ON THE USE OF A TRUTH-SPACE DIAGRAM FOR ASSESSING LINGUISTIC RULES

By

GAURAV ARORA

Bachelor of Science in Chemical Engineering

Guru Gobind Singh Indraprastha University

Delhi, India

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2007

ON THE USE OF A TRUTH-SPACE DIAGRAM FOR

ASSESSING LINGUISTIC RULES

Thesis Approved:

Dr. R. Russell Rhinehart

Dr. Gary Yen

Dr. Karen High

Dr. A. Gordon Emslie

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

NOMENCLATURE

Subscripts and Superscripts

H	The linguistic category ‘High’
i	Index for a point in the data set
j	Index for linguistic category (Low, Medium or High)
l	Index for the rule-set (Rule statement number)

Symbols

a	Lower fuzzy limit for linguistic category j
b	Upper fuzzy limit for linguistic category j
N, M	Number of inputs and outputs of the process
n, m	Number of linguistic values in antecedent and consequent
F1	Flow rate of hot water stream
F2	Flow rate of cold water stream
T1	Temperature of hot water stream
T2	Temperature of cold water stream
T3	Temperature of the mixed stream

F3	Flow rate of the mixed stream
T_a	Truth of Antecedent
T_c	Truth of Consequent
g	Number of consecutive data points with $T_a > 0.5$
x	Numerical value of point i in the dataset
μ	Membership function of the point i in the j^{th} linguistic category

CHAPTER I

INTRODUCTION

1.1 Issues in the process industry

1.1.1 What is process industry?

Process industries convert raw materials into products. These products are either consumed directly or reprocessed by a different process industry into other useful products. The process industries include chemical, food and beverage, pulp and paper, oil and gas, metal, water and wastewater treatment, forest products, etc.

1.1.2 Current challenges in the process industry and their consequences

In the author's knowledge, most of these industries are facing an increasingly challenging environment due to the highly competitive markets, dwindling resources, increasing demand for better quality products, stringent environmental regulations, etc. To stay competitive in this challenging environment, two trends are becoming quite visible in these industries: 1) more sophisticated and complex plants are being built, and 2) major revamps are being performed on the existing plants. The rapid rate of these changes in the process industry can easily be seen by the large number of the Front-End Engineering and Design (FEED) and Front-End Loading (FEL) projects gained by the

engineering, procurement and construction (EPC) companies such as Fluor, Bechtel, Burns & McDonnell, CDI, The Shaw Group, KBR, etc. in the recent years. In addition, this growth of the EPC industry is occurring world wide, as indicated by the heavy recruitment of people in these companies. Globally, the EPC industry is looking at a 60% increase of employees from 2005 to 2009 [1].

1.1.3 Need for a better process management

With the major changes occurring in the process industry, process management becomes very important to ensure a safe, economic, and environmental-friendly process which must also produce uniform products.

In addition, the statistics reveal that the management of the existing processes is not occurring at its optimum level. For example: consider the energy consumption and wastage of the chemical industries in 2001. According to one estimate [2], 37 percent of the total fuel and electricity delivered to chemical engineering facilities was lost in combustion, distribution, and energy conversion activities. At fuel prices of $7 per MMBtu, this meant a loss of around $26 billion. It was further mentioned in this study that 10 to 20 percent of this energy could be practically saved (the rest has to be lost due to the fundamental laws of physics and thermodynamics) through better process management. Thus, there is a lot of scope of improvement in this field and the current trends in the industry make it imperative to improve the process management.

1.1.4 Current process management methods and their shortcomings

The management of processes (unit operations, reactors and control strategies) is obtained through the use of cause-and-effect rules. These rules are usually derived from the phenomenological models or subjective experience of the experts and operators gained during years of trial-and-error. However, the rapid rate at which new processes are being built, existing processes are being revamped and personnel are switching companies makes it extremely difficult to manage them. This is because quite often the expert's knowledge will not be complete. In addition, the transfer of knowledge from the expert to the programmer (to create expert system (ES)) might create oversimplified rules which will be ineffective.

For these reasons, efforts have been made in recent years to gain mechanistic understanding of the process by using data-mining techniques (neural network, time-series analysis, fuzzy-neuro-stochastic techniques for fault detection). Software (such as Gensym G2) has been developed based on these techniques to gain knowledge from the data. However, current software is not utilizing the full potential of today's computers in extracting the knowledge from the data.

Several goals which are to be achieved by the data-mining techniques are to: 1) autonomously generate linguistic cause-and-effect rules from data, 2) incorporate the dynamic and temporal features of the process in these rules, 3) develop metrics and find their threshold values for determining the validity of these rules. These issues are being considered and resolved in the research at Oklahoma State University (OSU), and linguistic modeling of the processes, using fuzzy logic, is being performed to meet the listed goals. The advantages of modeling a process with the linguistic rules are:

- Linguistic rules can be used to model complex processes, for which the traditional models are hard to develop.
- Linguistic statements are easier to understand and interpret than the mathematical equations.
- Human Logic understanding can form the starting rule base.
- Independent mechanisms can be modeled together by using an "OR" operator.
- Linguistic statements can be easily validated against the logical understanding of the process.
- Defuzzification of the fuzzy output from the linguistic rules can predict the future value of the variables.

On the other hand, the disadvantages of fuzzy modeling are:

- Substantial experience with traditional modeling has developed user acceptance.
- Algebraic and calculus-based models are more precise as well as computationally efficient.
- First-principles models provide precise tests of claims about process knowledge.

It should be noted that the fuzzy modeling is not intended to replace the traditional modeling approaches, but, to complement them. The fuzzy modeling would prove to be very useful for complex processes for which little or no prior process knowledge is available. The autonomously generated cause-and-effect rules can be then used in various areas including:

- Operator assistance

- Process automation and control
- Process safety and maintenance
- Incremental process improvement
- Model improvement
- Training and education

1.2 The current research – The big picture

The current research at OSU seeks to develop software for the autonomous generation of linguistic cause-and-effect rules that will be able to incorporate the dynamics (Delay, persistence, etc.) of the process. For complex systems, a large number of relevant process variables are possible; and thus, the current research would use genetic algorithms to generate an initial population of the rules. This rule base would be constantly updated and managed until it discovers all the relevant variables and all the valid rules. The autonomously generated example rule might be:

IF reactor temperature is High AND pressure is Medium THEN final conversion is High.

In this rule, the sentence following the "If" is the antecedent and the sentence following the "then" is the consequent. The "reactor temperature" and "pressure" are input variables and "final conversion" is the output variable. The words "High" and "Medium" are fuzzy variables and are also referred to as linguistic values of the variables. The word "AND" is the conjunction.

The general structure of these rules is:

If (antecedent) Then (consequent)

To assess the validity of the generated rules, the truth-space diagram (TSD) was developed by Sharma [3]. The TSD is the plot between the truth of antecedent and the truth of the consequent, for a given rule. The *Truth* of a statement was defined as the degree of membership of a data set to the linguistic terms in that statement. The degree of membership is defined as the degree of belongingness of the data set to the linguistic category. Thus, for the example rule given above, the truth of the consequent would be equal to the degree of membership of the given data to the "High" category for the output variable, "final conversion".

The general appearance of the TSD plot is shown in Figure 1. The data points lying on the upper-right Quadrant of the TSD implied that both the antecedent and the consequent had high degree of truth. Thus, the data suggested that the rule was a "valid rule" since what the rule stated, did happen. The data points lying on the lower-right Quadrant of the TSD implied that the antecedent had a high degree of truth but the consequent had a low degree of truth. Thus, the data suggested that the rule was an "invalid" rule since what the rule stated, did not happen. Similarly, the data points in the lower-left Quadrant of the TSD implied that the both the antecedent and the consequent had a low degree of truth. Hence, the data was "indeterminate" in evaluating this rule. Finally, the data points lying on the upper-left Quadrant of the TSD implied a low degree of truth of antecedent but a high degree of truth of consequent. Thus, the data suggested that a "hidden mechanism" was present (in addition to the mechanism suggested by the rule statement) which was leading to the consequent. The TSD and work done by Sharma has been discussed in detail in Section 2.2.

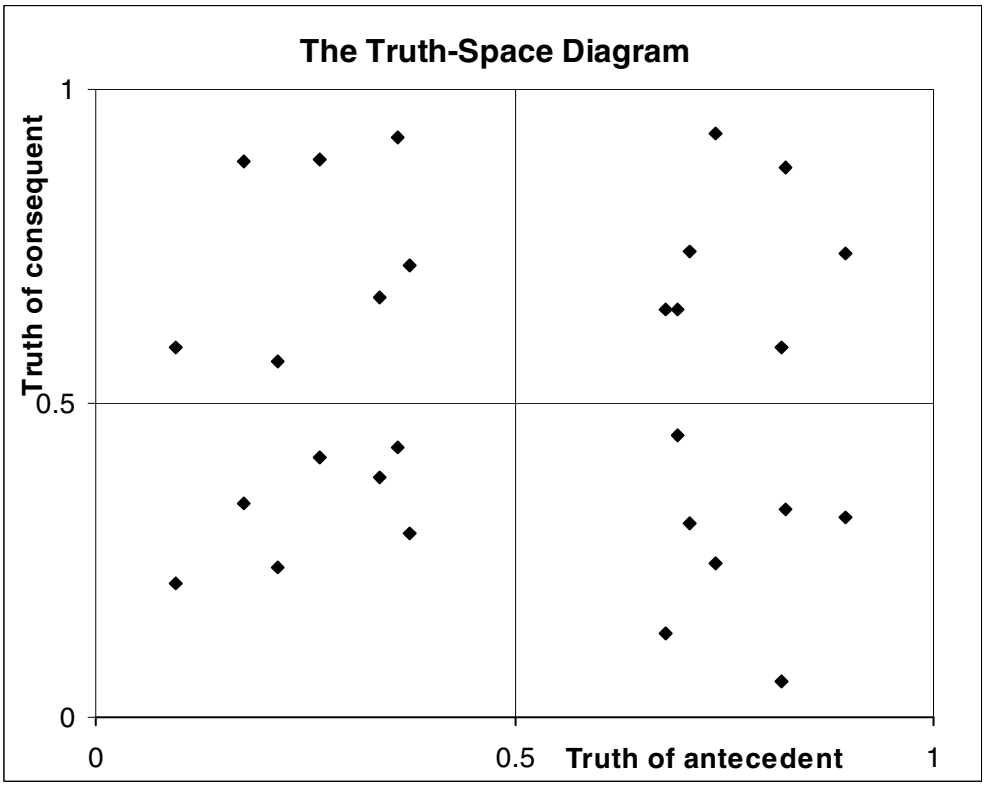


Figure 1: The general appearance of the Truth-Space Diagram

Kumar [4] proposed the concept of "trips" and developed metrics to be used as the selection criteria for the linguistic rules. A trip, within a Quadrant, was defined as the locus of path traced by points into and out of that Quadrant. The metrics developed were based on the number of trips in the upper-right-hand Quadrant and the lower-right-hand Quadrant of the TSD. The concept of trips and work done by Kumar has been discussed in detail in Section 2.3.

Thus, the OSU research is being performed in two different areas: 1) developing genetic algorithms, and 2) development of the TSD. These areas have been further divided into four parallel parts:

1. Development of rule-extraction mechanism (using genetic algorithms and clustering) which accommodates the dynamic features of the process.
2. Continuous management and updating of the rule set thus obtained.

Dr. Gary Yen, professor in the department of Electrical and Computer Engineering, Oklahoma State University, and his students are working on these two parts of the project.

3. Quantification of the dynamic and temporal features of the process.
4. Development and exploration of the TSD to assess the quality of the rules extracted in Steps 1 and 2 and to indicate the relevant input variables which affect the output.

Dr. R. Russell Rhinehart, Head of department of Chemical Engineering, Oklahoma State University, and his students, Ming Su and Gaurav Arora, are currently working on these two parts of the project.

Ming Su [5] is currently concentrating his research on the quantification of persistence by developing a mask, and on analyzing the effect of the missing rules on the efficiency of the rule base in predicting the output.

1.2.1 The present work

Sharma [3] and Kumar [4] devised the TSD and the metrics to assess the validity of a linguistic rule. The TSD, coupled with the concept of trips, was capable of handling dynamics in the process. However, there were several issues which needed to be resolved, to facilitate the application of this technique to a new process and to make accurate predictions of the consequent variables, using the selected rules. These issues

included: 1) analyzing the affect of over specification (the variables which were not relevant are added to the rule set) and under specification (the variables which were relevant are not added to the rule set) on the TSD, 2) quantifying the process delay, 3) improving the prediction technique of Kumar, to give one output for each input, and comparing it with the conventional prediction techniques. Thus, the goal of this research was to resolve these issues and involved:

- Developing a better understanding of the TSD and its attributes.
- Incorporating the delay of the process in the fuzzy rules.
- Accurately predicting the future values of the output variables.

The main contributions of this research are to:

1. Study the affect of the over specification and under specification in the antecedent and consequent of the rules, on the TSD.
2. Find the best mathematical operator to calculate the truth of antecedent (T_a) and consequent (T_c).
3. Quantify the delay.
4. Obtain a criterion based on which the rules can be compared.
5. Find the minimum number of rules required to describe the system completely.
6. Find the best value for the Quadrant size of the TSD.
7. Find the best metric and to obtain its universal value.
8. Make this technique applicable to a process with any number of process variables.
9. Find the belief in a rule.

10. Minimize the curse of dimensionality.

11. Decouple the consequent terms to make the predictions.

12. Choose the best mapping function from T_a to T_c in making the predictions.

13. Blend the delay values in making the predictions.

CHAPTER II

REVIEW OF LITERATURE

2.1 Existing Gaps in Literature

Findings of the on-going research at OSU, in autonomous generation of linguistic cause-and-affect rules reveals challenges of: 1) autonomous rule extraction and optimization, 2) rule attribute quantification, 3) rule base management and, 4) prediction from the rules.

The autonomous rule extraction involves the usage of the data-driven techniques to extract the valid rules from the process data. These techniques require several rule attributes to be defined and quantified, based on which selection criteria are proposed to assess the validity of the linguistic rules. Once the rule base is generated, it is important to continuously update and manage the rule base for further improvement and to incorporate any change in the process. Finally, the rules should be able to predict the future values of the output variables accurately to prove their validity.

However, several issues exist in the current state-of-the-art techniques which need to be resolved to autonomously generate these rules in the dynamic processes. These issues are related to the 1) extraction of rules in noisy data, 2) quantification of dynamic attributes of the system such as delay and persistence, 3) development of new selection

criteria, for the dynamic processes, to assess the validity of the rules, 4) universal applicability of these techniques, and 5) development of new prediction technique which uses the historical information of the process to predict the future.

Various vendors today provide software for automating decision-making. While there are many case studies citing benefits and utility, the techniques for modeling and decision support are mainly conventional, often primitive, and insufficient for complex processes. Most of the software are incapable of autonomously generating the rules and rely on the knowledge of the experts to generate the rule base. For example, Johnson [6] provided a case study in which the Proactive Controller Assistant (ProCA) based on Gensym G2 software was created to ensure pipeline efficiency and safety. However, the rule set was created based on the heuristic knowledge of "Gas Control" controllers. The G2 programmers had to constantly add/modify information to reflect current conditions. In addition, the future flow patterns were predicted by using linear regression techniques. Similarly, Mario, et al. [7] provides another Gensym case study describing the integration of an expert system with a fuzzy controller for start-up of a petroleum offshore platform. Again, the rules were generated based on operator's and engineer's knowledge. The fuzzy controller was used for opening the choke valve of the wells. The dynamic elements were not considered and the controlled variable included just level in tanks and pressure in separator and pipelines.

An attempt for generating fuzzy rules from numerical data was made by Wang [8]. The rules were obtained for each pair of desired input-output data. Due to the presence of a large number of data pairs, and each pair generated one rule, it was highly probable that some interacting rules, the rules with the same IF part but a different THEN

part, will be selected as "valid" rules. Out of all interacting rules, only the rule with the maximum degree, given by the product of the antecedent and consequent membership functions for the given rule, was allowed to be a part of the final rule base to resolve the conflict and to reduce the number of selected rules. Wang further mentioned that an expert should check given data pairs and assign a degree of importance to each data point. Then, the degree of a rule would be calculated by the product of the antecedent and consequent membership function for the given rule and the degree of importance of the data point. However, this technique would fail in dynamic and noisy systems as several invalid rules might be selected because of the noise in the process. In addition, it would take a great amount of time for the expert to look at each data point and assign a degree of importance to it. Again, the assigned degree of importance to the data point (and hence the rule) would depend on the subjective experience of the expert. Furthermore, in processes requiring continuous rule updating, this technique would not be practical since it would require the expert to continuously observe data and assign importance to it. To predict the output value from the given inputs, Wang assumed that truth of the consequent is equal to the truth of the antecedent. Though, it is a widely used assumption in fuzzy logic, but, there is no fundamental reasoning behind this assumption.

Hong [9] proposed a genetic, fuzzy, rule-mining algorithm to effectively construct a fuzzy rule base. The proposed approach consisted of three phases: fuzzy-rule generating, fuzzy-rule encoding and fuzzy-rule evolution. In the fuzzy-rule generating phase, N fuzzy rules were randomly generated with equal probability. A symbol * was introduced in rule base to represent the "don't care" value representing the absence of this attribute in the rule. In the encoding phase, the rules were encoded as a bit-string

chromosome and handled as an individual. In the evolution phase, the rules were evaluated on the basis of fitness functions. This technique used multiple fitness evaluation criteria involving *accuracy*, *utility*, and *coverage* for the fitness evaluation of the rule. *Accuracy* was defined as the ratio of fuzzy correctness cardinality to that of the sum of the fuzzy correctness cardinality and fuzzy incorrectness cardinality. Higher the value of the *accuracy,* the better is the rule. The *utility* of a rule represented its necessity in the process. If an event is correctly predicted by only one rule, then the *utility* value equaled 1 as this rule was necessary to describe the process. Thus, larger the *utility* of a rule is, the more inevitable the rule was. The *coverage* of a rule was based on the number of events which the rule could express. The rule which had the maximum value of the product of accuracy and utility, for a given data set, was selected and all the data points covered by this rule were removed. This process was continued until either all the data points were over or all the rules were evaluated. This process was proven to be useful in the classification of the objects. However, in this technique the total number of fuzzy rules was defined beforehand and its ability to learn from a noisy data was not discussed.

