. The Maximum Likelihood Rectification (MLR) method is a generalized framework that allows for various sensor behaviors and any number of process model constraints, including no constraints. The method uses historical plant data to find the prior probability distribution of the true plant states, using the RSDE methodology developed in Chapter
Table 4.3: MSE results for rectification of flow example data, $P(x)$ calibrated from data corrupted with 10% gross errors.

<table>
<thead>
<tr>
<th># Constraints</th>
<th>No Failures $\text{mse}=1.02$</th>
<th>Failure of F3 $\text{mse}=19.93$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MLR 0.433</td>
<td>MLR 0.490</td>
</tr>
<tr>
<td>1</td>
<td>MLR 0.395</td>
<td>MLR 0.428</td>
</tr>
<tr>
<td>2</td>
<td>MLR 0.338</td>
<td>MLR 0.365</td>
</tr>
<tr>
<td>3</td>
<td>MLR 0.263</td>
<td>MLR 0.277</td>
</tr>
</tbody>
</table>
Table 4.4: MSE values for data rectified using the MLR method with no constraints, for various combinations of b and p parameters, for 1000 point data set containing no gross errors.

<table>
<thead>
<tr>
<th>p</th>
<th>b = 5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>0.401</td>
<td>0.402</td>
<td>0.402</td>
<td>0.403</td>
<td>0.403</td>
</tr>
<tr>
<td>0.05</td>
<td>0.400</td>
<td>0.401</td>
<td>0.402</td>
<td>0.402</td>
<td>0.402</td>
</tr>
<tr>
<td>0.10</td>
<td>0.397</td>
<td>0.400</td>
<td>0.401</td>
<td>0.401</td>
<td>0.402</td>
</tr>
<tr>
<td>0.15</td>
<td>0.392</td>
<td>0.398</td>
<td>0.400</td>
<td>0.400</td>
<td>0.401</td>
</tr>
<tr>
<td>0.20</td>
<td>0.388</td>
<td>0.395</td>
<td>0.397</td>
<td>0.399</td>
<td>0.400</td>
</tr>
</tbody>
</table>
Table 4.5: MSE values for data rectified using the MLR method with no constraints, for various combinations of b and p parameters, for 1000 point data set containing gross errors in sensor F3.

<table>
<thead>
<tr>
<th>p=</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>0.583</td>
<td>0.578</td>
<td>0.569</td>
<td>0.564</td>
<td>0.571</td>
</tr>
<tr>
<td>0.05</td>
<td>0.529</td>
<td>0.531</td>
<td>0.524</td>
<td>0.515</td>
<td>0.523</td>
</tr>
<tr>
<td>0.10</td>
<td>0.511</td>
<td>0.501</td>
<td>0.502</td>
<td>0.495</td>
<td>0.503</td>
</tr>
<tr>
<td>0.15</td>
<td>0.482</td>
<td>0.476</td>
<td>0.465</td>
<td>0.468</td>
<td>0.479</td>
</tr>
<tr>
<td>0.20</td>
<td>0.436</td>
<td>0.441</td>
<td>0.444</td>
<td>0.452</td>
<td>0.463</td>
</tr>
</tbody>
</table>
5. This methodology is robust enough to find the distribution of the underlying states from historical data, even when there are measurement biases (gross errors) in the historical data. The MLR technique was demonstrated on two examples, and was able to obtain good estimates of the true plant state, even when no process model was used in the rectification. This constitutes a new class of problem that can now be solved.
Chapter 5:
Estimating State Probability Distributions from Noisy and Corrupted Data

The method of Recursive State Density Estimation (RSDE) is developed for determining the probability distribution of the states of a system from measurements that contain both random noise and gross errors. The technique is based on the Expectation Maximization (EM) algorithm and is iterative in nature. Similar to EM, at each iteration the likelihood of the distribution estimated by the Recursive State Density Estimation algorithm is guaranteed to increase, thus arriving at the most likely distribution of the true states, given the measurement data set and the algorithm initial conditions. Convergence of the algorithm to the correct solution, for a simple case where an analytical answer can be derived for comparison, is shown. Two chemical process examples, that have more complex distributions, are also shown. Once the probability distribution of the states has been determined, many monitoring and statistical process and quality control functions can be performed using the more accurate distributions of the process states, avoiding corruption of the distribution due to faulty measurements.
5.1 Introduction

In chemical and manufacturing plants, numerous measurements of state variables are made to obtain a picture of what is happening in the plant. The measurement systems used to observe the process introduce "small" random errors when operating normally (measurement noise) and can introduce "large" systematic errors when a failure of the measurement system occurs (gross errors). Historical plant data contain both type of errors. If we wish to derive the probability distribution of the true states from historical data, a way to remove these errors is required. Such a probability distribution can be used for various tasks including statistical process control (Owen, 1989), quality control (Montgomery, 1985), and more recently data rectification (Chapter 4). If reasonable assurances can be made that the fundamental processes that have led to the historical probability distribution of the states is unchanged, then the probability distribution of the states can be used for future data rectification, statistical process control, quality control, modeling, and prediction.

Casting the model of the plant in a probabilistic framework facilitates the use of all the standard tools of statistical and Bayesian analysis to make sound probabilistic statements about the validity of the solutions to the various problems being solved. This includes deciding when the process has deviated from the fundamental processes that were driving the historical distribution of the states, resulting in a change in the underlying probability distribution of the states. This ability to detect when the historical model is no longer valid is extremely important, as it warns the user that the results obtained from the model are unlikely to be correct.

Any parametric or nonparametric density estimation technique allows for the calculation of the probability density function for a given set of data. The goal of this
Chapter is to estimate the probability density function of the true plant states, given a data set that includes measurement errors. This leads to a dilemma, in that all that is available are the corrupted measurements, but what is required is the probability distribution of the states of the measured variables. This Chapter introduces a technique in which the probability distribution of the states can be estimated from the noisy and corrupted historical measurement sets. We use Expectation Maximization (Dempster et al, 1977) to link probability density function estimation, from Chapter 3, with maximum likelihood data rectification, from Chapter 4, to arrive at the probability distribution of the states using the measurements.

5.2 Expectation Maximization

The Expectation Maximization (EM) algorithm (Dempster, 1977) is a broadly applicable algorithm for computing maximum likely estimates from incomplete data. In many applications a large number of data vectors are collected for modeling and analysis purposes. Invariably in these large data sets some of the data vectors have "holes" in them, for reasons of a failed sensor, clerical omission, or other data collection error. The rest of the data vector does hold valuable information that would be lost if the entire data vector were thrown out because of the missing measurements. But to use standard modeling and analysis tools, a value must be supplied for the missing value. EM is used to calculate the most likely estimate of the relevant parameters, such as regression coefficients, means, variances, and other parameters of interest depending on the application, in spite of the missing data.

The EM algorithm consists of an expectation step, the E-step, and a maximization step, the M-step, which are iteratively repeated until convergence. If the model were known, then the missing values could be calculated by finding their
expected values given the known data and the model (the E-step). Conversely, if the data set were complete, then well known complete data techniques for finding the most likely parameter values could be used to find the parameters of the model (M-step). After an initial estimate of the missing values, the EM algorithm iteratively goes through the E and M steps until convergence is reached. In the E-step the sufficient statistics of the missing data are estimated, as opposed to the missing data themselves (Little and Rubin, 1987). Stable convergence is achieved as the M-step finds parameters that monotonically increase the likelihood of the parameters given the data at each iteration. This monotonic convergence ensures that the maximum likelihood estimate of the parameters given the data, and initial starting point of the algorithm, are arrived at (Meng and Rubin, 1994). The EM algorithm has been used in a broad range of applications, including estimates of parameters of density functions when there is missing data (Redner and Walker 1984, Meng and Rubin 1991, Yuille et al 1994), in evaluating synaptic transmissions (Stricker and Redman, 1994), as well as in expert systems (Jordan and Jacobs, 1994). In this application we use the EM algorithm to link data rectification and probability density estimation steps to arrive at the distribution of the plant states from the noisy and corrupted measurements.

5.3 Recursive State Density Estimation

The proposed application of the EM algorithm is different from previous applications in that in this case the "missing data" are the true states of the plant, not sporadic missing measurements in a data set. We call the application of the EM algorithm for finding the density estimate of the states from noisy and corrupted measurements Recursive State Density Estimation (RSDE). The best estimate of the true states of the plant at the beginning of the RSDE procedure are the measurements
themselves. The probability distribution $P(x)$ is found by fitting an appropriate
distribution, parametric or nonparametric, to the $y$ data by estimating the sufficient
statistics of $P(x)$ with $x=y$. This is the M-step. With an estimate of $P(x)$ the expected
value of the sufficient statistics of the "missing data" (the $x$ data) can be calculated.
This is the E-step. The M-step and E-step are then alternated between until the
convergence of the distribution is obtained. In the examples section, a simple example
with a known analytical answer is used to illustrate the procedure and demonstrate its
convergence.

The sufficient statistics of the missing data will depend on the form of the
distribution used for estimating $P(x)$. For example, if a simple one dimensional
Gaussian is used, the sufficient statistics are the sum of the $x$ values and the sum of
squared $x$ values. In the general case of process data, the distribution is often multi-
dimensional and multi-modal, and its form is usually unknown. Nonparametric density
estimators are capable of capturing the more complex distributions, but these
distributions depend on the data themselves for their form. Thus the sufficient statistics
of the nonparametric density estimators are the $x$ data themselves.

Casting the problem of estimating the distribution of the $x$ data from the
measurements $y$ in terms of the EM algorithm, and using a nonparametric density
estimator, leads to the following RSDE algorithm, which is graphically shown in Figure
5.1:

1) Set initial estimate of the $x$ data equal to the measurements $y$
2) Estimate $P(x)$ using an appropriate method
3) Use the MLR approach in Chapter 4 to find the maximum
   likely new $x$ values given the measurements and the current
   estimate of $P(x)$
Figure 5.1: Diagram of the Recursive State Density Estimation algorithm

\[
P(\hat{x}) \rightarrow \hat{x} = \max \text{MLR}(\hat{x})
\]
\[
\text{s.t. } f(\hat{x}) = 0; \; g(\hat{x}) < 0
\]

Initialize with \( \hat{x} = y \)
Using the new $x$ values from 3 go to step 2 until convergence is reached.

Convergence is reached when none of the $x$ values change upon subsequent iterations of the algorithm. An efficient solution method for solving the MLR problem in step 3 is given in Chapter 6.

The RSDE methodology applies to any form of parametric or nonparametric density estimator that is appropriate for the given data set. If a parametric distribution for $x$ is used, then the sufficient statistics of the $x$ data, and not the $x$ data themselves, need to be calculated in step 3. Through the MLR formulation, the problem can be implemented with any distribution of the measurement error, including ones that allow for sensor failure, making it broadly applicable. Application of the algorithm is shown in section 5.6.

5.4 Incorporating Linear Constraints

In the data rectification step of the RSDE algorithm, the MLR is solved subject to known constraints. If linear constraints are imposed on the rectification then there will be an exact linear correlation among the rectified values, which are then used to find $P(x)$ in the next iteration. If the mixture models with Gaussian basis functions, such as EBF estimators, are used to model $P(x)$, then the exact linear correlation in the data will cause a singularity in the covariance matrix of the Gaussian basis functions, and it will not be possible to find the inverse of the covariance matrix which is needed to find the density estimate. Although the data is observed in $n$ dimensions, the data is truly only $(n-m)$ dimensional, where $m$ is the number of independent linear model constraints. One way to avoid this singularity problem is to estimate $P(x)$ in $n$
dimensions and not enforce the linear equality constraints in MLR solution during the RSDE algorithm. The linear relationships are "rediscovered" in $P(x)$ as the Gaussian distributions will approximate the linear relationships by constructing very thin discs as substitutes for the plane that represent the linearly constrained data. The analytical linear equality constraints are then enforced when the derived $P(x)$ distribution is used to rectify future data.

The singularity problem is avoided, and estimation of $P(x)$ is improved, if the density estimation is done only in the subspace of the independent dimensions. For a system constrained by $m$ linear equations that satisfy the equation $Ax=0$, a probability distribution is constructed in the linear independent subspace defined by the $m$ by $n$ constraint matrix $A$,

$$P(x) = f(Px)$$  \hspace{1cm} (5.1)

where $f$ is the probability density estimation function. The matrix $P$ is a projection matrix that maps $x$ in $n$-space to a corresponding vector in $m$-space. The $n$ dimensional vector $x$ is recovered by multiplying the $m$ dimensional vector by $P^T$. The $(n-m)$ by $n$ projection matrix $P$ is formed by finding $(n-m)$ orthonormal $n$ dimensional basis vectors that span the null space of plane described by $Ax=0$ (Brown, 1991). Each row of $P$ corresponds to a basis vector. In general, the projection of an $n$ dimensional vector $u$ to an $m$ dimensional vector $v$ by multiplying by $P$, then projection of $v$ back to $n$-space by multiplication by $P^T$ does not result in the recovery of $u$. That is, for

$$v=Pu \text{ and } u'=P^Tv$$  \hspace{1cm} (5.2)
will not equal \( \mathbf{u} \) generally. This is because the projection matrix \( \mathbf{P} \) projects all vectors to the plane \( \mathbf{Ax} = 0 \). For the case where \( \mathbf{Au} = 0 \) then the transformation in equation 5.2 will result in \( \mathbf{u} = \mathbf{u}' \), because if the vector already lies on the plane then the projection does not move the vector but only changes its basis from an \( n \) dimensions to \( m \) dimensions. Thus for all valid rectified states (i.e. those for which \( \mathbf{Ax} = 0 \)) the matrix projection does not move \( \mathbf{x} \) but only temporarily changes its dimension for calculation of \( \mathbf{P}(\mathbf{x}) \) in a lower dimension than \( \mathbf{x} \).

5.5 Performance Measures

Three performance measures: log probability (\( J \) value), log likelihood, and mean squared error (MSE) are used to assess the performance of the RSDE algorithm in the section 5.6. The most important measure is the \( J \) value as it measures the goodness of fit of the calculated probability density function. The log likelihood and MSE measures should monotonically increase and monotonically decrease respectively at subsequent iterations of the RSDE algorithm.

5.5.1 Log Probability (\( J \) value)

The measure of how well a particular density estimator approximates a density function is the product of the probabilities of sample points independently and randomly drawn from the true underlying distribution (Traven, 1991). For a test set drawn from the true underlying distribution of the states of size \( n_t \), the overall fit measure is given by:

\[
J' = \prod_{i=1}^{n_t} P(\mathbf{x}_i)
\]  

(5.3)
For convenience of calculation, usually the normalized log of equation 5.3 is used:

\[ J = \frac{1}{n_l} \log(J') = \frac{1}{n_l} \sum_{i=1}^{n_l} \log(P(x_i)) \]  \hspace{1cm} (5.4)

For each of the three examples, a training set of the states \( x \) is calculated and the measurements \( y \) are created by adding noise and possibly gross errors to the \( x \) data. A probability distribution density function estimator is then fit to the \( x \) training data, the raw \( y \) training data, and by the RSDE method for each of the three examples. The \( J \) value for a separate 1000 point test set of \( x \) data is calculated for each estimator. The largest possible \( J \) value for an estimator is that of the estimator calculated from the \( x \) training data, because we cannot expect to produce an estimator that is any better than one that is fit to the true plant states. We also want the \( J \) value of the estimator to be better than the \( J \) value of the estimator fit to the raw \( y \) data. Thus the range of \( J \) values for a good estimator is between the \( J \) value of the estimator fit to the \( y \) data (on the low side) and the \( J \) value of the estimator fit to the \( x \) data (on the high side). A good estimator will be closer to \( J \) value of the estimator fit to the \( x \) data. The examples given are all from simulated chemical plants, and thus the true plant states are known and an estimator fit to the true plant states can be generated for comparison purposes. In practice the true plant states are never known and thus the RSDE technique needs to be used to find an estimate of the true plant states.

5.5.2 Log Likelihood

The likelihood of the rectified state for an individual measurement in the training set is the product of the probability of the rectified state \( P(x_k) \) and the probability the adjustment made to the measurement \( P(\delta_k) \):
\[ L' (x_k, y_k) = P(x_k)P(\delta_k) \] (5.5)

For the entire training set the likelihood is the product of the individual likelihoods:

\[ L' (X, Y) = \prod_{k=1}^{K} P(x_k)P(\delta_k) \] (5.6)

where \( X \) and \( Y \) are \( K \) by \( n \) matrices containing the \( K \) individual \( x \) and \( y \) row vectors.

For convenience usually the normalized log of equation 5.6 is used:

\[ L(X, Y) = \frac{1}{K} \sum_{k=1}^{K} \left[ \log(P(x_k)) + \log(P(\delta_k)) \right] \] (5.7)

In the EM algorithm the log likelihood function being maximized, in this case equation 5.7, is guaranteed to monotonically increase until convergence (Dempster, 1977).

Monotonic increase of the log likelihood function at each iteration of the RSDE algorithm is shown for each of the examples.

### 5.5.3 Mean Squared Error

One application of the probability distribution of the plant states estimated by the RSDE methodology is data rectification (Chapter 4), and data rectification is one of the steps of the RSDE algorithm. Thus the RSDE algorithm estimates the true plant states for the training set of data used to calibrate \( P(x) \), along with the desired probability density function. A measure of the accuracy of the rectified states is the mean squared error (MSE)
\[
\text{MSE} = \frac{1}{N*K} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{(\hat{x}_{n,k} - x_{n,k})^2}{\sigma_n^2}
\]  
(5.8)

where \(N\) is the dimension of the measurement vector, \(K\) is the number of measurement vectors used to calculate the MSE, \(\hat{x}\) is the rectified state, \(x\) is the true plant state, and \(\sigma_n\) is the standard deviation of the Gaussian noise on sensor \(n\). The MSE is structured such that if there are no sensor failures (i.e. the measurements only contain random Gaussian noise with standard deviations \(\sigma_n\)) then the MSE = 1.0 (see Chapter 4). If there are gross errors, the MSE can be greater than 1.0. If the rectification is perfect (i.e. the true state of the plant was found for every measurement vector) then the MSE = 0.0. Thus the goal is to drive the MSE towards zero. It is expected that the MSE will monotonically decrease at each RSDE iteration because we expect each subsequent RSDE iteration to have a more accurate \(P(x)\) which should lead to more accurate rectified values and this would be reflected by a lower MSE. Plots of the MSE as a function of the RSDE iteration showing the monotonic decrease are shown for the two chemical engineering system examples.

5.6 Examples

The RSDE methodology was first applied to a simple one dimensional distribution of states where the parameters of the state distribution can be analytically derived from the parameters of the measurement distribution, which can be easily calculated using standard techniques. The RSDE algorithm is then applied to find the probability distribution of the states for two chemical engineering processes from noisy and corrupt measurements. 

129
5.6.1 One Dimensional Gaussian Distribution

The distribution of the states of a one dimensional system are known to follow a Gaussian distribution with unknown mean $\mu$ and unknown variance $\sigma_x$. The system is observable through measurements which are recorded as $y$ and are related to $x$ by

$$y = x + \varepsilon$$

(5.9)

where $\varepsilon$ is a normally distributed random variable with mean zero and known variance $\sigma$. Further, $\varepsilon$ is independent of $x$. Given the set of $y$ data, the mean and variance of $y$ can be calculated using standard techniques. From the $y$ data, the distribution of the $x$ data can be analytically calculated. The mean of $x$ is equal to the mean of $y$, and the variance of $x$ is equal to the variance of $y$ minus the variance of $\varepsilon$ ($\sigma^2$). This is because the distribution of $y$ is formed by the sum of two independent Gaussians, and the mean of $y$ is the sum of the mean of $x$ and mean of $\varepsilon$, and the variance of $y$ is the sum of the variance of $x$ and the variance of $\varepsilon$. This same result can be achieved by applying the principles of EM in the RSDE algorithm. The advantage of using the RSDE algorithm is that it is not only applicable to this simple case, but it is also applicable to more complex systems where the deconvolution of the distribution of $x$ from the distribution of $y$ is extremely difficult or impossible to do.

To apply the EM algorithm, the complete data set is considered to be both $x$ and $y$. The "missing" data in the EM sense are the $x$ values. The joint distribution of $x$ and $y$, given the mean and variance of $x$ is:

$$f(x_i, y_i \mid \mu, \sigma_x) = f(y_i \mid x_i, \mu, \sigma_x) f(x_i \mid \mu, \sigma_x)$$

(5.10)
\[ f(x_i, y_i | \mu, \sigma_x) = \frac{\beta_i}{\sqrt{2\pi} \sigma_T} \exp \left[ -\frac{1}{2} \left( \frac{\left( x_i - \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2}{\sigma_T^2} \right) \right] \] (5.11)

thus the joint distribution for the complete data set of \( n \) measurements is

\[ f(X, Y | \mu, \sigma_x) = \prod_{i=1}^{n} \frac{\beta_i}{\sqrt{2\pi} \sigma_T} \exp \left[ -\frac{1}{2} \left( \frac{\left( x_i - \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2}{\sigma_T^2} \right) \right] \] (5.12)

which simplifies to

\[ f(X, Y | \mu, \sigma_x) = \exp \left[ -\frac{1}{2} \sum_{i=1}^{n} \left( \frac{x_i - \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2 \right] \prod_{j=1}^{n} \frac{\beta_j}{\sqrt{2\pi} \sigma_T} \] (5.13)

where,

\[ \beta_i = \frac{1}{\sqrt{2\pi} \sqrt{\sigma_x^2 + \sigma^2}} \exp \left[ -\frac{1}{2} \frac{\sigma_x^2 y_i^2 + \sigma^2 \mu^2}{\sigma^2 \sigma_x^2} \right] \exp \left[ -\frac{1}{2} \left( \frac{\sigma_x^2 + \sigma^2}{\sigma^2 \sigma_x^2} \right) \left( \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2 \right] \] (5.14)

and

\[ \sigma_T = \frac{\sigma \sigma_x}{\sqrt{\sigma + \sigma_x^2}} \] (5.15)

In the EM algorithm the missing data themselves are not estimated, but the sufficient statistics of the missing data are estimated. The sufficient statistics for the \( x \) data are the sum of the \( x \) values and the sum of the square of the \( x \) values, as the joint distribution is linear in these two quantities. Thus we want to estimate:

\[ \mathbb{E} \left[ \sum_{i=1}^{n} (x_i | y_i, \mu, \sigma_x) \right] = \sum_{i=1}^{n} \mathbb{E}[x_i | y_i, \mu, \sigma_x] \] (5.16)
and
\[ E \left[ \sum_{i=1}^{n} (x_i^2 \mid y_i, \mu, \sigma_x) \right] = \sum_{i=1}^{n} E [x_i^2 \mid y_i, \mu, \sigma_x] \]  \hspace{1cm} (5.17)

For the \( i \)-th measurement, the expected value of the corresponding state and state squared are

\[ E [x_i \mid y_i, \mu, \sigma_x] = \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \]  \hspace{1cm} (5.18)

and

\[ E [x_i^2 \mid y_i, \mu, \sigma_x] = \sigma_T^2 + \left( \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2 \]  \hspace{1cm} (5.19)

where we have used the fact that

\[ \text{Var}(x_i) = E [x_i^2] - (E [x_i])^2 \]  \hspace{1cm} (5.20)

thus

\[ E [x_i^2] = \sigma_T + (E [x_i])^2 \]  \hspace{1cm} (5.21)

Therefore, the expected values of the relevant sums are:

\[ E \left[ \sum_{i=1}^{n} (x_i \mid y_i, \mu, \sigma_x) \right] = \sum_{i=1}^{n} \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \]  \hspace{1cm} (5.22)

and

\[ E \left[ \sum_{i=1}^{n} (x_i^2 \mid y_i, \mu, \sigma_x) \right] = n \sigma_T^2 + \sum_{i=1}^{n} \left( \frac{\sigma_x^2 y_i + \sigma^2 \mu}{\sigma^2 + \sigma_x^2} \right)^2 \]  \hspace{1cm} (5.23)
This completes the E step. The M step calculates the usual Maximum Likelihood Estimates of the mean and variance given the sufficient statistics. Given the values of \( \mu \) and \( \sigma_x \) from the \( t \)th iteration, the values of \( \mu \) and \( \sigma_x \) at the \((t+1)\)th iteration are found by

\[
\mu^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \left( x_i | y_i, \mu^{(t)}, \sigma_x^{(t)} \right) = \frac{1}{n} \sum_{i=1}^{n} \frac{\sigma_x^{(t)} y_i + \sigma_{\mu}^{(t)}}{\sigma^2 + \sigma_x^{(t)}^2}
\]

which converges to

\[
\hat{\mu} = \frac{\sum_{i=1}^{n} y_i}{n}
\]

(5.25)

and

\[
\sigma_x^{(t+1)} = \frac{1}{n} \left( \sum_{i=1}^{n} \left( x_i^2 | y_i, \mu^{(t)}, \sigma_x^{(t)} \right) - \left( \sum_{i=1}^{n} \left( x_i | y_i, \mu^{(t)}, \sigma_x^{(t)} \right) \right)^2 \right)
\]

(5.26)

which simplifies to

\[
\sigma_x^{(t+1)} = \frac{\sigma_x^{(t)}^2}{\sigma^2 + \sigma_x^{(t)}^2} + \frac{1}{n} \sum_{i=1}^{n} \left( \frac{\sigma_x^{(t)} y_i + \sigma_{\mu}^{(t)}}{\sigma^2 + \sigma_x^{(t)}^2} \right)^2 - \frac{1}{n^2} \left( \sum_{i=1}^{n} \frac{\sigma_x^{(t)} y_i + \sigma_{\mu}^{(t)}}{\sigma^2 + \sigma_x^{(t)}^2} \right)^2
\]

(5.27)

and converges to

\[
\hat{\sigma}_x^2 = \sigma_y^2 - \sigma^2
\]

(5.28)

Thus the correct result for the mean and variance of \( x \) given the \( y \) data are achieved using the EM algorithm. This demonstrates that the RSDE algorithm, which casts the \( x \) values as "missing" data and then applies the principles of EM, converges to the maximum likely result for the mean and variance of the state variables given the measurements. The RSDE algorithm can now be applied to more complex systems,
where analytical results, as in the one dimensional Gaussian distribution case, are not readily available.

5.6.2 Linearly Constrained Flow Network

Figure 5.2 shows the flow network from Mah (1987). The total flow of each stream is measured. The linear plant constraints are \( Ax = 0 \), where \( A \) is composed of the three independent mass balances that constrain the process:

\[
A = \begin{bmatrix}
1 & -1 & 0 & 1 & 0 \\
0 & 1 & -1 & 0 & 0 \\
0 & 0 & 1 & -1 & -1 \\
\end{bmatrix}
\]

and \( x \) is a 5 by 1 vector containing the 5 flowrates. The plant was assumed to operate at a low and high flow rate with probabilities 0.45 and 0.55 respectively. At the low flow rate \( F_1 \) was drawn from \( N(100, 10) \) distribution, and at the high flow rate \( F_1 \) was drawn from \( N(150, 10) \) distribution. The ratio of \( F_4 \) to \( F_3 \) was drawn from a \( N(0.6, 0.05) \) distribution. Having randomly selected \( F_1 \) and the separator split fraction, the remaining flows were calculated, and the \( y \) values were generated by adding random, zero-centered, Gaussian noise with standard deviations 1.0, 4.0, 4.0, 3.0, and 1.0 respectively, for the five sensors. A 100 point data set was generated for calibration of \( P(x) \), and a separate 1000 point data set was generated for testing the accuracy of the fit distributions. The cross validation methodology in Chapter 3 was used to select the best density estimator, which was an EBF estimator with four basis functions.