Wang [10] proposed another method of self-generating fuzzy rule base via genetic algorithm. Firstly, an initial population of P chromosomes was generated randomly. Then, the fitness function, to evaluate the fitness of each chromosome, was defined based on the output error of each rule and the rules number of the i-th chromosome. Based on the value of the fitness functions, the rules were ranked. Finally, the new generation of the rules was generated by using reproduction, crossover and mutation operators. The advantage of this method was that there was no need to initialize the rules number, the positions of the antecedent and consequent fuzzy sets in the beginning of the GA.

However, this method required a specific length and structure of the chromosome. The fixed structure of chromosome is required in almost all the current techniques using genetic algorithms for the autonomous generation of rules. However, the need is for a technique which would continuously establish relationships between relevant variables and put them together in a rule set. Wang represented each rule R (j_1, $j_{2,}$,.. j_m) in the complete fuzzy rule base with m inputs and n outputs as:

If x_1 is $A_{(1,j1)}$ and x_2 is $A_{(1,j2)}$ … and x_m is $A_{(1,jm)}$

Then y_1 is B_1 (j_1, j_2, …, j_m) and y_2 is B_2 (j_1, j_2, …, j_m) … y_n is B_n (j_1, j_2, …, j_m)

Though, this rule structure would work, but, the computation effort required to extract rules in this rule set would be higher, than that of several rule sets consisting of m inputs and one output each. The advantage of breaking a multiple-input, multiple-output rule base into several multiple-input, single-output rule bases has been discussed in Section 3.2.4. In addition, the applicability of this technique to dynamic systems was not discussed in this work.

Umano [11] proposed a method to extract quantified fuzzy rules from numerical data. An example of the fuzzy rules developed was “Most data whose attribute A is large are small in the attribute B”. Here, “large” and “small” were fuzzy sets of attribute A and B, respectively, and “most” was a fuzzy quantifier. To extract the fuzzy rules, an algorithm was used which generated a fuzzy decision tree, using fuzzy sets defined by a user. The fuzzy decision tree consisted of nodes for testing attributes, edges for branching by test values of fuzzy sets and leaves for deciding classes with certainties. The fuzzy rules were extracted from fuzzy decision tree by evaluating its understandability and in formativeness. Of the rules extracted, the best rules were selected by evaluating them.

The candidate rules were evaluated using *certainty* of the rule, *value of the fuzzy quantifier*, the *number of attributes* for restriction and the *coverage* of the rule. The *certainty* was calculated as the membership value of fuzzy quantifier for the given proportion of the data set of a given class. The *values of the fuzzy quantifiers* were evaluated based on the membership values chosen for "More than", "Most" and "Almost all" fuzzy quantifiers. The greater the *number of attributes* for restriction is, the more detailed the rule could classify the data, but the more complex the rule is and the more difficult it is to understand. Thus, membership values were assigned to the *number of attributes* and these membership values decreased with the increase in number of attributes. The *coverage* of the extracted rules needed to greater than a predefined threshold value. The issue with this work was that it consisted of a large number of parameters whose values were based on the human discretion. In addition, the structure of the rules used in this work made the technique useful for classification problems but is hard to be understood by a plant operator.

Juuso [12, 13] designed linguistic equations for process analysis, process control, fault diagnosis, and forecasting for nonlinear multivariable systems. The insight to the process dynamic operation was described as the most important contribution of this work. The novelty of the approach was that the nonlinearity was handled through the membership definitions and not in the rule base. These membership definitions were generated directly from process data on the basis of operating area of the model and variation for each process case. However, the delays in the process were assumed to be fixed and crisp, which does not match real processes. Juuso acknowledged that this works

only for small systems and mentioned that an appropriate handling of the delays would extend the operating area of the model considerably.

Kermani [14] developed a fuzzy logic system with evolutionary variable rules. All the variables (output, cases, features, qualifiers, and operators) were continually evolved. The rules were created in real-time and were updated with time. It was claimed to be valuable in applications requiring constantly-updated fuzzy rules and in applications where fuzzy rules are difficult to pre-define, such as, stock market forecasting. The dynamics of stock market were introduced in the rules through variables D1, D2, D3…Dn. Here, Dn represented the normalized stock price change of (today – n). Here, n represented the number of days in the past and the values used were 1, 2, 3, 4, 5, 7 and 30. Although, this work provided an algorithm for updating the fuzzy rules with time, the handling of dynamics of the system was inconvenient because introducing a variable for every possible delay value increases the size of rule base exponentially and, thus, would make the technique impractical for process with variant delays.

In the Sections 2.2 and 2.3, the work done at OSU by Sharma [3] and Kumar [4] is discussed. The work discussed in this report is in continuation of the work done by these two researchers.

2.2 Prior work at OSU by Sharma

Sharma [3] proposed the concept of the "Truth Space Diagram" for evaluating the goodness of each rule. A truth space diagram was defined by Sharma as a "two-

dimensional space bounded by the truth of the antecedent and the truth of the consequent of a linguistic rule". The three steps used in the autonomous extraction of rules were:

1. Data generation and processing
2. Exhaustive search for initial rule base generation
3. Calculation of numerical metrics and rule base optimization

2.2.1 Data generation

Sharma used a Hot and Cold water simulator to explore the validity of his technique. The simulator incorporated real world dynamics such as transport and measurement delays and is shown in Figure 2.

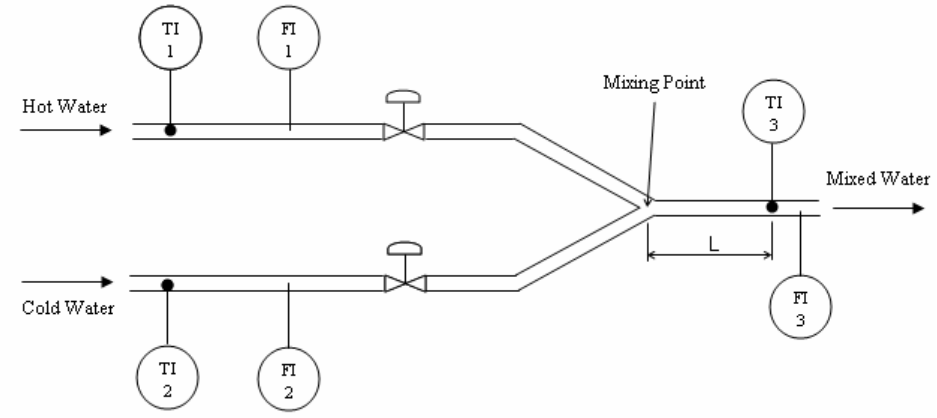


Figure 2: The hot and cold water-mixing simulator

For the purpose of generating data, three input variable were manipulated and the affect on one output variable was monitored. The input variables chosen were:

1. Temperature of the hot water stream ($0 \leq T1 \leq 100\ ^{o}C$).
2. Flow rate of the hot water stream ($0 \leq F1 \leq 30$ Kg/min).
3. Flow rate of the cold water stream ($0 \leq F2 \leq 30$ Kg/min).

The output variable chosen was:

1. Temperature of the mixed stream ($0 \leq T3 \leq 100\ ^{o}C$).

The temperature of the cold stream, T2 was not chosen as an input to keep the exhaustive search convenient. The algorithm simulated the mixing of two streams – one carrying the hot water and the other carrying cold water. It calculated the resultant temperature and delayed its measurement based on the mixing length L and the input flow rates. The simulator was run and the transient response to the inputs was obtained. Figure 3 showed the raw input data generated by Sharma. To find all possible process situations T1 was changed from a high temperature to a medium temperature and then again to a high temperature. Similarly, the two flow rates were also changed in a similar way, from high, to medium to low. The data was sampled at an interval of one second. The simulator code is in Appendix A.

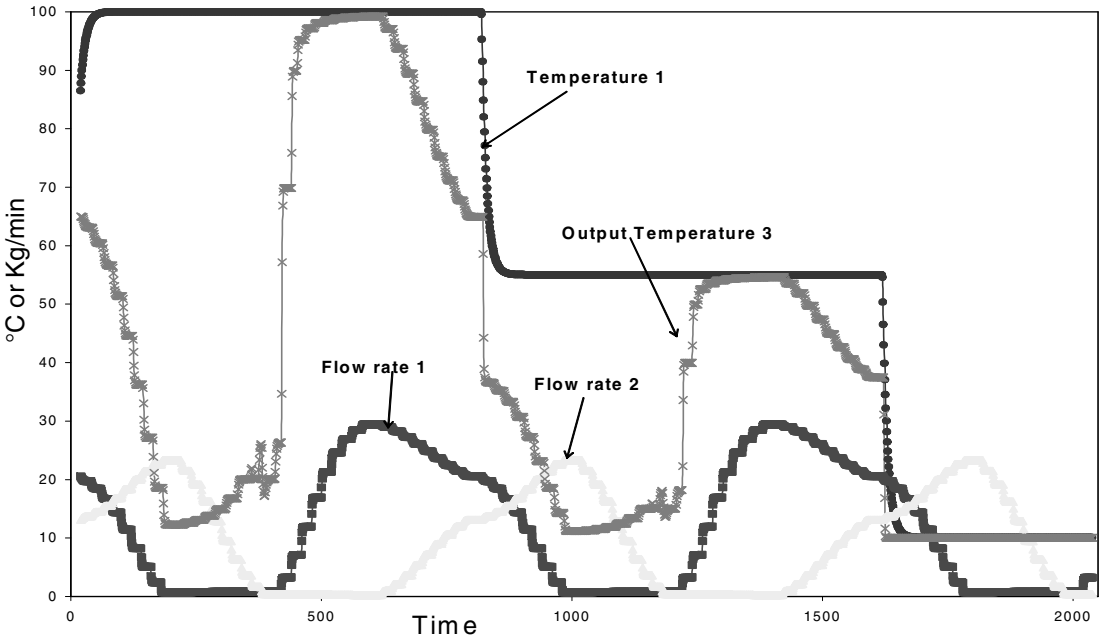


Figure 3: Transient Input-Output Data (reproduced from [3])

2.2.2 Data processing

The data from the simulator was processed in two steps: First the delayed output-data was un-delayed by shifting data backwards into three categories short, medium and long delay. The shifting was done by deleting a number of data-points from the top of the output data column and shifting the rest of the column. The number of time units by which the column was shifted equaled the delay. Figure 4 depicts this procedure schematically.

Time	T1	F1	F2	T3_Original Values	T3_Short Delay	T3_Medium Delay	T3_Long Delay
1	86.466	20.58	13.13	64.98472	64.9641	64.25688	63.09222
2	87.754	20.44	13.16	64.97263	64.95359	63.94546	63.08392
3	88.92	20.24	13.26	64.9641	64.85162	63.6954	62.92605
4	89.974	20.08	13.36	64.95359	64.58994	63.50988	62.51725
5	90.928	19.96	13.46	64.85162	64.25688	63.37791	62.00672
6	91.791	19.87	13.53	64.58994	63.94546	63.28615	61.54251
7	92.573	19.81	13.58	64.25688	63.6954	63.22311	61.18189
8	93.279	19.77	13.61	63.94546	63.50988	63.1801	60.92411
9	93.919	19.74	13.63	63.6954	63.37791	63.15086	60.74795
10	94.498	19.72	13.65	63.50988	63.28615	63.13103	60.63054
11	95.021	19.71	13.66	63.37791	63.22311	63.11759	60.55336
12	95.495	19.7	13.67	63.28615	63.1801	63.10849	60.50298
13	95.924	19.7	13.67	63.22311	63.15086	63.10234	60.47028
14	96.312	19.69	13.68	63.1801	63.13103	63.09817	60.4491
15	96.663	19.69	13.68	63.15086	63.11759	63.09536	60.43541
16	96.98	19.69	13.68	63.13103	63.10849	63.09348	60.42656
17	97.268	19.69	13.68	63.11759	63.10234	63.09222	60.42086
18	97.528	19.69	13.68	63.10849	63.09817	63.08392	60.41718
19	97.763	19.68	13.68	63.10234	63.09536	62.92605	60.41481
20	97.976	19.68	13.68	63.09817	63.09348	62.51725	60.41332
21	98.168	19.68	13.68	63.09536	63.09222	62.00672	60.41237
22	98.343	19.46	13.74	63.09348	63.08392	61.54254	60.40481
23	98.5	19.17	13.88	63.09222	62.92605	61.18189	60.18476
24	98.643	18.92	14.04	63.08392	62.51725	60.92411	59.56002
25	98.772	18.75	14.17	62.92605	62.00672	60.74795	58.7724
26	98.889	18.63	14.27	62.51725	61.54251	60.63054	58.07296
27	98.995	18.55	14.33	62.00672	61.18189	60.55336	57.55146
28	99.09	18.5	14.38	61.54251	60.92411	60.50298	57.19742
29	99.177	18.47	14.4	61.18189	60.74795	60.47028	
30	99.255	18.45	14.42	60.92411	60.63054	60.4491	
31	99.326	18.44	14.44	60.74795	60.55336	60.43541	
32	99.39	18.43	14.44	60.63054	60.50298	60.42656	
33	99.448	18.42	14.45	60.55336	60.47028	60.42086	
34	99.501	18.42	14.45	60.50298	60.4491	60.41718	
35	99.548	18.41	14.45	60.47028	60.43541	60.41481	
36	99.591	18.41	14.46	60.4491	60.42656	60.41332	
37	99.63	18.41	14.46	60.43541	60.42086	60.41237	
38	99.665	18.41	14.46	60.42656	60.41718	60.40481	
39	99.697	18.41	14.46	60.42086	60.41481	60.18476	
40	99.726	18.41	14.46	60.41718	60.41332	59.56002	
41	99.752	18.41	14.46	60.41481	60.41237	58.7724	
42	99.776	18.07	14.54	60.41332	60.40481	58.07296	
43	99.797	17.61	14.76	60.41237	60.18476	57.55146	
44	99.816	17.26	14.98	60.40481	59.56002	57.19742	
45	99.834	17.03	15.16	60.18476	58.7724		
46	99.85	16.88	15.29	59.56002	58.07296		
47	99.864	16.79	15.37	58.7724	57.55146		
48	99.877	16.73	15.42	58.07296	57.19742		
49	99.889	16.69	15.45	57.55146			
50	99.899	16.67	15.47	57.19742			

Figure 4: Example of Backward Shifting of Output variable T3 with Short Delay = 2 sec; Medium Delay =6 sec; Long Delay = 22 sec. Reproduced from [3].

Secondly, the crisp input-output data was fuzzified using triangular membership functions:

$$\mu^{i,j} = \frac{a_j - x_i}{a_j - b_j} \tag{1}$$

Where j=1 to 3 and i = 1 to n tot, data

n tot, data = total number of data-sets in the input-output data

x_i = crisp numerical value of the i^{th} input or output variable

$\mu^{i,j}$ = fuzzy membership value of xi in the j^{th} fuzzy category

a_j and b_j = fuzzy set break points for category j

Figure 5 shows an example of the fuzzy classification of output temperature T3, into three fuzzy categories of high, medium and low. In this example, for the category "low", j = 1, a_j = 10 °C and b_j = 50 °C and $\mu^{i,1}$ = 1 if $x_i \leq$ 10 °C. Similarly, for the category "medium" j = 2, a_j = 10 °C and b_j = 50 °C only if 10 < x_i < 50 °C while a_j = 50 °C and b_j = 95 °C if 50 °C < x_i < 95 °C; at x_i = 50 °C, $\mu^{i,2}$ = 1. Similarly for the category "high" j = 3, a_j = 50 °C and b_j = 90 °C and $\mu^{i,3}$ = 1 if $x_i \geq$ 95 °C. Triangular membership functions and only three fuzzy categories were used to keep the example simple since the number of rules in the initial rule base, which defines the size of the search space, increases exponentially with addition of each fuzzy category. However, it should be noted that the technique developed by Sharma would be applicable with other membership functions (trapezoidal, Gaussian, etc.), too.

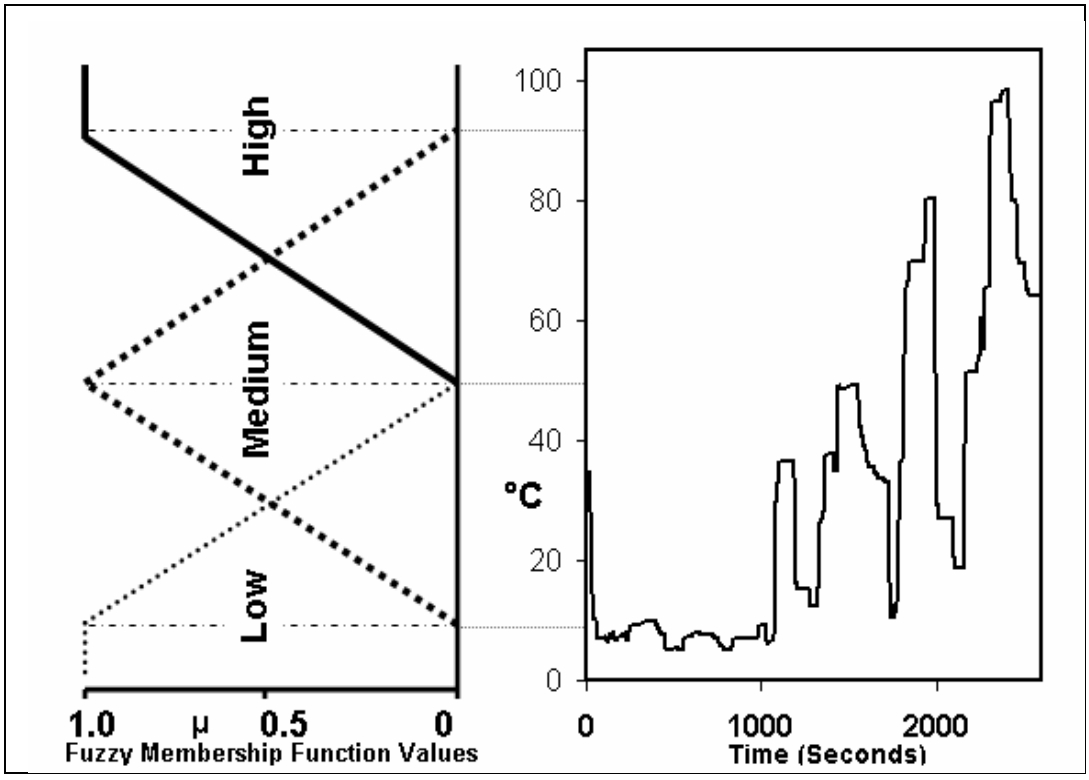


Figure 5: Fuzzy Classification of Output T3 (Reproduced from [3]).