The RSDE methodology was applied with \( P(x) \) fit in the linearly independent two-dimensional subspace of the flow network. The projection matrix \( P \) is found by finding the null space of \( A \), and is:
Figure 5.2: Flow network example.
\[
P = \begin{bmatrix}
-0.6124 & -0.2041 & -0.2041 & 0.4082 & -0.6124 \\
0 & 0.5714 & 0.5714 & 0.5174 & 0
\end{bmatrix}
\]

In addition to the RSDE estimator, a density estimator was fit to the generated \( x \) data and to the raw \( y \) data. To fit an estimator to the \( y \) data it was first projected to the constraint plane by multiplication by \( P \). Because \( y \) does not lie on the constraint plane, the raw \( y \) data vector cannot be recovered by \( P^\top P y \), as it can for the \( x \) values that do satisfy the constraints. Thus the projection to the plane reconciles but does not completely rectify the \( y \) data. The \( J \) value for each estimator is summarized in Table 5.1. The RSDE estimators \( J \) value of -7.47 was very close to the maximum expected \( J \) value for the estimator fit to the \( x \) data of -7.45 and much larger than the \( y \) estimator \( J \) value of -7.62.

Figure 5.3 shows that the log likelihood monotonically increases with RSDE iteration as expected. The raw \( y \) data does not satisfy the constraints and thus \( P(x) \) at \( x = y \) (the initial condition for the RSDE algorithm) is 0. Therefore the log likelihood at the start of the algorithm (iteration 0) is undefined and is not plotted. Figure 5.4 shows the monotonic decrease of the MSE from 1 to 0.32. Thus at each RSDE iteration more accurate estimates of the true plant states are being found.

The RSDE algorithm was also applied to a data set that was corrupted with gross errors in the F1 sensor. A gross error ranging from -30 to +30 was added to 10% of the F1 measurements in the 100 point training data set. The \( J \) value for the RSDE estimator was -7.48 which was essentially the same as the \( J \) value of the RSDE estimator fit to the data without the gross errors, and very close to the \( J \) value of -7.45 for the estimator fit to the \( x \) data. The estimator fit to the raw \( y \) data with the gross
Table 5.1: Log Probability (J) values for probability density functions (PDF) estimated from the true state variables, the measurements, and by the RSDE algorithm for the three examples.

<table>
<thead>
<tr>
<th>PDF estimated from</th>
<th>Flow Network</th>
<th>Heat Exchanger Network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Noise Only</td>
<td>10% Gross Errors</td>
</tr>
<tr>
<td>x data</td>
<td>-7.45</td>
<td>-7.45</td>
</tr>
<tr>
<td>RSDE</td>
<td>-7.47</td>
<td>-7.48</td>
</tr>
</tbody>
</table>
Figure 5.3: The monotonic increase of the log likelihood function during the iterations of the RSDE algorithm for the Flow Network data set with no gross errors.
Figure 5.4: The monotonic decrease of the MSE value during the iterations of the RSDE algorithm for the Flow Network data set with no gross errors.
errors was -7.74. Thus the RSDE algorithm was able to effectively reject the gross
errors and find a good estimate of P(x). This was also shown by the MSE being
reduced from 2.72 to 0.40.

The RSDE methodology was then applied by fitting P(x) in five dimensions
without forcing the known analytical constraints to be obeyed in the data rectification
step. It is not suggested that P(x) be estimated in the full dimension of the
measurements when there are known analytical model constraints, but this example is
shown to demonstrate how "unknown" process correlations can be captured by P(x).
In this case the linear correlations in the data are "discovered" by the RSDE
methodology and incorporated into P(x). Figure 5.5 shows the log likelihood at each
RSDE iteration and it monotonically increases. The log likelihood does not level off
because the estimated plant states are almost perfectly linearly correlated after six
iterations, and thus the covariance matrices of the basis functions are nearly singular.
Small changes in the estimated plant states, increasing the linear correlation among
them, cause the covariance matrices to become closer to singular and thus the P(x)
value drastically rises between iterations, causing the log likelihood to continually
increase. If RSDE algorithm is continued eventually the estimated plant states become
linearly correlated enough to cause the basis function covariance matrices to become
singular, and the algorithm will terminate as the P(x) function can no longer be
calculated. Estimating P(x) in the linearly independent subspace, as shown previously,
eliminates this problem. The MSE was reduced from 1 to 0.37 during the RSDE
algorithm showing that the P(x) distribution was capturing the process data correlation
without enforcement of the linear constraints. In this manner the P(x) distribution will
also capture all correlations in the data and then exploit these correlations in the data
rectification process.

140
Figure 5.5: The monotonic increase of the log likelihood function during the iterations of the RSDE algorithm for the Flow Network data set with no gross errors with $P(x)$ estimated in 5 dimensions.
5.6.3 Heat Exchanger Network

Figure 5.6 shows a heat exchanger network that was adapted from Tjoa and Biegler (1991). Process stream A is heated by process stream B and utility stream C and D. The flow of utility stream D is adjusted to maintain an outlet temperature of stream A (TA8) of 615°C. The instrumentation of the network is such that mass and energy balances cannot be completely written for the system. Table 5.2 shows the operating conditions of the network. The RSDE technique was applied to a data set that consisted of 120 points, and an EBF density estimator with 6 units was used to estimate the probability density function. A separate 1000 point test data set was generated to calculate the J values of density estimators calculated from the x data, raw y data, and by the RSDE algorithm.

The J values for each estimator are summarized in Table 5.1. The J value for the RSDE estimator was -28.73 which is close to the J value of -27.38 for the estimator fit to the x data and much larger than the J value of -31.37 for the estimator fit to the y data. Thus the implicit process constraints were captured, and a fairly accurate $P(x)$ was found using the RSDE algorithm. As in the previous example, the log likelihood monotonically increased with each iteration of the RSDE methodology, and thus the maximum likelihood distribution of the states given the measurements was achieved. The MSE monotonically decreased from 1.05 to 0.62, indicating that at each subsequent iteration the estimate of the state was closer to the true state. The data rectification in this case was done without mathematical constraints because the instrumentation on the network did not allow for mass and energy balance to be written. Nonetheless the data rectification was quite accurate and thus we have a good deal of confidence in the estimated probability distribution of the plant states.
Figure 5.6: Heat exchanger network example.
Table 5.2: Operating Conditions for the heat exchanger network.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC1</td>
<td>N(240,5)</td>
</tr>
<tr>
<td>TA1</td>
<td>N(450,5)</td>
</tr>
<tr>
<td>TB1</td>
<td>N(620,5)</td>
</tr>
<tr>
<td>TC1</td>
<td>N(670,5)</td>
</tr>
<tr>
<td>TD1</td>
<td>N(690,5)</td>
</tr>
<tr>
<td>FA3/FA2</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Probability = 0.3:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA1</td>
<td>N(800,5)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(200,5)</td>
</tr>
</tbody>
</table>

Probability = 0.4

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA1</td>
<td>N(900,5)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(250,5)</td>
</tr>
</tbody>
</table>

Probability = 0.3

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA1</td>
<td>N(990,5)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(320,5)</td>
</tr>
</tbody>
</table>
Another data set for the heat exchanger network was created with 120 points. For this data set a gross error between -10 and +10 C was present in 10% of the measurements of TA4. The J value for the estimator calculated by the RSDE algorithm was -29.61 which was larger than the J value calculated from the raw y data containing the gross errors of -32.16. The log likelihood monotonically increased throughout all iterations of the RSDE methodology, although the starting and final log likelihood values were lower than for the data set containing only noise because of the gross errors present. The gross errors also caused the starting MSE to be approximately 2.27, as opposed to the 1.05 starting value for the data set with no gross errors, but the RSDE methodology still reduced the MSE for the data set with gross errors to the same 0.62 level that was observed for the data set with no gross errors.

During the iterative RSDE process, the gross errors are rejected as being unlikely because they lie in regions of low probability. For an individual measurement vector, the sensor containing the gross error is drawn towards the correct value because by making a large adjustment in the erroneous sensor and small adjustments in the other sensors, which do not contain gross errors, the rectified state is moved into a more likely region of P(x) than the measurement. This adjustment lowers the P(δ) contribution to the objective function value (equation 4.14) but increases the P(x) contribution to the objective function value. The net result is an increase in the objective function value, and thus the log likelihood value monotonically increases during the algorithm. The RSDE methodology only makes adjustments that are probable given the information at hand. Figure 5.7 shows the measurements and the final rectified values plotted against the true values for sensor TA4, the sensor with the gross error. The end result is that gross errors are effectively rejected, and the rectified values obtained are close to the true values. This results in the probability distribution.
Figure 5.7: Plot of the measurement (+) and the final rectified value (o) of TA4 against the true value of TA4 for the data set in which 10% of the TA4 measurements are corrupted by gross errors.
estimated by the RSDE methodology being very close to the true probability
distribution of the states.

5.7 Conclusions

The Recursive State Density Estimation (RSDE) methodology is presented as an
application of the EM algorithm for finding the maximum likely distribution of the true
plant states given the noisy and corrupted measurements. As an application of the EM
algorithm, the convergence properties of the RSDE methodology follow that of the EM
algorithm and thus the maximum likely distribution of the states given the
measurements and the algorithm initial conditions is ensured. The RSDE methodology
was shown to converge to the correct analytic result for a simple one-dimensional
Gaussian distribution. The RSDE algorithm was also successfully applied to the more
complex distributions of two simulated chemical processes, specifically a flow network
and a heat exchanger network. In all cases the RSDE methodology performed well,
even when the data set was corrupted with 10% gross errors. For the examples shown,
the probability density functions obtained using the RSDE algorithm were better
estimators of the true state probability distribution than a probability density function
estimator fit to the raw measurement data, as measured by the log probability of an
independent test data set. For a linear flow network, the RSDE methodology was
applied in both the full space of the measurements and in the linearly independent
subspace of the processes, a distribution which accurately rectified the data was found
in both cases. The RSDE methodology yields the probability distribution of the true
plant states, that is most likely given the measurement information, and this distribution
can then be applied to safety, quality control, and statistical process control tasks.
Using the distribution of the true plant states, and not the distribution of the corrupted measurements, will improve the performance of each of these tasks.
Chapter 6:
An Efficient Algorithm for Maximum Likelihood Data Rectification

With simple assumptions, the maximum likelihood rectification objective function involves a sum of exponential functions, which can be time consuming to solve using nonlinear programming methods. We exploit the structure of the objective function by introducing two sets of indicator variables into the problem, reducing the objective function to a quadratic form, if the indicator variable values are known. We then apply the Expectation Maximization (EM) algorithm to iteratively estimate the values of the indicator variables, and to estimate the rectified state using the surrogate quadratic objective function, which has an analytical solution when there are no analytical constraints or when the constraints are linear (e.g. mass balances), which makes the algorithm computationally fast. Additionally, the objective function value is a measure of the likelihood of the rectified state given the measurement and the prior distribution of the plant states. The final objective function value is required to be above a calculated threshold, to ensure that the rectified states are likely and that the method has not terminated at a local maximum. An efficient algorithm for re-starting the EM optimization procedure based on the likelihood of candidate starting points drawn from the prior distribution of the states is developed. The final value of the indicator variables identify which measurements contain gross errors. The proposed solution technique was found to be approximately 20 times faster than conventional nonlinear programming methods, with no loss of accuracy, for the two chemical process examples investigated.
6.1 Introduction

The Maximum Likelihood Rectification (MLR) technique of Johnston and Kramer (1995), also presented in Chapter 4, seeks to find the set of plant states that are most likely given the observed plant measurements. Mathematically the states that maximize the conditional probability of the plant states given the measurements are sought:

\[
\max_x P(x|y) = \max_x \frac{P(y|x)P(x)}{P(y)} = \max_x P(y|x)P(x)
\]  

(6.1)

The last equality is achieved because \( P(y) \) is a normalization constant, for a given measurement, and does not depend on \( x \), and thus can be ignored.

The first term in equation 6.1, \( P(y|x) \), is the probability distribution of the measurement given the state. A measurement of a state can be modeled as the sum of the true state being measured and an additive error,

\[
y_i = x_i + \delta_i
\]

(6.2)

where \( \delta_i \) is the error. The usual case is that the error in the sensor is an additive random value, but multiplicative errors, where the size of the error is dependent on the size of the signal, are also possible (Liptak and Venczel, 1982). Equation 6.2 can be used to model multiplicative errors by allowing \( \delta_i \) to be a function of \( x_i \). For the case where \( \delta_i \) is independent of \( x_i \), then \( P(y|x_i) = P(\delta_i) \).

The \( P(\delta_i) \) represents the statistical distribution of the error of the sensor which may include many modes of operation, including a normal mode and any number of failure modes. The overall error distribution can be found by summing the product of
the probability distribution of a particular error mode and the prior probability of the mode, over all of the modes of operation.

\[
P(\delta_i) = \sum_j P(\delta_i|m_j) P(m_j)
\]  \hspace{1cm} (6.3)

When a sensor is in its normal mode of operation, the error term \(\delta_i\) represents the sensor "noise", and is usually modeled as a zero mean Gaussian distribution. The magnitude of the noise term is reported by sensor manufacturers and contains the effects of conformity, hysteresis, dead band, and repeatability errors (Liptak and Venczel, 1982). Because the noise term is a culmination of a number of different errors when the sensor is operating normally, the cumulative error will approach a Gaussian distribution as the number of error sources increase, by the central limit theorem. Thus a Gaussian distribution is appropriate for the noise contribution to equation 6.3. A common source of error to many sensors is miscalibration. Additional failure modes can also be characterized, such as failure to a fixed value (modeled as a delta function), and a failure to a random value (modeled as a uniform distribution). The exact distributions of many failure modes may be difficult to characterize. For example, it is reasonable to assume that small miscalibrations are more likely to occur than large ones, but it may not be possible to model the exact distribution of the miscalibrations. If smaller gross errors are more likely than larger gross errors, the distribution of the gross errors arising from the many sensor error modes might be approximated by a Gaussian distribution, whose standard deviation is greater than the standard deviation of the normal noise. Under this assumption, the total distribution of the error term \(\delta_i\) can be reasonably modeled by a bivariate Gaussian:

\[
P(\delta_i) = (1-p_i) \frac{1}{\sqrt{2\pi} \sigma_i} \exp\left(-\frac{(x_i-y_i)^2}{2\sigma_i^2}\right) + p_i \frac{1}{\sqrt{2\pi} \sigma_i} \exp\left(-\frac{(x_i-y_i)^2}{2b_i^2\sigma_i^2}\right)
\]  \hspace{1cm} (6.4)
In this expression, \( p_i \) is the probability of a gross error in sensor \( i \), \( \sigma_i \) is the standard deviation of the normal noise band in sensor \( i \), and \( b_i \) is the ratio of the standard deviation of the gross error distribution of sensor \( i \) to the standard deviation of the normal noise band for sensor \( i \). Further development of the sensor model is given in Chapter 4.

Assuming that sensor errors are uncorrelated, the joint distribution of the biases is the product of the individual biases:

\[
P(\delta) = \prod_i P(\delta_i)
\]

This reduces the MLR objective function in equation 6.1 to:

\[
\max_x P(x) P(\delta) \\
\text{s.t. } h(x) = 0 \\
g(x) \leq 0
\]

The objective function requires a tradeoff between likely adjustments to the measurements \( P(\delta) \), and arrival at a likely rectified state \( P(x) \), while satisfying known constraints. The tradeoff is discussed in further detail in the next section.

The final form of the objective function depends on the functional form used to model \( P(x) \). The Recursive State Density Estimation (RSDE) method is developed in Chapter 5 to estimate \( P(x) \) from noisy and corrupted measurements. Although any form of the distribution can be used, non-parametric estimators are often the best choice as the functional form of the probability distribution is typically unknown. Mixture models, which are a weighted sum of a number of Gaussian distributions, are one form of non-parametric density estimators that have been shown to asymptotically approach
the true distribution of the data, as the amount of data used in the estimation increases (Everitt and Hand, 1981). Chapter 3 proposes an algorithm for estimating probability density functions using an Elliptical Basis Function (EBF) network, which is a mixture of Gaussian shaped units. The form of the estimator is

$$\rho(x) = \sum_{h=1}^{H} w_h \psi_h(x)$$  \hspace{1cm} (6.7)

where

$$\psi(x) = \exp\left(- (x - m_h)Q_h^{-1}(x - m_h)\right)$$  \hspace{1cm} (6.8)

In the EBF estimator, all of the scale factors are contained in the $w_h$ term, and thus each individual $\psi(x)$ function is Gaussian in shape but not exactly a Gaussian distribution. The scale factors that are contained in the weighting value $w_h$ can be removed from $w_h$ and inserted instead into the function $\psi(x)$. This converts the EBF function estimator into a summation of Gaussians.

$$\rho(x) = \sum_{h=1}^{H} q_h \phi_h(x; m_h, Q_h)$$  \hspace{1cm} (6.9)

where

$$\phi(x; m_h, Q_h) = \frac{1}{\sqrt{(2\pi)^n \det(Q_h)}} \exp\left(- \frac{(x - m_h)Q_h^{-1}(x - m_h)}{2}\right)$$  \hspace{1cm} (6.10)

and $Q_h$ is a scaled version of $Q'_h$. Note that the mixture of Gaussian units is not used because the process truly operates in a number of Gaussian modes, but because density estimators using mixture of Gaussians are able to capture any type of distribution, in the limit an infinite number of infinitely small units. In practice, estimators with a modest number of units are able to estimate most commonly encountered distributions.
Using the same notation as equation 6.10, the mixture of two Gaussians sensor model in equation 6.11 can be represented as

\[ P(\delta_i) = (1-p_i)\phi(\delta_i; 0, \sigma_i^2) + p_i\phi(\delta_i; 0, b_i \sigma_i^2) \]  \hspace{1cm} (6.11)

where generally \( \phi(x; m, Q) \) denotes a Gaussian distribution evaluated at \( x \), centered at \( m \) with covariance matrix \( Q \), as given by equation 6.10. As discussed in Chapter 5, \( P(x) \) may be fit in a lower dimension than \( x \), by projection by the orthonormal vectors spanning the linearly independent subspace (matrix \( P \)). Thus, using equations 6.9 and 6.11 in equation 6.1, along with the projection of \( x \) by \( P \) for calculating \( P(x) \), the resulting maximum likelihood problem that need to be solved is:

\[
\max_x \left[ \prod_{i=1}^{n} \left( (1-p_i)\phi(x_i; y_i, \sigma_i^2) + p_i\phi(x_i; y_i, b_i \sigma_i^2) \right) \right] \left[ \sum_{h=1}^{H} q_h \phi_h(Px; m_h, Q_h) \right] 
\hspace{1cm} (6.12)
\]

\[
s.t. \quad h(x) = 0 \\
g(x) \leq 0
\]

If \( P(x) \) is estimated in the same dimension as \( x \), then \( P \) in equation 6.12 is set to the identity matrix, or equivalently \( P \) is set to the scalar 1. Equation 6.12 is the objective function form for \( P(x) \) described by a mixture of Gaussian distributions, and the measurement error \( P(\delta) \) described by a bivariate Gaussian distribution. This chapter focuses on this form of the MLR objective function to develop an efficient maximization algorithm.
6.2 Surrogate Objective Function

It is generally computationally expensive to find the solution to equation 6.12. When the objective function is a single exponential function or a product of exponentials, taking the natural logarithm converts the problem into one that is quadratic in the argument \( x \), and greatly simplifying the computations. Because of the summations that appear in both terms of equation 6.12 this approach cannot simplify the computation. We propose an alternative approach that exploits the simplifications that arise when the objective function is a product of the exponential functions. In particular, we introduce new variables which allow us to represent the objective function in equation 6.12 as a product of exponentials with the same global maximum as equation 6.12 (Jordan and Jacobs, 1994). Taking the natural log of this surrogate objective function results in a quadratic function of \( x \). The unconstrained and linearly constrained maximum of this function has an analytical solution, and thus the rectified values are easily obtained.

A mixture distribution like equation 6.9 can be thought of as the marginal distribution of \( x \) derived from \( P(x, z) \), the joint probability distribution of \( x \) and \( z \), where \( z \) is a vector of dimension \( H \). Each \( z_h \) value represent the probability of \( x \) being from component \( h \) of the mixture distribution (equation 6.9). Each of the indicator variables, \( z_h \), can only take on the value one or zero, and only one \( z_h \) may be non-zero at a time. Thus the single \( z_h \) that is non zero "indicates" from which of the \( H \) component distributions \( x \) belongs to. The joint pdf of \( x \) and \( z \) is then

\[
P(x, z) = \prod_{h=1}^{H} [q_h P(x|lh)]^{z_h}
\]  

(6.13)
Because $z_h$ are binary variables, $P(x)$ is formed from $P(x, z)$ by summing over all $H$ of the possible combinations of $z_h$. The equality between equation 6.13 and the actual mixture distribution $P(x)$ is verified by doing this sum:

$$P(x) = \sum_z P(x, z) = \sum_z \prod_{h=1}^H [q_h P(x|h)]^{z_h} = \sum_{h=1}^H q_h P(x|h)$$

$$= \sum_{h=1}^H q_h \phi_h (P_x; m_h, Q_h)$$

(6.14)

where $\sum_z$ denotes the summation over the $H$ possible combinations of the indicator variables (i.e. $z={z_1=1, z_2=0, \ldots z_H=0}$, $z={z_1=0, z_2=1, \ldots z_H=0}$, \ldots $z={z_1=0, z_2=0, \ldots z_H=1}$).

Similarly, the addition in the distribution of the error, $P(\delta)=P(y|x)$ (first term of equation 6.12), can be removed so that only a product of Gaussians remains, by introducing a second set of indicator variables analogous to the $z$ indicator variables introduced for the $P(x)$ term. The $i^{\text{th}}$ indicator variable, $\zeta_i$, has a value of one when the $i^{\text{th}}$ sensor contains a gross error and has a value of 0 when the $i^{\text{th}}$ sensor error is only the instrument noise. Thus the product form of the distribution of $P(y|x)$ using the indicator variables is:

$$P(y, \zeta | x) = \prod_{i=1}^n (1-p_i)\phi(x; y, \sigma_i^2)^{(1-\zeta_i)} p_i \phi(x; y, b_i \sigma_i^2)^{\zeta_i}$$

(6.15)

Similar to equation 6.14, the verification that equation 6.15 reduces to the form of $P(y|x)$ is obtained by summing over all the $2^n$ possible combinations of the indicator variables.

156
Using 6.13 and 6.15 in 6.12 leads to the surrogate MLR objective function using the indicator variables:

$$\max_{x, z, \xi} \left( \left[ \prod_{h=1}^{H} q_h \phi(Px; m_h, Q_h) \right]^{2a} \right)$$

$$\left[ \prod_{i=1}^{n} \left( (1-p_i) \phi(x; y, \sigma_i^2) \right)^{(1-\xi_i)} \left( p_i \phi(x; y, b_i \sigma_i^2) \right)^{\xi_i} \right]$$

(6.16)

Taking the natural logarithm of equation 6.16 leads to a sum of the powers in the exponentials, which are quadratic in x.

$$\max_{x, z, \xi} \left( -\sum_{h=1}^{H} z_h (Px - m_h)^T Q_h^{-1} (Px - m_h) + (P^T Px - y)^T \Lambda (P^T Px - y) + C \right)$$

(6.17)

where C is a constant independent of x and for i=1 to n

$$\Lambda^{-1} = \text{diag} \left( \frac{(1-\xi_i)}{\sigma_i^2} + \frac{\xi_i}{b_i^2 \sigma_i^2} \right)$$

(6.18)

In equation 6.17, the $P^T Px$ term ensures that the rectified x values satisfy the linear constraints that the P matrix was derived from (see Chapter 5).

By ignoring terms that are independent of x, completing the square in x, and minimizing the negative of the function, equation 6.17 can be written in a standard least squares format:

$$\min_{x, \omega} (Px - \omega)^T S^{-1} (Px - \omega)$$

(6.19)

where
\[ S = \left( \sum_{h=1}^{H} z_h Q_h^{-1} + PA^{-1} P^T \right)^{-1} \]  \hfill (6.20)

and

\[ \omega = S \left( \sum_{h=1}^{H} z_h Q_h^{-1} m_h + PA^{-1} y \right) \]  \hfill (6.21)

Thus, by the introduction of indicator variables \( z \) and \( \zeta \) the complex objective function in equation 6.12 has been converted to a standard quadratic form. Although the surrogate objective function in equation 6.19 is a least squares formulation it retains all of the advantages of the parent objective function of equation 6.12. The introduction of the indicator variables has resulted in an objective function that is quadratic in \( x \), but now the optimization is over \( x, z \) and \( \zeta \). The structure of the surrogate objective function is such that if the \( z \) and \( \zeta \) variables were known, then the objective function is quadratic in \( x \), with an analytical solution if there are no or linear equality constraints (such as mass balances). The approach is to use the Expectation Maximization algorithm to iteratively estimate the indicator variables and solve the quadratic objective function for \( x \).

6.3 Solution Using Expectation Maximization

Expectation Maximization (EM) has been applied in a wide variety applications as a means of optimizing a likelihood function. Originally proposed by Dempster et al. (1977) as a means of dealing with incomplete data sets, applications have expanded to include density estimation (Meng and Rubin, 1991; Specht, 1991; Redner and Walker, 1984), clustering analysis (Cheseman et al., 1988; Nowlan, 1991), expert systems (Jordan and Jacobs, 1994), and in analysis of synaptic transmission (Stricker and Redman, 1994). In each of the above applications, the optimization of the likelihood
function would be greatly simplified if a set of unknown values were known. In the case of dealing with incomplete data sets, maximizing the likelihood function to find the parameters of interest would be greatly simplified if the missing values of the data were known. The EM algorithm consists of two steps: the Expectation (E) step and the Maximization (M) step. In the E step, the set of unknown values that would simplify the optimization, called the missing data, is estimated by finding the expected value of the missing data using the model parameters found in the M step. In the M step, a new set of model parameters are estimated using the expected value of the missing data found in the E step. The algorithm alternates between the two steps until convergence. It is guaranteed that after each EM iteration the set of model parameters arrived at will be at least as likely as the set of model parameters arrived at in the previous iteration (Dempster et al., 1977). The method converges when the likelihood no longer increases in subsequent iterations. This monotonic increase of the likelihood function ensures that the result arrived at is the most likely result given the data available and the initial starting point chosen for the algorithm (i.e. it is a local optimum).

In the solution of the MLR problem, the maximization in equation 6.12 is greatly simplified by the introduction of indicator variables. The transformed objective function in equation 6.19 is quadratic in \( \mathbf{x} \), once the values of the indicator variables, \( z \) and \( \zeta \), are known. The expected value of \( z \) and \( \zeta \) are found using \( \hat{\mathbf{x}}^{(k)} \), which is the estimate of \( \mathbf{x} \) at the \( k \)th iteration, in the E step.

\[
\hat{z}^{(k)} \equiv E\{z|\hat{\mathbf{x}}^{(k)}, y\} = \sum_z zP\{z|\hat{\mathbf{x}}^{(k)}, y\} \tag{6.22}
\]

\[
\hat{\zeta}^{(k)} \equiv E\{\zeta|\hat{\mathbf{x}}^{(k)}, y\} = \sum_\zeta \zeta P\{\zeta|\hat{\mathbf{x}}^{(k)}, y\} \tag{6.23}
\]
The conditional expectations of the indicators are derived by taking the summations in equations 6.22 and 6.23 over all H possible combinations of z, and all \(2^n\) possible combinations of \(\zeta\):

\[
\hat{z}^{(k)} = \sum_z z P(z \mid \hat{x}^{(k)}, y) = \sum_z z \frac{P(z, \hat{\hat{x}}^{(k)} \mid y)}{P(\hat{x}^{(k)} \mid y)} = \sum_z z \frac{P(z, \hat{\hat{x}}^{(k)})}{P(\hat{x}^{(k)})} \quad (6.24)
\]

The last equality in equation 6.24 arises by assuming that \(P(y \mid x, z) = P(y \mid x)\). That is, information about which component of the prior distribution (\(P(x)\)) that \(x\) is drawn from (knowing \(z\)) does not affect the probability distribution of the error \(P(\delta) = P(y \mid x)\).