2.2.3 Initial Rule Base and Calculations

The dynamic information was included in the linguistic rules using Persistence and Delay. Persistence was included in the antecedent and the Delay was included in the consequent side of the rule. The general structure of for a given rule, R, in the initial rule base was:

IF (T1 is L/M/H AND F1 is L/M/H AND F2 is L/M/H AND Persistence is L/M/H) THEN (after L/M/H delay T3 is L/M/H); Where L/M/H refers to either Low, Medium or High.

The total number of possible rules was 3^6= 729. An exhaustive search was done to generate all these rules. Depending on the rule statement R_l (where $1 \leq l \leq 729$) the

persistence of the antecedent was calculated for all data points in the input-output data set. The persistence of each linguistic label was measured by the number of time units the membership value of variable had persisted in the fuzzy category. The combined persistence was defined as the minimum persistence of any of the three input parts (T1, F1, F2) of the rule antecedent. Once the combined persistence was known then it was fuzzified to find the fuzzy membership value of the persistence variable to be used in the truth space calculations described below.

The *Truth* of any statement was defined as the degree of membership of any data set or example to the linguistic terms in that statement. The truth of the antecedent $Ta_{i,l}$ and the Truth of the consequent $Tc_{i,l}$ were calculated for each rule statement for each point x_i in the input-output data using a geometric operator as:

$$Ta_{i,l} = \left(\mu_{T1}^{i,j} \times \mu_{F1}^{i,j} \times \mu_{F2}^{i,j} \times \mu_{Persistence}^{i,j}\right)^{\frac{1}{4}} \tag{2}$$

$$Tc_{i,l} = \mu_{T3}^{i,j} \tag{3}$$

These values were then used for the construction of the Truth-Space Diagram (TSD). The TSD was bounded by the region {T: $0 \leq T \leq 1$, where T = truth of antecedent/consequent}, where a truth equal to 0 means absolutely false, and truth equal to 1 means absolutely truth. The TSD diagram was divided into four Quadrants and each Quadrant provided different information about each rule as discussed in Section 1.2. This information was assessed by looking at the number of data points lying in a given Quadrant. In addition, metrics were proposed by Sharma, based on the number of points in the four Quadrants, to assess the validity of the rule. Kumar [4] pointed out that these metrics would give problems in the rule selection when the data is noisy because the data

points might get placed in the Quadrants due to noise. In addition, Kumar mentioned that the long persistence of a certain event would have caused many points to be placed in the TSD of the rule depicting the event. To overcome this issue, Kumar proposed new set of metrics and only the metrics developed by her will be discussed in this report (metrics developed by Sharma will not be discussed here).

2.3 Prior work done by Kumar

Kumar [4] introduced the concept of trips to eliminate the inherent disadvantages in Sharma's work. She defined a trip within a Quadrant as the locus of a path traced by points into and out of the Quadrant. Thus, it is a combination of monotonous increasing and decreasing behavior of Ta and Tc value of points. A path is obtained by connecting the data points which appear consecutively with time in the TSD. When a threshold number of these points appear consecutively in a single Quadrant, then a trip is said to be made in that Quadrant. The implication of the Quadrants in the TSD based on the concept of trips is illustrated in Figure 6.

All the Quadrants were intuitively chosen to be of size 0.5 x 0.5 each. The Quadrants on the upper-left, upper-right, lower-left, and lower-right were named as QI, QII, QIII, and QIV respectively. Kumar suggested that each data point should not be considered as a separate event. Instead, all the loci should be counted as independent trips that corroborate the statement of the rule as demonstrated in Figure 6. The implications of the presence of trips in these four Quadrants have been discussed in Section 2.3.2.

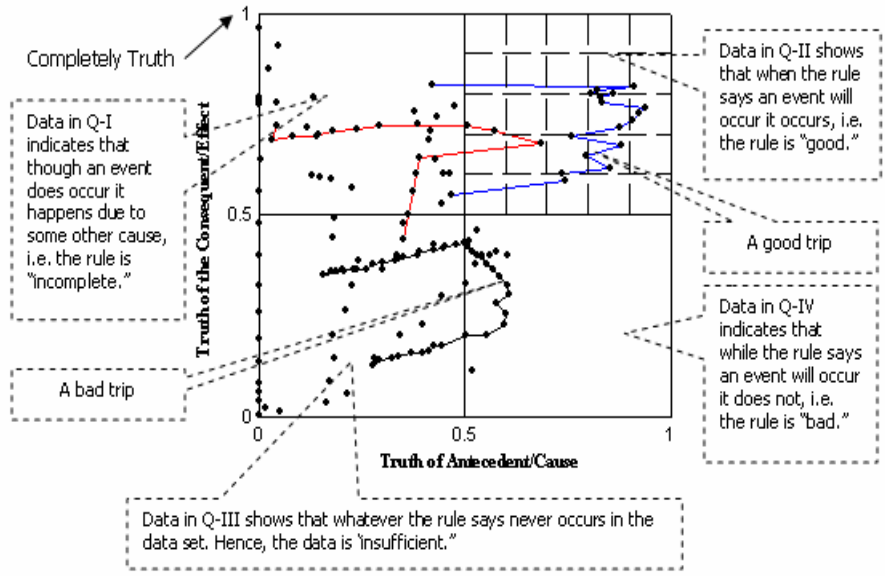


Figure 6: TS Diagram (Reproduced from [4]).

2.3.1 Threshold Condition

In order to exclude spurious events from being called as trip, an arbitrary value was chosen for the *Minimum Time* that a path into a Quadrant needed to stay in the Quadrant. It was a user defined value and was given by:

$$Threshold = \frac{Minimum\,Time}{Sampling\,Time} \tag{4}$$

The sampling time was defined as the time interval between two consecutive data samples, assumed to remain constant throughout the process. Thus, Threshold was defined as the least number of successive points within a Quadrant that can be termed as a trip in the Quadrant. A larger value of the threshold would imply more stringent requirements to qualify a path as a trip and vice-versa. Kumar choose an intuitive value of 5 points for the Threshold.

For a better understanding, consider Figure 7, It shows three paths traced into Quadrant II but only two were called as trips. The third path did not qualified as a trip since it had only three points in Quadrant II, which is less than the threshold value of five points.

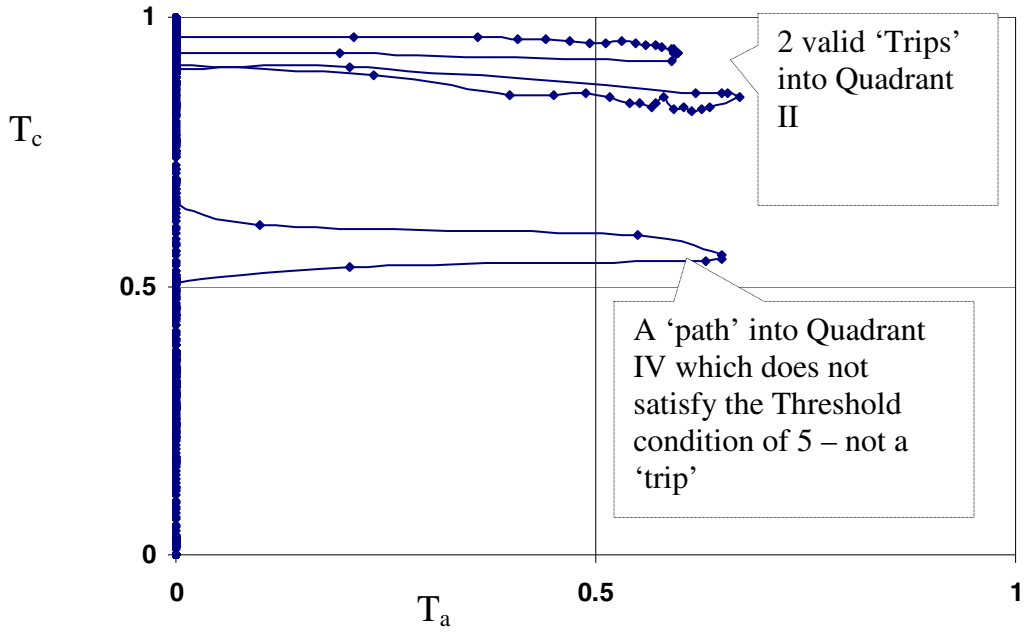


Figure 7: A Truth Space Diagram depicting 3 paths traced into Quadrant II, of which only 2 are 'Trips' (Reproduced from [4]).

2.3.2 Corroboration

The minimum number of trips which a rule had to make into a Quadrant for it to provide sufficient evidence of 'corroboration' was called as corroboration. Kumar choose an intuitive value of 2 for corroboration.

2.3.2.1 Trips in Quadrant II

Trips in the second Quadrant ($0.5 \leq T_c \leq 1.0$ and $0.5 \leq T_a \leq 1.0$) implied that the consequent of the rule was actually caused by the antecedent expressed in the rule. Thus, more trips in Quadrant II suggested that the rule was good.

2.3.2.2 Trips in Quadrant IV

Trips in the fourth Quadrant ($0.5 \leq T_a \leq 1.0$ and $0 \leq T_c \leq 0.5$) implied that the consequent of the rule didn't comply with the effect expected by the antecedent expressed in the rule. Thus, trips in Quadrant IV suggested that the rule was bad.

2.3.2.3 Trips in both Quadrants (II, IV)

If the trips were made in both the Quadrant II and Quadrant IV, and the threshold condition is satisfied in both the Quadrants, then a trip was said to be made in both. If the threshold condition was satisfied in only one Quadrant then the trip was said to be made in the Quadrant in which the threshold condition was satisfied. For example: consider the Figure 8:

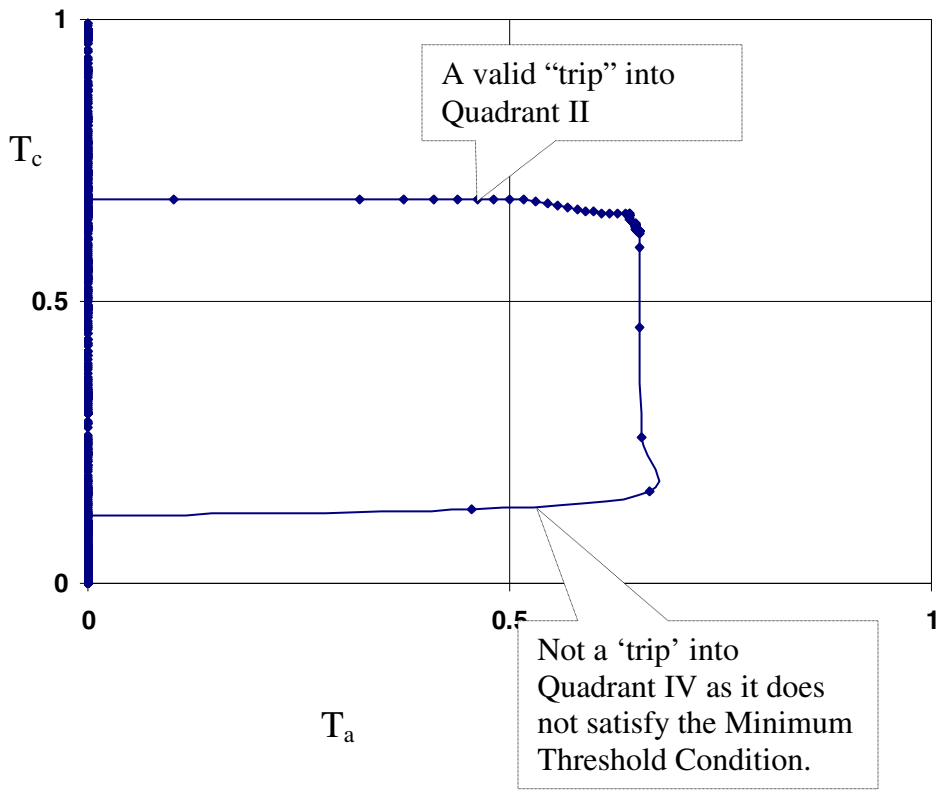


Figure 8: A Truth Space Diagram depicting the difference between a valid 'trip' and an invalid 'trip' (Reproduced from [4]).

2.3.3 The Selection Metric: Merit

Merit was defined as the difference between the number of good trips and the number of bad trips. Thus,

$$Merit = No.\,of\ \ Good\ Trips - No.\,Of\ \ Bad\ Trips \tag{5}$$

Higher value of Merit provided higher evidence of the rule being observed often. The advantage of this metric was that it was independent of the number of data points in the Quadrant. It was used in combination with the minimum corroboration condition to select valid rules.

2.3.4 The Prediction Mode

In the prediction mode, the T_a of the new data was analyzed to determine the rules being activated, and to predict the T_c of the output. To predict the future outcomes an expectation metric was proposed. The expectation was calculated based on the information gathered from the historical data. For its calculation, the Quadrants II and IV were divided into grids as shown in Figure 9. Thus, there were five T_a zones and 10 T_c zones.

From the historical data, the data distribution in each of the ten consequent zones for the five antecedent zones for each rule in the initial data-base was analyzed using the histogram and was normalized as shown in Figure 9.

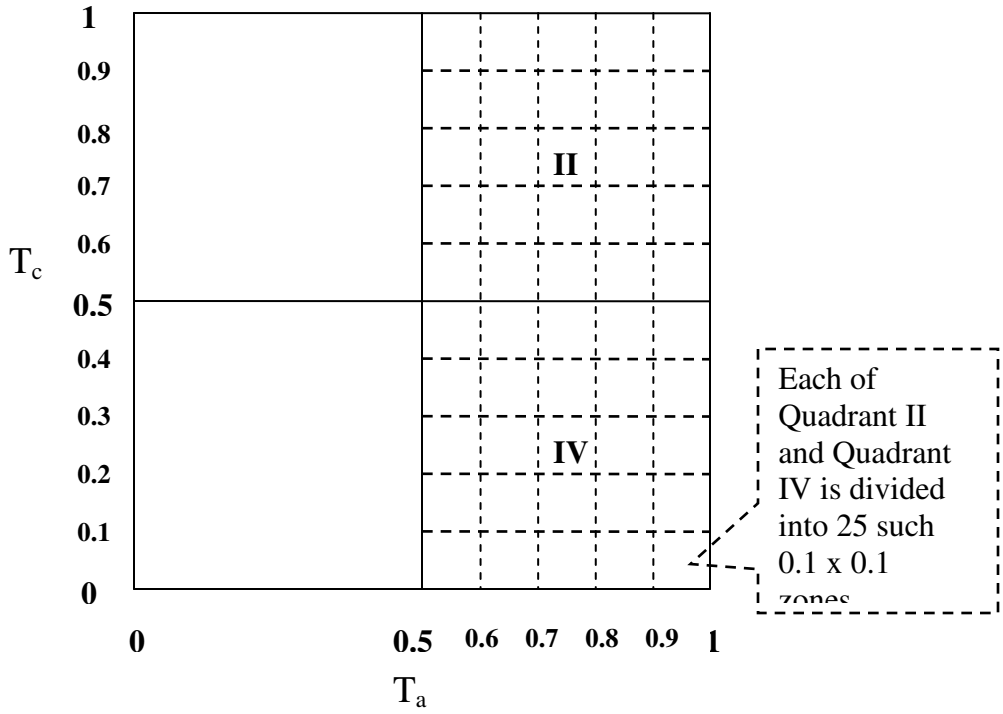


Figure 9: A TSD showing the division of Quadrants II, IV into a total of 50 zones of size 0.1 x 0.1. The Ta axis is divided into five zones of size 0.1 each (0.5-1), and the Tc axis is divided into ten zones of size 0.1 each (0-1). (Reproduced from [3]).

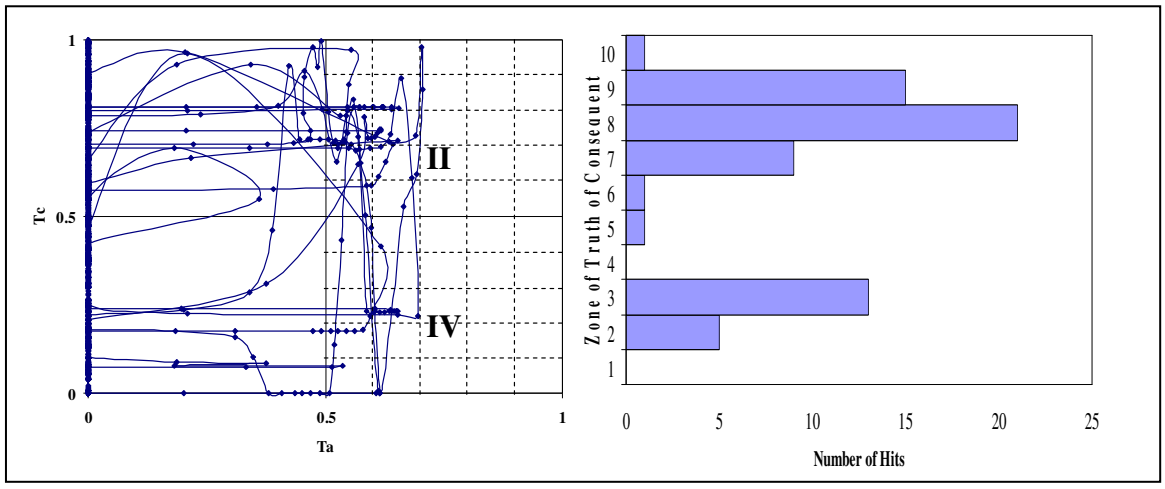


Figure 10: Example of distribution of points in the II, IV Quadrants based on 'Historical Data'. The adjoining histogram represents cumulative historical hits in each of the ten consequent zones. Data in individual Ta zones is to be normalized and used in conjunction with the antecedent hits in the 'New Data' to calculate the 'Expectations' as shown in Figure 11. Note: Only points that contribute to making trips(good or bad) are considered for all calculation purposes. (Reproduced from [4]).

For the new data, every time the T_a appeared in any of the five T_a zones it was recorded and called as a hit in that zone. For example: If T_a was 0.65 then a hit was said to be made in the second T_a zone. This information about the hits was used in conjunction with the historical normalized data to give the normalized expected occurrences of the truth of the consequent. An example has been shown in Figure 11 where antecedent hits were made in two zones. Based on the antecedent hits in the two zones, expectations were calculated for each of the two zones to give the cumulative absolute 'Expectation' of occurrences of the T_c in the ten zones. These values were then normalized to give the cumulative normalized 'Expectations' for each of the ten consequent zones.

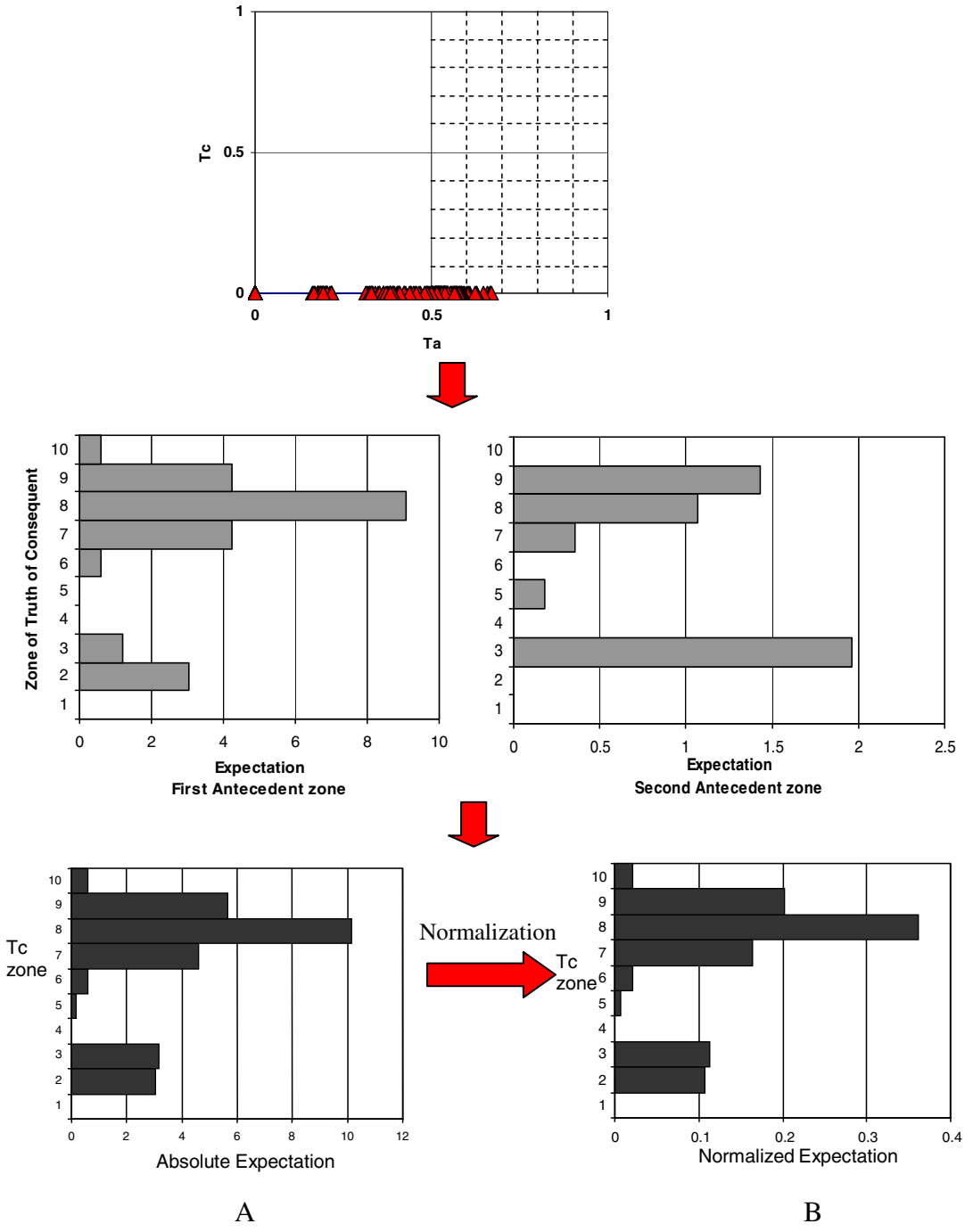


Figure 11: Based on the antecedent hits in the two Ta zones, expectations are calculated for each of the two Ta zones to yield the cumulative absolute 'Expectation' of Occurrences of Truth of Consequent in the ten zones, based on the Antecedent Hits of 'New Data'. The values of these 'Expectations' are then normalized in the range (0-1) to yield the cumulative normalized 'Expectations' for each of the ten consequent zones. (Reproduced from [4]).

2.3.5 Calculations for the prediction mode

The calculations were performed in two stages by Kumar. In the first stage, the historical data was analyzed and the results were stored to be used in the prediction mode. In the second stage, the future outcomes were predicted for the consequent variables using the information stored in the first stage. The calculations performed in these two stages by Kumar have been discussed next.

2.3.5.1 From Historical Data

The historical data was processed in the following steps:

1. A column vector *Hits* was used to record the number of points or hits made in each of the five zones of the antecedent.
2. Then, a 10 x 5 matrix *Numpoints* was used to store the number of hits in each of the ten T_c zones for each of the five Ta zones. Thus, each element in the matrix the number of hits in each of the 50 zones.
3. Then, each column of the matrix *Numpoints* was separated into five column vectors represented by $\vec{V}_i$ and then normalized in the range 0-1 as shown below:

$$\vec{V}_{i\,norm} = \left[\frac{\vec{V}_i}{\vec{1}^{\,T} \cdot \vec{V}_i} \right] \tag{6}$$

Here, i = 1 to 5 and $\vec{1}$ was defined as $\vec{1} = [1,1,1,1,1,1,1,1,1,1]$

The five columns vectors were then concatenated back to give the normalized matrix *NormNumPts* (10 x 5) as shown in Equation (7):

$$NormNumPts_{10\times5} = \left[\left(\vec{V}_{1norm}\right), \left(\vec{V}_{2norm}\right), \left(\vec{V}_{3norm}\right), \left(\vec{V}_{4norm}\right), \left(\vec{V}_{5norm}\right) \right]_{10\times5} \tag{7}$$

2.3.5.2 From New Data

1. Similar to the first step in the processing of the historical data, a column vector Pred*Hits* was used to record the number of points or hits made in each of the five zones of the antecedent for the new data. This vector was converted into a 5 x 5 matrix, *PredDiagHits* for the purpose of performing the mathematical operation of product of two matrixes as discussed in the next step.

2.3.5.3 Calculation of the expectation matrix

1. The matrixes *NormNumPts* and *PredDiagHits* were multiplied to give a 10 x 5 matrix named *product* as shown in Equation (8):

$$product_{10x5} = NormNumPts_{10x5} \cdot PredDiagHits_{5x5} \tag{8}$$

2. The elements of the product matrix were then added along the row to yield the E*xpectation* matrix (10 x 1) as shown in Equation (9):

$$Expectation_{10x1} = \sum_{row-wise} product_{10x5} \tag{9}$$

3. The expectation matrix was then normalized to give the *NormExpectation matrix* as shown in Equation (10):

$$NormExpectation_{10x1} = \frac{Expectation_{10x1}}{\sum_{column-wise} Expectation_{10x1}} \tag{10}$$

2.3.5.4 Weighted mean average

The weighted mean average of the expectations was finally calculated to give the final output and was calculated by Equation (11):

$$PredMean = \frac{(0.05 xExpectation(1,1) + 0.15 xExpectation(2,1) + ... + 0.95 xExpectation(10,1))}{(TotSum)} \tag{11}$$

Where,

$$Totsum = \sum_{Column-wise} Expectation_{10x1} \tag{12}$$

2.4 Issues with Sharma's and Kumar's Work

Kumar and Sharma each provided strong tools in the form of the TSD and trips respectively to autonomously generate cause-and-effect rules in dynamic processes. However, there were several issues which needed to be resolved before this technique could be applied to complex real-world problems. Some of these issues were:

1. Interpretation of the TSD: Sharma assigned the meaning to the four Quadrants of the TSD. Kumar redefined these Quadrants based on the concept of the trips as discussed in the Section 2.3.3. This novel approach assisted in the selection of valid rules in dynamic systems. However, the meaning of these Quadrants was not compatible with the genetic algorithm approach for the selection of rules because the assigned meaning of the Quadrants assumed that all the relevant variables describing the process were known beforehand i.e. it assumed that there was no over specification and under specification in the antecedent and the consequent. However, when genetic algorithms will be used to select rules, all the antecedents and the consequents will not be known beforehand. Instead, with time, the

relationships will be established between the antecedents and the consequents. In other words, the rules will be evolved. Hence, the interpretation of the Quadrants should include the effect of over specification and the under specification. This issue has been considered in this work and discussed in the Chapter 3.

2. Delay Quantification: Sharma and Kumar each used an over simplistic technique to incorporate the delay in the process. As discussed in the Section 2.2.2, the data was shifted backwards into three categories short, medium and long delay. This analysis allows the delay to take only three values (one corresponding to each category). In addition, because of predefined shifting of the data, there is a good possibility that the shifted values of the T_c would give a trip when it should not and vice-versa. Thus, it is necessary to quantify Delay based on the relationship between T_a and T_c. This quantified delay should then be used to shift the data backwards in time.
3. Calculation of the T_a and T_c: Sharma and Kumar each used a geometric mean operator to calculate the truth of the antecedent as given by Equation (2). This geometric mean operator has never been used in the literature. The rationale for using this operator was that it would make it easier to select the rules using lesser number of data points. However, the minimum operator and the product operator have been widely used in the literature for the truth calculations with considerable success. Thus, it is necessary to compare the geometric operator with the minimum and product operator. The best operator should then be used for the calculations of T_a. In addition, the calculation for the truth of the consequent did not involve the membership value of Delay. Since, the Delay was allowed to have

just three values thus its membership function value was a Boolean variable and not a fuzzy variable. The membership value for delay in the each category was used as unity for the calculations of T_c. However, delay is one of the variables in the rules and like other variables its membership value should be used in the calculation of the truth of the consequent. This gives another rationale to quantify the delay.

4. Quadrant Size in the TSD: Kumar and Sharma each used an arbitrarily chosen value of $T_a = 0.5$ and $T_c = 0.5$ for the TSD. However, this value was completely intuitive and no justification was provided as to why this Quadrant size should be chosen. Kumar acknowledged this issue in her work. The larger values of T_a and T_c for the Quadrant size would make the rule selection process strict and vice-versa. This gives rise to question that should this value be based on the process and expert discretion or can a universal value for the Quadrant size be obtained? In this work, this issue has been addressed, and justification for using the chosen Quadrant size has been given.
5. Selection of the Metrics and their universal values: Sharma's metrics were based on the number of the data points in each of the four Quadrants of the TSD. However, these metrics were not suitable for the noisy data. Hence, Kumar developed the concept of the threshold, trips and provided to new metrics: Corroboration and Merit. A threshold value of 5 was chosen by Kumar to qualify a path in a Quadrant as a trip. This is a common practice in the process control to choose this value of 5 to qualify an event as change in the process. In addition, intuitive values for the metrics were chosen. A value of 2 was chosen for the

corroboration and a value of 1 was chosen for Merit. However, these values were intuitive, not based on any fundamental phenomena or analysis. Moreover, the justification of using two metrics, instead of one, was not provided. Kumar acknowledged the need to find the universal value of the metrics in her work. Hence, this work would discuss the issues related to the selection of the metrics and their universal values.

6. Predicting the value of the consequent in the future: Kumar developed the technique for predicting the future values of the consequent based on the historical data as discussed in the Section 2.3.4. Her work provided a novel approach to predict the future outcomes by dividing the Quadrants II and IV into grids. However, each value of the T_a did not predict a value of T_c. The entire new data was used to register hits and then those hits were used to predict the consequent values by using it in conjunction with the historical data. Thus, several hits were made in the rules but each rules predicted just one output. In addition, delay was not accounted for while making predictions. Thus, this technique needed refinement to predict one consequent value for every antecedent and the delay value was required to be incorporated to predict the time in the future at which the predicted value will be realized. In this work, this work has been done. In addition, this work compares the prediction technique of Kumar with the conventional prediction method used in the literature of fuzzy logic.

CHAPTER III

METHODOLOGY

3.1 Terms used in the research

In this work, some of the terms and conventions used by Sharma [3] and Kumar [4] have been replaced by new terms and conventions. These are discussed below:

- *Change in the Quadrant names*: The convention used in the naming the Quadrant has been changed to better coincide with convention. The new names of the Quadrants are shown in Figure 12:

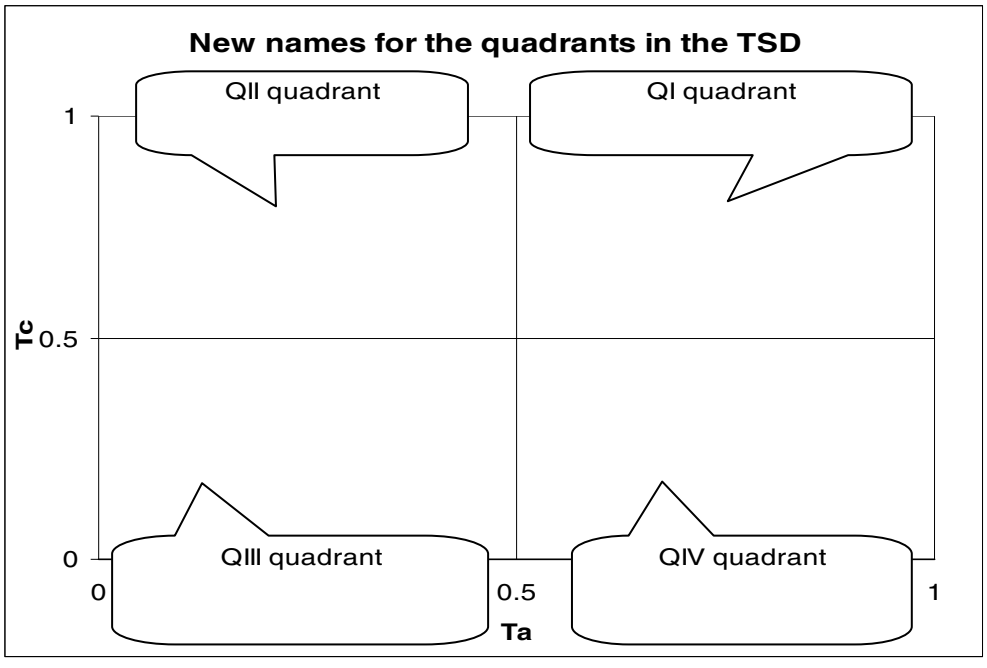


Figure 12: TSD diagram showing the new names for the Quadrants.

These new Quadrant names will be used in the rest of this report.

- *Change in the trip names*: In the previous research, the trips in the Quadrants QI and QIV were known as good and bad trips respectively. However, these names seem too simple and incomplete and will not be used in this report. The trips in QI, QII, QIII and QIV will be simply referred to as trips in QI, trips in QII, trips in QIII and trips in QIV respectively.
- *Number of antecedent variables*: In the previous work, four antecedent variables, T1, F1, F2, persistence, and two consequent variables, Delay and T3, were used. Thus, a total of 3^6 = 729 possible rules were obtained. In this work, five antecedent variables have been used which are T1, F1, T2, F2 and Persistence. The output variables are delay and T3. Thus, there will a total of 3^7=2187 possible rules.
- *Truth of antecedent and consequent*: In the previous work the geometric operator was used to calculate the truth of the antecedent as given by Equation (2):

$$Ta_{i,l} = \left(\mu_{T1}^{i,j} \times \mu_{F1}^{i,j} \times \mu_{F2}^{i,j} \times \mu_{Persistence}^{i,j}\right)^{\frac{1}{4}} \tag{2}$$

In this work, the minimum operator will be used for the reasons discussed in Section 3. The truth of antecedent will be calculated using Equation (13):

$$Ta_{i,l} = \min\left(\mu_{T1}^{i,j1}, \mu_{F1}^{i,j2}, \mu_{F2}^{i,j3}, \mu_{T2}^{i,j4}, \mu_{Persistence}^{i,j5}\right) \tag{13}$$

The superscript j in the Equation has been replaced by j1, j2, j3, j4, j5 in Equation (13) to acknowledge that the variables can have different linguistic categories. The truth of the consequent was earlier calculated using Equation (3):

$$Tc_{i,l} = \mu_{T3}^{i,j} \tag{3}$$

In this work, the truth of the consequent will be calculated as:

$$Tc_{i,l} = \min\left(\mu_{T3}^{i,j6}, \mu_{Delay}^{i,j7}\right) \quad (14)$$

The value of μ_{delay} requires the delay to be quantified. A statistical technique is proposed to quantify delay and will be discussed in Section 3.4.

- *Dominating variable*: A variable in the antecedent of a rule, which has the lowest membership value as compared to the membership values of the other variables in the antecedent of the given rule, for a given data point, will be called a *dominant variable* in the antecedent. For example: consider the antecedent A1 as:

T1 is Low **AND** F1 is Med **AND** F2 is Low **AND** T2 is Low **AND** Persistence is Med

Consider that $\mu_{T1} = 0.55$, $\mu_{F1} = 0.59$, $\mu_{F2} = 0.83$, $\mu_{T2} = 0.57$ and $\mu_{persistence} = 0.65$ for this rule. Then, T1 will be the dominating antecedent variable since it has the lowest membership value for this antecedent. Similarly, a variable in the consequent of a rule, which has the lowest membership value as compared to the membership values of the other variables in the consequent of the given rule, for a given data point, will be called a *dominant variable* in the consequent. It should be noted that the dominating variable controls the calculation of the truth of the antecedent and consequent.

- *Interacting rules*: The rules with the same antecedent but different consequents (Here, same and different imply same and different linguistic values of variables), in a given rule set, will be called interacting rules. An example of two interacting rules is:

If F1 is High **AND** F2 is High then after Short delay F3 is High

If F1 is High **AND** F2 is High then after Med delay F3 is Low

- *Non-interacting rules*: The rules with different antecedents will be called *non-interacting rules*. They may or may not have the same consequent. An example of two non-interacting rules is:

If F1 is Low **AND** F2 is High then after Short delay F3 is High

If F1 is High **AND** F2 is High then after Med delay F3 is Low

- *Non-interacting group*: A set of rules consisting of all possible interacting rules describing a particular phenomenon in the process (these do not interact with any rule outside their group) would be called a *non-interacting group*. This will be discussed in further detail in Section 3.5.
- *Over specification*: If a variable is not a relevant variable for the given process, but is included in the rule base then this condition will be called *over specification*. If this irrelevant variable is included in the antecedent of the rules then this condition will be called as over specification in the antecedent and if this irrelevant variable is included in the consequent of the rules then this condition will be called as over specification in the consequent.
- *Under specification*: If a variable is a relevant variable for the given process, but is not included in the rule base then this condition will be called *under specification*. If this relevant variable is missing from the antecedent of the rules then this condition will be called as under specification in the antecedent and if this relevant variable is missing from the consequent of the rules then this condition will be called as over specification in the consequent.
- *Complete rule*: A rule in which the antecedent part includes all the possible mechanisms which can lead to the consequent will be called a complete rule. It

should be noted that this definition of a complete rule is not based on the under specification of variable in the antecedent, but it is based on the presence of an "OR" operator in the rule (to reflect different phenomena leading to the consequent). Thus, a rule will be complete, even if there is an under specification in the antecedent, as long as it covers all the possible mechanisms for the process.