This is a result of the measurement errors being uncorrelated with the true value of \(x\).

The \(i^{th}\) component of \(\hat{z}\) is found by substituting equations 6.13 and 6.14 into equation 6.24 and noting that when \(\hat{z}_i = 0\), all terms in the numerator vanish, and when \(\hat{z}_i = 1\) only the \(i^{th}\) term in the numerator remains:

\[
\hat{z}^{(k)}_i = \frac{q_i \phi_i(P \hat{x}^{(k)}; m_i, Q_i)}{\sum_{h=1}^H q_h \phi_h(P \hat{x}^{(k)}; m_h, Q_h)} \quad (6.25)
\]

Note that \(\sum_{i=1}^H \hat{z}^{(k)}_i = 1\).

A similar derivation leads to the following expression for \(\hat{\zeta}^{(k)}\):  

\[
\hat{\zeta}^{(k)} = \sum_z z P(z \hat{x}^{(k)}, y) = \sum_z z \frac{P(\zeta, y \mid \hat{x}^{(k)})}{P(y \mid \hat{x}^{(k)})} \quad (6.26)
\]
The \( i \)th component of \( \hat{\zeta} \) is found by substituting equations 6.11 and 6.5 into equation 6.26 and noting that when \( \hat{\zeta}_i \) is 0 all terms in the numerator vanish and when \( \hat{\zeta}_i \) is 1 the \( i \)th term in the numerator is accompanied by a factor that cancels out all of the factors in the sum in the denominator except the \( i \)th one:

\[
\hat{\zeta}_i^{(k)} = \frac{p_i \phi(\hat{x}_i^{(k)}; y_i, b_i^2 \sigma_i^2)}{(1-p_i)\phi(\hat{x}_i^{(k)}; y_i, \sigma_i^2) + p_i \phi(\hat{x}_i^{(k)}; y_i, b_i^2 \sigma_i^2)} \tag{6.27}
\]

Equations 6.25 and 6.27 compose the E step of the EM algorithm. With \( \hat{\mathbf{Z}} \) and \( \hat{\zeta} \) known, the objective function in equation 6.19 is a simple quadratic in \( \mathbf{x} \), which has an analytical optimum at \( \mathbf{x} = \omega \) if there are no constraints. Thus \( \hat{\mathbf{x}}^{(k+1)} \) is calculated in the M step of the EM algorithm by:

\[
\hat{\mathbf{x}}^{(k+1)} = \mathbf{P}^T \left( \mathbf{S}^{(k)} \left[ \sum_{h=1}^{H} \hat{\mathbf{z}}_h^{(k)} \mathbf{Q}_h^{-1} \mathbf{m}_h + \Lambda^{-1(k)} \mathbf{y} \right] \right) \tag{6.28}
\]

where \( \mathbf{S} \) and \( \Lambda \) are as defined in equations 6.18 and 6.20 and are evaluated at \( \hat{\mathbf{Z}}^{(k)} \) and \( \hat{\zeta}^{(k)} \). The multiplication of the bracketed term, which is in the dimension of the linearly independent subspace, by \( \mathbf{P}^T \) ensures that the rectified values will lie on the plane of constraints from which \( \mathbf{P} \) was derived (see Chapter 5).

If \( \mathbf{P}(\mathbf{x}) \) is found in the full dimension of \( \mathbf{x} \), and there are known linear \( \mathbf{A} \) constraints of the form \( \mathbf{A} \mathbf{x} = \mathbf{c} \) that are imposed on the solution of the rectification problem (i.e. were not imposed in finding \( \mathbf{P}(\mathbf{x}) \), then equation 6.19 also has an analytical solution. The M-Step becomes:
$$\hat{x}^{(k+1)} = \omega^{(k)} - S^{(k)}A^T(AS^{(k)}A^T)^{-1}(A\omega^{(k)} - c)$$  \hspace{1cm} (6.29)$$

where $\Lambda^{(k)}$, $S^{(k)}$, and $\omega^{(k)}$ are found by evaluating equations 6.18, 6.20, and 6.21 respectively at $\hat{x}^{(k)}$, and with $P$ equal to the identity matrix. If there are linear constraints but no prior distribution of $x$, then the M-Step is still equation 6.29, but in the evaluation of $S$ and $\omega$ (equations 6.20 and 6.21) only the terms involving $\Lambda$ and $y$ are used.

The algorithm proceeds by alternating between the E step (equations 6.25 and 6.27) and the M step (equation 6.28 or 6.29 depending on existence of constraints) until convergence is reached. As guaranteed by the EM algorithm, the solution $x^{(k+1)}$ is as likely or more likely than the solution $\hat{x}^{(k)}$ for all $k$. This results in the most likely set of states $x$ being found for the given measurement vector and the starting point. The algorithm initial conditions are given in the next section.

The solution in equation 6.28 gives us some insight into the nature of the estimate of the true value of $x$. Equation 6.28 shows that the estimate of $x$ is a weighted average between the measurement, $\Lambda^{-1}y$, and the prior information of the process contained in the distribution of the states, $\sum_{h=1}^{H} z_h Q_{h^{-1}} m_h$. This can be clearly seen by writing equation 6.28 in terms of a weighting function:

$$\hat{x} = P^T Wy + P^T (I - W)\mu$$  \hspace{1cm} (6.30)$$

where

$$W = S(P\Lambda^{-1}P^T),$$  \hspace{1cm} (6.31)$$

$$\mu = [I - S(P\Lambda^{-1}P^T)]^{-1} S \left[ \sum_{h=1}^{H} z_h Q_{h^{-1}} m_h \right]$$  \hspace{1cm} (6.32)$$

162
with \( A \) and \( S \) calculated from equations 6.18 and 6.20 respectively. The weight matrix \( W \) is a function of the posterior probability that a gross error has occurred in a measurement \( \zeta \). The diagonal elements of \( PT^T W \) give the weight that the measurement of state \( i \) (\( y_i \)) is given in the estimation of state \( i \) (\( x_i \)). If \( \zeta_i \) is close to one then it is likely that a gross error has occurred in sensor \( i \), and consequently the \( w_{ii} \) element is small, giving little or no weight to the corrupted sensor in the estimation of \( x_i \). In this case the estimation of \( x_i \) is primarily through the use of the other uncorrupted measurements and the prior probability distribution of the states. Also for the corrupt sensor \( i \), indicated by \( \zeta_i \) being close to 1, the \( w_{ji} \) weights will be small as the corrupt measurement is given little or no weight in the estimation of the other states.

6.4 Initial Conditions

The best available initial guess for the true state of the plant is the measurement \( y \). The initial value of \( z \) is found by evaluating equation 6.25 at \( y \):

\[
\hat{z}^{(0)}_i = \frac{q_i \phi_i(Py; m_i, Q_i)}{\sum_{h=1}^{H} q_h \phi_h(Py; m_h, Q_h)}
\]

(6.33)

In determining the initial value for \( \zeta \) both the measurement and the information contained in \( P(x) \) are used. Evaluating \( \zeta \) at \( y \) in equation 6.27 would result in all \( \zeta^{(0)}_i \) being close to zero because setting \( \hat{x} = y \) indicates that there are no gross errors in the measurement. Using this starting value for \( \zeta \) biases the algorithm toward there being no gross errors present. The \( P(x) \) distribution holds information that can be exploited to avoid this bias. If the measurement contains only noise it is likely to lie in a probable
region of $P(x)$ and if a gross error is present then it is likely to lie in a zero or extremely low probable region of $P(x)$. Thus the best initial guess for $\zeta$ is found by finding the expected value of $\zeta$ given the measurement and the distribution $P(x)$,

$$
\zeta^{(0)} = E\{\zeta \mid y, z_h^{(0)}, m_h, Q_h\}
$$

(6.34)

Therefore,

$$
\zeta^{(0)}_i = \frac{p_i \phi(y_i; u_i, \nu_i^2 + b_i^2 \sigma_i^2)}{(1-p_i)\phi(y_i; u_i, \nu_i^2 + \sigma_i^2) + p_i \phi(y_i; u_i, \nu_i^2 + b_i^2 \sigma_i^2)}
$$

(6.35)

where $\nu_i^2$ is the $i$th element on the diagonal of $V$ which is the matrix:

$$
V = \sum_{h=1}^{H} z_h (P^T Q_h^{-1} P)
$$

(6.36)

and $u_i$ is the $i$th element of the vector $u$:

$$
u = \left[\sum_{h=1}^{H} z_h (P^T Q_h^{-1} P)\right]^{-1} \left[\sum_{h=1}^{H} z_h P^T Q_h^{-1} m_h\right]
$$

(6.37)

### 6.5 Finding the Global Optimum

The optimization problem in equation 6.12 is balance between likely adjustments to the measurements $P(\delta)$ and a likely rectified state $P(x)$, as conceptually shown in Figure 6.1. The cross represents the measurement point and the concentric circles around the cross represent the equal probability contours of $P(\delta)$ around the point, the highest probability of $P(\delta)$ being at the point. The elliptical contours conceptually represent a $P(x)$ distribution in the $x_1$-$x_2$ state space. The objective
Figure 6.1: Schematic representation of MLR problem formulation showing tradeoff between likely adjustments ($P(\delta)$) and likely rectified states ($P(x)$).
function is the product of the two contours. The nature of most probability functions (except those with hard boundaries such as uniform distributions) including a mixture of Gaussian distributions, is that they asymptotically approach zero probability as they move away from the bulk of the probability mass. The result is that in the tails of the probability function the value of the probability function is changing slowly. In Figure 6.1 the measurement lies in the tails of the probability distribution and because \( P(x) \) is changing very slowly in this region while \( P(\delta) \) goes through a maximum at the measurement point, the product of \( P(x) \) and \( P(\delta) \) can display local maxima. Similarly in the tails of \( P(\delta) \), where it is changing very slowly, \( P(x) \) is changing much more quickly and thus the product of \( P(\delta) \) and \( P(x) \) often shows a maximum in this region. There also can be maxima that appear between these two extreme. This results in local maxima appearing in the objective function in equation 6.12.

The presence of the local maxima can lead to a solution at a local maximum point rather than the global maximum, which would result in an erroneous solution. One approach to finding the global optimum is to begin the optimization at a number of randomly selected starting points and choose the objective function value that is the highest. This can require a large amount of computation as a complete optimization must be done for each starting point. To improve upon the random start re-solution we take a two part approach. First, we evaluate the likelihood of the local solution to determine if it is a likely global optimum. Second, if it is not likely that the solution reached is a global solution then a new starting point is selected from the prior distribution of rectified plant states \( P(x) \). The procedure is outlined in Figure 6.2.

The objective function in equation 6.12 is the likelihood of the rectified state given the measurement and the prior distribution of the states \( P(x) \). As such, the value of the objective function reflects how likely a particular solution is. A histogram of
Figure 6.2: Outline of solution algorithm. Measurement $y$ is used as first starting point and solution to the MLR found using EM. The likelihood (LH) of the solution is compared to the acceptable likelihood threshold ($\tau$) and if LH is less than $\tau$ then a new starting point is generated.

- Initialize $\hat{x}^{(0)} = y$
- New Starting Point Drawn From $P(x)$
- Solve MLR Using EM
  - LH < $\tau$
  - Maximum Iterations
- Rectify Based on Analytical Constraints Only
  - Yes
  - No
  - Yes
  - Valid Solution
final objective function values found in estimating $P(x)$ from a set of calibration data using the RSDE technique in Chapter 5 is formed, and lower objective function limit is set to included a certain percentage of the calibration cases (e.g. 97.5%). If the final objective function value is below this limit then it is concluded that the final solution reached is a local optimum and a new starting point is required.

The first starting point used is the measurement $y$, using the initial guesses for $z$ and $\zeta$ given in the previous section. If the likelihood of the solution obtained is below the threshold then the problem is resolved using a new initial guess and the EM optimization algorithm is started at the E-step (equation 6.25 and 6.26), instead of using the initialization equations for $z$ and $\zeta$ in the previous section. All valid solutions to the rectification problem will have to lie in $P(x)$ and thus instead of selecting a new starting point completely at random, candidate starting points are randomly drawn from $P(x)$. To avoid wasting computational time solving the optimization from starting points that are unlikely to yield a valid solution a number of starting points are drawn from $P(x)$. The likelihood of each candidate point being a solution to the optimization problem for the measurement $y$ is evaluated using equation 6.12. The candidate points are then ranked according to their likelihoods and the most likely point is chosen as the starting point. If the solution obtained from this starting value is below the objective function threshold then the next most likely starting point in the candidate set is used as the initial guess and the problem is re-solved. This continues until the solution reached is above the objective function threshold or all of the points in the candidate set have been used. At this point either another candidate set of starting points can be drawn from $P(x)$ or the method can stop and no valid rectified state is obtained. If no valid rectified state is obtained it is likely that the true state of the plant that generated the measurement lies in the extreme tails of the $P(x)$ distribution, and thus the true plant
state itself is very unlikely. That is, that the $P(x)$ plant model is near its limit of applicability and thus cannot be applied reliably to rectify the data. In this case the data can still be reconciled to the known analytical model constraints, although measurements that do not appear in the analytical model will not be reconciled at all.

Computational efficiency is also improved by having a pre-selected candidate set of starting points instead of generating a candidate set from $P(x)$ when required. A good choice for this candidate set are the rectified values obtained from the RSDE procedure in Chapter 5, applied to a calibration set of historical plant data. In finding the likelihood of a particular candidate starting point $P(x)$ and $P(\delta)$ are required. $P(\delta)$ depends on the particular measurement being rectified, but a starting point will have the same $P(x)$ for all measurements. The $P(x)$ for each candidate starting point is calculated off-line as part of the $P(x)$ estimation procedure, and thus only $P(\delta)$ for each starting point needs to be calculated for a particular measurement to find the likelihood of that starting point. This greatly reduces the number of function evaluations required to find and rank the likelihoods of the candidate starting points.

6.7 Examples

In this section, two chemical engineering examples are presented to demonstrate data rectification by the MLR technique and EM solution algorithm. First the EM solution algorithm is applied without constraints (M-step given by equation 6.27) to rectify the data from a heat exchanger network (Tjoa and Biegler, 1991; Johnston and Kramer, 1995). The EM solution algorithm is then applied with linear mass balance constraints (M-step from equation 6.29) to a flow network taken from Mah (1987). In all cases, the performance criterion used to gauge the efficacy of the rectification is the mean squared error:
$$\text{MSE} = \frac{1}{N \cdot K} \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{(\hat{x}_{n,k} - x_{n,k})^2}{\sigma_n^2}$$ (6.38)

where $N$ is the dimension of the measurement vector, $K$ is the number of measurement vectors used to calculate the MSE, $\hat{x}$ is the rectified state, $x$ is the true plant state, and $\sigma_n$ is the standard deviation of the Gaussian noise on sensor $n$. The MSE is structured such that if there are no sensor failures (i.e. the measurements only contain random Gaussian noise with standard deviations $\sigma_n$) then the MSE = 1.0, for $\hat{x} = y$ (see Chapter 4). If there are gross errors, the MSE can be greater than 1.0. If the rectification is perfect (i.e. the true state of the plant was found for every measurement vector) then the MSE = 0.0. Thus the goal is to drive the MSE towards zero. In all examples $P(\delta_t)$ is characterized by a bivariate Gaussian distribution (equation 6.4) with $p_t = 0.10$, and $b_t = 20$.

### 6.7.1 Heat Exchanger Example

Figure 6.3 shows a heat exchanger network that was modified from Tjoa and Biegler (1991). Process stream A is heated by process stream B in two exchangers, and with utility stream C and D in two other exchangers. The flow of stream D is adjusted so that the outlet temperature of stream A, $T_{A8}$, will be 615 C. The plant operates at low, medium, and high capacity levels where process streams A and B are at low, medium, and high flow rates. The temperatures of all inlet streams are at constant values, with some process noise. The operating conditions are summarized in Table 6.1. Using these operating conditions, a training set of 120 points was generated, from which $P(x)$ was estimated using the RSDE technique in Chapter 5. The noise added to the training and test sets had a standard deviation of 0.75 for the temperature
Figure 6.3: Heat exchanger network example.
Table 6.1: Operating condition for Heat Exchanger network example.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Plant Operating Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low Flow</td>
</tr>
<tr>
<td></td>
<td>Probability = 0.3</td>
</tr>
<tr>
<td>FA1</td>
<td>N(800, 5)</td>
</tr>
<tr>
<td>FB1</td>
<td>N(200, 5)</td>
</tr>
<tr>
<td>FC1</td>
<td>N(240, 5)</td>
</tr>
<tr>
<td>TA1</td>
<td>N(450, 5)</td>
</tr>
<tr>
<td>TB1</td>
<td>N(620, 5)</td>
</tr>
<tr>
<td>TC1</td>
<td>N(670, 5)</td>
</tr>
<tr>
<td>TD1</td>
<td>N(690, 5)</td>
</tr>
<tr>
<td>FA3/FA2</td>
<td>0.42</td>
</tr>
</tbody>
</table>
measurements, and standard deviations of 4.0, 2.0, 3.0 and 3.0 for flows FA1, FB1, FC1, and FD1 respectively. In this case the density estimator that best fit the data was an elliptical basis function estimator with six units and overlap of 1. Two separate data sets were created for performance testing. One set had only Gaussian noise in the sensors and the other set simulated a failure in sensor TA4, by adding a gross error in the range of -30 to +30 C.

The MLR was done without any constraints because the instrumentation on the network does not allow for any mass or energy balances to be written. Again the MLR shows its power in estimating the state of the plant, without any prior constraints. The MSE was reduced from 1.0 to 0.64 and from 10.48 to .73 for the two data sets. Even with failed sensors the MLR and EM solution technique were able to find the rectified values and lower the MSE below the random noise level of 1.0. For the test set with only noise, approximately 98% of the points reached a viable solution using the measurement as the initial condition, 1% reached a viable solution with less than 10 new starting points, and 1% of the points were unresolved. For the test data set containing gross errors approximately 93% of the test data reached a viable solution starting at the measurement. Approximately 4% of the test data required less than 15 starting points to reach a viable solution, and a further 1% of the points reached a viable solution starting at more than 15 starting points. No viable solution was found for the remaining 3% of the test data. The measurements that did not result in a viable solution all had true plant states that were in the extreme tails of the P(x) distribution, and thus the true state of the plant itself had a low likelihood. This can be interpreted as the process operating at the very edge of its normal regime and the likelihood threshold can be thought of as a distinguishing between the normal operating regime of the plant and novel operating states.

173
6.7.2 Flow Network Example

Figure 6.4 shows the flow network from Mah (1987). The total flow of each stream is measured. Data were generated by sampling $F_1$ from $U(15,40)$, calculating the remaining flows, and generating the $y$ values by adding random, zero-centered, Gaussian noise with standard deviations 1.0, 4.0, 4.0, 3.0, and 1.0 respectively, for the five sensors. A training data set of 100 points was generated to estimate $P(x)$ using the Recursive State Density Estimation technique in Chapter 5. $P(x)$ was fit in five dimensions and the known mass balance constraints were analytically enforced in the MLR solution. A separate 1000-point test set was also generated for the performance studies. A multivariate Gaussian distribution was used for $P(x)$, which, in spite of the mismatch from the actual distribution, was found to be the best estimator for this data set using the cross-validation technique in Chapter 3. Another 1000-point data set was also created, and in this set a failure of the $F_3$ sensor was simulated by adding a miscalibration bias ranging from -75% to +75% of the nominal flowrate.

The three known mass balance constraints were applied to the MLR problem, and the linearly constrained M step (equation 6.29) was used. For the data set containing only noise the MSE was reduced from 1.0 to 0.25. In approximately 99% of the cases a viable solution was obtained using the measurement as the starting point. No viable solution was found for the remaining data points. For the data set with a gross error in the $F_3$ sensor the MSE was reduced from 20.4 to .36 by the MLR technique. For this data set approximately 98% of the points were successfully rectified starting from the measurement and no viable solution was found for the remaining points. The states that generated the measurements that did not reach viable solutions all corresponded to states that were in the extreme tails of the $P(x)$ distribution.
Figure 6.4: Flow network used in data rectification.
Previous data rectification schemes are only able to reconcile data to a set of known constraints and cannot incorporate the knowledge contained in the historical database of the plant. To provide a comparison to conventional techniques, the flow data was also reconciled using the global test (Tamhane and Mah, 1985) in conjunction with serial elimination to identify the gross errors (Nogita, 1972). This method reduced the MSE from 1.0 to 0.43 for the data set containing only noise and reduced the MSE from 20.4 to 0.86 for the data set containing gross errors. These are both larger than the MSE obtained using the MLR. The MLR was also applied to the data set without using the information in P(x). This resulted in MSE values of 0.40 and 0.60 for the noise only and gross error containing data, respectively. Approximately 95% of the points in the data set containing the gross error reached a viable solution using the measurement as the starting point. The other 5% of the points reached a viable solution using less than 10 starting points. The results of the MLR without P(x) are somewhat better than the conventional method, particularly the gross error data set, but is inferior to the performance of the MLR using P(x). The results are summarized in Table 6.2.

6.8 Algorithm Efficiency

The proposed solution algorithm offers increased speed over conventional methods of solving the nonlinear optimization problem posed in equation 6.12. For comparison purposes, the MATLAB Optimization Toolbox (Mathworks, 1995) implementation of the BFGS Quasi-Newton method with a mixed quadratic and cubic line search procedure was used to solve the nonlinear optimization problem in equation 6.12, and compared to the time for the solution obtained using the proposed EM implementation. The BFGS method was supplied with a function to analytically calculate the partial derivatives of the objective function. We recorded the average time
Table 6.2: MSE results for the flow network.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Raw Data</th>
<th>MLR with P(x)</th>
<th>MLR without P(x)</th>
<th>Weighted Least Squares with Serial Elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise Only</td>
<td>1.00</td>
<td>0.25</td>
<td>0.40</td>
<td>0.43</td>
</tr>
<tr>
<td>Gross Error in F₃</td>
<td>20.4</td>
<td>0.36</td>
<td>0.60</td>
<td>0.86</td>
</tr>
</tbody>
</table>
to reach a solution starting at the measurement for all the points that reached a viable solution using this starting point. Thus the results are the average time for a single optimization. The results showed that each of the methods arrived at the same solution for each point, and thus those points that required more than one starting value would require the same number of starting values in either method, since both are local optimizers. Although the MATLAB implementation of the BFGS method may not be the fastest implementation or the fastest method, it gives a benchmark to which other methods can be compared. The EM implementation was also coded in MATLAB and thus the basic operations used by the two algorithms should have the same speed (function evaluation, matrix inversion, etc.). For the heat exchanger network data the BFGS method took an average of 3.4 seconds per point over the 1000 point data set, while the proposed EM solution method took an average of only 0.2 seconds per point over the same 1000 point data set. In this case the EM implementation is approximately 17 times faster than the MATLAB BFGS method. The results are summarized in Table 6.3.

For the flow data case in which \( P(x) \) was estimated, the optimization in equation 6.12 is constrained. The MATLAB Optimization Toolbox implementation of the constrained BFGS Quasi-Newton method (Mathworks, 1995) took an average of 2.1 seconds per point over the 1000 point data set to rectify the flow data. The proposed EM implementation took an average of 0.1 seconds per point over the same 1000 point data set. Thus the EM approach is approximately 21 times faster than the constrained BFGS method in this case. Both methods arrived at the same rectified states. If the \( P(x) \) information is ignored, and MLR is applied to the data using the EM solution technique, each point takes 0.1 seconds to rectify. The MLR method only took 0.03 seconds per point longer than the conventional linear reconciliation technique but gave...
Table 6.3: Average time over a 1000 point test set to rectify a one measurement vector with methods coded in MATLAB run on a Sun Sparc Station 10.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of variables</th>
<th>Average Time per point (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained MLR Using EM</td>
<td>13</td>
<td>0.2</td>
</tr>
<tr>
<td>BFGS Quasi-Newton</td>
<td>13</td>
<td>3.4</td>
</tr>
<tr>
<td>Constrained MLR Using EM</td>
<td>5</td>
<td>0.1</td>
</tr>
<tr>
<td>Constrained BFGS Quasi-Newton</td>
<td>5</td>
<td>2.1</td>
</tr>
</tbody>
</table>
significantly better rectified values. The algorithm speed results are summarized in Table 6.2. With only a modest computational time increase we are able to utilize all the benefits of the MLR formulation, including rectifying measurements for which there are no analytical constraints.

6.9 Conclusions

The complex form of the Maximum Likelihood Rectification (MLR) objective function was reduced to a quadratic form by introducing indicator variables. The resulting surrogate objective function is quadratic in the plant states and has an analytical solution in the absence of constraints or when the constraints are linear, if the values of the indicator variables are known. The Expectation Maximization (EM) algorithm was applied to iteratively estimate the indicator variables and estimate the states. The solution was shown to be 17 to 21 times faster than the conventional BFGS Quasi-Newton non-linear optimization method on the examples with 13 and 5 sensors respectively. The MLR technique was also successful in simultaneously removing both noise and gross errors from measurement data for two example processes. The proposed optimization technique is also able to diagnose the quality of the solution obtained by calculating the solution likelihood and comparing it to a likelihood threshold. This eliminates unlikely solutions which correspond to local maxima. A resampling technique based on the probability distribution of the process is proposed for finding new starting points. This method led to the exclusion of local maxima solutions, and resulted in most of the data (>97%) to be rectified to a likely solution.
Chapter 7:
Case Study - Cycloheaxane Process Plant

7.1 Introduction

To assess the performance of a data rectification scheme simulated data must be used. The true states of an operating chemical plant are never known, and thus application of data rectification schemes to real plant data cannot be quantified for accuracy. The benefits of data rectification to an operating plant would only be seen over long term improved plant operation, but simulated data can be used quickly assess the accuracy of data rectification schemes. Process simulation can be used to generate a set of true plant states, and then measurements can be simulated by adding errors to the true process state data. Data rectification schemes can be applied to the simulated measurements, and the efficacy of a method for removing errors can be measured by how well the true plant states are estimated. The methodologies for estimating the state probability density function from noisy and corrupted data, and application of this distribution to the rectification of process data are applied to a simulated chemical process with 30 measurements.
7.2 Process Description

The process chosen for the case study is the production of Cyclohexane from Benzene. This process is used as a demonstration process for the AspenPlus Process Simulator, and the example code was used to simulate the plant (AspenTech, 1988). AspenPlus release 9.2-1 (AspenTech, 1995) was used to simulate the process. A process diagram is shown in figure 7.1. The feed Benzene stream is taken as pure Benzene, but the Hydrogen stream is contaminated with methane and nitrogen. The benzene and hydrogen feed streams are mixed with a primarily hydrogen (stream 6) recycle stream and a primarily benzene (stream 7) recycle stream. The mixture is preheated then fed to a reactor that is kept isothermal by cooling with boiler feed water. In the reactor, equilibrium is reached and the reactor effluent is cooled and flashed in a phase separator, which sends most of the hydrogen overhead to be recycled. Part of the separator overhead stream is purged to prevent buildup of nitrogen in the recycle loop. The liquid from the separator contains the cyclohexane product and unreacted benzene. Part of this stream is recycled to the front end of the reactor and the remaining portion is sent to a 15 ideal stage distillation column. The products of the column are the vapor light ends overhead (stream 10) and the cyclohexane product in the bottoms (stream 11). The column is operated such that the purity of cyclohexane in the bottoms product is greater than 99.5%. The utility side of the reactor preheater and the column reboiler are supplied from a common steam header, and the utility side of the reactor effluent cooler and the column condenser are supplied by a common cooling water header.