- *Incomplete rule*: A rule in which the antecedent part does not involve all the possible mechanisms which can lead to the consequent will be called as an incomplete rule. This means that an "OR" operator is missing in the rule.
- *Defined system*: A system for which *all the possible mechanisms, relevant antecedent and consequent variables* form a part of the rule base will be called a defined system. A system for which all the possible mechanisms, relevant antecedent and consequent variables are not known beforehand is not a defined system.
- *Change in the rule names*: Previously, a rule was defined as a good or a bad rule based on the number of trips in QI and QIV. However, these names are over simplistic since a good rule might not be completely good and vice versa. In this research, the terms "valid rule" and "invalid rule" will substitute the terms "good rule" and "bad rule" respectively. In addition, a rule was allowed to be called a "valid rule" or an "invalid rule" even if the system was not a defined system. However, in this work, the terms "valid rule" or an "invalid rule" will be used only if the system is a defined system.
- *Shifting of trips*: The situation in which the Quadrant/Quadrants in which the trips should have been made do not register a trip. These trips shift to some other

Quadrants. This situation would occur when the system is not a defined system or when the delay calculations are inaccurate or the membership functions are not properly chosen.

3.2 Reinvestigating the TSD

The significance of the trips in the four Quadrants of the TSD, as provided by Kumar [4], has been discussed in the Section 2.3.2. This assigned meaning to the Quadrants of the TSD is based on the assumption that all the possible relevant antecedent and consequent variables were known beforehand. However, the broad goal of this research is to use the genetic algorithms to establish relationships between the variables and then to extract the valid rules. The genetic algorithms may not be able to find all the relevant variables or may include a variable which is not relevant in the initial rule base. Thus, the role of the TSD should be to indicate an over specification or under specification of the variables in the rules. The current understanding of the TSD doesn't acknowledges this possible over specification or/and under specification of the variables. However, these factors would influence the TSD as discussed in Sections 3.2.1, 3.2.2, and 3.2.3:

3.2.1 Over specification in antecedent

Consider the hot-cold water simulator of Figure 2. The five relevant input variables are T1, F1, F2, T2 and Persistence and the two output variables are T3 and delay. Assume that another variable, pressure in the atmosphere, has been accidentally

included in the antecedent of the rules by the genetic algorithm. This may change the calculation for the truth of the antecedent as $\mu_{Pressure}$ will be used in its calculation. When $\mu_{Pressure}$ is the dominating antecedent variable then the value of T_a will be affected by this irrelevant variable. The truth of the antecedent and consequent will be calculated as:

$$Ta_{i,l} = \min\left(\mu_{T1}^{i,j1}, \mu_{F1}^{i,j2}, \mu_{F2}^{i,j3}, \mu_{T2}^{i,j4}, \mu_{Persistence}^{i,j5}, \mu_{Pressure}^{i,j6}\right) \quad (15)$$

$$Tc_{i,l} = \min\left(\mu_{T3}^{i,j6}, \mu_{Delay}^{i,j7}\right) \quad (14)$$

This can affect the trips in QI and QIV in three possible ways:

- *No effect*: If the pressure is not a dominating variable, then it will not affect the trip. This is because the position of the trip in the TSD is based on the value of the T_a and the value of the T_a is decided by the dominating variable.
- *Same quadrant effect*: If the pressure is a dominating variable with $\mu_{pressure} > 0.5$ then it will not change the Quadrant of the trips in either QI or QIV but would change the T_a. For example: Assume that $\mu_{T1} = 0.72$, $\mu_{F1} = 0.81$, $\mu_{F2} = 0.83$, $\mu_{T2} = 0.68$ and $\mu_{persistence} = 0.74$, $\mu_{pressure} = 0.55$, $\mu_{delay} = 0.34$ and $\mu_{T3} = 0.59$. In this case, the pressure is dominating variable. If the pressure would not have been mistakenly included in the rule base then the T_a would have been equal to 0.68, as given by Equation (13), and a data point would have been seen in the fourth Quadrant (because $T_c = 0.34$) of the TSD for the given rule. However, since the pressure is the dominating variable and forms the part of the rule base the T_a would be equal to 0.55, as given by Equation (15), and the data point would continue to appear in QIV. Thus, the inclusion of the pressure doesn't change the

Quadrant of the point but does changes the T_a value. This is the reason that this situation is called the *same quadrant effect*. Similarly, if the pressure is a dominating variable with $\mu_{pressure} < 0.5$ and the membership values of all the other variables are less than 0.5, then it will not change the Quadrant of the trips in QII and QIII, but would change the T_a. For example: Assume that $\mu_{T1} = 0.43$, $\mu_{F1} = 0.41$, $\mu_{F2} = 0.33$, $\mu_{T2} = 0.28$ and $\mu_{persistence} = 0.34$, $\mu_{pressure} = 0.16$, $\mu_{delay} = 0.54$ and $\mu_{T3} = 0.59$. In this example, the $T_a = 0.16$ and $T_c = 0.54$. Thus, a point would have appeared in the QII. If the pressure would not have been mistakenly included in the rule base then T_a would be 0.28 and T_c would have remained equal to 0.54. This point would again lie in the QII, but the truth of the antecedent has changed. Thus, due to the same quadrant effect the trips in QI, QII, QIII, and QIV do not change their Quadrants but shift towards the left. The same quadrant effect is not highly undesirable since it will not cause any shifting of trips from one Quadrant to the other and wouldn't create problems in the selection of the valid rules.

- *Different quadrant effect*: If the pressure is a dominating variable with $\mu_{pressure} < 0.5$ and other variables have a membership value greater than 0.5 then it will change the Quadrant of the trips. As an example: Assume that $\mu_{T1} = 0.53$, $\mu_{F1} = 0.61$, $\mu_{F2} = 0.63$, $\mu_{T2} = 0.78$ and $\mu_{persistence} = 0.74$, $\mu_{pressure} = 0.36$, $\mu_{delay} = 0.54$ and $\mu_{T3} = 0.59$. This point would appear in QII since $T_a = 0.36$ and $T_c = 0.54$. However, if the pressure would not have been mistakenly included in the rule base then this point would have appeared in QI ($T_a = 0.53$, $T_c = 0.54$). Thus, due to the different quadrant effect, the trips which should have been in Quadrants QI and QIV would show up in QII and QIII respectively. The different quadrant

effect is undesirable because it leads to shifting of the trips from the Quadrants on the right (QI and QIV) to the Quadrants on the left (QII and QIII) and thus affects the rule-selection process. Figure 13 shows the different quadrant effect due to the over specification in the antecedent. A trip which should have been QI was made in QII, and a trip which should have been in QIV was made in QIII due to this affect. The tails of the arrows start from the data points which should have appeared, if there was no over specification in antecedent, and the head of the arrows point towards the data points which actually appeared in the TSD due to the over specification in the antecedent.

Different Quadrant Effect due to over specification in the antecedent

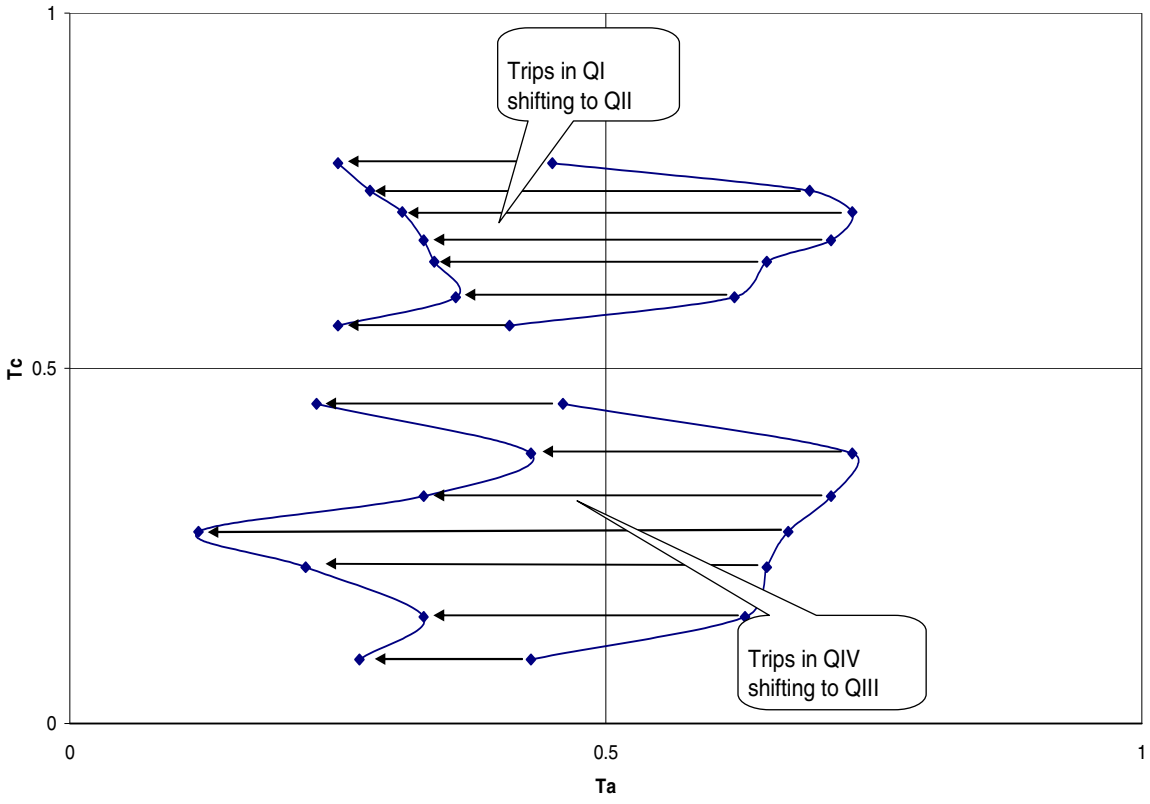


Figure 13: TSD showing the different quadrant effect, of over specification in the antecedent, on the trips. The trips in QI move to QII.

Figure 14 shows the all the three affects (No effect, Same quadrant effect, Different quadrant effect) of the over specification in the antecedent. The tails of the arrows start from the data points which should have appeared, if there was no over specification in antecedent, and the head of the arrows point towards the data points which actually appeared in the TSD, due to the over specification in the antecedent.

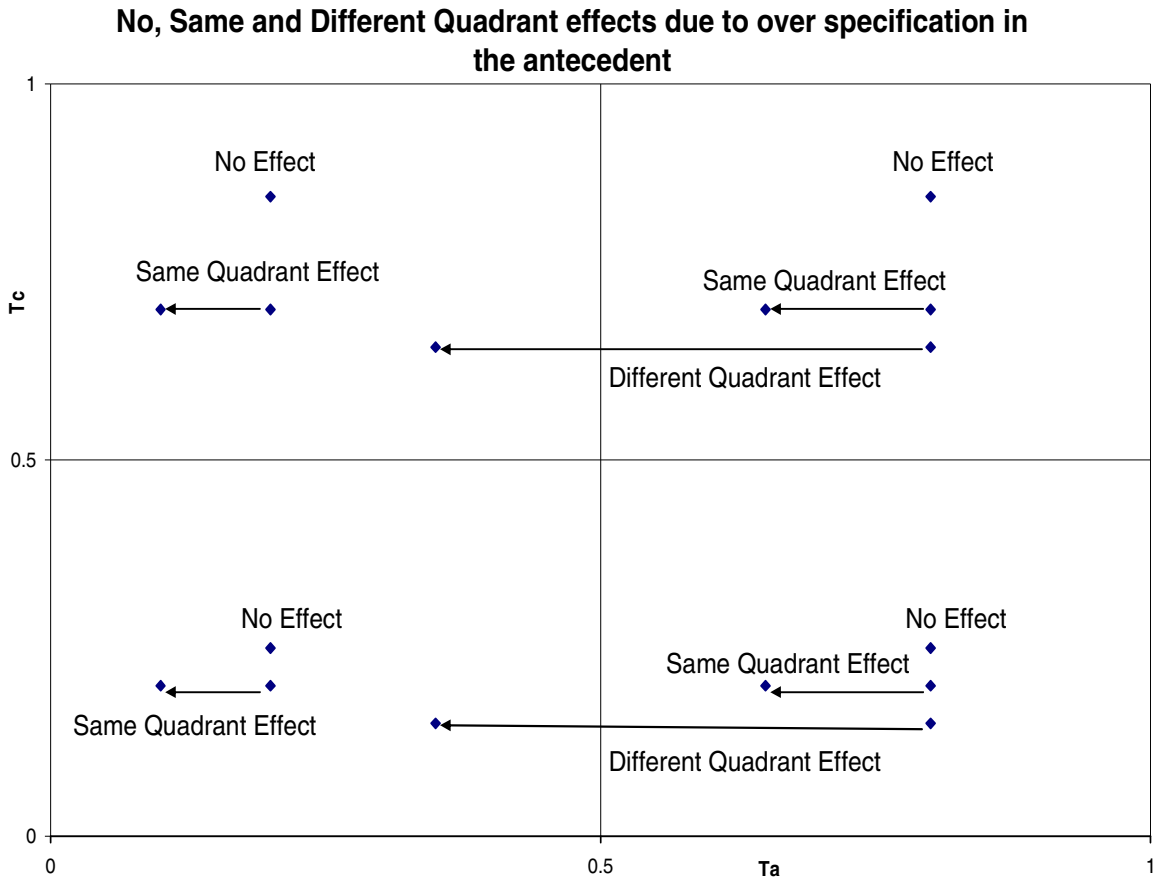


Figure 14: TSD showing the three different effects of the over specification in the antecedent on the positioning of the points in the four Quadrants.

3.2.2 Under specification in the antecedent

Assume that the genetic algorithm was not able to discover one of the antecedent variables, T2 and thus this variable will not be included in the initial rule base. This leads to an under specification in the antecedent, and the truth of the antecedent will be calculated as:

$$Ta_{i,l} = \min\left(\mu_{T1}^{i,j1}, \mu_{F1}^{i,j2}, \mu_{F2}^{i,j3}, \mu_{Persistence}^{i,j5}\right) \quad (16)$$

The missing variable, T2, may affect the trips in following three ways:

- *No effect*: If the variable, T2, is not a dominating variable then its absence will not affect the trip.
- *Same quadrant effect*: If T2 is a dominating variable with $\mu_{T2} < 0.5$ and other variables (in the antecedent) have a membership value less than 0.5, then the absence of T2 will not change the Quadrant of the trip, but would change the T_a. Similarly, if T2 is a dominating variable with $\mu_{T2} > 0.5$ and other variables have a membership greater than 0.5, then it will not change the Quadrant of the trip, but would change the T_a. Thus, this same quadrant effect will shift the trips from left to right, but will not cause any shifting of trips from one Quadrant to the other. This affect will not create any problem in the rule-selection process.
- *Different quadrant effect*: If T2 is a dominating variable with $\mu_{T2} < 0.5$ and other variables have a membership value greater than 0.5, then the absence of T2 will change the Quadrant of the trip. For example: Assume that $\mu_{T1} = 0.53$, $\mu_{F1} = 0.61$, $\mu_{F2} = 0.63$, $\mu_{T2} = 0.30$ and $\mu_{persistence} = 0.74$, $\mu_{delay} = 0.54$ and $\mu_{T3} = 0.59$. Thus, the actual T_a should be equal to 0.3, and this point should appear in QII. However, if the variable T2 was not included in the rule base, the T_a from the Equation (16)

will be 0.53 and the trip would be made in the QI. Thus, trips which should have been in QII and QIII would shift to QI and QIV respectively. This affect is undesirable because it will create problems in the rule-selection process.

Thus, an under specification in the antecedent can shift the trips from left to right. Figure 15 shows the all the three affects (No effect, Same quadrant effect, Different quadrant effect) of the under specification in the antecedent. The tails of the arrows start from the data points which should have appeared, if there was no under specification in antecedent, and the head of the arrows point towards the data points which actually appeared in the TSD, due to the under specification in the antecedent.

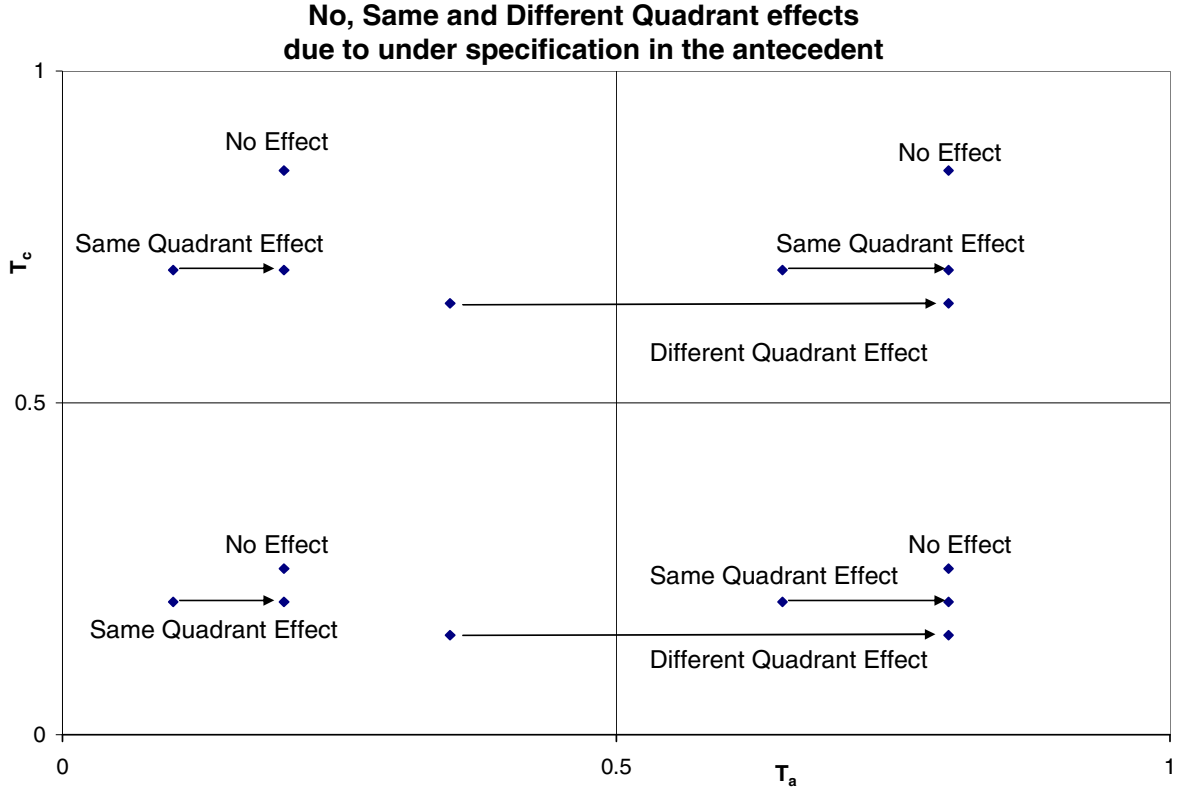


Figure 15: TSD showing the three different effects of the under specification in the antecedent on the positioning of the points in the four Quadrants.

3.2.3 Over specification in the consequent

In addition to the delay and the temperature T3, if an extraneous variable is added to the consequent which is not relevant to the process, then it can lead to misleading values of the truth of the consequent. Assume that another variable, pressure in the atmosphere, has been accidentally included in the consequent of the rules by the genetic algorithm. In that case, the T_c will be calculated as:

$$Tc_{i,l} = \min\left(\mu_{T3}^{i,j6}, \mu_{Delay}^{i,j7}, \mu_{\Pr essure}^{i,j8}\right) \tag{17}$$

The extraneous variable, Pressure, may affect the trips in following three ways:

- *No effect*: If the variable, Pressure, is not a dominating variable then its presence will not affect the trip.
- *Same quadrant effect*: If Pressure is a dominating variable with $\mu_{pressure} > 0.5$ and other variables (in the consequent) have a membership value greater than 0.5, then the presence of the variable, Pressure, will not change the Quadrant of the trip, but would change the T_c. Similarly, if Pressure is a dominating variable with $\mu_{Pressure} < 0.5$ and other variables have a membership value less than 0.5, then it will not change the Quadrant of the trip, but would change the T_c. Thus, this *same quadrant effect* will shift the trips from top to bottom, but will not cause any shifting of trips from one Quadrant to the other. This affect will not create any problem in the rule-selection process.
- *Different quadrant effect*: If Pressure is a dominating variable with $\mu_{Pressure} < 0.5$ and other variables have a membership value greater than 0.5, then the presence of the variable, Pressure, will change the Quadrant of the trip. Thus, trips which

should have been in QI and QII would shift to QIV and QIII respectively. This affect is undesirable because it will create problems in the rule-selection process.

Thus, an over specification in the consequent can shift the trips from top to bottom. Figure 16 shows the all the three affects (No effect, Same quadrant effect, Different quadrant effect) of the over specification in the consequent. The tails of the arrows start from the data points which should have appeared, if there was no under specification in antecedent, and the head of the arrows point towards the data points which actually appeared in the TSD, due to the over specification in the consequent.

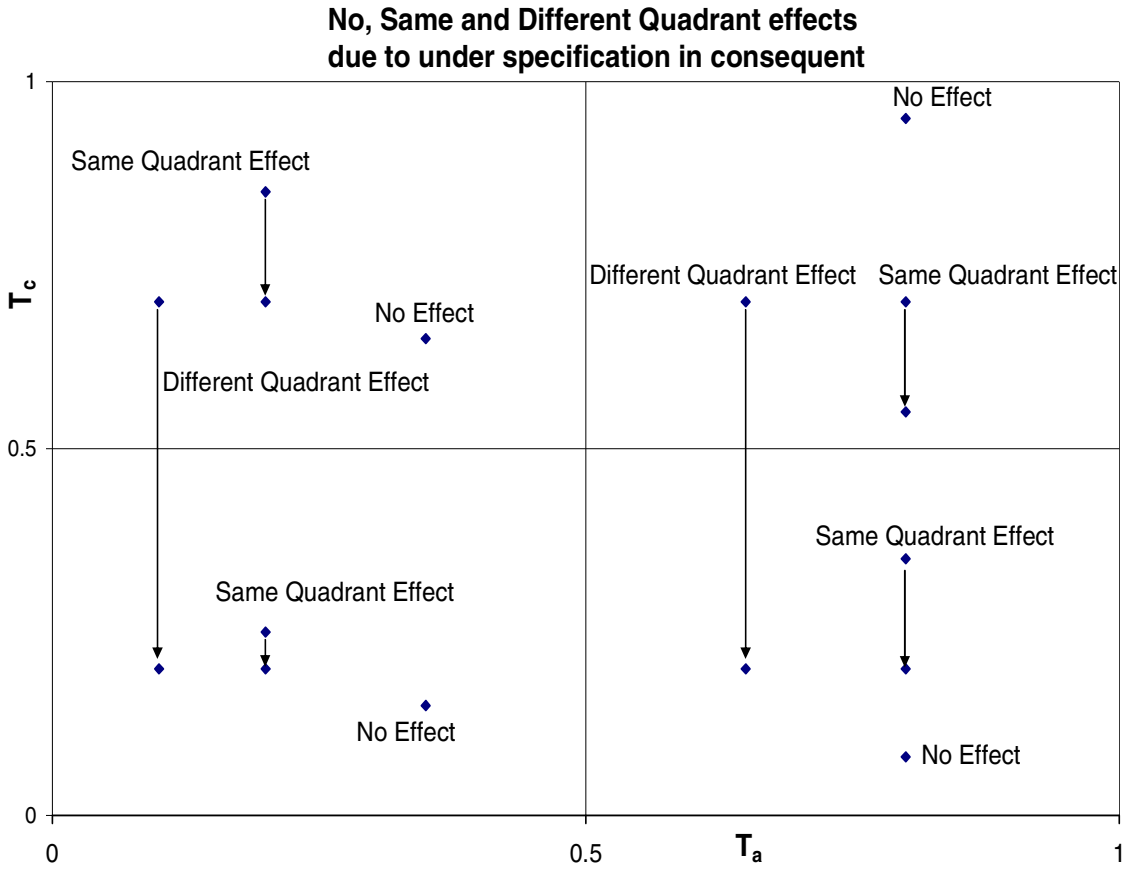


Figure 16: TSD showing the three different effects of the over specification in the consequent on the positioning of the points in the four Quadrants.

3.2.4 Combination of any of the above cases

Genetic algorithms may obtain rules which are affected from over specification as well as under specification of the antecedent and/or consequent. In that case, the analysis of the shifting of the trips becomes quite complex and several possible cases can be obtained by combining the cases discussed above.

Other factors which may lead to the shifting of the trips are the membership functions of the variables and inaccurate calculation of the delay. These issues are not considered in this work.

The two main goals of the TSD are:

- To assess the validity of a rule.
- To indicate the over specification and under specification of the antecedent and the consequent.

To increase the efficiency of the TSD in terms of indicating over specification and under specification of the variables and in assessing the validity of a rule, it is essential to develop a methodology for indicating over specification and under specification. The issues related to the over specification in the consequent can be solved by considering the rule bases with different consequents as different rule sets. Only one output variable, in addition to the delay, should be allowed to be a part of the rule set. For example: In the hot-cold water simulator, the input variables are: T1, F1, F2, T2 and persistence and the output variables are T3, F3, Delay1 (for T3) and Delay2 (for F3). To solve the problem of over specification in the consequent, two separate rule sets should be used. In one system, rules will consist of the five given inputs and, T3 and Delay1 as the consequent. In the second system, the rules will consist of the five given inputs and, F3 and Delay2 as the

consequent. This will help in eradicating the problem of over specification of the consequent. In addition, it will require lesser data to obtain rules because the membership values of both the variables, F3 and T3, do not have to be greater than 0.5 to get T_c to be greater than 0.5.

Another major advantage of using two different rule sets instead of one comprehensive rule set is that the computational effort required for the management of two rule sets is lesser than that for a single comprehensive rule set. For example: Assume that two rule sets, Fuzzy1 and Fuzzy2, were employed in the hot-cold water simulator. Fuzzy1 consists of rules with 7 variables (T1, T1, F1, F2, persistence, Delay1, T3) and Fuzzy2 also consists of 7 variables (T1, T1, F1, F2, persistence, Delay2, F3). Then, the total number of rules in both the rule sets will be $3^7 + 3^7 = 4374$. Now, let's compare it with a single rule set, Fuzzy3, which consists of rules with 9 variables (T1, T1, F1, F2, persistence, Delay1, Delay2, T3, and F3). The total number of possible rules in Fuzzy3 will be $3^9 = 19683$. Hence, it would require less computational effort if different consequents are considered as different rule sets rather than using all the consequent variables together in a single rule set.

This can be generalized for a process with N inputs and M outputs. The antecedent of the rules will have (N + 1) terms because persistence is also included in the antecedent of the rules. Similarly, the consequent of the rules will have (2M) terms because one delay is associated with each consequent variable. Assume that all the antecedent variables and all the consequent variables (in the rules) have n and m linguistic values respectively. Thus, the total number of possible antecedents is n^{N+1} and the total number of possible consequents is m^{2M}. Now, if a single rule base is used, then it

will consist of $(n^{N+1}. m^{2M})$ rules. However, if M rule bases are used (each rule base will consist of two consequents, Delay and one output variable), then it will consist of $(n^{N+1}.M.m^2)$ rules. The ratio, R_{1M}, of the number of rules from the single rule base to that of the total number of rules from M rule bases is given by Equation 18:

$$R_{1M} = \frac{m^{2M}}{M.m^2} = \frac{m^{2(M-1)}}{M} \tag{18}$$

The saving in the computational effort which will be achieved by using M rule bases instead of a single rule base can be seen in Figure 17. As the number of output variables increase, R_{1M} increase exponentially.

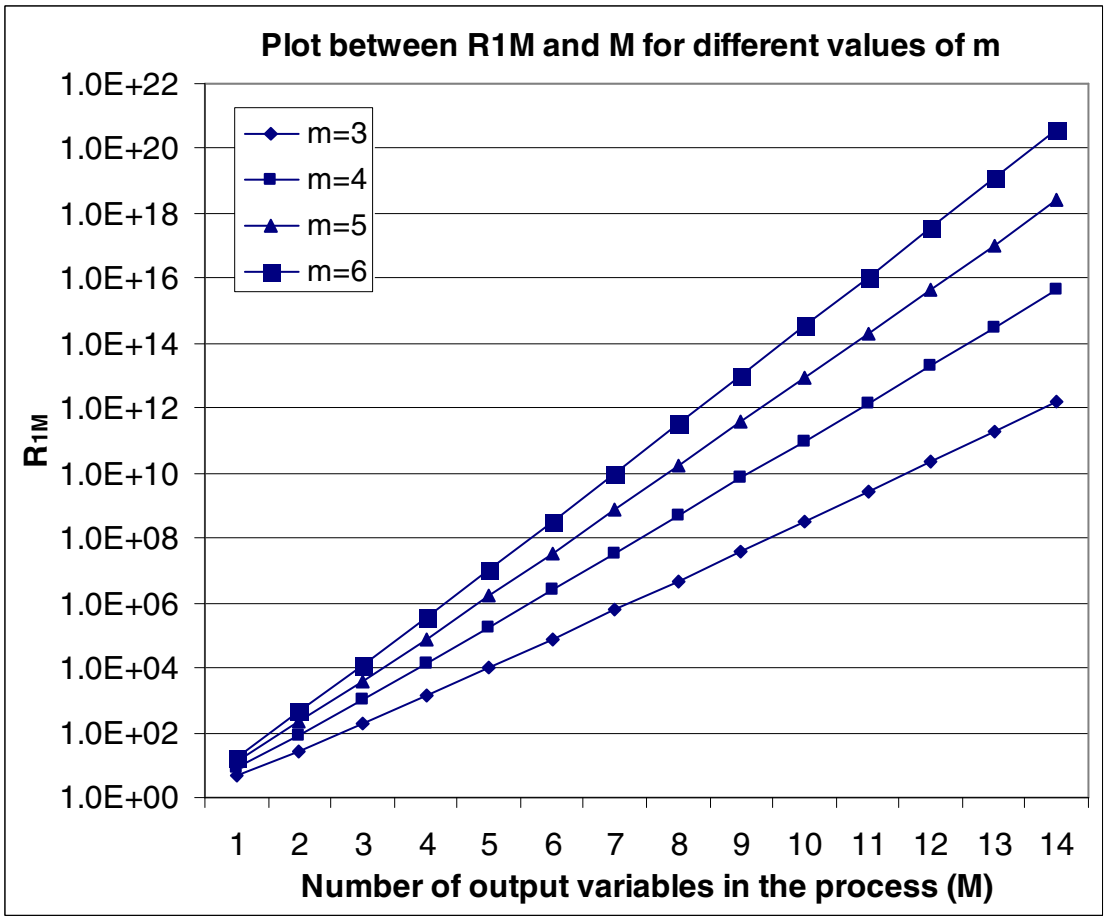


Figure 17: Plot between R_{1M} and M, showing the advantage of using M rule sets with two consequent variables each (delay and one output variable), instead of using one comprehensive rule set, consisting of 2M consequent variables.

Once the problem of over specification in the consequent has been solved, the TSD will be used in indicating the over specification and under specification of the antecedent in all the rule sets. The trips appearing in QI would indicate that either the rule is "valid" or that the trips are made in this Quadrant because of the under specification or/and over specification in the antecedent. The trips in QII would indicate that either the rule is "incomplete" i.e. there is an additional "hidden mechanism" governing the process or/and it may imply an over specification or/and under specification in the antecedent. The trips in QII will also show up because of the rules with similar consequents but different antecedents. A trip in QI, for a rule, will lead to a trip in QII for the rules which have the same consequent but different antecedents from the given rule. The trips in QIII suggest that the data is "indeterminate" i.e. neither the validity nor the invalidity of this rule can be determined from the given data in evaluating this rule. The trips in QIII may also may be present because of either the over specification or/and the under specification in the antecedent of the rules. Thus, the trips in QIII indicate that either the data is "indeterminate" for this rule or that all the relevant variables are not properly specified. Similarly, the trips appearing in QIV indicate that either the rule is "invalid" or that the trips are made in this Quadrant because of the under specification or/and over specification in the antecedent. The significance of the four Quadrants in the TSD can be seen in Table 1.

It should be noted that the trips in QI or QIV can not appear because of the different quadrant effect of the over specification in the antecedent. Similarly, the trips in QII and QIII can not appear because of the different quadrant effect of the under specification in the antecedent.

Table 1: Possibilities which can lead to trips in a given Quadrant.

Trips in	Could be because:
QI	1. Rule is a "valid" rule 2. Under specification in antecedent. This may be because of: • No effect • Same quadrant effect • Different quadrant effect 3. Over specification in the antecedent. This may be because of: • No effect • Same quadrant effect 4. Combination of over specification and under specification in the antecedent.
QII	1. Rule is "incomplete" i.e. hidden mechanism is present 2. Under specification in antecedent. This may be because of: • No effect • Same quadrant effect 3. Over specification in the antecedent. This may be because of: • No effect • Same quadrant effect • Different quadrant effect 4. Combination of "hidden mechanism" and/or over specification and/or under specification in the antecedent. 5. Rules with same antecedents but different consequents.
QIII	1. Data is "indeterminate" in evaluating the rule. 2. Under specification in antecedent. This may be because of: • No effect • Same quadrant effect 3. Over Specification in antecedent. This may be because of: • No effect • Same quadrant effect • Different quadrant effect 4. Combination of "indeterminate data" and/or over specification and/or under specification in the antecedent.
QIV	1. Rule is an "invalid" rule. 2. Under specification in antecedent. This may be because of: • No effect • Same quadrant effect • Different quadrant effect 3. Over specification in the antecedent. This can be: • No effect • Same quadrant effect 4. Combination of over specification and under specification in the antecedent.

Due to the several possible cases, discussed in Table 1, which may place a trip in a given Quadrant in the TSD, it seems difficult to indicate the presence of "hidden mechanisms", under specification or/and over specification of antecedent using the TSD. However, a closer analysis of the trips in QI and QIV can help resolve this issue. A *valid* rule should have trips in QI (if data expresses it), but should not have trips in QIV (assuming the delay calculations are accurate and optimum membership functions are used). Even if a rule is over specified in antecedent, then also trips should not appear in QI as well as QIV because the over specification can not lead to trips in QIV due to *different quadrant effect* (as shown in Table 1) or because of *No effect* and the *same quadrant effect,* as these affects would require data points to originate in QIV. Similarly, the *No effect* and the *same quadrant effect,* due to under specification in the antecedent, should not lead to trips in QIV as they require data points to originate in QIV. Thus, if trips are made in QI as well as QIV, then it should be because of the *different quadrant effect* of under specification in the antecedent. Hence, a rule having trips in both QI and QIV will indicate under specification in the antecedent. The human operator or genetic algorithms can then be used to find the under specified variable in the antecedent. This process will continue to find missing antecedents till the rules having trips in QI will almost stop registering trips in QIV. At this point, the under specification problem will be solved. After this stage, the meaning of the TSD will simplify and is shown in Figure 18:

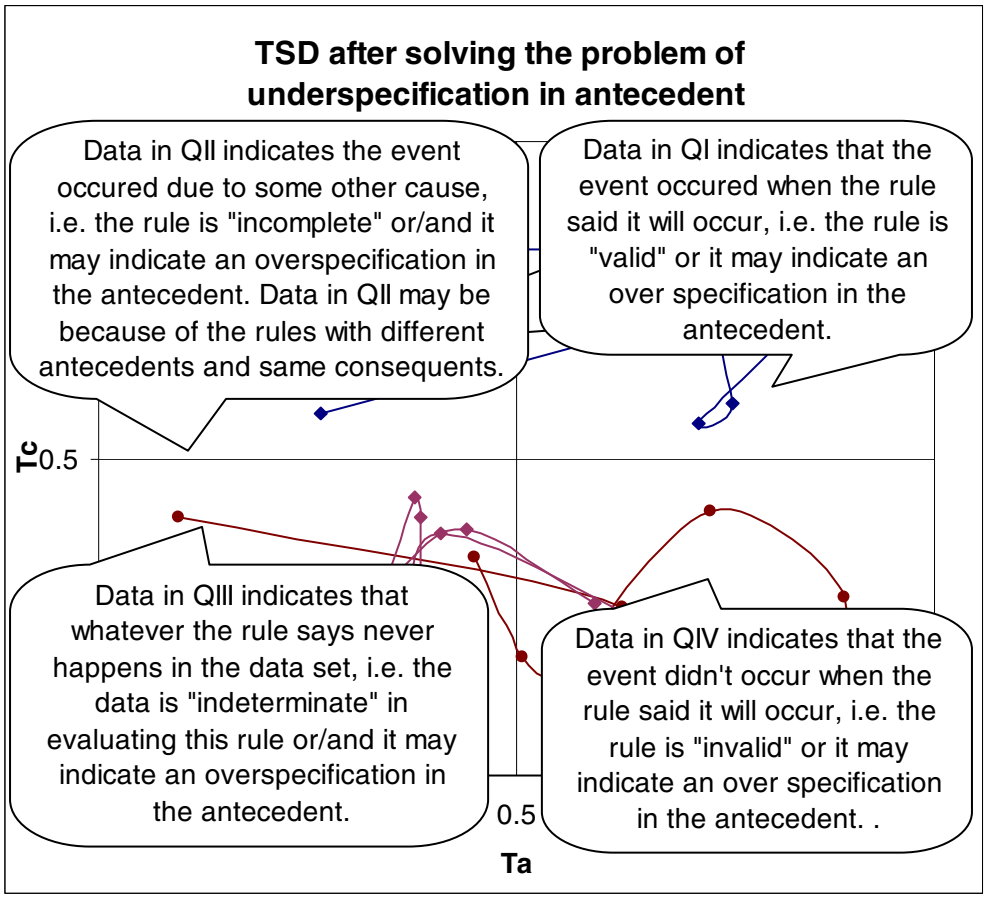


Figure 18: Interpretation of the TSD after solving the problem of under specification in the antecedent.