Measurements are made as indicated in figure 7.1. The simulation of the measurements is given in the next section. The "F" sensors measure total flow in lb/min, the "T" sensors measure temperature in degrees Fahrenheit and the "V" sensors indicate the valve position, in % open, of the utility stream supply control valve.
7.3 Process Simulation

The AspenPlus process simulator was used to simulate the states of the plant. From the states the measurements were simulated by adding noise and in some cases gross errors to the state data. The base case Aspen input file for generating the process states is in Appendix B, and it gives the specifics of each unit operation in the process flow sheet. The Aspen simulator solves rigorous mass and energy balances for the flowsheet to calculate the flows, temperatures, pressures, and compositions of all streams as well as the duties of all heat exchangers. The utility flows and subsequent valve positions were calculated from the heat duties. For each of the heat exchangers the valve position was related to the flow of the utility stream by

\[ F = k \cdot VP \cdot \sqrt{\Delta P} \]  \hspace{1cm} (7.1)

where \( \Delta P \) is the pressure drop in psi, \( VP \) is the valve position in percent open, and \( k \) is a constant such that the maximum expected utility flowrate is obtained at 90% open. The MATLAB files that calculate the valve position are in Appendix B.

7.4 Process Operating Conditions

Chemical processing plants do not operate at a single operating point but rather in certain regions of operation. This was simulated by varying a number of input parameters to the process simulator, to generate a set of plant state data. The variables varied were the temperature and flowrate of the feed streams, the purity of the hydrogen feed stream, the reactor inlet temperature and pressure, the separator temperature and pressure, hydrogen and benzene recycle rates, column reflux ratio and pressure, cooling
water header pressure and temperature, and steam header pressure. The process was assumed to operate at a low and high flow rate with probability 0.4 and 0.6 respectively. Operating conditions at each level of operation for the input parameters to the process simulator are summarized in Table 7.1.

Using these distributions 1500 sets of input data were generated and the AspenPlus process simulator was used to calculate the process states for each set. 500 of these data points were used as a training set, and 1000 of these points were used as a testing set. The states, corresponding to the 30 measurements shown in figure 7.1, were taken from the AspenPlus output. Measurements from these states were generated by adding zero-centered Gaussian noise to the states. All temperature measurements had a noise standard deviation of 0.5 degrees F, pressure sensors had a noise standard deviation of 2 psi, and the flow sensors had noise standard deviations of 2.0, 4.0, 5.0, 4.0, 4.0, 5.0, 5.0, 1.0, 1.0, 4.0, and 10.0 lb/min for streams 1 to 11 respectively.

7.5 Calibration of P(x)

The Recursive State Density Estimation technique described in Chapter 5 was used to find P(x) from the measurement data. There are 5 independent mass balances that can be written for the process:

\[
\begin{align*}
0 &= F_1 + F_2 - F_3 + F_6 + F_7 \\
0 &= F_3 - F_4 - F_5 \\
0 &= F_4 - F_6 - F_9 \\
0 &= F_5 - F_7 - F_8 \\
0 &= F_8 - F_{10} - F_{11}
\end{align*}
\]
Table 7.1: Plant Operating Conditions at Low and High productions rates. $N(m, s)$ denotes a Normal distribution centered at $m$ with standard deviation $s$, and $U(l, u)$ denotes a Uniform distribution between lower bound $l$ and upper bound $u$.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Units</th>
<th>Low Production Rate (Probability = 0.4)</th>
<th>High Production Rate (Probability = 0.6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benzene Feed Flow (FBz)</td>
<td>lbmol/hr</td>
<td>$N(100, 10)$</td>
<td>$N(180, 10)$</td>
</tr>
<tr>
<td>Hydrogen Feed Flow</td>
<td>lbmol/hr</td>
<td>$FBz\cdot U(3.05, 3.18)$</td>
<td>$FBz\cdot U(3.05, 3.18)$</td>
</tr>
<tr>
<td>Methane Impurity Feed Flow</td>
<td>lbmol/hr</td>
<td>$N(10, 2)$</td>
<td>$N(18, 2)$</td>
</tr>
<tr>
<td>Benzene Inlet Temperature</td>
<td>F</td>
<td>$N(100, 4)$</td>
<td>$N(100, 4)$</td>
</tr>
<tr>
<td>Hydrogen Inlet Temperature</td>
<td>F</td>
<td>$N(120, 4)$</td>
<td>$N(120, 4)$</td>
</tr>
<tr>
<td>Reactor Inlet Temperature</td>
<td>F</td>
<td>$N(400, 5)$</td>
<td>$N(400, 5)$</td>
</tr>
<tr>
<td>Reactor Inlet Pressure</td>
<td>psi</td>
<td>$N(330, 7)$</td>
<td>$N(330, 7)$</td>
</tr>
<tr>
<td>Reactor Pressure Drop</td>
<td>psi</td>
<td>$5 + \ln(0, 3)l$</td>
<td>$10 + \ln(0, 3)l$</td>
</tr>
<tr>
<td>Separator Temperature</td>
<td>F</td>
<td>$N(120, 4)$</td>
<td>$N(120, 4)$</td>
</tr>
<tr>
<td>Separator Pressure Drop</td>
<td>psi</td>
<td>$5 + \ln(0, 3)l$</td>
<td>$8 + \ln(0, 3)l$</td>
</tr>
<tr>
<td>Purge Split Fraction</td>
<td></td>
<td>$N(.13, .01)$</td>
<td>$N(.08, .01)$</td>
</tr>
<tr>
<td>Benzene Recycle Split Fraction</td>
<td></td>
<td>$N(0.4, .02)$</td>
<td>$N(0.3, .02)$</td>
</tr>
<tr>
<td>Column Pressure</td>
<td>psi</td>
<td>$N(200, 5)$</td>
<td>$N(200, 5)$</td>
</tr>
<tr>
<td>Column Reflux Ratio</td>
<td></td>
<td>$1.2 + \ln(0, 0.08)l$</td>
<td>$1.2 + \ln(0, 0.08)l$</td>
</tr>
<tr>
<td>Steam Header Pressure</td>
<td>psi</td>
<td>$N(600, 10)$</td>
<td>$N(560, 10)$</td>
</tr>
<tr>
<td>Cooling Water Temperature</td>
<td>F</td>
<td>$N(50, 4)$</td>
<td>$N(50, 4)$</td>
</tr>
<tr>
<td>Cooling Water Header Pressure</td>
<td>psi</td>
<td>$N(170, 5)$</td>
<td>$N(200, 5)$</td>
</tr>
</tbody>
</table>
Orthornormal basis vectors that span the null space of the mass balance equations were found as in Chapter 5, and these were used to construct the projection matrix $P$. The measured variables that do not appear in the mass balances, the pressures, temperatures and valve positions, are assigned an identity entry in the $P$ matrix, and thus projection by $P$ does not change their value. The probability distribution of the states $P(x)$ was fit in the 25 dimensional subspace that results from the projection of the 30 dimensional measurement vector by $P$. The cross validation methodology discussed in Chapter 3 indicated that an EBF with 12 basis functions was the density estimator that gave the best fit.

For all sensors the probability of a gross error was 0.05 and the value of $b$ in the bivariate Gaussian error distribution was 20. The log likelihood value calculated as in Chapter 6 for each RSDE iteration is shown in figure 7.2. As expected the log likelihood increases at each RSDE iteration until convergence is reached. The MSE monotonically decreases from its starting value of 1 to 0.48 during the RSDE algorithm, as shown in figure 7.3. Thus the estimated $P(x)$ is capturing the correlation in the data and exploiting these correlations to rectify the data.

7.6 MLR Performance

A test set of 1000 data points was generated as described in section 7.4. The MLR methodology described in Chapter 4 was applied with the solution method described in Chapter 6, to the test measurement data with only noise and with simulated gross errors. The gross errors for each sensor were generated by adding a bias in the range of ± 30 times the sensor noise standard deviation to the state value. Table 7.2 gives that starting and final MSE values for each data set. The MLR performed well on the data set containing only noise, reducing the MSE from 1.0 to 0.47. The MLR also
Figure 7.2  Monotonic increase of log likelihood at each RSDE iteration
Figure 7.3  Monotonic decrease of MSE at each RSDE iteration
Table 7.2 MSE results for 1000 points test data set using the MLR method and the Conventional method of the global test and serial elimination.

<table>
<thead>
<tr>
<th>Sensor with Gross Error</th>
<th>Starting MSE</th>
<th>MLR Final MSE</th>
<th>Global Test and Serial Elimination Final MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>1.00</td>
<td>0.47</td>
<td>0.82</td>
</tr>
<tr>
<td>F1</td>
<td>4.45</td>
<td>0.57</td>
<td>1.43</td>
</tr>
<tr>
<td>F2</td>
<td>4.21</td>
<td>0.64</td>
<td>15.34</td>
</tr>
<tr>
<td>F3</td>
<td>4.43</td>
<td>0.65</td>
<td>0.97</td>
</tr>
<tr>
<td>F4</td>
<td>4.39</td>
<td>0.63</td>
<td>0.99</td>
</tr>
<tr>
<td>F5</td>
<td>4.22</td>
<td>0.71</td>
<td>1.02</td>
</tr>
<tr>
<td>F6</td>
<td>4.39</td>
<td>0.62</td>
<td>1.02</td>
</tr>
<tr>
<td>F7</td>
<td>4.06</td>
<td>0.64</td>
<td>0.97</td>
</tr>
<tr>
<td>F8</td>
<td>4.27</td>
<td>0.79</td>
<td>2.77</td>
</tr>
<tr>
<td>F9</td>
<td>4.39</td>
<td>0.57</td>
<td>3.06</td>
</tr>
<tr>
<td>F10</td>
<td>4.09</td>
<td>0.58</td>
<td>1.51</td>
</tr>
<tr>
<td>F11</td>
<td>4.62</td>
<td>0.57</td>
<td>23.59</td>
</tr>
<tr>
<td>P3</td>
<td>4.36</td>
<td>0.68</td>
<td>4.00</td>
</tr>
<tr>
<td>P Reactor</td>
<td>4.49</td>
<td>0.63</td>
<td>4.62</td>
</tr>
<tr>
<td>P Separator</td>
<td>4.18</td>
<td>0.67</td>
<td>4.49</td>
</tr>
<tr>
<td>T1</td>
<td>4.19</td>
<td>1.79</td>
<td>4.08</td>
</tr>
<tr>
<td>T2</td>
<td>4.83</td>
<td>1.88</td>
<td>3.96</td>
</tr>
<tr>
<td>T3</td>
<td>4.20</td>
<td>0.55</td>
<td>4.26</td>
</tr>
<tr>
<td>T Reactor</td>
<td>4.37</td>
<td>0.55</td>
<td>4.20</td>
</tr>
<tr>
<td>T Separator</td>
<td>4.19</td>
<td>0.56</td>
<td>4.38</td>
</tr>
<tr>
<td>T9</td>
<td>4.32</td>
<td>0.55</td>
<td>4.16</td>
</tr>
<tr>
<td>T10</td>
<td>4.43</td>
<td>0.68</td>
<td>4.23</td>
</tr>
<tr>
<td>T11</td>
<td>4.83</td>
<td>0.62</td>
<td>3.91</td>
</tr>
<tr>
<td>TC2</td>
<td>4.69</td>
<td>0.71</td>
<td>4.47</td>
</tr>
<tr>
<td>TC9</td>
<td>4.19</td>
<td>0.62</td>
<td>3.89</td>
</tr>
<tr>
<td>TC12</td>
<td>4.61</td>
<td>0.63</td>
<td>4.04</td>
</tr>
</tbody>
</table>
effectively rejected the gross errors and reduced the MSE from a starting value of approximately 4.5 to below 1, which corresponds to the noise level, for all but two sensors. Gross errors in the inlet temperature of the hydrogen and benzene feed streams were not all effectively rejected, having final MSE values of 1.79 and 1.88 respectively. The median squared error for these test sets was 0.99 and 0.96 indicating that over half of the measurements with the gross errors were effectively removed. This shows while there is some correlation in the data that involves the two inlet temperatures, it is not strong to conclusively reject all gross errors in these measurements that are encountered. Figures 7.4 to 7.8 show the removal of the gross errors from the measurement data set for 5 different measurements: cyclohexane product flow, reactor pressure, reactor temperature, separator temperature, and column feed tray temperature. Overall the MLR performed well in removing both noise and gross errors from the measurement.

For comparison the data was also reconciled to the five mass balances, given in section 7.5, using the global test and serial elimination for gross error detection and identification. Table 7.2 also shows the final MSE for the data reconciled in this manner. The conventional method was unable to remove gross errors from streams 2 or 11. Investigation into this showed that the serial elimination procedure was incorrectly identifying a gross error in streams 1 and 10 respectively for these cases. In only 3 cases was the conventional methodology able to lower the MSE below the noise level of 1. When gross errors were present in variables other than the flows the conventional methodology was obviously unable to detect and remove the gross errors. In all case the MLR outperformed the conventional method.

Table 7.3 shows the starting and final MSE values for the test data set when the MSE is calculated using only the 11 flow measurements. Again the MLR outperforms
Figure 7.4: Plot of the measurement (•) and the final rectified value (○) of the cyclohexane product flowrate ($F_{11}$) against the true value of $F_{11}$ for a data set with gross errors in $F_{11}$. Perfect rectified values would lie on the rectified value equals true value line shown.
Figure 7.5: Plot of the measurement (+) and the final rectified value (○) of the reactor pressure against the true value of the reactor pressure for a data set with gross errors in the reactor pressure. Perfectly rectified values lie on the rectified value equals true value line shown.
Figure 7.6: Plot of the measurement (+) and the final rectified value (o) of the reactor temperature against the true value of the reactor temperature for a data set with gross errors in the reactor temperature. Perfectly rectified values lie on the rectified value equals true value line shown.
Figure 7.7: Plot of the measurement (+) and the final rectified value (o) of the separator temperature against the true value of the separator temperature for a data set with gross errors in the separator temperature. Perfectly rectified values lie on the rectified value equals true value line shown.
Figure 7.8: Plot of the measurement (+) and the final rectified value (o) of the column feed tray temperature against the true value of the column feed tray temperature for a data set with gross errors in the column feed tray temperature. Perfectly rectified values lie on the rectified value equals true value line shown.
Table 7.3  MSE results calculated only on the flow measurements for 1000 points test data set using the MLR method and the Conventional method of the global test and serial elimination.

<table>
<thead>
<tr>
<th>Sensor with Gross Error</th>
<th>Starting MSE</th>
<th>MLR Final MSE</th>
<th>Global Test and Serial Elimination Final MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>1.0</td>
<td>0.39</td>
<td>0.60</td>
</tr>
<tr>
<td>F1</td>
<td>9.87</td>
<td>0.49</td>
<td>2.16</td>
</tr>
<tr>
<td>F2</td>
<td>9.06</td>
<td>0.77</td>
<td>40.10</td>
</tr>
<tr>
<td>F3</td>
<td>11.32</td>
<td>0.65</td>
<td>0.9</td>
</tr>
<tr>
<td>F4</td>
<td>9.11</td>
<td>0.76</td>
<td>0.98</td>
</tr>
<tr>
<td>F5</td>
<td>10.24</td>
<td>0.84</td>
<td>1.04</td>
</tr>
<tr>
<td>F6</td>
<td>10.90</td>
<td>0.66</td>
<td>1.04</td>
</tr>
<tr>
<td>F7</td>
<td>10.17</td>
<td>0.61</td>
<td>0.92</td>
</tr>
<tr>
<td>F8</td>
<td>11.08</td>
<td>0.92</td>
<td>5.83</td>
</tr>
<tr>
<td>F9</td>
<td>10.25</td>
<td>0.64</td>
<td>6.63</td>
</tr>
<tr>
<td>F10</td>
<td>10.32</td>
<td>0.52</td>
<td>2.39</td>
</tr>
<tr>
<td>F11</td>
<td>10.59</td>
<td>0.50</td>
<td>62.61</td>
</tr>
</tbody>
</table>
the conventional method for all data sets. The MLR effectively rejects the gross errors in each data set resulting in final MSE below the noise level of 1. The conventional method is unable to correctly identify and remove the gross errors from the data sets, and the MSE is reduced below 1 in only 3 cases.

7.7 Identification of Sensor Miscalibration

A 100 point test data set was created and a -5 degree F bias was added to the reactor temperature sensor for all points in the data set. The data set was rectified using the MLR methodology. Figure 7.9 plots the difference between the measurement and the rectified temperature. The MLR rectifies the data identifying a bias in the reactor temperature between -4 and -6 degrees F for 99% of the data set. In an operating plant this would indicate to the operator that the reactor temperature sensor is consistently reading low and that the thermocouple should be re-calibrated to avoid a process upset or a safety incident.

7.8 Conclusions

The methodology developed in Chapters 3 to 6 of this thesis were employed to successfully estimate the probability distribution of the plant states, and successfully rectify future data from a simulated chemical plant with 30 measurements. Improvements over conventional data rectification schemes was shown, especially when gross errors are present in the data. Application of the MLR method to estimating systematic biases in measurement data was also shown by the identification of a -5 degree F bias in the reactor temperature sensor.
Figure 7.9  Bias in reactor temperature estimated by MLR for data set with a -5 C bias in the reactor temperature.
Chapter 8: Conclusions and Future Directions

8.1 Conclusions

This research has demonstrated a new approach to data rectification that is applicable to more systems, and achieves better results, than current data rectification technology. The work is based on the maximum likelihood principle, and poses the data rectification problem in a probabilistic framework by maximizing the probability of the true plant states given the measurements. This statement is the maximization of the probability of what we want to know, the true plant states, given what we observe, the error containing measurements. Applying Bayes theorem to the problem formulation results in the Maximum Likelihood Rectification (MLR) objective function, which is the product of the probability of the rectified plant state and the probability of the adjustments made to the measurement to reach the rectified state. The most likely adjustment to the measurement is no adjustment, but this rectified state may not be likely. Thus there is a tradeoff between likely rectified states and the adjustments made to the measurements. The resulting maximum likely rectified state yields the best estimate of the true plant state, given the measurement and prior information about the process.
The prior information about the process states was effectively captured in the probability distribution of the plant states, using the Recursive State Density Estimation algorithm. This algorithm used the principles of Expectation Maximization (EM), to link probability density function estimation and data rectification, to obtain the probability distribution of the true process states from a set of noisy and corrupted measurements. The data for most chemical processes do not follow parametric forms of probability distributions (e.g. Gaussian), and thus the Elliptical Basis Function (EBF) density estimator was developed. This is a nonparametric density estimation technique that was shown to converge to any arbitrary distribution in the limit of infinite data. The EBF estimator was demonstrated to adequately approximate a number of distributions for finite data sets.

The developed MLR framework was also able to reject gross-errors in rectifying the data, without the requirement of a gross error detection step. This was achieved by implementing a robust sensor model, that probabilistically accounted for the possible distribution of gross-errors in the sensor. The robust sensor model, and the prior information contained in the estimate of the plant state probability distribution, resulted in bounded influence curves for the rectification. Thus, as the error in a sensor becomes large, its influence on the solution tends to zero, and it is effectively eliminated from the measurement set. Traditional data rectification approaches, using the weighted least squares objective function, have unbounded influence curves. Thus, as the error in a sensor becomes large it exerts a larger and larger influence on the solution. To avoid this problem auxiliary techniques to remove the gross-errors are applied to the data before applying conventional data reconciliation. This data pre-processing is avoided in the MLR framework.
The MLR and RSDE methodologies were able to effectively remove both noise and gross errors from the 5, 13 and 30 dimensional chemical engineering examples that were examined. In all cases the MLR outperformed conventional data reconciliation techniques. The MLR framework is also capable of solving a new class of problem, one in which no analytical constraints are known. The MLR not only captures the correlation among all variables, its framework allows for the incorporation of any known process models. This is applicable to the common industrial situation where a partial model of the plant is available.

The MLR objective function, with the probability distribution of the plant states described by an EBF and using a robust sensor model, leads to a computationally complex optimization problem. The problem was successfully decomposed into a quadratic objective function by the introduction of two sets of indicator variables. The EM algorithm is used to iteratively estimate the values of the indicator variables and then estimate the rectified state solving the much simpler surrogate objective function. Results showed that the EM solution methodology was approximately 20 times faster than conventional nonlinear programming techniques, without the loss of accuracy.

8.2 Future Directions

The current research approach is one that has many applications to both data rectification and other areas of chemical and systems engineering.

8.2.1 Dynamic System Data Rectification

The developed MLR framework is for steady-state systems, but the principles of posing the data rectification problem in terms of maximizing the plant states given the measurements, can be applied to dynamic systems. In dynamic systems, there are added complications of fitting the probability density function of the trajectory of the
plant states over time. For batch processes, this could be achieved by treating time as just another variable, and applying the approach developed in this thesis. General dynamic processes that do not have a start or finish, are more difficult to deal with, but the basic methods developed in this thesis should be applicable with some modification to account for the dynamics.

8.2.2 Process Modeling

Before the computer age engineers often modeled processes by drawing lines to represent an input-output relationship between two or more variables. Even as computers arrived, and their power increased, engineers continued to model using deterministic lines. That is, a given input generates a deterministic output. Even complex modeling tools, such as cubic splines and neural networks, just represent a complex line or surface. With the currently available computer power, it is now feasible to model systems using probability distributions, as shown in this work.

Deterministic models of processes are a form of probability distribution, in that the probability is one if the inputs and outputs satisfy the model, and zeros otherwise. Probabilistic modeling is capable of capturing both the deterministic relationships and the statistical correlations in the data. Thus regions of the model inapplicability can be immediately identified by a low probability density, and the spread of the model about a correlation (i.e. a ridge or a plateau) are also determined. These properties are extremely important when applying derived process models to process monitoring and control tasks. The probabilistic modeling tools developed in this thesis could be used to estimate the probability distribution of all the states of a system. If it was desired to predict a particular state from the others in future applications, that state could be found by finding its maximum likely, or expected value, given all of the other measurements.
8.2.3 Multivariate Statistical Process Control

The probabilistic approach to modeling and data rectification in this work makes the developed framework ideal for multivariate statistical process control (MSPC). The goal of MSPC is to improve process performance by identifying when a process is outside its normal operating region, and take action to bring the process back in control. Conventional SPC examines a single variable at a time, despite the correlation between variables. The correlation between the variables, as captured in a multi-dimensional probability distribution, could be exploited to do better statistical process control in a MSPC framework.

8.2.4 Fault Detection and Fault Diagnosis

The probabilistic framework developed in this thesis could also be extended to detecting and identifying process faults. When the process fundamentally changes, the probability distribution of the states will also fundamentally change. Thus rectifying the data using the old probability distribution of the states will result in an erroneous solution. But the value of the objective function will be very low, because a number of large adjustments to the measurement will have to be made to rectify the data to the old probability distribution of the states. A process fault will cause a fundamental change in the probability distribution of the plant states, and thus could be identified in this manner. The probabilistic framework also allows for the posing of questions such as “what is the most likely fault to have occurred to generate the measurements observed”, which would lead to fault diagnosis applications.
Chapter 9: References


Appendix A:
MATLAB Source Code

Appendix A.1  EBF Code
Appendix A.2  RSDE Code
Appendix A.3  EM Solution to MLR Problem
Appendix A.1 EBF Code

make_rho.m    - file makes EBF density estimator
run_rho.m     - file calculates density estimates for an EBF
cv_rho.m      - Uses cross validation to find predicted log likelihood
my_cov.m      - finds weighted covariance matrix for EBF
my_ep_heur.m  - finds shape matrices for EBF
cluster.m     - finds centers for EBF
dist2.m       - finds Euclidian squared distance between points
layer1.m      - calculates activation for single EBF unit
mahal2.m      - finds Mahalanobis distance between points
shuffle.m     - utility function for randomizing a set of integers
function [m,shape,w_rho,rho_95] = make_rho(x,h,p,s)

%MAKE_RHO
%[m,shape,w_rho,rho_95] = make_rho(x,h,p,s)
%
% MAKE_RHO creates the density estimation portion of a validity
% index network. The input parameters are the training data
% x[ntrain,nin], h the number of rbf nodes, and p the overlap
% parameter used in the p-nearest neighbor heuristic.
%
% The outputs are m[h,nin] the centers of the h rbf units (see
% function cluster for details), the rbf unit shapes shape[h,1]
% or shape[h*nin,nin] (depending on whether spherical or elliptical
% units are used), w_rho[h,1] the weights for calculating the
% density.
%
% Uses elliptical units exclusively
%
% rho_p5 is the 95th percentile cutoff calculated from s fold
% cross validation. If s is not supplied then rho_95 is not
% calculated.
%
% Copyright (c) 1992 by Process Science, Inc.
% modified by Lloyd Johnston 9-10-92
% modified by Lloyd Johnston 7-21-93
% modified by Lloyd Johnston 6-20-94
[ntrain,nin] = size(x);
% Check dimensions
%
if(h > ntrain)
   error('More centers than training points specified');
end
if(p > h)
   error('Overlap parameter greater than number of centers');
end
%
% Determine unit centers
%
m = cluster(x,h);
shape = my_ep_heur(m,p,x);
a = layer1(m,shape,x);
nh = sum(a);
tot=sum(nh);
for i=1:h
    start = (i-1)*nin+1;
    finish = start+nin-1;
    shape_i = shape(start:finish,:);
    vol = sqrt(pi*nin)/det(shape_i);
    w_rho(i) = (nh(i)/tot)/vol;
end
%
% Finding 95th percentile cutoff

219
% if nargin == 4
disp('Finding 95th Percentile Cutoff Value');
x_rh = [ ];
a = shuffle(ntrain);
ntest = floor(ntrain/s)*ones(1,s);
for i = 1:rem(ntrain,ntest)
    ntest(i) = ntest(i) + 1;
end
if(any(ntest > ntrain-h))
    error('Too many centers specified for size of training subset. Increase s or decrease h.');
end
%
% s-fold CV loop
%
for i = 1:s
    disp(['CV_iteration = ',int2str(i),', of ',int2str(s)])
    if(i==1)
        start_test = 1;
        end_test = ntest(1);
    else
        start_test = start_test + ntest(i-1);
        end_test = end_test + ntest(i);
    end
    x_train = zeros(ntrain-ntest(i),nin);
    x_test = zeros(ntest(i),nin);
%
% Set up training and testing subsets
%
itrain = 0;
ittest = 0;
for j = 1:ntrain
    if(j<start_test | j>end_test)
        itrain=itrain+1;
        x_train(itrain,:) = x(a(j,:));
    else
        itest = itest+1;
        x_test(itest,:) = x(a(j,:));
    end
end
{mmm,ssshape,rrrho}=make_rho(x_train,h,p);
x_rh = [x_rh; run_rho(x_test,mmm,ssshape,rrrho)];
clear x_train;
clear x_test;
end
x_rh=sort(x_rh);
indx=round(0.05*ntrain);
rho_95=x_rh(indx);
end
function [x_rho,fail] = run_rho(x,m,shape,w_rho,rho_95)

%RUN_RHO
%[x_rho fail] = run_rho(x,m,shape,w_rho,rho_95)
%
% Function to get density estimates from a rho network.
% Inputs are the cases to run, x[k,nin], the unit centers
% m[h,nin], the unit shapes shape, and the weights for
% calculating the densities wrho[h,1]. The output is the
% density estimates for each x, x_rho[k,1].
% RUN_RHO internally determines whether shape is [h,1] vector
% of unit widths for spherical units or [h*nin,nin] for
% elliptical units.
% fail is a returned vector that gives the index of all
% points whose density is below the rho_95 test limit. If
% rho_95 is not supplied then fail is not calculated.
%
% Copyright (c) 1992 by Process Science, Inc.
% modified by Lloyd Johnston 9-10-93

A=layer1(m,shape,x);
x_rh0=A*w_rho;
if nargin==5
disp('looking for low densities')
f=x_rh0<rho_95;
fail=find(f);
end

function logprob = cv_rho(x,h,p,s)

%CV_RHO
%logprob = cv_rho(x,h,p,s)
%
% CV_RHO determines the log of the probability of the data
% for a density estimation (rho) network. logprob measures
% how well the network architecture suits the density
% estimation problem. Optimal h and p maximize logprob.
%
% Inputs are the training data x[k,nin], a number of
% choices of h[nh], the number of rbf units, and p[np],
% the overlap parameter used in the p-nearest neighbor
% heuristic. s is the number of subsets used in cross
% validation. The output is logprob[nh,np] the logprob
% for each feasible architecture.
%
% Calls make_rho.m which is currently setup for elliptical units only

221
function cv = cross_val(ntrain, nh, x, p, s)

% Copyright (c) 1992 by Process Science, Inc.
% Modified Lloyd P. M. Johnston 9-15-93
% Modified Lloyd P. M. Johnston 6-20-94

[ntrain,nin] = size(x);
np = length(p);
h = length(h);
logprob = zeros(nh,np);
for ip=1:np
    for ih=1:nh
        if(p(ip) < h(ih))
            disp(['h =',int2str(h(ih)),' p = ',int2str(p(ip))])
            a = shuffle(ntrain);
            ntest = floor(ntrain/s)*ones(1,s);
            for i=1:rem(ntrain,ntest)
                ntest(i) = ntest(i) + 1;
            end
            if(any(ntest > ntrain-h(ih)))
                error('Too many centers specified for size of training subset. Increase s or decrease h.');
            end
            disp(['CV_iteration = ',int2str(i) of ',',int2str(s)])
            if(i==1)
                start_test = 1;
            else
                start_test = start_test + ntest(i-1);
            end
            end_test = end_test + ntest(i);
            end
            x_train = zeros(ntrain-ntest(i),nin);
            x_test = zeros(ntest(i),nin);
            disp('Set up training and testing subsets')
            itrain = 0;
           ittest = 0;
            for j=1:ntrain
                if(j<start_test || j>end_test)
                    itrain=itrain+1;
                    x_train(ietrain,:) = x(a(j,:));
                else
                   ittest = ittest+1;
                    x_test(ittest,:) = x(a(j,:));
                end
            end
            [m,shape,rho] = make_rho(x_train,h(ih),p(ip));
function Qmine=my_coy(wt,x)

% Qmine=my_coy(wt,x) finds weighted covariance matrix
%
%
% Copyright(c) 1996 Lloyd P. M. Johnston

wt=wt/mean(wt); % this just scales weights - no change in shape of covariance
% matrix, only in size of numbers. Found it to be numerically
% advantageous. Covariances rescaled anyway to get unit shape
% matricies.