Having solved the issue of under specification in the antecedent, the next step will be to identify the over specification in the antecedent or/and any possible hidden mechanisms. This can be done by observing trips in QI and QII. A rule having no over specification in antecedent or/and hidden mechanism should not give trips in QII. Thus, a rule having trips in QI as well as QII would indicate the possibility of over specification in the antecedent or the presence of a hidden mechanism which is affecting the output. At this stage, the human operator or genetic algorithms can be employed to discover any possible hidden mechanisms or to discard any irrelevant variables in the antecedent. This process will end when the rules will almost stop registering trips in QII. However, it

should be noted that few data points would still appear in QII as a result of the errors in delay calculation and because of the membership function of the variables. The meaning of the TSD at the end of this process is shown in the Figure 19:

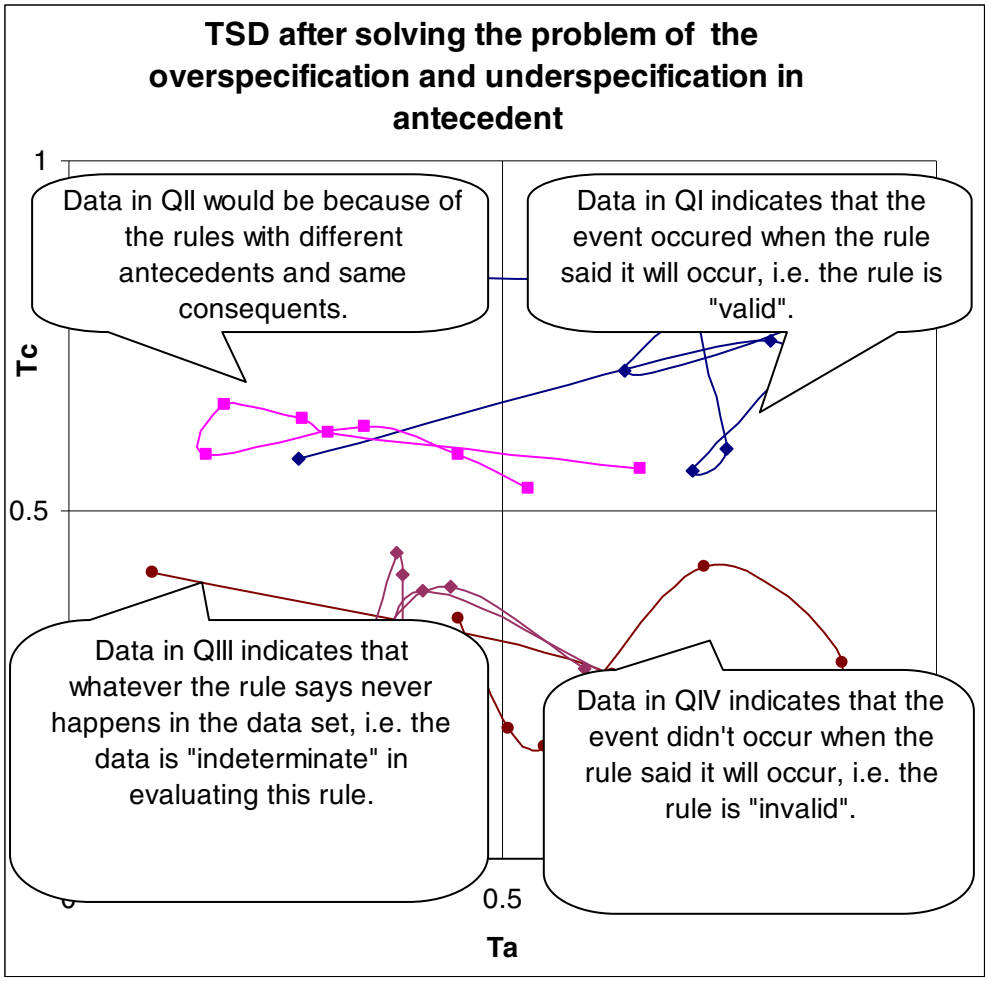


Figure 19: TSD for a defined system. The trips do not appear in QII because the problem of over specification and under specification in the antecedent, and the hidden mechanisms has been solved.

The rest of this work will assume the TSD shown in Figure 19 as the basis.

3.3 Choice of the mathematical operator to calculate T_a and T_c

The two main operators which are used in the literature of fuzzy logic to calculate T_a and T_c are minimum operator and product operator. In this research, a geometric mean operator was also explored. These three operators are used as:

Minimum Operator

$$Ta_{i,l} = \min\left(\mu_{T1}^{i,j1}, \mu_{F1}^{i,j2}, \mu_{F2}^{i,j3}, \mu_{T2}^{i,j4}, \mu_{Persistence}^{i,j5}\right) \tag{13}$$

$$Tc_{i,l} = \min\left(\mu_{T3}^{i,j6}, \mu_{Delay}^{i,j7}\right) \tag{14}$$

Product Operator

$$Ta_{i,l} = \left(\mu_{T1}^{i,j1} \times \mu_{F1}^{i,j2} \times \mu_{F2}^{i,j3} \times \mu_{T2}^{i,j4} \times \mu_{Persistence}^{i,j5}\right) \tag{18}$$

$$Tc_{i,l} = \left(\mu_{T3}^{i,j6} \times \mu_{Delay}^{i,j7}\right) \tag{19}$$

Geometric mean operator

$$Ta_{i,l} = \left(\mu_{T1}^{i,j1} \times \mu_{F1}^{i,j2} \times \mu_{F2}^{i,j3} \times \mu_{T2}^{i,j4} \times \mu_{Persistence}^{i,j5}\right)^{\frac{1}{5}} \tag{20}$$

$$Tc_{i,l} = \left(\mu_{T3}^{i,j6} \times \mu_{Delay}^{i,j7}\right)^{\frac{1}{2}} \tag{21}$$

In the literature, a product operator has been recommended when the requirement is for the most robust *in the average* fuzzy logic and minimum operator has been recommended when the need is for the most robust *in the worst case* fuzzy logic. However, in the TSD the minimum operator would be more suitable than the product operator because when the number of variables would become large, then it would become difficult to extract rules using product operator. For example: Consider a process

with 15 input variables. Assume that for a given rule each of these 15 antecedents had a membership value of 0.95. Then, using product operator the T_a would be $(0.95)^{15}$=0.46. Thus, even with such high truth in individual antecedents, the rule would not give a trip in QI or QIV (assuming the Quadrant size is set at $T_a = 0.5$ and $T_c = 0.5$). On the other hand, minimum operator would give a value of 0.95 for the same example. Thus, it is computationally efficient to obtain trips using the minimum operator. However, it should be noted that the minimum operator is discontinuous and hard to differentiate during the optimization process.

Due to the increase in inefficiency of the product operator, in obtaining trips, with increasing variables, the geometric operator was suggested. However, the problem with the geometric operator was that it could accept semi-valid rules. For example: let's assume a rule with $T_a > 0.5$ which has two consequents with membership values .35 and .8. Then, the geometric operator would give $T_c = (0.35*0.8)^{0.5}$=0.53. It would be considered as a trip in the QI Quadrant. Thus, the final rule base could consist of many rules which are not the best in describing the process.

In addition, the minimum operator assists in performing a clearer analysis of the system. For example: the minimum operator supports the analysis of the over specification and under specification discussed in the Section 3.2. On the other hand, the geometric mean operator does not possess the ability to clear distinguish the over specification and under specification, because the concept of the dominating variable will no longer exist. Thus, the meanings assigned to the Quadrants in Figures 18 and 19 will no longer hold true. Furthermore, the minimum operator would also prove useful in

giving a universal applicability to the technique (as explained in the next few sections) by:

- Providing the fundamental tool based on which the comparison of rules will be performed.
- Assisting in identifying the Quadrant size in the TSD
- Obtaining the metric to be used and in calculating its universal value.
- Reducing the curse of dimensionality.

Hence, the minimum operator would be used to calculate T_a and T_c in this work.

3.4 Delay Quantification

The process delay is used in the linguistic rules to capture the dynamics of the process. However, the delay is not a measurable quantity like T1, T2, F1, F2 and T3. In addition, the scope of this research is not limited to a given process, but it should be applicable to any process which is governed by the cause-and-affect rules. This requires that delay calculation should not be based on a mechanistic model of the process. As explained in the Section 2.2.2, Sharma [3] assumed constant and intuitive values for short, medium and long delays. These crisp values of the delays could not be fuzzified since the ranges of delay values were not available. Only three possible values for the delay were used. Due to this, the membership value of delay could not be used in the calculation of the T_c. Thus, a technique is required to quantify delay values which will acknowledge the possibility of variant delays in the process.

The delay establishes a relationship between the inputs and outputs of the process. The delay quantification can be performed if we can capture this relationship between the inputs and outputs. In fuzzy logic, the relation between T_a and the membership of the desire output variable can be used to quantify delay. In this work, a technique has been developed for establishing this relationship and is explained in the following steps:

1) Ask the user to provide the value of the maximum possible delay (M) in the process.
2) For a given rule r, compute the truth of antecedent (T_a) and the membership value of the output (μ_{T3}), for each data point.
3) Scan all the T_a values for the regions which have five or more consecutive points with $T_a > 0.5$. These regions will either give a trip in QI or QIV depending on the value of the calculated delay and μ_{T3}. Let the number of points in a given region be g (By convention, integers are represented by the letters i, j, k, l, m, or n but these symbols have been already used for other purposes in this work. Thus, the symbol g is used).
4) For each region, find the value of the Pearson's correlation coefficient ($-1 \le r \le 1$) between the g points of T_a, and the g points of μ_{T3} delayed backwards by a value equal to the sampling time. Continue finding the value of the Pearson's correlation coefficient with μ_{T3} delayed by 2, 3, 4……..M times the sampling time. The steps involved in the calculation of the Pearson's correlation coefficient are:
    - Calculate $S(T_a)$, the sum of the g points of T_a and $S(\mu_{T3})$, the sum of the g points of μ_{T3} as shown in the Equations (22) and (23). In addition,

calculate $S(T_a^{\,2})$, the sum of the squares of the g points of T_a; $S(\mu_{T3}^{\,2})$, the sum of the squares of the g points of μ_{T3}; and $S(T_a\mu_{T3})$, the sum of the product of T_a and μ_{T3} for the g points of the region as shown in Equations (24), (25), and (26) respectively.

$$S(T_a) = \sum_{i=1}^{g} T_a[i] \tag{22}$$

$$S(\mu_{T3}) = \sum_{i=1}^{g} \mu_{T3}[i] \tag{23}$$

$$S(T_a^{\,2}) = \sum_{i=1}^{g} T_a^{\,2}[i] \tag{24}$$

$$S(\mu_{T3}^{\,2}) = \sum_{i=1}^{g} \mu_{T3}^{\,2}[i] \tag{25}$$

$$S(\mu_{T3}T_a) = \sum_{i=1}^{g} T_a[i]\mu_{T3}[i] \tag{26}$$

- Calculate the variance of T_a, $Var(T_a)$; the variance of μ_{T3}, $Var(\mu_{T3})$; and the covariance of T_a and μ_{T3}, $Co\,\mathrm{var}(T_a\mu_{T3})$ as shown in Equations (27), (28), and (29) respectively.

$$Var(T_a) = \frac{S(T_a^{\,2})}{g} - \frac{(S(T_a))^2}{g^2} \tag{27}$$

$$Var(\mu_{T3}) = \frac{S(\mu_{T3}^{\,2})}{g} - \frac{(S(\mu_{T3}))^2}{g^2} \tag{28}$$

$$Co\,\mathrm{var}(T_a\mu_{T3}) = \frac{S(\mu_{T3}T_a)}{N} - \frac{S(T_a)S(\mu_{T3})}{N^2} \tag{29}$$

- Calculate the square of the Pearson's correlation coefficient (r^2) as given by Equation (30). The square of Pearson's correlation coefficient (r^2) is used instead of Pearson's correlation coefficient (r) because in delay calculations only the strength of the relationship between T_a and μT3 is of interest, not whether it has an either a positive or negative nature of the correlation.

$$r^2 = \frac{(Co\,\mathrm{var}(T_a \mu_{T3}))^2}{Var(T_a \mu_{T3})} \tag{30}$$

5) Compare the value of these correlation coefficients to each other and find the value of the sampling time, d, at which the correlation coefficient is the maximum.
6) This value d will be referred to as the delay associated with that given trip.
7) Repeat the steps 1-6 for all the regions in rule r with $T_a > 0.5$.
8) Repeat the steps 1-7 for all the rules in the process.

A good estimate for the value of the maximum possible delay associated with the process, M, would provide robustness to this technique. A value of M which is either too low or too high would give inaccurate estimates for the delay value. If the value of M is much lower than the actual value of the maximum possible delay in the process then this technique will not be able to give a good estimate (as the correlation coefficient are compared only for time intervals from 1,2…….M) of Delay. Thus, if the actual delay for a trip is greater than M then this technique would not be able to obtain it.

If the value of M is much higher than the actual value then this might also lead to inaccurate estimates of the value of delay. This is because the process variables (and thus,

T_a and T_c) do not generally vary monotonically i.e. these process variables keep varying in a cyclic manner. Thus, if the value of M is very high then it is possible that for a given region of T_a the T_c would finish its one cycle (increase, decrease and again increase or vice versa) and then it might lead to a higher value of the correlation coefficient at the other cycle (even though no such correlation exists). An example of such a scenario is shown in Figure 20. Assume that a value of 10 was chosen for M. Again, assume that the input variable affect the output variable but after a delay value of 1 second. Thus, this delay quantification should find the maximum value of the correlation coefficient between the input and output variables at a delay value of 1 second. However, the output variable is varying in a cyclic manner. Thus, it is possible that a large value of the correlation coefficient would be obtained at a delay of 8 seconds, too. If this value of the correlation coefficient is greater than the value of the correlation coefficient at the delay of 1 second then this technique would assign a delay value of 8 second for this case (which is incorrect). However, if we have a good estimate of M for this process (Let's say M= 5 sec) then the correlation at 8 seconds would not be found. Hence, a good estimate for the M value would provide robustness to this delay quantification technique.

Thus, in this technique of delay quantification the entire system is divided into several smaller regions (trips) and a single value of delay is associated with each trip. In the Table 2, the results for delay quantification for a trip in a given rule are shown. N.R. represents that the value was *not required* in the calculations.

Firstly, the entire T_a column in the Table 2 was scanned in search for prospective trips in QI or QIV ($T_a > 0.5$). Then, a value for M = 50 second was chosen based on the knowledge of the process. One such prospective trip was found and it consisted of 8

points (g=8). Then, the correlation coefficients were calculated between the g points of T_a and the g points of μ_{T3} delayed by 1, 2, 3……50 seconds. The maximum value of the correlation coefficient was obtained at a delay value of 26 seconds. This value of the delay was then assigned to the given trip. Based on a triangular membership function, the delay value was then fuzzified and the value of μ_{delay} was found to be 0.3. The μ_{T3} values were undelayed by shifting the μ_{T3} values backward in time by 26 seconds. The truth of the consequent was then calculated by using the minimum operator and a value of 0.3 was calculated for it. Thus, this trip resulted in a trip in QIV.

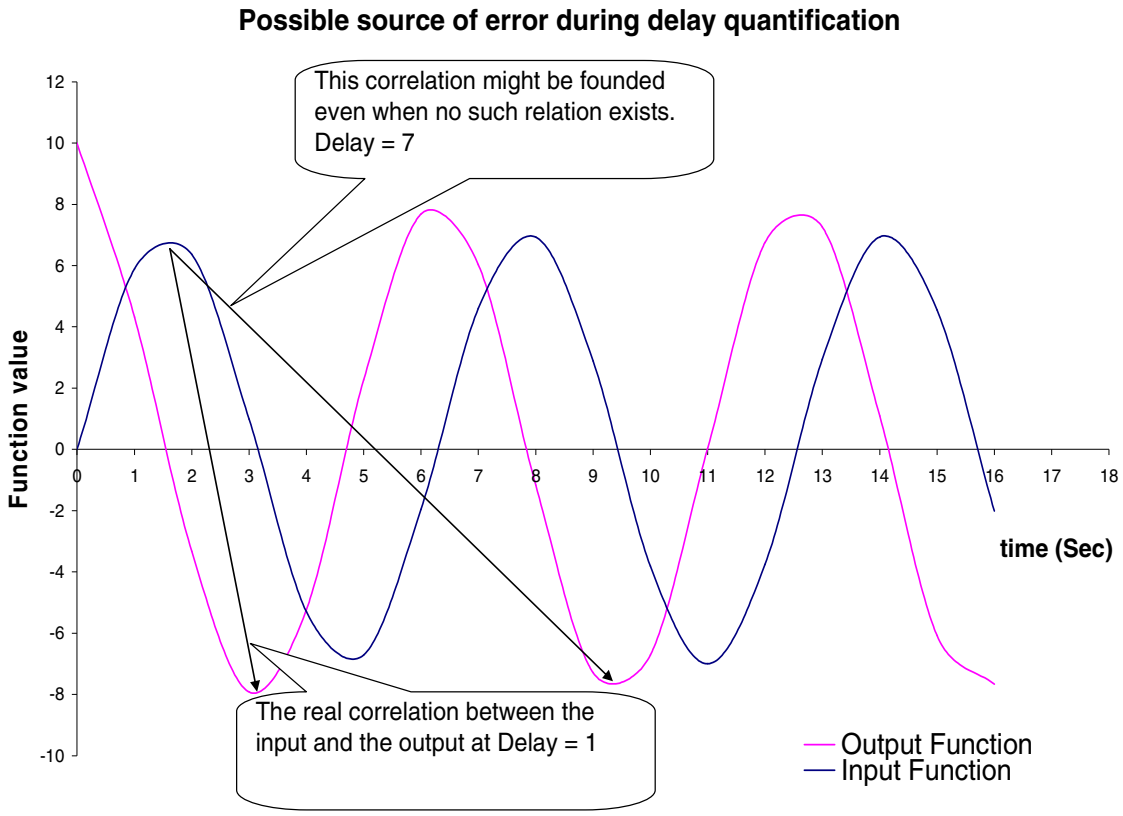


Figure 20: Affect of choosing a large value for M on delay quantification.