[m,n]=size(x);
temp=x.*(wt*ones(1,n));
av=sum(temp)/sum(wt);
y1=(x-ones(m,1)*av);

Qmine=y1'*y1/(m-1);

function shape = my_ep_heur(m,p,x)

% MY_EP_HEUR
% shape = my_ep_heur(m,p,x)
% Elliptical version of p-nearest neighbor heuristic. Given hidden
% unit locations m[h,nin], overlap parameter p, and data x[k,nin],
% EP_HEUR establishes ellipses by local adaption to the covariance
% structure of the x data, and sizes the ellipses so that there is
% the desired overlap. Output is returned as matrix shape[h*nin,nin]
% which contains the inverse covariance matrices of the hidden units.
%
% Copyright (c) 1992 by Process Science, Inc.
% modified by Lloyd Johnston 12-10-92
%
[h,nin] = size(m);
[ntrain,nin] = size(x);
sigma = p_heur(m,p);
a = layer1(m,sigma,x);
shape=f;
for i=1:h
    wcov = my_cow(a(:,i),x);
    shape = [shape; inv(wcov)];
end
for i=1:h
    start = (i-1)*nin+1;
    finish = start+nin-1;
    shape_i = shape(start:finish,:);
    sqdist = mahal2(ones(h,1)*m(:,i),m,shape_i);
    sqdist=sort(sqdist);
    sigmasq=mean(sqdist(2:p+1));
    shape(start:finish,:) = shape_i/sigmasq;
end

function m = cluster(x,h,iplot)

%CLUSTER
%m = cluster(x,h)
%
% CLUSTER returns the matrix m(h,nin) of h cluster centers for the data matrix x[ntrain,nin]. The k-means clustering algorithm is used.
%
% m=cluster(x,h,1) plots the movement of the cluster centers if nin=2.
%
% Copyright (c) 1992 by Process Science, Inc.
[ntrain,nin] = size(x);
if(nargin==3)
    if(nin==2)
        iplot=0;
    else
        plot(x(:,1),x(:,2),'+')
        hold on
    end
else
    iplot=0;
end
%
% choose h random points for initial cluster centers
%
m = x(unique_r(h,1,ntrain,:));
if(ierr==1); plot(m(:,1),m(:,2),'*'); end
%
% main loop
rel_change = 99;
while(rel_change > 0.0001)
dist = dist2(x,m);
if(h == 1)
    b=ones(1,ntrain);
else
    b = -(dist - ones(h,1)*min(dist));
end
ncluster = sum(b);
if(~any(ncluster == 0))
    % locate centroid of each cluster (b.x divided elementwise by ncluster)
    mnew=(b.*x)/((ones(nin,1)*ncluster));
    rel_change = sum(sum(((m-mnew).^2))/sum(sum(m.^2));
    if(iprint==1); plot(mnew(:,1),mnew(:,2),'o'); end
    m = mnew;
else
    break;
end
end
if(any(ncluster == 0))
disp('ncluster = 0...restarting clustering algorithm')
hold off
m = cluster(x,h);
end
end
if(iprint==1); hold off; end

function sqdist=dist2(x1,x2)

% DIST2
% sqdist=dist2(x1,x2)
% DIST2 calculates the squared distance between k pairs of points,
% x1[k,nin] to x2[k,nin]. Output are the squared distances,
% sqdist[k].
%
% Copyright (c) 1992 by Process Science, Inc.
[n1,nx] = size(x1);
[n2,nnx] = size(x2);
if(nnx ~= nx)
    error('Vector lengths incommensurate in dist2');
end
for i=1:n2
    diff = x1 - ones(n1,1)*x2(i,:);
    if(nnx==1)
```matlab
sqdist(i,:) = (diff.^2);
else
    sqdist(i,:) = sum((diff.^2));
end
end

function ah=layer1(m,shape,x)

% LAYER1
% ah=layer1(m,shape,x)
% % Calculates activation at hidden nodes for inputs x[k,nin]
% in radial or elliptical basis function networks. Inputs:
% the unit centers m[h,nin] and shapes [h,1] or [h*nin,nin].
% Output is ah[k,h], the activations of the hidden units.
% % LAYER1 automatically determines whether spherical or elliptical
% % units are used.
% %
% Copyright (c) 1992 by Process Science, Inc.
[k,nin] = size(x);
[h,nin] = size(m);
[hx,xx] = size(shape);
%
% spherical units; Euclidean distance
if(hx==h)
    dist = dist2(x,m);
    ah = (exp(-dist./((shape.^2)*ones(1,k))));
%
% elliptical units; Mahalanobis distance
elseif(hx==h*nin)
    if (hx==h*nin)
        for i=1:h
            start = (i-1)*nin+1;
            finish = start+nin-1;
            invcov_i = shape(start:finish,:);
            sqdist = mahal2(ones(k,1)*m(i,:),x,invcov_i);
            ah = [ah (exp(-sqdist))];
        end
    else
        error('shape is incorrectly specified');
    end
end
```
function sqdist = mahal2(x1,x2,shape)

%MAHAL2
% sqdist = mahal2(x1,x2,shape)
% Calculates the Mahalanobis distances between k pairs of points,
% x1[k,nin] to x2[k,nin], based on the inverse covariance matrix in
% shape. Output are the squared Mahalanobis distances, sqdist[k].
%
% Copyright (c) 1992 by Process Science, Inc.
[k,nin] = size(x1);
for i=1:k
   sqdist(i) = (x1(i,:)-x2(i,:))*shape*(x1(i,:)-x2(i,:))';
end

function a = shuffle(n)

%SHUFFLE
%a=shuffle(n)
% SHUFFLE returns a list of integers 1 through n (inclusive)
% in random order through a process of swapping. The output
% is a row vector a[1,n] randomized integers.
%
% Copyright (c) 1992 by Process Science, Inc.
% Modified by Lloyd Johnston 6-20-94
a = [1:n];
% do 2*n random swaps
% for i=1:2*n
   r1 = floor(n*rand(1))+1;
   r2 = floor(n*rand(1))+1;
   temp = a(r1);
   a(r1) = a(r2);
   a(r2) = temp;
end
Appendix A.2  RSDE Code

rsde.m  - the RSDE algorithm for finding $P(x)$
run_pca.m  - projects data using matrix P
log_obj_n.m  - calculates log likelihood objective function values
msqerr.m  - finds mean squared error.
function [m,shape,w_rho,dd,lh,xh,mse,lpd,lp]=rsde(y,tol,maxiter,sig,p_ge,b,h,p,x,P,st,stflg)

% RSDE
% [m,shape,w_rho,dd,lh,xh,mse]=rsde(y,tol,maxiter,sig,p_ge,b,h,p,x,P,st,stflg)
% This uses Recursive State Density Estimation method of training to
% train an ebf rectifier, the y data is adjusted to constraints held
% in PCA projection matrix P, and then an ebf is
% fit to the projected data the projected space. The program ends
% when the newly rectified x value is within tol of the last rectified
% x values (on a fractional basis). dd is vector of the convergence
% criterion, and lh is vector of the loglikelihood values at each iteration.
% If there are no known constraints set P=1;
%
% Copyright(c) 1996 Lloyd P. M. Johnston
%
[n,nin]=size(y);
if nargin==10
   st=y;
   stflg=1;
end
dd=[];
lh=[];
mse=[];
base=max(y)-min(y);
psc=run_pca(y,P);
[m,shape,w_rho]=make_rho(pcy,h,p);
yyyy=run_pca(pcy,P);
[Smix,qmix]=ebftomix(m,shape,w_rho);
mmix=m;
[lh,lpd,lp]=log_obj_n(y,yyy,mmix,qmix,Smix,p_ge,sig,b,P);
lh=[lh sum(lh)/n]
lpd=[lpd sum(lpd)/n]
lp=[lpx sum(lpx)/n]
ms=msqerr(yyy,x,sig.^2);
mse=[mse ms]
xh=f_linpx_emr(y,mmix,qmix,Smix,p_ge,sig,b,x,st,-1e300,[],[],P,stflg);
lastxh=y;
d=sum(sum(abs(xh-lastxh))/n)/base)/nin;
dd=[dd,d];
[lh,lpd,lp]=log_obj_n(xh,y,mmix,qmix,Smix,p_ge,sig,b,P);
lh=[lh sum(lh)/n]
lpd=[lpd sum(lpd)/n]
lp=[lpx sum(lpx)/n]
ms=msqerr(xh,x,sig.^2);
mse=[mse ms]
iter=0;
while (d>tol & iter<maxiter)
   iter=iter+1;
disp(iter)
   lastxh=xh;
   pch=run_pca(xh,P);
   [m,shape,w_rho]=make_rho(pch,h,p);

229
function [T]=run_pca(x,P)

% %RUN_PCA  [T]=run_pca(x,P) returns scores matrix T for x data using P
% matrix for the pca.
%
% Copyright(c) 1996 Lloyd P. M. Johnston

[n,m]=size(x);
T=[];
for i=1:n
    Tstar=x(i,:)*P';
    T=[T;Tstar];
end

function [llh,lpd,lp]=log_obj_n(xh,y,mmix,qmix,shapemix,p_ge,sig,b,P);

% % [llh,lpd,lp]=log_obj_n(xh,y,mmix,qmix,shapemix,p_ge,sig,b,P);
% Calculates objective function values, as well as contributions to objective
% function log(P(d)) and log(P(x)).
%
% Copyright(c) 1996 Lloyd P. M. Johnston

if nargin==9
    P=1;
end
[n,nin]=size(xh);
x=xh;
y=y;
for i=1:n
    x=xh(i,:);
y=y(i,:);
zphi=my_zphi((P*xh)',mmix,shapemix);
px=sum(zphi.*qmix);
zetaphi=my_zetaphi(xh,y,sig);
zetaphib=my_zetaphi(xh,y,b.*sig);
log_pdx=sum(log((1-p_ge).*zetaphi + p_ge.*zetaphib));
log_pdx=log(pdx);
llh(i)=log_pdx + log_pdx;
lpd(i)=log_pdx;
lpd(i)=log_pdx;
end

function mse=msqerr(x,xhat,vr)

% MSQERR mse=msqerr(x,xhat,vr) Finds a normalized mean squared error. vr is vector of measurement variances (i.e. sigma.^2)
% Copyright(c) 1996 Lloyd P. M. Johnston

[n_test,nin]=size(x);
[n,m]=size(vr);
if m==1
    mse=sum(sum((x-xhat).^2)/(n_test*nin*vr));
elself m==nin
    mse=sum(sum((x-xhat).^2./vr)/(n_test*nin));
else
    mse=['vr must be 1 by 1 or 1 by nin'];
end
if n_test==1
    mse=mean((x-xhat).^2./vr);
end

231
Appendix A.3  EM Solution to MLR Problem

f_linx_emr.m  - applies overall algorithm (calls linpx_emr.m)
linpx_emr.m  - EM optimization algorithm
my_E_step.m  - does E step
my_M_step_lin.m  - does M step with matrix projection by P
ebfomix.m  - converts EBF to mixture of Gaussians
emr_initial.m  - ranks starting points by objective function value
my_full_s_u.m  - calculates parameters for EM algorithm initialization
my_lambda.m  - finds \( \Lambda \) matrix
my_S.m  - finds S matrix for M step
my_zetaphi.m  - calculates Gaussian distribution values for finding zeta
my_zphi.m  - calculates Gaussian distribution values for finding z
function
[xh,zt,z,ms,msl,n_em,bad]=f_linx_emr(y,mm,qm,Sm,p_ge,sig,b,x,st,Ta,xtr,prtr,P,stflag)

This is the solution algorithm in chapter 6. The actual EM optimization is done in
linx_emr.m which this function calls.

y (n by nin) are the measurements, mm, qm, and Sm are Gaussian mixture model centers,
weights, and shape matrices for P(x). p_ge is probability of a gross error (1 by nin),
sig is vector of standard deviations (1 by nin), b is vector of b (1 by nin) parameters for
bivariate Gaussian error distribution, x (n by nin) are the true values, st are values to start
at (n by nin), Ta (1 by 1) is the log likelihood cutoff value, xtr are the candidate set of
start points and prtr are the corresponding P(x) values, P is the projection matrix derived
from the linear constraints, if there are no linear constraints set P=1, stflag is the start flag
if stflag=1 use initialization formulas, if stflag=0 do E step first from st.

Copyright(c) 1996 Lloyd P. M. Johnston

msqerr(y,x,sig.^2)
tic
[n,nin]=size(y);
[ntr,nin]=size(xtr);
if P==[]
P=1;
end
if stflag==[]
stflag=1;
end
for i=1:n
yy=y(i,:);
xx=x(i,:);
xh=st(i,:);
[xh,zt,z,ms,msl]=linx_emr(yy,mm,qm,Sm,p_ge,sig,b,xx,xh,P,stflag);
[TT,lpd,lpx]=log_obj_n(xh,yy,mm,qm,Sm,p_ge,sig,b,P);
nn_em=1;
count=0;
nststr=999999999999999;
if TT<Ta
disp([TT ms])
xstr,st_lh]=emr_initial(yy,p_ge,sig,b,xtr,prtr,ntr,100);
if nststr==size(xstr);
count=1;
while (TT<Ta) & (count <=nststr)
strp=xstr(count,:);
[xh,zt,z,ms,msl]=linx_emr(yy,mm,qm,Sm,p_ge,sig,b,xx,strp,P,0);
[TT,lpd,lpx]=log_obj_n(xh,yy,mm,qm,Sm,p_ge,sig,b,P);
disp([count, st_lh(count) TT msqerr(strp,xx,sig.^2) msqerr(xh,xx,sig.^2))]
count=count+1;
nn_em=nn_em+1;
nnum_ge=length(find(zt>0.7));
end
iiinndd=find(zt>0.7);
if (count-1==nstrt)
    ms=1;
    xh=yy;
    bad=[bad, I];
end
disp([i ms nn_em TT length(iiinnd)])
xhh(i,:)=xh;
zzt(i,:)=zt;
if mm==[]
    zz(i,:)=z;
end
mmsl(i)=ms;
n_em(i,:)=[nn_em ms TT lpd lpx];
end
ms=mean(mmsl);
msl=mmsl;
xh=xhh;
zt=zztt;
z=zz;
run_time=toc;
disp('MSE  Run Time')
ms=[ms run_time]

function [xh,zt,z,ms,msl]=linpx_emr(yy,mm,qm,Sm,p_ge,sig,b,xx,xh,P,flag)

[nin,nin]=size(yy);
base2=yy;
vr=sig.^2;
tol=1e-6;
maxiter=100;
if nargin==11
    flag=0;
end
if mm==[]
    if flag
[zz,zzeta]=my_z_init(xh,yy,mm,qm,Sm,p_ge,sig,b,P,xh,yy);
else
[zz,zzeta]=my_E_step(xh,yy,mm,qm,Sm,p_ge,sig,b,P,xh,yy);
end
[xh]=my_M_step_lin(zz,zzeta,yy,mm,Sm,sig,p_ge,b,P,yy);
dif=999999999.99999999;
:i=er=0;
while (dif>tol) & (iter <= maxiter)
    iter=iter+1;
    xhlast=xh;
    [zz,zzeta]=my_E_step(xhlast,yy,mm,qm,Sm,p_ge,sig,b,P,xhlast,yy);
    [xh]=my_M_step_lin(zz,zzeta,yy,mm,Sm,sig,p_ge,b,P,yy);
    dif=sum(abs(xh-xhlast))/base2/nin;
    if iter==maxiter
        disp('maxiter reached')
    end
end
end

% only linear constraints no P(x)
else
    if flag
        [zz,zzeta]=my_z_init_nopx(xh,yy,p_ge,sig,b);
    else
        [zz,zzeta]=my_z_init_2_nopx(xh,yy,p_ge,sig,b);
    end
    [xh]=my_M_step_lin_nopx(zz,zzeta,yy,sig,p_ge,b,A,c);
dif=999999999.99999999;
iter=0;
while (dif>tol) & (iter <= maxiter)
    iter=iter+1;
    xhlast=xh;
    [zz,zzeta]=my_E_step(xhlast,yy,mm,qm,Sm,p_ge,sig,b);
    [xh]=my_M_step_lin_nopx(zz,zzeta,yy,sig,p_ge,b,A,c);
    iind=find(xh==0);
    if length(iind)==0
        denom=base2;
    else
        denom=xh;
        end
    dif=sum(abs(xh-xhlast))/denom/nin;
    if iter==maxiter
        disp('maxiter reached')
    end
end
zt=zzeta;
z=zz;
ms=mssqerr(xh,xx,vr);
msl=ms;
function [z,zeta]=my_E_step(xh,y,mmix,qmix,shapemix,p_ge,sig,b,P,xhs,ys)

  % performs E-Step of EM optimization algorithm (finds z and zeta)
  %
  % Copyright(c) 1996 Lloyd P. M. Johnston

  if nargin==8
    P=1;
    xhs=xh;
    ys=y;
  end
  small=1e-300;
  [nh,nin]=size(mmix);
  zphi=my_zphi((P*xh)',mmix,shapemix);
  zetaphi_sig=my_zetaphi(xh,y,sig);
  zetaphi_bsig=my_zetaphi(xh,y,b.*sig);
  if sum(zphi)<small
    zphi=ones(1,nh)*small;
  end
  z=(qmix.*zphi)/sum(qmix.*zphi);
  zeta=(p_ge.*zetaphi_bsig)./(1-p_ge).*zetaphi_sig + p_ge.*zetaphi_bsig);
  ind=find(((1-p_ge).*zetaphi_sig + p_ge.*zetaphi_bsig)==0);
  if ind==[]
    zeta(ind)=zeros(1,length(ind));
  end

function xh=my_M_step_lin(z,zeta,y,mmix,qmix,shapemix,sig,p_ge,b,P,ys)

  % Does M-step of EM algorithm for constraints contained in projection matrix
  % P. If no constraints set P=1.
  %
  % Copyright(c) 1996 Lloyd P. M. Johnston

  if nargin==8
    P=1;
    ys=y;
  end
  [m,nin]=size(mmix);
  Linv=my_lambda(zeta,sig,p_ge,b);
  [S,Qm]=my_S(z,mmix,shapemix,Linv,P);
  w=S*(Qm+.*((Linv.*y)'));
  xh=(P*w);
function [shapemix,p] = ebftomix(mu,shape,w_rho):

%EBFTOMIX
% [shapemix,p] = ebftomix(mu,shape,w_rho);
%
% Converts the EBF (rho-net) weights to prior probabilities p for
% a mixture model of normal distributions, where sum(p)=1.
%
% From Michael L. Thompson (MIT Ph.D. Thesis 1996)

[h,nin] = size(mu);
last = 0;
for i=1:h,
    first = last + 1;
    last = last + nin;
    vol = (pi^nin/det(shape(first:last,:)))^(-0.5);
    p(i) = w_rho(i)/vol;
end
shapemix = shape*2;

function [x_start,st_lh]=emr_initial(yy,p_ge,sig,b,xhtr,prtr,ntr,nst)

%
%x_start=emr_initial(yy,mmix,qmix,shpmix,p_ge,sig,b,xhtr,prtr,ntr,nst)
%
% Creates list of starting points ranked by their likelihood.
%
% Copyright(c) 1996 Lloyd P. M. Johnston

pofx=1;
if prtr==[]
    pofx=0;
end
rand_ind=shuffle(ntr);
for i=1:ntr
    xxh=xhtr(rand_ind(i,:));
    zetaphi=my_zetaphi(xxh,yy,sig);
    zetaphib=my_zetaphi(xxh,yy,b.*sig);
    log_pd=sum(log((1-p_ge).*zetaphi + p_ge.*zetaphib));
    if pofx
        llh(i)=log_pd + log(prtr(rand_ind(i)));
    else
        llh(i)=log_pd;
    end
end
[sllh,ind]=sort(-llh);
x_start=xhtr(rand_ind(ind(1:nst),:));
st_lh=-sllh(1:nst);
function [s,u]=my_full_s_u(zphi,Smix,mmix,z,P)

% Finds components for initial conditions of EM algorithm used in finding
% zeta(0). In chapter 6 v is used instead of s.

% Copyright(c) 1996 Lloyd P. M. Johnston

if nargin <=5
  P=1;
end
[H,mnin]=size(mmix);
if length(P)==1
  [dum,nin]=size(P);
else
  nin=mnin;
end
start=1;
SSinv=zeros(nin,nin);
SM=zeros(nin,1);
for i=1:H
  stop=start+mnin-1;
  Qinv=Smix(start:stop,:);
  start=start+1;
  SSinv=SSinv+z(i)*(P*Qinv*P);
  SM=SM+z(i)*(P*Qinv)*mmix(i,:);
end
SSinv;
SS=pinv(SSinv);
s2=diag(SS);
s=sqrt(s2);
u=SS*SM;
u=u';

function Linv=my_lambda(zeta,sig,p_ge,b);

% Finds diagonal matrix Lambda as a vector

% Copyright(c) 1996 Lloyd P. M. Johnston

s2=sig.^2;
Linv=(1-zeta)./s2 + zeta./(b.^2.*s2);
function [S,zQm]=my_S(z,mmix,shapemix,Linv,P);

% Finds S matrix and sum over the H units of z*Qinv*m

% Copyright(c) 1996 Lloyd P. M. Johnston

if nargin ==4
    P=1;
end
[m,mnin]=size(mmix);
if length(P)==1
    [dum,nin]=size(P);
else
    nin=mnin;
end
Ssum=zeros(mnin,mnin);
zQmsum=zeros(mnin,1);
start=1;
for i=1:m
    stop=start+mnin-1;
    Qinv=shapemix(start:stop,:);
    start=start+1;
    Ssum=Ssum+z(i)*Qinv;
    zQmsum=zQmsum+z(i)*Qinv*mmix(i,:);
end
S=inv(Ssum+P*diag(Linv)*P);
zQm=zQmsum;

function zetaphi=my_zetaphi(xh,y,sig);

% Calculates zero centered Gaussian distribution value for xh-y with standard
% deviation sig. Where zetaphi and sig are both vectors 1 by nin

% Copyright(c) 1996 Lloyd P. M. Johnston

[n,nin]=size(xh);
D=sqrt(2*pi)*sig;
um=((xh-y)./sig).^2;
zetaphi=exp(-0.5*num)./D;

function zphi=my_zphi(xh,mmix,shapemix)

% Calculates Gaussian distribution, with means mmix and inverse covariance
% matrices in shapemix, at xh. Where zphi is a vector 1 by number of
% basis functions in mixture model.

% Copyright(c) 1996 Lloyd P. M. Johnston

239
function [z,zeta]=my_z_init(xh,y,mmix,qmix,shapemix,p_ge,sig,b,P,xhs,ys)

% [z,zeta]=my_z_init(xh,y,mmix,qmix,shapemix,p_ge,sig,b,P,xhs,ys)
% Initial conditions for EM algorithm starting at measurement y

% Copyright(c) 1996 Lloyd P. M. Johnston

small=1e-300;
[n,nin]=size(y);
[nh,ninm]=size(mmix);
small=1e-300;
if nargin==8
    P=1;
    xhs=xh;
    ys=y;
end
zphi=my_zphi((P*xhs'),mmix,shapemix);
if sum(zphi)<=small;
    zphi=zeros(1,nh);
    xxx=((P*xhs'));
    start=1;
    for i=1:nh
        stop=start+ninm-1;
        Sh=shapemix(start:stop,:);
        start=start+1;
        sqdist(i)=mahal2(xxx,mmix(i,:),Sh);
    end
    [dum,iii]=min(sqdist);
    zphi(iii)=1;
end
z=(qmix.*zphi)/sum(qmix.*zphi);
% FLAG = 1 to use full form of s and u,
% FLAG = 0 for approx method of using max z
flag=1;
if flag
    [s,u]=my_full_s_u(zphi,shapemix,mmix,z,P);
else
    [mz,ind]=max(z);
end
\begin{verbatim}
start=(ind-1)*nin+1;
stop=start+nin-1;
Q=shapemix(start:stop,:);
SS=inv(P'*Q*P);
s2=diag(SS);
s=sqrt(s2);
u=(P'*mmix(ind,:))';
end
zetaphi_sig=my_zetaphi(y,u,(sig+s));
zetaphi_bsig=my_zetaphi(y,u,(b.*sig+s));
[ind]=find(((1-p_ge).*zetaphi_sig + p_ge.*zetaphi_bsig)<small);
zetaphi_bsig(ind)=small*ones(1,length(ind));
zeta=(p_ge.*zetaphi_bsig)./(1-p_ge).*zetaphi_sig + p_ge.*zetaphi_bsig);
\end{verbatim}
Appendix B:
Case Study Source Code

Appendix B.1  ASPENPLUS Input File
Appendix B.2  Sample ASPENPLUS Output File
Appendix B.3  Case Study MATLAB Source Code
Appendix B.1 ASPEN Input File

**equil.inp** - ASPENPLUS input file for the base case and 10 of the 1500 “case studies” (ASPENPLUS option for generating multiple outputs with different input conditions) done to generate the data for the cyclohexane case study problem in chapter 7.

**fwriter.f** - Fortran source code. fwriter.f is called by the ASPENPLUS to write the desired to a file in a form readable by MATLAB
TITLE 'HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE'

IN-UNITS ENG

DEF-STREAMS CONVEN ALL

DIAGNOSTICS
   HISTORY SYS-LEVEL=3 SIM-LEVEL=3 PROP-LEVEL=3 STREAM-LEVEL=3 &
   CONV-LEVEL=3

SYS-OPTIONS INTERPRET=NO

RUN-CONTROL MAX-TIME=10000.0

DATABANKS PURECOMP / AQUEOUS / SOLIDS / INORGANIC / &
   NOASPNPCD

PROP-SOURCES PURECOMP / AQUEOUS / SOLIDS / INORGANIC

COMPONENTS
   H2 H2 H2 /
   N2 N2 N2 /
   CH4 CH4 CH4 /
   C6H6 BZ /
   C6H12-1 CH

FLOWSHEET
   BLOCK HP-SEP IN=4 OUT=5 6
   BLOCK L-FLOW IN=6 OUT=8 9
   BLOCK V-FLOW IN=5 OUT=7 10
   BLOCK COL IN=9 OUT=11 12
   BLOCK FD-MX IN=7 1 2 8 OUT=3
   BLOCK RX-EQ IN=3 OUT=4 14

PROPERTIES SYSOP3

PROP-REPLACE SYSOP3 RKS-BM
   PROP MUVMX MUVMX08
   PROP MULMX MULMX02
   PROP KVXM KVXM07
   PROP KLXM KLXM01
   PROP DLMX DLMX05
   PROP DL DL03
   PROP DV DV01
   PROP MUL MUL01
   PROP MUVMUV07
   PROP KV KV06
   PROP KL KL01
STREAM 1
SUBSTREAM MIXED TEMP=120.0 PRES=335.0
MOLE-FLOW H2 310.0 / N2 .40 / CH4 10.0

STREAM 2
SUBSTREAM MIXED TEMP=100.0 PRES=15.0 MOLE-FLOW=100.0
MOLE-FRAC BZ 1.0

STREAM 3
SUBSTREAM MIXED TEMP=400.0 PRES=330.0 MOLE-FLOW=662.6
MOLE-FRAC H2 .630 / N2 .007 / CH4 .14 / BZ .15 / &
   CH .073

BLOCK L-FLOW FSPLIT
   DESCRIPTION "L-FLOW"
   FRAC 8 .30

BLOCK V-FLOW FSPLIT
   DESCRIPTION "V-FLOW"
   FRAC 10 .080

BLOCK FD-MX HEATER
   DESCRIPTION "FEED-MIX"
   PARAM TEMP=400 PRES=330.0

BLOCK HP-SEP FLASH2
   DESCRIPTION "HP-SEP"
   PARAM TEMP=120.0 PRES=.50

BLOCK COL RADFRAC
   DESCRIPTION "COLUMN"
   PARAM NSTAGE=15
   FEEDS 9 8 ON-STAGE
   PRODUCTS 11 1 V / 12 15 L
   P-SPEC 1 200.0
   COL-SPECS B:F=0.98 DP-COL=15 MOLE-RDV=1.0 MOLE-RR=1.20
   DB:F-PARAMS COMPS=CH

BLOCK RX-EQ REQUIL
   PARAM NREAC=1 TEMP=400 PRES=320 NPHASE=1
   STOIC 1 H2 -3 * / BZ -1 * / CH 1 *

CONV-OPTIONS
   WEGSTEIN MAXIT=100

FORTRAN F-1
   DEFINE F1 STREAM-VAR STREAM=1 SUBSTREAM=MIXED &
      VARIABLE=MASS-FLOW
   DEFINE F2 STREAM-VAR STREAM=2 SUBSTREAM=MIXED &
      VARIABLE=MASS-FLOW
   DEFINE F3 STREAM-VAR STREAM=3 SUBSTREAM=MIXED &
      VARIABLE=MASS-FLOW
   DEFINE F4 STREAM-VAR STREAM=4 SUBSTREAM=MIXED &
      VARIABLE=MASS-FLOW
DEFINE F5 STREAM-VAR STREAM=5 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F6 STREAM-VAR STREAM=6 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F7 STREAM-VAR STREAM=7 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F8 STREAM-VAR STREAM=8 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F9 STREAM-VAR STREAM=9 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F10 STREAM-VAR STREAM=10 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F11 STREAM-VAR STREAM=11 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE F12 STREAM-VAR STREAM=12 SUBSTREAM=MIXED & VARIABLE=MASS-FLOW
DEFINE T1 STREAM-VAR STREAM=1 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T2 STREAM-VAR STREAM=2 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T3 STREAM-VAR STREAM=3 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T4 STREAM-VAR STREAM=4 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T5 STREAM-VAR STREAM=5 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T6 STREAM-VAR STREAM=6 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T7 STREAM-VAR STREAM=7 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T8 STREAM-VAR STREAM=8 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T9 STREAM-VAR STREAM=9 SUBSTREAM=MIXED VARIABLE=TEMP
DEFINE T10 STREAM-VAR STREAM=10 SUBSTREAM=MIXED & VARIABLE=TEMP
DEFINE T11 STREAM-VAR STREAM=11 SUBSTREAM=MIXED & VARIABLE=TEMP
DEFINE T12 STREAM-VAR STREAM=12 SUBSTREAM=MIXED & VARIABLE=TEMP
DEFINE P1 STREAM-VAR STREAM=1 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P2 STREAM-VAR STREAM=2 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P3 STREAM-VAR STREAM=3 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P4 STREAM-VAR STREAM=4 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P5 STREAM-VAR STREAM=5 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P6 STREAM-VAR STREAM=6 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P7 STREAM-VAR STREAM=7 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P8 STREAM-VAR STREAM=8 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P9 STREAM-VAR STREAM=9 SUBSTREAM=MIXED VARIABLE=PRES
DEFINE P10 STREAM-VAR STREAM=10 SUBSTREAM=MIXED & VARIABLE=PRES
DEFINE P11 STREAM-VAR STREAM=11 SUBSTREAM=MIXED & VARIABLE=PRES
DEFINE P12 STREAM-VAR STREAM=12 SUBSTREAM=MIXED & VARIABLE=PRES
DEFINE TC1 BLOCK-VAR BLOCK=COL VARIABLE=TEMP & SENTENCE=PROFILE ID1=1
DEFINE TC2 BLOCK-VAR BLOCK=COL VARIABLE=TEMP & SENTENCE=PROFILE ID1=2
DEFINE TC3 BLOCK-VAR BLOCK=COL VARIABLE=TEMP & SENTENCE=PROFILE ID1=3
DEFINE TC4 BLOCK-VAR BLOCK=COL VARIABLE=TEMP & SENTENCE=PROFILE ID1=4
DEFINE TC5 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
DEFINE TC6 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=6
DEFINE TC7 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=7
DEFINE TC8 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=8
DEFINE TC9 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=9
DEFINE TC10 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=10
DEFINE TC11 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=11
DEFINE TC12 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=13
DEFINE TC13 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=13
DEFINE TC14 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=14
DEFINE TC15 BLOCK-VAR BLOCK=COL VARIABLE=TEMP &
SENTENCE=PROFILE ID1=15
DEFINE QFD BLOCK-VAR BLOCK=FD-MX VARIABLE=QCALC &
SENTENCE=PARAM
DEFINE QRX BLOCK-VAR BLOCK=RX-EQ VARIABLE=QCALC &
SENTENCE=RESULTS
DEFINE QSEP BLOCK-VAR BLOCK=HP-SEP VARIABLE=QCALC &
SENTENCE=PARAM
DEFINE QCBTM BLOCK-VAR BLOCK=COL VARIABLE=REB-DUTY &
SENTENCE=RESULTS
DEFINE QCTOP BLOCK-VAR BLOCK=COL VARIABLE=COND-DUTY &
SENTENCE=RESULTS
DEFINE XB1 MOLE-FRAC STREAM=12 COMPONENT=CH
DEFINE XB2 MOLE-FRAC STREAM=12 COMPONENT=BZ
DEFINE XB3 MOLE-FRAC STREAM=12 COMPONENT=H2
DEFINE XB4 MOLE-FRAC STREAM=12 COMPONENT=CH4
DEFINE XB5 MOLE-FRAC STREAM=12 COMPONENT=N2

CALL FWRITE(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,
F & T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,
F & P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,
F & TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8,TC9,
F & TC10,TC11,TC12,TC13,TC14,TC15,
F & QFD,QRX,QSEP, QCBTM, QCTOP,
F & XB1,XB2,XB3,XB4,XB5)

EXECUTE LAST

CONV-OPTIONS
PARAM CHECKSEQ=NO

STREAM-REPOR MOLEFLOW MOLEFRAC
CASE-STUDY
VARY MOLE-FLOW STREAM=1 SUBSTREAM=MIXED COMPONENT=H2
VARY MOLE-FLOW STREAM=1 SUBSTREAM=MIXED COMPONENT=CH4
VARY STREAM-VAR STREAM=2 VARIABLE=MOLE-FLOW
VARY STREAM-VAR STREAM=1 VARIABLE=TEMP
VARY STREAM-VAR STREAM=2 VARIABLE=TEMP
VARY BLOCK-VAR BLOCK=FD-MX SENTENCE=PARAM VARIABLE=TEMP
VARY BLOCK-VAR BLOCK=FD-MX SENTENCE=PARAM VARIABLE=PRES
VARY BLOCK-VAR BLOCK=RX-EQ SENTENCE=PARAM VARIABLE=TEMP
VARY BLOCK-VAR BLOCK=RX-EQ SENTENCE=PARAM VARIABLE=PRES
VARY BLOCK-VAR BLOCK=HP-SEP SENTENCE=PARAM VARIABLE=TEMP
VARY BLOCK-VAR BLOCK=HP-SEP SENTENCE=PARAM VARIABLE=PRES
VARY BLOCK-VAR BLOCK=V-FLOW SENTENCE=FRAC VARIABLE=FRAC & ID1=10
VARY BLOCK-VAR BLOCK=L-FLOW SENTENCE=FRAC VARIABLE=FRAC & ID1=8
VARY BLOCK-VAR BLOCK=COL SENTENCE=COL-SPECT & VARIABLE=MOLE-RR
VARY BLOCK-VAR BLOCK=COL SENTENCE=COL-SPECT & VARIABLE=B:F
VARY BLOCK-VAR BLOCK=COL SENTENCE=P-SPEC VARIABLE=PRES & ID1=1

CASE  1  329.2519  10.3993  106.2493  121.6381  &
      100.2117  395.0989  328.4018  395.0989  &
      319.2883  121.5621  -5.7287  0.1269  &
      0.4110  1.3932  0.9582  195.6335
CASE  2  315.6030  7.4023  102.4353  119.6243  &
      98.4865  399.0380  332.4934  399.0380  &
      326.3927  121.5790  -5.4098  0.1309  &
      0.3901  1.2881  0.9595  197.8741
CASE  3  324.5112  8.3574  102.1190  116.0342  &
      98.4918  408.9316  331.0316  408.9316  &
      321.2528  114.0953  -6.3321  0.1121  &
      0.4198  1.3677  0.9152  194.0977
CASE  4  512.1748  16.3563  165.7398  120.0588  &
      100.1339  397.3764  334.4129  397.3764  &
      323.7175  120.6134  -9.8916  0.0883  &
      0.2850  1.2453  0.9580  190.6400
CASE  5  607.1642  19.1684  187.2518  124.1719  &
      100.0916  403.8663  318.2619  403.8663  &
      305.4672  118.7662  -10.8380  0.0718  &
      0.2838  1.5826  0.9764  200.5638
CASE  6  330.5088  9.5672  105.4176  115.3136  &
      98.5444  391.3105  325.7056  391.3105  &
      315.5768  120.7683  -8.6139  0.1104  &
      0.4223  1.4062  0.9337  198.0890
CASE  7  331.2423  7.8377  107.0027  119.4885  &
      102.2748  390.8882  347.2407  390.8882  &
      339.5656  120.1123  -5.6100  0.1346  &
      0.3827  1.4542  0.9372  205.5577
CASE  8  336.6488  9.2071  106.3892  118.6308  &
      98.7685  398.1729  331.5402  398.1729  &
      324.2715  120.0278  -5.2748  0.1282  &
      0.4119  1.5351  0.9064  199.8721

248
subroutine fwriter(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12, &
  T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12, &
  P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12, &
  TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8,TC9, &
  TC10,TC11,TC12,TC13,TC14,TC15, &
  QFD,QRX,QSEP,QCBTM,QCTOP, &
  XB1,XB2,XB3,XB4,XB5)

  implicit real*8 (a-h,o-z)

  OPEN(ACCESS='APPEND',UNIT=75, FILE = '/users/lloyd/F.dat')
  OPEN(ACCESS='APPEND',UNIT=76, FILE = '/users/lloyd/T.dat')
  OPEN(ACCESS='APPEND',UNIT=77, FILE = '/users/lloyd/TC.dat')
  OPEN(ACCESS='APPEND',UNIT=78, FILE = '/users/lloyd/P.dat')
  OPEN(ACCESS='APPEND',UNIT=79, FILE = '/users/lloyd/Q.dat')
  OPEN(ACCESS='APPEND',UNIT=71, FILE = '/users/lloyd/X.dat')

  WRITE(75,99) F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12
  WRITE(76,99) T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12
  WRITE(77,97) TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8,TC9,TC10, &
               TC11,TC12,TC13,TC14,TC15
  WRITE(78,99) P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12
  WRITE(79,98) QFD,QRX,QSEP,QCBTM,QCTOP
  WRITE(71,91) XB1,XB2,XB3,XB4,XB5

  99 FORMAT(12(1X,F12.2))
  98 format(5(1x,f18.2))
  97 format(15(1x,f12.2))
  91 FORMAT(5(1x,f12.8))

  close(75)
  close(76)
  close(77)
  close(78)
  close(79)
  close(71)
  return

end
Appendix B.2    Sample ASPENPLUS Output File

ASPENPLUS output file for the base case.
ASSEN PLUS (TM) IS A PROPRIETARY PRODUCT OF ASPEN TECHNOLOGY, INC.
(ASIENTECH), AND MAY BE USED ONLY UNDER AGREEMENT WITH ASPENTECH.
REVIEWED RIGHTS LEGEND: USE, REPRODUCTION, OR DISCLOSURE BY THE
U.S. GOVERNMENT IS SUBJECT TO RESTRICTIONS SET FORTH IN
(i) FAR 52.227-14, Alt. III, (ii) FAR 52.227-19, (iii) DFARS
252.227-7013(c)(1)(ii), or (iv) THE ACCOMPANYING LICENSE AGREEMENT,
AS APPLICABLE. FOR PURPOSES OF THE FAR, THIS SOFTWARE SHALL BE DEEMED
TO BE "UNPUBLISHED" AND LICENSED WITH DISCLOSURE PROHIBITIONS.
CONTRACTOR/SUBCONTRACTOR: ASPEN TECHNOLOGY, INC. TEN CANAL PARK,
CAMBRIDGE, MA 02141.

TABLE OF CONTENTS

RUN CONTROL SECTION............................. 1
  RUN CONTROL INFORMATION........................ 1

FLOWSHEET SECTION................................. 2
  FLOWSHEET CONNECTIVITY BY STREAMS .......... 2
  FLOWSHEET CONNECTIVITY BY BLOCKS .......... 2
  CONVERGENCE STATUS SUMMARY ................ 2
  FORTRAN BLOCK: F-1............................ 2
  CONVERGENCE BLOCK: SOLVER01 ............... 4
  COMPUTATIONAL SEQUENCE ..................... 4
  OVERALL FLOWSHEET BALANCE ................. 5
  CASE-STUDY BLOCK............................. 5

PHYSICAL PROPERTIES SECTION....................... 10
  COMPONENTS ................................... 10

U-O-S BLOCK SECTION............................... 11
  BLOCK: COL MODEL: RADFRAC .................. 11
  BLOCK: FD-MX MODEL: HEATER ............... 15
  BLOCK: HP-SEP MODEL: FLASH2 ............... 16
  BLOCK: L-FLOW MODEL: FSPLIT ............... 16
  BLOCK: RX-EQ MODEL: REQUIL ............... 17
  BLOCK: V-FLOW MODEL: FSPLIT ............... 18

STREAM SECTION.................................. 20
  1 10 11 12 14.................................. 20
  2 3 4 5 6...................................... 21
  7 8 9.......................................... 22

PROBLEM STATUS SECTION.......................... 23
  BLOCK STATUS .................................. 23
# Hydrogenation of Benzene to Form Cyclohexane

## Table of Contents

**CASE NO. 1**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOWSHEET SECTION</td>
<td>24</td>
</tr>
<tr>
<td>FLOWSHEET CONNECTIVITY BY STREAMS</td>
<td>24</td>
</tr>
<tr>
<td>FLOWSHEET CONNECTIVITY BY BLOCKS</td>
<td>24</td>
</tr>
<tr>
<td>CONVERGENCE STATUS SUMMARY</td>
<td>24</td>
</tr>
<tr>
<td>FORTRAN BLOCK: F-1</td>
<td>24</td>
</tr>
<tr>
<td>CONVERGENCE BLOCK: $SOLVER01$</td>
<td>26</td>
</tr>
<tr>
<td>COMPUTATIONAL SEQUENCE</td>
<td>27</td>
</tr>
<tr>
<td>OVERALL FLOWSHEET BALANCE</td>
<td>27</td>
</tr>
<tr>
<td>U-O-S BLOCK SECTION</td>
<td>28</td>
</tr>
<tr>
<td>BLOCK: COL MODEL: RADFRAC</td>
<td>28</td>
</tr>
<tr>
<td>BLOCK: FD-MX MODEL: HEATER</td>
<td>32</td>
</tr>
<tr>
<td>BLOCK: HP-SEP MODEL: FLASH2</td>
<td>33</td>
</tr>
<tr>
<td>BLOCK: L-FLOW MODEL: FSPLIT</td>
<td>33</td>
</tr>
<tr>
<td>BLOCK: RX-EQ MODEL: REQUIL</td>
<td>34</td>
</tr>
<tr>
<td>BLOCK: V-FLOW MODEL: FSPLIT</td>
<td>35</td>
</tr>
<tr>
<td>STREAM SECTION</td>
<td>37</td>
</tr>
<tr>
<td>1 10 11 12 14</td>
<td>37</td>
</tr>
<tr>
<td>2 3 4 5 6</td>
<td>38</td>
</tr>
<tr>
<td>7 8 9</td>
<td>39</td>
</tr>
<tr>
<td>PROBLEM STATUS SECTION</td>
<td>40</td>
</tr>
<tr>
<td>BLOCK STATUS</td>
<td>40</td>
</tr>
</tbody>
</table>

**CASE NO. 2**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOWSHEET SECTION</td>
<td>41</td>
</tr>
<tr>
<td>FLOWSHEET CONNECTIVITY BY STREAMS</td>
<td>41</td>
</tr>
<tr>
<td>FLOWSHEET CONNECTIVITY BY BLOCKS</td>
<td>41</td>
</tr>
<tr>
<td>CONVERGENCE STATUS SUMMARY</td>
<td>41</td>
</tr>
<tr>
<td>FORTRAN BLOCK: F-1</td>
<td>41</td>
</tr>
<tr>
<td>CONVERGENCE BLOCK: $SOLVER01$</td>
<td>43</td>
</tr>
<tr>
<td>COMPUTATIONAL SEQUENCE</td>
<td>44</td>
</tr>
<tr>
<td>OVERALL FLOWSHEET BALANCE</td>
<td>44</td>
</tr>
<tr>
<td>U-O-S BLOCK SECTION</td>
<td>45</td>
</tr>
<tr>
<td>BLOCK: COL MODEL: RADFRAC</td>
<td>45</td>
</tr>
<tr>
<td>BLOCK: FD-MX MODEL: HEATER</td>
<td>49</td>
</tr>
<tr>
<td>BLOCK: HP-SEP MODEL: FLASH2</td>
<td>50</td>
</tr>
<tr>
<td>BLOCK: L-FLOW MODEL: FSPLIT</td>
<td>50</td>
</tr>
<tr>
<td>BLOCK: RX-EQ MODEL: REQUIL</td>
<td>51</td>
</tr>
<tr>
<td>BLOCK: V-FLOW MODEL: FSPLIT</td>
<td>52</td>
</tr>
</tbody>
</table>

253
FLOWSHEET CONNECTIVITY BY STREAMS

<table>
<thead>
<tr>
<th>STREAM</th>
<th>SOURCE</th>
<th>DEST</th>
<th>STREAM</th>
<th>SOURCE</th>
<th>DEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>----</td>
<td>FD-MX</td>
<td>1</td>
<td>----</td>
<td>FD-MX</td>
</tr>
<tr>
<td>5</td>
<td>HP-SEP</td>
<td>V-FLOW</td>
<td>6</td>
<td>HP-SEP</td>
<td>L-FLOW</td>
</tr>
<tr>
<td>8</td>
<td>L-FLOW</td>
<td>FD-MX</td>
<td>9</td>
<td>L-FLOW</td>
<td>COL</td>
</tr>
<tr>
<td>7</td>
<td>V-FLOW</td>
<td>FD-MX</td>
<td>10</td>
<td>V-FLOW</td>
<td>----</td>
</tr>
<tr>
<td>11</td>
<td>COL</td>
<td>----</td>
<td>12</td>
<td>COL</td>
<td>----</td>
</tr>
<tr>
<td>3</td>
<td>FD-MX</td>
<td>RX-EQ</td>
<td>4</td>
<td>RX-EQ</td>
<td>HP-SEP</td>
</tr>
<tr>
<td>14</td>
<td>RX-EQ</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FLOWSHEET CONNECTIVITY BY BLOCKS

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>INLETS</th>
<th>OUTLETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP-SEP</td>
<td>4</td>
<td>5 6</td>
</tr>
<tr>
<td>L-FLOW</td>
<td>6</td>
<td>8 9</td>
</tr>
<tr>
<td>V-FLOW</td>
<td>5</td>
<td>7 10</td>
</tr>
<tr>
<td>COL</td>
<td>9</td>
<td>11 12</td>
</tr>
<tr>
<td>FD-MX</td>
<td>7 1 2 8</td>
<td>3</td>
</tr>
<tr>
<td>RX-EQ</td>
<td>3</td>
<td>4 14</td>
</tr>
</tbody>
</table>

CONVERGENCE STATUS SUMMARY

TEAR STREAM SUMMARY

<table>
<thead>
<tr>
<th>STREAM</th>
<th>MAXIMUM</th>
<th>VARIABLE</th>
<th>CONV</th>
<th>ID</th>
<th>ERROR</th>
<th>TOLERANCE</th>
<th>ERR/TOL</th>
<th>ID</th>
<th>STAT</th>
<th>BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>.50467E-07</td>
<td>.10612E-06</td>
<td>.47556</td>
<td>MASS ENTHALPY</td>
<td>#</td>
<td>SOLVER01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# = CONVERGED
* = NOT CONVERGED

FORTRAN BLOCK: F-1

SAMPLED VARIABLES:

F1 : TOTAL MASSFLOW IN STREAM 1 SUBSTREAM MIXED
F2 : TOTAL MASSFLOW IN STREAM 2 SUBSTREAM MIXED
F3 : TOTAL MASSFLOW IN STREAM 3 SUBSTREAM MIXED
F4 : TOTAL MASSFLOW IN STREAM 4 SUBSTREAM MIXED
F5 : TOTAL MASSFLOW IN STREAM 5 SUBSTREAM MIXED
F6 : TOTAL MASSFLOW IN STREAM 6 SUBSTREAM MIXED
F7 : TOTAL MASSFLOW IN STREAM 7 SUBSTREAM MIXED
F8 : TOTAL MASSFLOW IN STREAM 8 SUBSTREAM MIXED
F9 : TOTAL MASSFLOW IN STREAM 9 SUBSTREAM MIXED
F10 : TOTAL MASSFLOW IN STREAM 10 SUBSTREAM MIXED
F11 : TOTAL MASSFLOW IN STREAM 11 SUBSTREAM MIXED
F12 : TOTAL MASSFLOW IN STREAM 12 SUBSTREAM MIXED
T1 : TEMPERATURE IN STREAM 1 SUBSTREAM MIXED

255
FORTRAN BLOCK: F-1 (CONTINUED)

T2 : TEMPERATURE IN STREAM 2 SUBSTREAM MIXED
T3 : TEMPERATURE IN STREAM 3 SUBSTREAM MIXED
T4 : TEMPERATURE IN STREAM 4 SUBSTREAM MIXED
T5 : TEMPERATURE IN STREAM 5 SUBSTREAM MIXED
T6 : TEMPERATURE IN STREAM 6 SUBSTREAM MIXED
T7 : TEMPERATURE IN STREAM 7 SUBSTREAM MIXED
T8 : TEMPERATURE IN STREAM 8 SUBSTREAM MIXED
T9 : TEMPERATURE IN STREAM 9 SUBSTREAM MIXED
T10 : TEMPERATURE IN STREAM 10 SUBSTREAM MIXED
T11 : TEMPERATURE IN STREAM 11 SUBSTREAM MIXED
T12 : TEMPERATURE IN STREAM 12 SUBSTREAM MIXED
P1 : PRESSURE IN STREAM 1 SUBSTREAM MIXED
P2 : PRESSURE IN STREAM 2 SUBSTREAM MIXED
P3 : PRESSURE IN STREAM 3 SUBSTREAM MIXED
P4 : PRESSURE IN STREAM 4 SUBSTREAM MIXED
P5 : PRESSURE IN STREAM 5 SUBSTREAM MIXED
P6 : PRESSURE IN STREAM 6 SUBSTREAM MIXED
P7 : PRESSURE IN STREAM 7 SUBSTREAM MIXED
P8 : PRESSURE IN STREAM 8 SUBSTREAM MIXED
P9 : PRESSURE IN STREAM 9 SUBSTREAM MIXED
P10 : PRESSURE IN STREAM 10 SUBSTREAM MIXED
P11 : PRESSURE IN STREAM 11 SUBSTREAM MIXED
P12 : PRESSURE IN STREAM 12 SUBSTREAM MIXED
TC1 : SENTENCE=PROFILE VARIABLE=TEMP ID1=1 IN UOS BLOCK COL
TC2 : SENTENCE=PROFILE VARIABLE=TEMP ID1=2 IN UOS BLOCK COL
TC3 : SENTENCE=PROFILE VARIABLE=TEMP ID1=3 IN UOS BLOCK COL
TC4 : SENTENCE=PROFILE VARIABLE=TEMP ID1=4 IN UOS BLOCK COL
TC5 : SENTENCE=PROFILE VARIABLE=TEMP ID1=5 IN UOS BLOCK COL
TC6 : SENTENCE=PROFILE VARIABLE=TEMP ID1=6 IN UOS BLOCK COL
TC7 : SENTENCE=PROFILE VARIABLE=TEMP ID1=7 IN UOS BLOCK COL
TC8 : SENTENCE=PROFILE VARIABLE=TEMP ID1=8 IN UOS BLOCK COL
TC9 : SENTENCE=PROFILE VARIABLE=TEMP ID1=9 IN UOS BLOCK COL
TC10 : SENTENCE=PROFILE VARIABLE=TEMP ID1=10 IN UOS BLOCK COL
TC11 : SENTENCE=PROFILE VARIABLE=TEMP ID1=11 IN UOS BLOCK COL
TC12 : SENTENCE=PROFILE VARIABLE=TEMP ID1=12 IN UOS BLOCK COL
TC13 : SENTENCE=PROFILE VARIABLE=TEMP ID1=13 IN UOS BLOCK COL
TC14 : SENTENCE=PROFILE VARIABLE=TEMP ID1=14 IN UOS BLOCK COL
TC15 : SENTENCE=PROFILE VARIABLE=TEMP ID1=15 IN UOS BLOCK COL
QFD : SENTENCE=PARAM VARIABLE=QCALC IN UOS BLOCK FD-MX
QRX : SENTENCE=RESULTS VARIABLE=QCALC IN UOS BLOCK RX-EQ
QSEP : SENTENCE=PARAM VARIABLE=QCALC IN UOS BLOCK HP-SEP
QCBTM : SENTENCE=RESULTS VARIABLE=REB-DUTY IN UOS BLOCK COL
QCTOP : SENTENCE=RESULTS VARIABLE=COND-DUTY IN UOS BLOCK COL
XB1 : CH MOLEFRAC IN STREAM 12 SUBSTREAM MIXED
XB2 : BZ MOLEFRAC IN STREAM 12 SUBSTREAM MIXED
XB3 : H2 MOLEFRAC IN STREAM 12 SUBSTREAM MIXED
XB4 : CH4 MOLEFRAC IN STREAM 12 SUBSTREAM MIXED
XB5 : N2 MOLEFRAC IN STREAM 12 SUBSTREAM MIXED
FORTRAN STATEMENTS:
C  COMMON /USRFL0/ F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12
    CALL FWRITER(F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,
&    T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,
&    P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,
&    TC1,TC2,TC3,TC4,TC5,TC6,TC7,TC8,TC9,

ASLEN PLUS VER: HP-UX REL: 9.2-1 INST: MIT-AIX 04/17/96 PAGE 4
HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE
FLOWSHEET SECTION

FORTRAN BLOCK: F-1 (CONTINUED)
&    TC10,TC11,TC12,TC13,TC14,TC15,
&    QFD,QRX,QSEP,QCSTM,QCTOP,
&    XB1,XB2,XB3,XB4,XB5)

EXECUTE LAST

CONVERGENCE BLOCK: SOLVER01

----------------------------------
Tear Stream :  3
Tolerance used:  .100E-03
Trace molefrac:  .100E-05

MAXIT= 30 WAIT 1 ITERATIONS BEFORE ACCELERATING
QMAX = .00E+00 QMIN = -.50
METHOD: WEGSTEIN  STATUS: CONVERGED
TOTAL NUMBER OF ITERATIONS:  8

*** FINAL VALUES ***

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
<th>PREV VALUE</th>
<th>ERR/TOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.6626341E+03</td>
<td>.6626369E+03</td>
</tr>
<tr>
<td>H2 MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.4182196E+03</td>
<td>.4182221E+03</td>
</tr>
<tr>
<td>N2 MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.4509389E+01</td>
<td>.4509516E+01</td>
</tr>
<tr>
<td>CH4 MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.9327946E+02</td>
<td>.9328028E+02</td>
</tr>
<tr>
<td>BZ MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.1000001E+03</td>
<td>.1000001E+03</td>
</tr>
<tr>
<td>CH MOLEFLOW</td>
<td>LBMOL/HR</td>
<td>.4662561E+02</td>
<td>.4662501E+02</td>
</tr>
<tr>
<td>PRESSURE</td>
<td>PSI</td>
<td>.3300000E+03</td>
<td>.3300000E+03</td>
</tr>
<tr>
<td>MASS ENTHALPY</td>
<td>BTU/LB</td>
<td>.4562656E+02</td>
<td>.4562439E+02</td>
</tr>
</tbody>
</table>

*** ITERATION HISTORY ***

TEARSTREAMS:

<table>
<thead>
<tr>
<th>ITERATION</th>
<th>MAX-ERR/TOL</th>
<th>STREAM ID</th>
<th>VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1640.</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>2</td>
<td>143.0</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>3</td>
<td>19.20</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>4</td>
<td>-127.2</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>5</td>
<td>-44.96</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>6</td>
<td>12.38</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>7</td>
<td>-4.013</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
<tr>
<td>8</td>
<td>.4756</td>
<td>3</td>
<td>MASS ENTHALPY</td>
</tr>
</tbody>
</table>
COMPUTATIONAL SEQUENCE

------------------------

SEQUENCE USED WAS:
SOLVER01 RX-EQ HP-SEP V-FLOW L-FLOW FD-MX
(RETURN SOLVER01)
COL F-1

AS PEN PLUS VER: HP-UX REL: 9.2-1 INST: MIT-AIX 04/17/96 PAGE 5
HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE
FLOWSHEET SECTION

OVERALL FLOWSHEET BALANCE

------------------------

*** MASS AND ENERGY BALANCE ***
IN  OUT  GENERATION  RELATIVE DIFF.

CONVENTIONAL COMPONENTS
(LBMOL/HR)
H2  310.000  10.0028 -300.000  -.794973E-05
N2  .400000  .400128 -.860492E-15  -.319393E-03
CH4  10.0000  10.0008  .000000E+00  -.813090E-04
BZ  100.000  120.700E-03 -99.9999  .220710E-09
CH  .000000E+00  99.9999  99.9999  .598536E-05

TOTAL BALANCE
MOLE(LBMOL/HR)  420.400  120.403 -300.000  -.667654E-05
MASS(LB/HR)  860.92  860.89  .310531E-05
ENTHALPY(BTU/HR)  .197292E+07  -.550427E+07  1.35843

CASE-STUDY BLOCK

-----------------

CASE-STUDY VARIABLES:
VARY  1: H2 MOLECULAR FLOW IN STREAM 1 SUBSTREAM MIXED
VARY  2: CH4 MOLECULAR FLOW IN STREAM 1 SUBSTREAM MIXED
VARY  3: TOTAL MOLECULAR FLOW IN STREAM 2 SUBSTREAM MIXED
VARY  4: TEMPERATURE IN STREAM 1 SUBSTREAM MIXED
VARY  5: TEMPERATURE IN STREAM 2 SUBSTREAM MIXED
VARY  6: SENTENCE=PARAM VARIABLE=TEMP IN UOS BLOCK FD-MX
VARY  7: SENTENCE=PARAM VARIABLE=PRES IN UOS BLOCK FD-MX
VARY  8: SENTENCE=PARAM VARIABLE=TEMP IN UOS BLOCK RX-EQ
VARY  9: SENTENCE=PARAM VARIABLE=PRES IN UOS BLOCK RX-EQ
VARY 10: SENTENCE=PARAM VARIABLE=TEMP IN UOS BLOCK HP-SEP
VARY 11: SENTENCE=PARAM VARIABLE=PRES IN UOS BLOCK HP-SEP
VARY 12: SENTENCE=FRAC VARIABLE=FRAC ID1=10 IN UOS BLOCK V-FLOW
VARY 13: SENTENCE=FRAC VARIABLE=FRAC ID1=8 IN UOS BLOCK L-FLOW
VARY 14: SENTENCE=COL-SPECS VARIABLE=MOLE-RR IN UOS BLOCK COL
VARY 15: SENTENCE=COL-SPECS VARIABLE=B:F IN UOS BLOCK COL
VARY 16: SENTENCE=P-SPEC VARIABLE=PRES ID1=1 IN UOS BLOCK COL

CASE-STUDY VALUES:

258
CASE-STUDY BLOCK (CONTINUED)

### CASE NO. 1

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>329.252</td>
</tr>
<tr>
<td>2</td>
<td>10.3993</td>
</tr>
<tr>
<td>3</td>
<td>106.249</td>
</tr>
<tr>
<td>4</td>
<td>121.638</td>
</tr>
<tr>
<td>5</td>
<td>100.212</td>
</tr>
<tr>
<td>6</td>
<td>395.099</td>
</tr>
<tr>
<td>7</td>
<td>328.402</td>
</tr>
<tr>
<td>8</td>
<td>395.099</td>
</tr>
<tr>
<td>9</td>
<td>319.288</td>
</tr>
<tr>
<td>10</td>
<td>121.562</td>
</tr>
<tr>
<td>11</td>
<td>-5.72870</td>
</tr>
<tr>
<td>12</td>
<td>0.12690</td>
</tr>
<tr>
<td>13</td>
<td>0.41100</td>
</tr>
<tr>
<td>14</td>
<td>1.39320</td>
</tr>
<tr>
<td>15</td>
<td>0.95820</td>
</tr>
<tr>
<td>16</td>
<td>195.634</td>
</tr>
</tbody>
</table>

### CASE NO. 2

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>315.603</td>
</tr>
<tr>
<td>2</td>
<td>7.40230</td>
</tr>
<tr>
<td>3</td>
<td>102.435</td>
</tr>
<tr>
<td>4</td>
<td>119.624</td>
</tr>
<tr>
<td>5</td>
<td>98.4865</td>
</tr>
<tr>
<td>6</td>
<td>399.038</td>
</tr>
<tr>
<td>7</td>
<td>332.493</td>
</tr>
<tr>
<td>8</td>
<td>399.038</td>
</tr>
<tr>
<td>9</td>
<td>326.393</td>
</tr>
<tr>
<td>10</td>
<td>121.579</td>
</tr>
<tr>
<td>11</td>
<td>-5.40980</td>
</tr>
<tr>
<td>12</td>
<td>0.13090</td>
</tr>
<tr>
<td>13</td>
<td>0.39010</td>
</tr>
<tr>
<td>14</td>
<td>1.28810</td>
</tr>
<tr>
<td>15</td>
<td>0.95950</td>
</tr>
<tr>
<td>16</td>
<td>197.874</td>
</tr>
</tbody>
</table>

### CASE NO. 3

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>324.511</td>
</tr>
<tr>
<td>2</td>
<td>8.35740</td>
</tr>
<tr>
<td>3</td>
<td>102.119</td>
</tr>
<tr>
<td>4</td>
<td>116.034</td>
</tr>
<tr>
<td>5</td>
<td>98.4918</td>
</tr>
<tr>
<td>6</td>
<td>408.932</td>
</tr>
<tr>
<td>7</td>
<td>331.032</td>
</tr>
<tr>
<td>8</td>
<td>408.932</td>
</tr>
<tr>
<td>9</td>
<td>321.253</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>10</td>
<td>114.095</td>
</tr>
<tr>
<td>11</td>
<td>-6.33210</td>
</tr>
</tbody>
</table>

**ASPN PLUS VER: HP-UX REL: 9.2-1 INST: MIT-AIX 04/17/96 PAGE 7**

**HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE**

**FLOWSHEET SECTION**

**CASE-STUDY BLOCK (CONTINUED)**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.11210</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.41980</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.36770</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.91520</td>
<td>PSI</td>
</tr>
<tr>
<td>16</td>
<td>194.098</td>
<td>PSI</td>
</tr>
</tbody>
</table>

**CASE NO. 4**

**VARIABLE VALUE**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>512.175</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>2</td>
<td>16.3563</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>3</td>
<td>165.740</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>4</td>
<td>120.059</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>100.134</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>397.376</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>334.413</td>
<td>PSI</td>
</tr>
<tr>
<td>8</td>
<td>397.376</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>323.718</td>
<td>PSI</td>
</tr>
<tr>
<td>10</td>
<td>120.613</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>-9.89160</td>
<td>PSI</td>
</tr>
<tr>
<td>12</td>
<td>0.088300</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.28500</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.24530</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.95800</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>190.640</td>
<td>PSI</td>
</tr>
</tbody>
</table>

**CASE NO. 5**

**VARIABLE VALUE**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>607.164</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>2</td>
<td>19.1684</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>3</td>
<td>187.252</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>4</td>
<td>124.172</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>100.092</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>403.866</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>318.262</td>
<td>PSI</td>
</tr>
<tr>
<td>8</td>
<td>403.866</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>305.467</td>
<td>PSI</td>
</tr>
<tr>
<td>10</td>
<td>118.766</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>-10.8380</td>
<td>PSI</td>
</tr>
<tr>
<td>12</td>
<td>0.071800</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.28380</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.58260</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.97640</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>200.564</td>
<td>PSI</td>
</tr>
</tbody>
</table>

**CASE NO. 6**

**VARIABLE VALUE**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>330.509</td>
<td>LBMOL/HR</td>
</tr>
</tbody>
</table>

260
<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9.56720 LBMOL/HR</td>
</tr>
<tr>
<td>3</td>
<td>105.418 LBMOL/HR</td>
</tr>
<tr>
<td>4</td>
<td>115.314 F</td>
</tr>
<tr>
<td>5</td>
<td>98.5444 F</td>
</tr>
<tr>
<td>6</td>
<td>391.311 F</td>
</tr>
<tr>
<td>7</td>
<td>325.706 PSI</td>
</tr>
<tr>
<td>8</td>
<td>391.311 F</td>
</tr>
<tr>
<td>9</td>
<td>315.577 PSI</td>
</tr>
<tr>
<td>10</td>
<td>120.768 F</td>
</tr>
<tr>
<td>11</td>
<td>-8.61390 PSI</td>
</tr>
<tr>
<td>12</td>
<td>0.11040</td>
</tr>
<tr>
<td>13</td>
<td>0.42230</td>
</tr>
<tr>
<td>14</td>
<td>1.40620</td>
</tr>
<tr>
<td>15</td>
<td>0.93370</td>
</tr>
<tr>
<td>16</td>
<td>198.089 PSI</td>
</tr>
</tbody>
</table>

**CASE NO. 7**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>331.242 LBMOL/HR</td>
</tr>
<tr>
<td>2</td>
<td>7.83770 LBMOL/HR</td>
</tr>
<tr>
<td>3</td>
<td>107.003 LBMOL/HR</td>
</tr>
<tr>
<td>4</td>
<td>119.489 F</td>
</tr>
<tr>
<td>5</td>
<td>102.275 F</td>
</tr>
<tr>
<td>6</td>
<td>390.888 F</td>
</tr>
<tr>
<td>7</td>
<td>347.241 PSI</td>
</tr>
<tr>
<td>8</td>
<td>390.888 F</td>
</tr>
<tr>
<td>9</td>
<td>339.566 PSI</td>
</tr>
<tr>
<td>10</td>
<td>120.112 F</td>
</tr>
<tr>
<td>11</td>
<td>-5.610000 PSI</td>
</tr>
<tr>
<td>12</td>
<td>0.13460</td>
</tr>
<tr>
<td>13</td>
<td>0.38270</td>
</tr>
<tr>
<td>14</td>
<td>1.45420</td>
</tr>
<tr>
<td>15</td>
<td>0.93720</td>
</tr>
<tr>
<td>16</td>
<td>205.558 PSI</td>
</tr>
</tbody>
</table>

**CASE NO. 8**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>336.649 LBMOL/HR</td>
</tr>
<tr>
<td>2</td>
<td>9.20710 LBMOL/HR</td>
</tr>
<tr>
<td>3</td>
<td>106.389 LBMOL/HR</td>
</tr>
<tr>
<td>4</td>
<td>118.631 F</td>
</tr>
<tr>
<td>5</td>
<td>98.7685 F</td>
</tr>
<tr>
<td>6</td>
<td>398.173 F</td>
</tr>
<tr>
<td>7</td>
<td>331.540 PSI</td>
</tr>
<tr>
<td>8</td>
<td>398.173 F</td>
</tr>
<tr>
<td>9</td>
<td>324.272 PSI</td>
</tr>
<tr>
<td>10</td>
<td>120.028 F</td>
</tr>
<tr>
<td>11</td>
<td>-5.27480 PSI</td>
</tr>
<tr>
<td>12</td>
<td>0.12820</td>
</tr>
<tr>
<td>13</td>
<td>0.41190</td>
</tr>
</tbody>
</table>

261
### CASE NO. 9

**Variable**

<table>
<thead>
<tr>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>305.569</td>
</tr>
</tbody>
</table>

**CASE-STUDY BLOCK (CONTINUED)**

<table>
<thead>
<tr>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7.64340</td>
</tr>
<tr>
<td>3</td>
<td>99.4592</td>
</tr>
<tr>
<td>4</td>
<td>124.374</td>
</tr>
<tr>
<td>5</td>
<td>99.5092</td>
</tr>
<tr>
<td>6</td>
<td>392.222</td>
</tr>
<tr>
<td>7</td>
<td>327.672</td>
</tr>
<tr>
<td>8</td>
<td>392.222</td>
</tr>
<tr>
<td>9</td>
<td>319.845</td>
</tr>
<tr>
<td>10</td>
<td>122.479</td>
</tr>
<tr>
<td>11</td>
<td>-7.52400</td>
</tr>
<tr>
<td>12</td>
<td>0.13190</td>
</tr>
<tr>
<td>13</td>
<td>0.41330</td>
</tr>
<tr>
<td>14</td>
<td>1.43190</td>
</tr>
<tr>
<td>15</td>
<td>0.97380</td>
</tr>
<tr>
<td>16</td>
<td>204.169</td>
</tr>
</tbody>
</table>

### CASE NO. 10

**Variable**

<table>
<thead>
<tr>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>314.624</td>
</tr>
<tr>
<td>2</td>
<td>13.3443</td>
</tr>
<tr>
<td>3</td>
<td>101.073</td>
</tr>
<tr>
<td>4</td>
<td>109.072</td>
</tr>
<tr>
<td>5</td>
<td>98.8922</td>
</tr>
<tr>
<td>6</td>
<td>409.740</td>
</tr>
<tr>
<td>7</td>
<td>324.216</td>
</tr>
<tr>
<td>8</td>
<td>409.740</td>
</tr>
<tr>
<td>9</td>
<td>317.493</td>
</tr>
<tr>
<td>10</td>
<td>118.242</td>
</tr>
<tr>
<td>11</td>
<td>-8.70560</td>
</tr>
<tr>
<td>12</td>
<td>0.13060</td>
</tr>
<tr>
<td>13</td>
<td>0.37980</td>
</tr>
<tr>
<td>14</td>
<td>1.47580</td>
</tr>
<tr>
<td>15</td>
<td>0.91180</td>
</tr>
<tr>
<td>16</td>
<td>201.470</td>
</tr>
</tbody>
</table>

**CASE STUDIES WILL BEGIN WITH BLOCK: SOLVER01**
COMPONENTS

---------

<table>
<thead>
<tr>
<th>ID</th>
<th>TYPE</th>
<th>FORMULA</th>
<th>NAME OR ALIAS</th>
<th>REPORT NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>C</td>
<td>H2</td>
<td>H2</td>
<td>H2</td>
</tr>
<tr>
<td>N2</td>
<td>C</td>
<td>N2</td>
<td>N2</td>
<td>N2</td>
</tr>
<tr>
<td>CH4</td>
<td>C</td>
<td>CH4</td>
<td>CH4</td>
<td>CH4</td>
</tr>
<tr>
<td>BZ</td>
<td>C</td>
<td>C6H6</td>
<td>C6H6</td>
<td>BZ</td>
</tr>
<tr>
<td>CH</td>
<td>C</td>
<td>C6H12-1</td>
<td>C6H12-1</td>
<td>CH</td>
</tr>
</tbody>
</table>

BLOCK: COL  MODEL: RADFRAC

-------------------------------

COLUMN

INLETS  - 9  STAGE  8
OUTLETS - 11  STAGE  1
12  STAGE  15
PROPERTY OPTION SET: SYSOP3  REDLICH-KWONG-SOAVE EQUATION OF STATE

*** MASS AND ENERGY BALANCE ***

IN   OUT   RELATIVE DIFF.
TOTAL BALANCE
MOLE(LBMOL/HR)  103.185  103.185  .275445E-15
MASS(LB/HR )  8435.89  8435.89  .215625E-15
ENTHALPY(BTU/HR ) -.661536E+07  -.526286E+07  -.204449

********************************

***** INPUT DATA *****

********************************

***** INPUT PARAMETERS *****

NUMBER OF STAGES  15
ALGORITHM OPTION  STANDARD
ABSORBER OPTION  NO
INITIALIZATION OPTION  STANDARD
HYDRAULIC PARAMETER CALCULATIONS  NO
INSIDE LOOP CONVERGENCE METHOD  BROYDEN
DESIGN SPECIFICATION METHOD  NESTED
MAXIMUM NO. OF OUTSIDE LOOP ITERATIONS  25
MAXIMUM NO. OF INSIDE LOOP ITERATIONS  10
MAXIMUM NUMBER OF FLASH ITERATIONS  50
FLASH TOLERANCE  0.00010000
OUTSIDE LOOP CONVERGENCE TOLERANCE  0.00010000

263
**** COL-SPECS ****

MOLAR VAPOR DIST / TOTAL DIST 1.00000
MOLAR REFLUX RATIO 1.20000
BOTTOMS TO FEED RATIO 0.98000

ASPN PLUS  VER: HP-UX  REL: 9.2-1  INST: MIT-AIX  04/17/96 PAGE 12
HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE
U-O-S BLOCK SECTION

BLOCK: COL MODEL: RADFRAC (CONTINUED)

**** PROFILES ****

P-SPEC STAGE 1 PRES, PSI 200.000

***********************
**** RESULTS ****
***********************

*** COMPONENT SPLIT FRACTIONS ***

OUTLET STREAMS

11 12
COMPONENT:
H2 1.0000 .76252E-10
N2 1.0000 .24040E-09
CH4 1.0000 .15262E-07
BZ .28232E-01 .97177
CH .20001E-01 .98000

*** SUMMARY OF KEY RESULTS ***

TOP STAGE TEMPERATURE F 282.277
BOTTOM STAGE TEMPERATURE F 402.070
TOP STAGE LIQUID FLOW LBMOL/HR 6.62284
BOTTOM STAGE LIQUID FLOW LBMOL/HR 97.6657
TOP STAGE VAPOR FLOW LBMOL/HR 5.51903
BOTTOM STAGE VAPOR FLOW LBMOL/HR 161.675
MOLAR REFLUX RATIO 1.20000
MOLAR BOILUP RATIO 1.65540
CONDENSER DUTY (W/O SUBCOOL) BTU/HR -94,041.9
REBOILER DUTY BTU/HR 1,446,550.

**** MAXIMUM FINAL RELATIVE ERRORS ****

DEW POINT .16677E-04 STAGE= 5
BUBBLE POINT .14801E-03 STAGE= 8
COMPONENT MASS BALANCE .54388E-05 STAGE= 2 COMP= H2
ENERGY BALANCE .22341E-03 STAGE= 1
**** PROFILES ****

**NOTE** REPORTED VALUES FOR STAGE LIQUID AND VAPOR RATES ARE THE FLOWS FROM THE STAGE EXCLUDING ANY SIDE PRODUCT. FOR THE FIRST STAGE, THE REPORTED VAPOR FLOW IS THE VAPOR DISTILLATE FLOW. FOR THE LAST STAGE, THE REPORTED LIQUID FLOW IS THE LIQUID BOTTOMS FLOW.

<table>
<thead>
<tr>
<th>STAGE TEMPERATURE</th>
<th>PRESSURE</th>
<th>ENTHALPY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>282.28</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>347.80</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>358.32</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>358.84</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>395.61</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>400.91</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>401.49</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>402.07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAGE FLOW RATE</th>
<th>FEED RATE</th>
<th>PRODUCT RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBMOL/HR</td>
<td>LBMOL/HR</td>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>LIQUID VAPOR</td>
<td>LIQUID VAPOR</td>
<td>MIXED LIQUID VAPOR</td>
</tr>
<tr>
<td>1</td>
<td>6.623</td>
<td>5.519</td>
</tr>
<tr>
<td>2</td>
<td>8.466</td>
<td>12.14</td>
</tr>
<tr>
<td>7</td>
<td>8.810</td>
<td>14.32</td>
</tr>
<tr>
<td>8</td>
<td>207.9</td>
<td>14.33</td>
</tr>
<tr>
<td>9</td>
<td>251.5</td>
<td>110.2</td>
</tr>
<tr>
<td>13</td>
<td>258.7</td>
<td>160.3</td>
</tr>
<tr>
<td>14</td>
<td>259.3</td>
<td>161.0</td>
</tr>
<tr>
<td>15</td>
<td>97.67</td>
<td>161.7</td>
</tr>
</tbody>
</table>

**** MASS FLOW PROFILES ****

<table>
<thead>
<tr>
<th>STAGE FLOW RATE</th>
<th>FEED RATE</th>
<th>PRODUCT RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB/HR</td>
<td>LB/HR</td>
<td>LB/HR</td>
</tr>
<tr>
<td>LIQUID VAPOR</td>
<td>LIQUID VAPOR</td>
<td>MIXED LIQUID VAPOR</td>
</tr>
<tr>
<td>1</td>
<td>547.8</td>
<td>216.2</td>
</tr>
<tr>
<td>2</td>
<td>705.8</td>
<td>764.0</td>
</tr>
<tr>
<td>7</td>
<td>735.3</td>
<td>951.0</td>
</tr>
<tr>
<td>8</td>
<td>.1735E+05</td>
<td>951.5</td>
</tr>
<tr>
<td>9</td>
<td>.2115E+05</td>
<td>9129.</td>
</tr>
<tr>
<td>13</td>
<td>.2177E+05</td>
<td>.1349E+05</td>
</tr>
<tr>
<td>14</td>
<td>.2183E+05</td>
<td>.1355E+05</td>
</tr>
<tr>
<td>15</td>
<td>8220.</td>
<td>.1361E+05</td>
</tr>
</tbody>
</table>

265
## BLOCK: COL  MODEL: RADFRAC (CONTINUED)

### **** MOLE-X-PROFILE ****

<table>
<thead>
<tr>
<th>STAGE</th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1861E-02</td>
<td>0.14002E-03</td>
<td>0.19395E-01</td>
<td>0.15657E-05</td>
<td>0.97900</td>
</tr>
<tr>
<td>2</td>
<td>0.9517E-03</td>
<td>0.84729E-04</td>
<td>0.10408E-01</td>
<td>0.15044E-05</td>
<td>0.98854</td>
</tr>
<tr>
<td>7</td>
<td>0.8918E-03</td>
<td>0.77439E-04</td>
<td>0.91959E-02</td>
<td>0.13105E-05</td>
<td>0.98983</td>
</tr>
<tr>
<td>8</td>
<td>0.8989E-03</td>
<td>0.78012E-04</td>
<td>0.92531E-02</td>
<td>0.12941E-05</td>
<td>0.98977</td>
</tr>
<tr>
<td>9</td>
<td>0.4643E-04</td>
<td>0.46466E-05</td>
<td>0.92480E-03</td>
<td>0.13213E-05</td>
<td>0.99902</td>
</tr>
<tr>
<td>13</td>
<td>0.21712E-09</td>
<td>0.37751E-10</td>
<td>0.57286E-07</td>
<td>0.12754E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.10261E-10</td>
<td>0.20446E-11</td>
<td>0.51281E-08</td>
<td>0.12441E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.48044E-12</td>
<td>0.10940E-12</td>
<td>0.44786E-09</td>
<td>0.11967E-05</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

### **** MOLE-Y-PROFILE ****

<table>
<thead>
<tr>
<th>STAGE</th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.11150</td>
<td>0.80532E-02</td>
<td>0.51928</td>
<td>0.61525E-06</td>
<td>0.36117</td>
</tr>
<tr>
<td>2</td>
<td>0.51478E-01</td>
<td>0.37369E-02</td>
<td>0.24662</td>
<td>0.11337E-05</td>
<td>0.69817</td>
</tr>
<tr>
<td>7</td>
<td>0.43508E-01</td>
<td>0.31504E-02</td>
<td>0.20571</td>
<td>0.10558E-05</td>
<td>0.74763</td>
</tr>
<tr>
<td>8</td>
<td>0.43492E-01</td>
<td>0.31493E-02</td>
<td>0.20566</td>
<td>0.10427E-05</td>
<td>0.74770</td>
</tr>
<tr>
<td>9</td>
<td>0.16955E-02</td>
<td>0.14714E-03</td>
<td>0.17452E-01</td>
<td>0.13804E-05</td>
<td>0.98070</td>
</tr>
<tr>
<td>13</td>
<td>0.16481E+00</td>
<td>0.11284E-08</td>
<td>0.10532E-05</td>
<td>0.12356E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.34854E-09</td>
<td>0.60585E-10</td>
<td>0.91764E-07</td>
<td>0.12323E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.16169E-10</td>
<td>0.32137E-11</td>
<td>0.79554E-08</td>
<td>0.12272E-05</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

### **** K-VALUES ****

<table>
<thead>
<tr>
<th>STAGE</th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.76292</td>
<td>0.57512</td>
<td>0.26773</td>
<td>0.39297</td>
<td>0.36891</td>
</tr>
<tr>
<td>2</td>
<td>0.53542</td>
<td>0.44100</td>
<td>0.23693</td>
<td>0.75357</td>
<td>0.70626</td>
</tr>
<tr>
<td>7</td>
<td>0.48815</td>
<td>0.40705</td>
<td>0.22380</td>
<td>0.80561</td>
<td>0.75530</td>
</tr>
<tr>
<td>8</td>
<td>0.48415</td>
<td>0.40396</td>
<td>0.22238</td>
<td>0.80572</td>
<td>0.75544</td>
</tr>
<tr>
<td>9</td>
<td>0.36508</td>
<td>0.31662</td>
<td>0.18869</td>
<td>1.0447</td>
<td>0.98166</td>
</tr>
<tr>
<td>13</td>
<td>0.34286</td>
<td>0.29891</td>
<td>0.18027</td>
<td>1.0637</td>
<td>1.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.33968</td>
<td>0.29631</td>
<td>0.17894</td>
<td>1.0636</td>
<td>1.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.33654</td>
<td>0.29374</td>
<td>0.17763</td>
<td>1.0635</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

### **** MASS-X-PROFILE ****

<table>
<thead>
<tr>
<th>STAGE</th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.35617E-04</td>
<td>0.47422E-04</td>
<td>0.37619E-02</td>
<td>0.14786E-05</td>
<td>0.99615</td>
</tr>
<tr>
<td>2</td>
<td>0.2324E-04</td>
<td>0.28470E-04</td>
<td>0.20028E-02</td>
<td>0.14095E-05</td>
<td>0.99794</td>
</tr>
<tr>
<td>7</td>
<td>0.21541E-04</td>
<td>0.25993E-04</td>
<td>0.17677E-02</td>
<td>0.12266E-05</td>
<td>0.99818</td>
</tr>
<tr>
<td>8</td>
<td>0.21715E-04</td>
<td>0.26187E-04</td>
<td>0.17788E-02</td>
<td>0.12113E-05</td>
<td>0.99817</td>
</tr>
<tr>
<td>9</td>
<td>0.11131E-05</td>
<td>0.15479E-05</td>
<td>0.17643E-03</td>
<td>0.12273E-05</td>
<td>0.99982</td>
</tr>
<tr>
<td>13</td>
<td>0.52006E-11</td>
<td>0.12566E-10</td>
<td>0.10920E-07</td>
<td>0.11838E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>14</td>
<td>0.24577E-12</td>
<td>0.68057E-12</td>
<td>0.97752E-09</td>
<td>0.11547E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>15</td>
<td>0.11508E-13</td>
<td>0.36416E-13</td>
<td>0.85370E-10</td>
<td>0.11107E-05</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
**** MASS-Y-PROFILE ****

<table>
<thead>
<tr>
<th>STAGE</th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.57372E-02</td>
<td>.57584E-02</td>
<td>.21264</td>
<td>.12267E-05</td>
<td>.77586</td>
</tr>
<tr>
<td>2</td>
<td>.16492E-02</td>
<td>.16637E-02</td>
<td>.62876E-01</td>
<td>.14073E-05</td>
<td>.93381</td>
</tr>
<tr>
<td>7</td>
<td>.13209E-02</td>
<td>.13292E-02</td>
<td>.49704E-01</td>
<td>.12421E-05</td>
<td>.94764</td>
</tr>
<tr>
<td>8</td>
<td>.13204E-02</td>
<td>.13286E-02</td>
<td>.49686E-01</td>
<td>.12266E-05</td>
<td>.94766</td>
</tr>
<tr>
<td>9</td>
<td>.41267E-04</td>
<td>.49765E-04</td>
<td>.33804E-02</td>
<td>.13019E-05</td>
<td>.99653</td>
</tr>
<tr>
<td>13</td>
<td>.17831E-09</td>
<td>.37560E-09</td>
<td>.19685E-06</td>
<td>.12592E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>14</td>
<td>.83484E-11</td>
<td>.20166E-10</td>
<td>.17492E-07</td>
<td>.12281E-05</td>
<td>1.0000</td>
</tr>
<tr>
<td>15</td>
<td>.38729E-12</td>
<td>.10697E-11</td>
<td>.15164E-08</td>
<td>.11813E-05</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

ASPN PLUS VER: HP-UX REL: 9.2-1 INST: MIT-AIX 04/17/96 PAGE 15
HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE
U-O-S BLOCK SECTION

BLOCK: FD-MX MODEL: HEATER

---------------------------------------------------
FEED-MIX
INLET STREAMS:  7  1  2  8
OUTLET STREAM:  3
PROPERTY OPTION SET:  SYSOP3 REDLICH-KWONG-SOAVE EQUATION OF STATE

**** MASS AND ENERGY BALANCE ****

<table>
<thead>
<tr>
<th>TOTAL BALANCE</th>
<th>IN</th>
<th>OUT</th>
<th>RELATIVE DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOLE(LBMOL/HR)</td>
<td>662.634</td>
<td>662.637</td>
<td>-423583E-05</td>
</tr>
<tr>
<td>MASS(LB/HR)</td>
<td>14201.3</td>
<td>14201.3</td>
<td>202595E-05</td>
</tr>
<tr>
<td>ENTHALPY(BTU/HR)</td>
<td>-.363843E+07</td>
<td>647965.0</td>
<td>-1.17809</td>
</tr>
</tbody>
</table>

**** INPUT DATA ****

TWO PHASE TP FLASH
SPECIFIED TEMPERATURE  F  400.00
SPECIFIED PRESSURE    PSI  330.00
MAXIMUM NO. ITERATIONS 30
CONVERGENCE TOLERANCE  0.00010000

**** RESULTS ****

OUTLET TEMPERATURE  F  400.00
OUTLET PRESSURE    PSI  330.00
HEAT DUTY          BTU/HR  .42864E+07
OUTLET VAPOR FRACTION  1.0000

V-L PHASE EQUILIBRIUM:

<table>
<thead>
<tr>
<th>COMP</th>
<th>F(I)</th>
<th>X(I)</th>
<th>Y(I)</th>
<th>K(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>.63115</td>
<td>.59662E-01</td>
<td>.63115</td>
<td>27.410</td>
</tr>
<tr>
<td>N2</td>
<td>.68052E-02</td>
<td>.75462E-03</td>
<td>.68052E-02</td>
<td>23.369</td>
</tr>
<tr>
<td>CH4</td>
<td>.14077</td>
<td>.26615E-01</td>
<td>.14077</td>
<td>13.731</td>
</tr>
<tr>
<td>BZ</td>
<td>.15091</td>
<td>.60394</td>
<td>.15091</td>
<td>.65414</td>
</tr>
<tr>
<td>CH</td>
<td>.70364E-01</td>
<td>.30903</td>
<td>.70364E-01</td>
<td>.59572</td>
</tr>
</tbody>
</table>

267
** BLOCK: HP-SEP  MODEL: FLASH2 **

---

** HP-SEP **
- **INLET STREAM**: 4
- **OUTLET VAPOR STREAM**: 5
- **OUTLET LIQUID STREAM**: 6
- **PROPERTY OPTION SET**: SYSOP3  REDLICH-KWONG-SOAVE EQUATION OF STATE

---

** *** MASS AND ENERGY BALANCE *** ***

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>OUT</th>
<th>RELATIVE DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td>** TOTAL BALANCE**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOLE (LB/MOL/HR)</td>
<td>362.637</td>
<td>362.637</td>
<td>-.156750E-15</td>
</tr>
<tr>
<td>MASS (LB/HR)</td>
<td>14201.3</td>
<td>14201.3</td>
<td>.735912E-09</td>
</tr>
<tr>
<td>ENTHALPY (BTU/HR)</td>
<td>-.869306E+07</td>
<td>-.124681E+08</td>
<td>.302776</td>
</tr>
</tbody>
</table>

---

** *** INPUT DATA *** ***

- **TWO PHASE TP FLASH**
  - SPECIFIED TEMPERATURE F: 120.00
  - PRESSURE DROP PSI: 5.00000
  - MAXIMUM NO. ITERATIONS: 30
  - CONVERGENCE TOLERANCE: 0.00010000

---

** *** RESULTS *** ***

- **OUTLET TEMPERATURE F**: 120.00
- **OUTLET PRESSURE PSI**: 315.00
- **HEAT DUTY BTU/HR**: -.37744E+07
- **VAPOR FRACTION**: .59351

---

** V-L PHASE EQUILIBRIUM :**

<table>
<thead>
<tr>
<th>COMP</th>
<th>F(I)</th>
<th>X(I)</th>
<th>Y(I)</th>
<th>K(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>.32601</td>
<td>.59637E-02</td>
<td>.54520</td>
<td>91.419</td>
</tr>
<tr>
<td>N2</td>
<td>.12435E-01</td>
<td>.43074E-03</td>
<td>.20657E-01</td>
<td>47.957</td>
</tr>
<tr>
<td>CH4</td>
<td>.25723</td>
<td>.27775E-01</td>
<td>.41437</td>
<td>14.919</td>
</tr>
<tr>
<td>BZ</td>
<td>.48846E-06</td>
<td>.11656E-05</td>
<td>.24683E-07</td>
<td>.21176E-01</td>
</tr>
<tr>
<td>CH</td>
<td>.40433</td>
<td>.96583</td>
<td>.19770E-01</td>
<td>.20470E-01</td>
</tr>
</tbody>
</table>

---

268
BLOCK: L-FLOW  MODEL: FSPLIT

L-FLOW
INLET STREAM:  6
OUTLET STREAMS: 8  9
PROPERTY OPTION SET: SYSOP3  REDLICH-KWONG-SOAVE EQUATION OF STATE

*** MASS AND ENERGY BALANCE ***
IN    OUT    RELATIVE DIFF.
TOTAL BALANCE
MOLE(LBMOL/HR)   147.407 147.407   .000000E+00
MASS(LB/HR)     12051.3 12051.3   .000000E+00
ENTHALPY(BTU/HR) -.945052E+07 -.945052E+07   .000000E+00

*** INPUT DATA ***
FRACTION OF FLOW
STRM=8  FRAC= 0.30000

*** RESULTS ***
STREAM= 8  SPLIT= 0.30000  KEY= 0
         0.70000  0

BLOCK: RX-EQ  MODEL: REQUIL

INLET STREAM:  3
OUTLET VAPOR STREAM: 4
OUTLET LIQUID STREAM: 14
PROPERTY OPTION SET: SYSOP3  REDLICH-KWONG-SOAVE EQUATION OF STATE

*** MASS AND ENERGY BALANCE ***
IN    OUT    GENERATION    RELATIVE DIFF.
TOTAL BALANCE
MOLE(LBMOL/HR)  662.637 362.637  -300.000  -.171567E-15
MASS(LB/HR)    14201.3 14201.3   -.144449E-06
ENTHALPY(BTU/HR)  647965. -.869306E+07   1.07454

269
BLOCK: RX-EQ  MODEL: REQUIL (CONTINUED)

*** INPUT DATA ***
ONE PHASE TP FLASH SPECIFIED PHASE IS VAPOR
SPECIFIED TEMPERATURE F  400.000
SPECIFIED PRESSURE  PSI  320.000
MAXIMUM NO. ITERATIONS  30
CONVERGENCE TOLERANCE  0.00010000
LIQUID ENTRAINMENT  0.00000E+00
SOLID SPLIT FRACTIONS:
SUBSTREAM NO. = 1 MIXED SUBSTREAM, NO SOLID SPLITS.

*** RESULTS ***
OUTPUT TEMPERATURE  F  400.00
OUTPUT PRESSURE  PSI  320.00
HEAT DUTY BTU/HR  -.93416E+07
VAPOR FRACTION  1.0000

BLOCK: V-FLOW  MODEL: FSPLIT

-----------------------------
V-FLOW
INLET STREAM:  5
OUTLET STREAMS:  7 10
PROPERTY OPTION SET: SYSO3 REDLICH-KWONG-Soave EQUATION OF STATE

*** MASS AND ENERGY BALANCE ***
IN   OUT   RELATIVE DIFF.
TOTAL BALANCE
MOLE(LBMOL/HR) 215.231 215.231 .132052E-15
MASS(LB/HR )  2150.00 2150.00 .211510E-15
ENTHALPY(BTU/HR ) -.301760E+07 -.301760E+07 .000000E+00

*** INPUT DATA ***
FRACTION OF FLOW  STRM=10 FRAC= 0.080000

ASPE PLUS  VER: HP-UX  REL: 9.2-1  INST: MIT-AIX  04/17/96 PAGE 19
HYDROGENATION OF BENZENE TO FORM CYCLOHEXANE
U-O-S BLOCK SECTION

BLOCK: V-FLOW  MODEL: FSPLIT (CONTINUED)

*** RESULTS ***
STREAM= 7  SPLIT= 0.92000  KEY= 0
 10  0.080000  0

270
HYPHENATION OF BENZENE TO FORM CYCLOHEXANE
STREAM SECTION

110 11 12 14

(note stream 14 is liquid product from reactor which is a
required output, but in this case there is no liquid output
so stream is classed "missing")

<table>
<thead>
<tr>
<th>STREAM ID</th>
<th>1</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM :</td>
<td>---</td>
<td>V-FLOW</td>
<td>COL</td>
<td>COL</td>
<td>RX-EQ</td>
</tr>
<tr>
<td>TO :</td>
<td>FD-MX</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

SUBSTREAM: MIXED

PHASE: VAPOR VAPOR MIXED LIQUID MISSING

COMPONENTS: LBMOL/HR

<table>
<thead>
<tr>
<th></th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>310.0000</td>
<td>9.3874</td>
<td>0.615</td>
<td>4.6923-11</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4000</td>
<td>0.3556</td>
<td>4.4446-02</td>
<td>1.0685-11</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>10.0000</td>
<td>7.1348</td>
<td>2.8659</td>
<td>4.3740-08</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4.2500-07</td>
<td>3.3956-06</td>
<td>1.1688-04</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.3404</td>
<td>1.9932</td>
<td>97.6656</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

COMPONENTS: MOLE FRACT

<table>
<thead>
<tr>
<th></th>
<th>H2</th>
<th>N2</th>
<th>CH4</th>
<th>BZ</th>
<th>CH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9675</td>
<td>0.5452</td>
<td>0.1115</td>
<td>4.8044-13</td>
<td>0.0</td>
</tr>
<tr>
<td>1.2484-03</td>
<td>2.0657-02</td>
<td>8.0532-03</td>
<td>1.0940-13</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>3.1211-02</td>
<td>0.4143</td>
<td>0.5192</td>
<td>4.4786-10</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2.4683-08</td>
<td>6.1525-07</td>
<td>1.1967-06</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>1.9770-02</td>
<td>0.3611</td>
<td>1.0000</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

TOTAL FLOW:

| LBMOL/HR | 320.4000 | 17.2184 | 5.5190 | 97.6657 | 0.0 |
| LB/HR | 796.5557 | 172.0001 | 216.2214 | 8219.6714 | 0.0 |
| CUFT/HR | 6030.4328 | 340.7538 | 208.2817 | 261.5547 | 0.0 |

STATE VARIABLES:

| TEMP F | 120.0000 | 120.0000 | 282.2769 | 402.0697 | 0.0 |
| PRES PSI | 335.0000 | 315.0000 | 200.0000 | 215.0000 | 320.0000 |
| VFRAC | 1.0000 | 1.0000 | 0.9999 | 0.0 | MISSING |
| LFRAC | 0 | 0 | 7.3045-05 | 1.0000 | MISSING |
| SFRAC | 0 | 0 | 0 | 0 | MISSING |

ENTHALPY:

| BTU/LBMOL | -692.9518 | -1.4040+04 | -3.2626+04 | -5.2014+04 | MISSING |
| BTU/LB | -278.7272 | -1.403.5363 | -832.2931 | -618.3832 | MISSING |
| BTU/HR | -2.2291+05 | -2.4126+05 | -1.8049+05 | -5.0873+06 | MISSING |

ENTROPY:

| BTU/LBMOL-R | -5.9920 | -1.42837 | -52.9834 | -125.1363 | MISSING |
| BTU/LB-R | -2.4101 | -1.4299 | -1.3524 | -1.4868 | MISSING |

DENSITY:

<p>| LBMOL/CUFT | 5.3131-02 | 5.0530-02 | 2.6498-02 | 0.3734 | MISSING |
| LB/CUFT | 0.1320 | 0.5047 | 1.0381 | 31.4261 | MISSING |
| AVG MW | 2.4861 | 9.9893 | 39.1774 | 84.1612 | MISSING |</p>
<table>
<thead>
<tr>
<th>STREAM ID</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM:</td>
<td>FD-MX</td>
<td>RX-EQ</td>
<td>HP-SEP</td>
<td>HP-SEP</td>
<td></td>
</tr>
<tr>
<td>TO:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V-FLOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L-FLOW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBSTREAM: MIXED</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENTS: LBMOL/HR</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>H2</td>
</tr>
<tr>
<td>N2</td>
</tr>
<tr>
<td>CH4</td>
</tr>
<tr>
<td>BZ</td>
</tr>
<tr>
<td>CH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPONENTS: MOLE FRAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
</tr>
<tr>
<td>N2</td>
</tr>
<tr>
<td>CH4</td>
</tr>
<tr>
<td>BZ</td>
</tr>
<tr>
<td>CH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOTAL FLOW:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBMOL/HR</td>
</tr>
<tr>
<td>LB/HR</td>
</tr>
<tr>
<td>CUFT/HR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STATE VARIABLES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP F</td>
</tr>
<tr>
<td>PRES PSI</td>
</tr>
<tr>
<td>VFRAC</td>
</tr>
<tr>
<td>LFRAC</td>
</tr>
<tr>
<td>SFRAC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENTHALPY:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTU/LBMOL</td>
</tr>
<tr>
<td>BTU/LB</td>
</tr>
<tr>
<td>BTU/HR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENTROPY:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTU/LB-R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DENSITY:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBMOL/CUFT</td>
</tr>
<tr>
<td>LB/CUFT</td>
</tr>
<tr>
<td>AVG MW</td>
</tr>
</tbody>
</table>
STREAM ID
FROM: V-FLOW  L-FLOW  L-FLOW
TO: FD-MX  FD-MX  COL

SUBSTREAM: MIXED
PHASE: VAPOR LIQUID LIQUID

COMPONENTS: LBMOL/HR
H2  107.9558  0.2637  0.6153
N2  4.0903  1.9048-02  4.4446-02
CH4  82.0512  1.2282  2.8639
BZ  4.8876-06  5.1547-05  1.2028-04
CH  3.9146  42.7109  99.6589

COMPONENTS: MOLE FRAC
H2  0.5452  5.9637-03  5.9637-03
N2  2.0657-02  4.3074-04  4.3074-04
CH4  0.4143  2.7775-02  2.7775-02
BZ  2.4683-08  1.1656-06  1.1656-06
CH  1.9770-02  0.9658  0.9658

TOTAL FLOW:
LBMOL/HR  198.0120  44.2220  103.1847
LB/HR  1978.0013  3615.3826  8435.8929
CUFT/HR  3918.6697  83.3250  194.4250

STATE VARIABLES:
TEMP F  120.0000  120.0000  120.0000
PRES PSI  315.0000  315.0000  315.0000
VFRAC  1.0000  0.0  0.0
LFRAC  0.0  1.0000  1.0000
SFAC  0.0  0.0  0.0

ENTHALPY:
BTU/LBMOL  -1.4020+04  -6.4112+04  -6.4112+04
BTU/LB  -1403.5340  -784.1921  -784.1921
BTU/HR  -2.7762+06  -2.8352+06  -6.6154+06

ENTROPY:
BTU/LBMOL-R  -14.2837  -139.4773  -139.4773
BTU/LB-R  -1.4299  -1.7060  -1.7060

DENSITY:
LBMOL/CUFT  5.0530-02  0.5307  0.5307
LB/CUFT  0.5947  43.3889  43.3889
AVG MW  9.9893  81.7552  81.7552
BLOCK STATUS

- CALCULATIONS WERE COMPLETED NORMALLY
- ALL UNIT OPERATION BLOCKS WERE COMPLETED NORMALLY
- ALL STREAMS WERE FLASHED NORMALLY
- ALL CONVERGENCE BLOCKS WERE COMPLETED NORMALLY
- ALL FORTRAN BLOCKS WERE COMPLETED NORMALLY
- THE CASE STUDY BLOCK HAS COMPLETED 0 CASE(S)
Appendix B.3  Case Study MATLAB Source Code

new_CH_casedat.m  - generates input data for ASPENPLUS code from specified distributions
make_VP.m  - computes valve positions from heat duty data
get_stm_F.m  - calculates steam flow from heat duty
get_stm_VP.m  - calculates valve position from steam flow
get_wat_F.m  - calculates cooling water flow from heat duty
get_wat_VP.m  - calculates valve position from cooling water flow
function [indat]=new_CH_casedat(num)

% [indat]=new_CH_casedat(num)
% 
% This function makes the matrix indat which is then sent to infilewrt.m
% to make the case study input portion of the ASPEN input file for the
% Cyclohexane from Benzene case study. num is
% the number of cases to generate.
% 
ui=rand(num,1);  
for i=1:num
    uui=ui(i);  
    TH2=randn(1,1)*4+120;  
    TBZ=randn(1,1)*4+100;  
    TInRx=randn(1,1)*5+400;  
    PInRx=randn(1,1)*7+330;  
    TX=TInRx;  
    TFish=randn(1,1)*4+120;  
    RR= 1.2+(abs(randn(1,1)*.2));  
    BF= 0.98-rand(1,1)*.08;  
    if uui < 0.4
        FBZ=randn(1,1)*10+100;  
        FH2=(abs(randn(1,1)*0.13)+3.05)*FBZ;  
        FCH4=randn(1,1)*2+10;  
        Prg_sp=randn(1,1)*0.01+.13;  
        Rcy_sp=randn(1,1)*0.02+0.4;  
        PRx=PInRx-(5+abs(randn(1,1)*3));  
        FFlsh=-(5+abs(randn(1,1)*3));  
    else
        FBZ=randn(1,1)*10+180;  
        FH2=(rand(1,1)*0.3+3.05)*FBZ;  
        FCH4=randn(1,1)*2+18;  
        Prg_sp=randn(1,1)*0.01+.08;  
        Rcy_sp=randn(1,1)*0.02+0.3;  
        PRx=PInRx-(10+abs(randn(1,1)*3));  
        FFlsh=-(8+abs(randn(1,1)*3));  
    end
end
Pcol=200+randn(1,1)*5;  
indat(i,:)=indat(i,:)+(i FH2 FCH4 FBZ TH2 TBZ TInRx PInRx TRx PRx TFish FFlsh Prg_sp Rcy_sp RR BF Pcol);  
clear FH2 FCH4 FBZ TH2 TBZ Prg_sp Rcy_sp PRx TRx TInRx PInRx TFish FFlsh  
clear RR BF Pcol

eend
function [VP]=make_VP(Q,F);

% Gets Valve positions using calculated heat duty. F is used to determine if plant
% Is at high or low production rate

[n,dum]=size(F);
cutoff=180;
min_Sp=560; %Plant steam pres. (psi) at HIGH prod. rate (low due to usage)
max_Sp=600; %Plant steam pres. (psi) at LOW prod. rate (high, less steam used)
std_Sp=10; %Steam pres. std. dev
min_Wp=200; %Cooling water header pres. (psi) at HIGH prod. rate
max_Wp=170; %Cooling water header pres. (psi) at LOW prod. rate
std_Wp=5; %Cooling water pres. std. dev.
mean_Wt=50; %Cooling Water Mean Temperature
std_Wt=4; %Cooling Water temperature std. dev.
max_Tout=200; %Exit Water Temp for Reactor cooling (boiler water pre-heater)

% VALVE POSITION CONSTANTS

cst_vp1=7.45; % for preheater exch. 1e7 Btu/hr for del_P=500 and 90% open
cst_vp2=120; % for reactor cooler

cst_vp3=100; % for separator exch.
cst_vp4=2.137; % for column reboiler 3e6 Btu/hr for del_P=500 and 90% open
cst_vp5=2.0; % for column condensor 1.6e5 Btu/hr for del_P=150 and 90% open

for i=1:n
    WtT=randn(1,1)*std_WtT+mean_WtT;
    QQ=Q(i,:);
    FF=F(i,2);
    if FF > cutoff % Plant running at HIGH rate
        SpT=randn(1,1)*std_SpT+min_SpT; % thus steam header pressure low
        WpT=randn(1,1)*std_WpT+min_WpT; % and Cooling Water header pressure low
    else
        SpT=randn(1,1)*std_SpT+max_SpT;
        WpT=randn(1,1)*std_WpT+max_WpT;
    end
    SpT1=get_stm_F(QQ(:,1),SpT);
    WtT2=get_wat_F(QQ(:,2),WtT,max_Tout);
    WtT3=get_wat_F(QQ(:,3),WtT);
    SpT4=get_stm_F(QQ(:,4),SpT);
    WtT5=get_wat_F(QQ(:,5),WtT);
    VP1=get_stm_VP(SpT1,SpT,cst_vp1);
    VP2=get_wat_VP(WtT2,WtT,cst_vp2);
    VP3=get_wat_VP(WtT3,WtT,cst_vp3);
    VP4=get_stm_VP(SpT4,SpT,cst_vp4);
    VP5=get_wat_VP(WtT5,WtT,cst_vp5);
    VP(i,:)=[VP1 VP2 VP3 VP4 VP5];
end
function F = get_stm_F(Q,StP);

    F = get_stm_F(Q,StP);  
    Takes in Q in Btu/hr and returns F in lbm/hr. Currently StP is not used 
    delta_H of steam taken as an average value. Could modify to look-up 
    delta_H evap based on steam pressure. 
    % Btu/lbm avgave value for steam in the 550 to 650 psi range 
    delta_H = 730;  
    F = Q / delta_H;  
    % F in lbm/hr

function VP = get_stm_VP(F,StP,const);

    gives steam valve position VP from needed flowrate and Steam header pressure 
    it is assumed the pressures are gauge and outlet pressure is atmospheric 
    therefore delta_P = StP. Uses a F = const*VP*sqrt(delta_P) correlation 
    and VP is 0-100 and is percent open.  
    % const = 7.45;  % corresponds to having a max flow rate of 1.5e4 lbm/hr 
    %    % equals about 1e7 Btu/hr with del_P-500 psi and valve 90% open 
    if nargin == 3  
        const = 7.45;  % corresponds to having a max flow rate of 1.5e4 lbm/hr 
        %    % equals about 1e7 Btu/hr with del_P-500 psi and valve 90% open 
    end 
    VP = (F/sqrt(StP))/const;

function F = get_wat_F(Q,WtT,max_WtT);

    F = get_wat_F(Q,WtT);  
    Takes in Q in Btu/hr and returns F in lbm/hr. WtT is water temperature 
    outlet water temp is set at a maximum value in the file. 
    if nargin == 3  
        max_WtT = 140;  % max water outlet temp (F) 
    end 
    del_T = max_WtT - WtT;  
    Cp = 1.0;  % Btu/(lbm F) average value for water around max_WtT 
    F = -Q / (Cp * del_T);  
    % F in lbm/hr
function VP = get_wat_VP(F, WtP, const);

% VP = get_wat_VP(F, WtP, const);
% gives cooling water valve position VP from needed flowrate and Cooling
% water header pressure.
% it is assumed the pressures are gauge and outlet pressure is atmospheric
% therefore delta P = WtP. Uses a \( F = const \times VP \times \sqrt{delta_P} \) correlation
% and VP is 0-100 and is percent open.
% If const is not supplied then
% const = 244.95; % corresponds to having a max flow rate of 2.7e5 lbm/hr
% equals about 2e7 Btu/hr with del_P = 150 ps and valve 90% open

if nargin == 3
    const = 244.95; % corresponds to having a max flow rate of 2.7e5 lbm/hr
    % equals about 2e7 Btu/hr with del_P = 150 ps and valve 90% open
end

VP = (F / sqrt(WtP)) / const;