Table 2: Delay quantification for a trip in a given rule in the hot-cold water simulator. N.R. represents that the value was *not required* in the calculations.

Rule 1789: IF Temp1 is MED & F1 is LOW & F2 is LOW & Temp2 is HIGH & Persistence is LOW THEN after LONG delay Temp3 will be MED

Time	T_a	μ_{T3}	Delay	μ_{Delay}	$\mu_{T3}^{delayed}$	T_c
1	0.0590	0.8390	N.R.	N.R.	N.R.	N.R.
2	0.3925	0.8388	N.R.	N.R.	N.R.	N.R.
3	0.6493	0.8386	26	0.3	0.8398	0.30
4	0.8193	0.8385	26	0.3	0.8399	0.30
5	0.7895	0.8384	26	0.3	0.8399	0.30
6	0.7368	0.8383	26	0.3	0.8399	0.30
7	0.6842	0.8382	26	0.3	0.8398	0.30
8	0.6316	0.8381	26	0.3	0.8398	0.30
9	0.5789	0.8381	26	0.3	0.8397	0.30
10	0.5263	0.8380	26	0.3	0.8396	0.30
11	0.4737	0.8380	N.R.	N.R.	N.R.	N.R.
12	0.4211	0.8380	N.R.	N.R.	N.R.	N.R.
13	0.3684	0.8380	N.R.	N.R.	N.R.	N.R.
14	0.3158	0.8380	N.R.	N.R.	N.R.	N.R.
15	0.2632	0.8380	N.R.	N.R.	N.R.	N.R.
16	0.2105	0.8381	N.R.	N.R.	N.R.	N.R.
17	0.1579	0.8382	N.R.	N.R.	N.R.	N.R.
18	0.1053	0.8383	N.R.	N.R.	N.R.	N.R.
19	0.0526	0.8384	N.R.	N.R.	N.R.	N.R.
20	0.0000	0.8385	N.R.	N.R.	N.R.	N.R.
21	0.0000	0.8387	N.R.	N.R.	N.R.	N.R.
22	0.0000	0.8388	N.R.	N.R.	N.R.	N.R.
23	0.0000	0.8390	N.R.	N.R.	N.R.	N.R.
24	0.0000	0.8392	N.R.	N.R.	N.R.	N.R.
25	0.0000	0.8394	N.R.	N.R.	N.R.	N.R.
26	0.0000	0.8395	N.R.	N.R.	N.R.	N.R.
27	0.0000	0.8396	N.R.	N.R.	N.R.	N.R.
28	0.0000	0.8397	N.R.	N.R.	N.R.	N.R.
29	0.0000	0.8398	N.R.	N.R.	N.R.	N.R.
30	0.0000	0.8399	N.R.	N.R.	N.R.	N.R.
31	0.0000	0.8399	N.R.	N.R.	N.R.	N.R.
32	0.0000	0.8399	N.R.	N.R.	N.R.	N.R.
33	0.0000	0.8398	N.R.	N.R.	N.R.	N.R.
34	0.0000	0.8398	N.R.	N.R.	N.R.	N.R.
35	0.0000	0.8397	N.R.	N.R.	N.R.	N.R.
36	0.0000	0.8396	N.R.	N.R.	N.R.	N.R.
37	0.0000	0.8395	N.R.	N.R.	N.R.	N.R.
38	0.0000	0.8394	N.R.	N.R.	N.R.	N.R.
39	0.0000	0.8393	N.R.	N.R.	N.R.	N.R.
40	0.0000	0.8392	N.R.	N.R.	N.R.	N.R.
41	0.0000	0.8391	N.R.	N.R.	N.R.	N.R.
42	0.0000	0.8390	N.R.	N.R.	N.R.	N.R.
43	0.0000	0.8389	N.R.	N.R.	N.R.	N.R.
44	0.0000	0.8388	N.R.	N.R.	N.R.	N.R.
45	0.0000	0.8387	N.R.	N.R.	N.R.	N.R.
46	0.0000	0.8386	N.R.	N.R.	N.R.	N.R.
47	0.0000	0.8386	N.R.	N.R.	N.R.	N.R.
48	0.0000	0.8385	N.R.	N.R.	N.R.	N.R.
49	0.0000	0.8384	N.R.	N.R.	N.R.	N.R.
50	0.0000	0.8385	N.R.	N.R.	N.R.	N.R.
51	0.0000	0.8416	N.R.	N.R.	N.R.	N.R.

3.5 Minimum number of rules required to describe the system completely

Consider the antecedent A1 as:

T1 is Low **AND** F1 is Med **AND** F2 is Low **AND** T2 is Low **AND** Persistence is Med

The nine possible rules (assuming that three linguistic values are assigned to each variable) with this antecedent are:

IF A1 **THEN** after short delay T3 is Low

IF A1 **THEN** after short delay T3 is Med

IF A1 **THEN** after short delay T3 is High

IF A1 **THEN** after Med delay T3 is Low

IF A1 **THEN** after Med delay T3 is Med

IF A1 **THEN** after Med delay T3 is High

IF A1 **THEN** after Large delay T3 is Low

IF A1 **THEN** after Large delay T3 is Med

IF A1 **THEN** after Large delay T3 is High

If the system under consideration is a defined system i.e. a system for which *all the possible mechanisms, relevant antecedent and consequent variables* are known, then at least one, out of these nine rules, will be a valid rule for the system because a cause in a process must lead to an effect. The total number of possible rules for the entire system is 3^7=2187. Thus, the minimum number of valid rules required to completely describe this process is 2187/9=243. However, if the data would not capture every possible

phenomenon occurring in the process, then the number of extracted valid rules will be less than 243.

By convention, only one out of these nine rules is allowed to become a part of the final rule base. This makes sure that the size of the rule base does not become too large; and thus, it saves the computational effort required in the management of these rules. Hence, in this work only one out these possible nine rules was allowed to form the part of the final rule base. Thus, the total number of rules which are required to describe the process completely is 243 in the case of the hot-cold water simulator.

3.6 Comparison of rules

In Kumar's work [3], the rules were not compared to each other and the rule which passed the criteria of the acceptance of the rules was termed as a valid rule. However, Kumar acknowledged that there was need to rank the rules which required comparing them to each other to select the better rules. Furthermore, Kumar proposed that the rules of distinctly different antecedents and consequents should not be compared. In fact, it is more logical to compare the rules with the same antecedents to each other as they conflict with each other. These rules with the same antecedent but different consequents, in a given rule set, will be called as interacting rules. Consider the following two rules:

If F1 is High **AND** F2 is High **AND** Persistence is High then after large delay F3 is High

If F1 is High **AND** F2 is High **AND** Persistence is High then after short delay F3 is low

These two rules have the same antecedent but different consequents. Both these rules are describing the same phenomenon (same antecedent) but are conflicting each other. Hence, the statements of these two rules are contradictory to each other. Thus, such rules should be compared to each other and as discussed in the Section 3.5 only one out of all the interacting rules should be allowed to form a part of the final rule base. It should be noted that the nine rules discussed in the Section 3.5 were also interacting rules as all of them had the same antecedents but different consequents. A rule will not compete with any other rule beside its interacting rules since all the other possible rules will have different antecedents. A group consisting of all possible interacting rules which do not interact with any rule outside their group would be called as a non-interacting group. In the hot-cold water simulator, we have 243 non-interacting groups and each such group will have 9 interacting rules in it.

On the other hand, the rules with different antecedents describe different conditions (events) in the process and do not conflict with each other. Hence, such rules should not be compared with each other. For example: consider the two rules:

If F1 is High **AND** F2 is High **AND** Persistence is High then after large delay F3 is High

If F1 is Low **AND** F2 is Low **AND** Persistence is Low then after short delay F3 is Med

These two rules should not be compared to each other as they are describing completely non-interacting events, they are representing completely different phenomenon. These rules will be called as non-interacting rules.

Thus, in this work only interacting rules are compared to each other, and only one out of all the possible interacting rules is selected to be a part of the final rule base. The metric 'corroboration' (number of trips in QI) proposed by Kumar will be used in this work to compare the interacting rules and this has been discussed in further detail in the Section 3.8. The basis of using *corroboration* for comparing the interacting rules is that a trip in QI in one interacting rule becomes a trip in QIV in the rest of the interacting rules. Similarly, since the non-interacting rules are not to be compared to each other, a trip in QI in one rule should not become a trip in QI or QIV in the rules which are not in the non-interacting group of this rule. These constraints will provide a constraint in choosing the value for the Quadrant size in TSD as explained in the Section 3.7.

3.7 Choice of the Quadrant size in the TSD

Sharma and Kumar used a fixed Quadrant size during the search for the valid rules. Specifically, the TSD was split into four Quadrants of size 0.5 x 0.5 each. Kumar acknowledged the need for finding an optimum value for this Quadrant size. It was also recommended that the value for the size of the TSD should be such that it will assist in the rule selection process. From the discussion in the previous section, two constraints affect the selection for the Quadrant size in the TSD:

1. A trip in QI for a rule should give a trip in QIV in its remaining interacting rules.
2. A trip in QI for a rule should not give a trip either in QI or in QIV in its non-interacting rules.

In addition, the minimum operator is a very strict operator i.e. it requires a high value of truth to extract "valid rules" rules. Thus, it is proposed that the Quadrant size of the TSD should be as lenient as possible. Thus, we need the minimum value of T_a and T_c, for the Quadrant size, which will satisfy the above two constraints. It was found that a value of $T_a = 0.5$ and $T_c = 0.5$ for the Quadrant size would fulfill these two constraints, as discussed below.

Consider again the 9 interacting rules discussed in the Section 3.5. For a given data point the sum of the used triangular membership values for the three categories low, medium and high is unity. Applying this knowledge to all the variables (F1, F2, T1, T2, Delay, Persistence, and T3) in the process, we get the following equations:

$$\mu_{T1}^{i,1} + \mu_{T1}^{i,2} + \mu_{T1}^{i,3} = 1 \tag{31}$$

$$\mu_{T2}^{i,1} + \mu_{T2}^{i,2} + \mu_{T2}^{i,3} = 1 \tag{32}$$

$$\mu_{F1}^{i,1} + \mu_{F1}^{i,2} + \mu_{F1}^{i,3} = 1 \tag{33}$$

$$\mu_{F2}^{i,1} + \mu_{F2}^{i,2} + \mu_{F2}^{i,3} = 1 \tag{34}$$

$$\mu_{T3}^{i,1} + \mu_{T3}^{i,2} + \mu_{T3}^{i,3} = 1 \tag{35}$$

$$\mu_{Pers}^{i,1} + \mu_{Pers}^{i,2} + \mu_{Pers}^{i,3} = 1 \tag{36}$$

$$\mu_{Delay}^{i,1} + \mu_{Delay}^{i,2} + \mu_{Delay}^{i,3} = 1 \tag{37}$$

Consider the first constraint which states that a trip in QI for a rule must give a trip in QIV in the remaining interacting rules. The T_a value for all these interacting rules will be equal as they have the same antecedents. Thus, to obtain a trip in QI or QIV in

these interacting rules the T_a must be greater than 0.5. When the value of $T_c > 0.5$ for a given rule (out of these nine) then it implies (from Equation (14)) that both $\mu_{T3}^{i,j6}$ and $\mu_{Delay}^{i,j7}$ are greater than 0.5 for that rule since the minimum operator is used for performing the truth calculations. It further implies, from Equations (36) and (37), that any other interacting rule with a different consequent in the remaining 8 rules must have a value less than 0.5. Thus, all the other interacting rules will have a $T_a > 0.5$ and a $T_c < 0.5$. Hence, when $T_c > 0.5$ and the number of number of points are equal to threshold for a given rule then it would give a trip in QIV in all the other interacting rules.

Consider the second constraint which states that trip QI for a rule should not give a trip either in QI or in QIV in a non-interacting rule. Again, to obtain a trip in QI or QIV in a rule the T_a must be greater than 0.5. It implies, from Equation (13), that $\mu_{T1}^{i,j1}, \mu_{F1}^{i,j2}, \mu_{F2}^{i,j3}, \mu_{F2}^{i,j3}$ $\mu_{T2}^{i,j4}$ and $\mu_{Persistence}^{i,j5}$ must all be greater than 0.5 for that rule. It means, from Equations (31), (32), (33), (34), and (35), that for any other non-interacting rule, the value of T_a would be less than 0.5 (since any other possible antecedent will have T_a less than 0.5). Hence, when $T_a > 0.5$ and the number of number of points is equal to the threshold, to count as a trip for a given rule, then it would not give a trip in QI or QIV in any non-interacting rule.

The only possible way to satisfy these two constraints is to set the value for the Quadrant size as $T_a = 0.5$ and $T_c = 0.5$. It should be noted that if the sum of membership values do not equal to unity for a variable then the membership values must be normalized before using this value for the Quadrant size.

3.8 Choice of Metric and its universal value

Two metrics, corroboration and merit, were invented and recommended in the previous research by Kumar [4] to check the validity of a rule. Corroboration was defined as the minimum number of trips in QI required to provide sufficient evidence of the 'corroboration'. Merit for a rule was defined as the difference between the number of trips in QI and the number of trips in QIV. The acceptance criteria for a rule to be classified as *valid* were:

1. A value of *two* was chosen as a reasonable number for the corroboration.
2. Merit of a rule was required to be greater than or equal to 1.

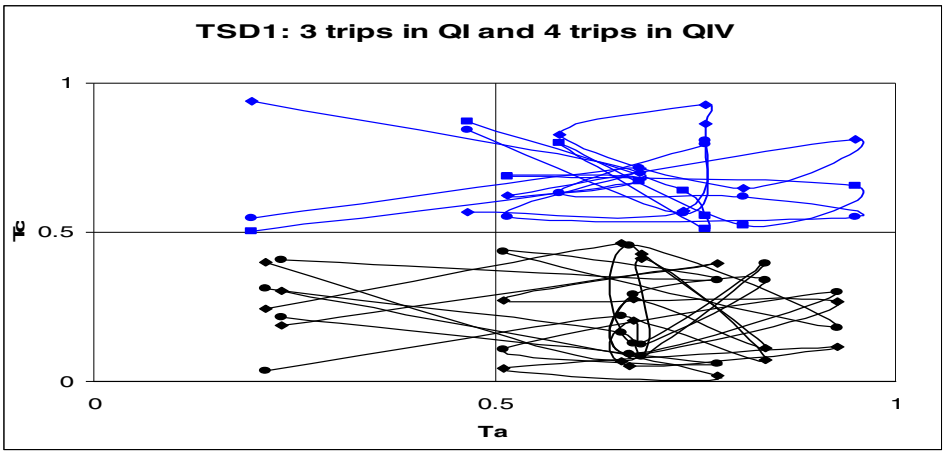
However, these threshold values for these metrics were intuitive. Thus, to provide universal applicability to that technique, an objective of this study is to find the threshold values to be used for these metrics.

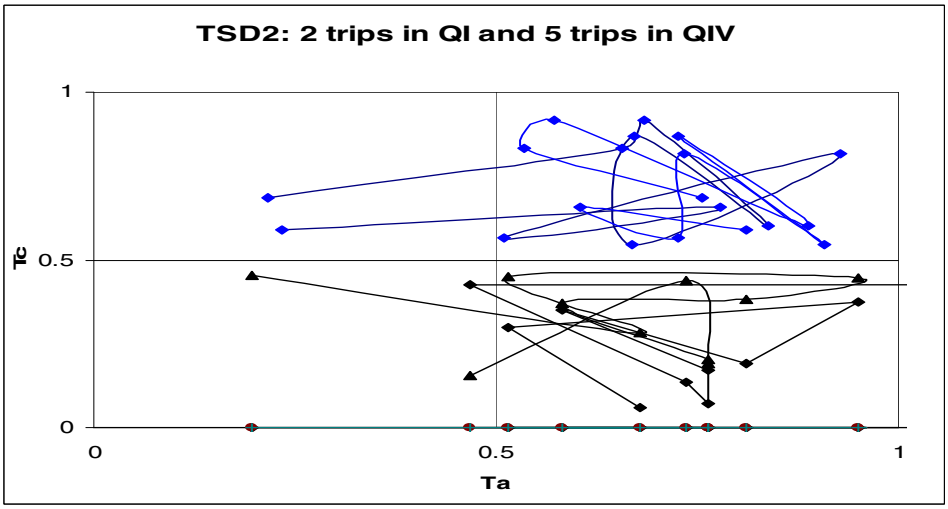
Recall, a rule competes only with its interacting rules. A trip in QI in a rule means a trip in QIV in its interacting rules. Thus, a single trip in QI in one rule would establish its validity over the other interacting rules if the other interacting rules have not registered any trip in QI. Hence, the value which should be used for the minimum number of trips in QI to confirm the validity of a rule is unity. However, it is quite possible (due to shifting of trips from one Quadrant to another because of membership value of variables or due to error is delay quantification) that a trip in QI would be made in more than one interacting rule in a given non-interacting group. In that situation, only the rule with the maximum number of trips in QI would be selected from the non-interacting group. In addition, if the number of trips in QI is equal (and maximum in the

non-interacting group) for more than one interacting rule then any one out of those rules would be selected randomly to maintain the size of the rule base.

In the previous work, merit was chosen as a metric to compare rules based on both the number of trips in QI and the number of trips in QIV. However, the current research has revealed that the number of trips in QI and the number of trips in QIV in the rules in a non-interacting group are not independent of each other. This is because a trip in QI in one rule implies a trip in QIV in other interacting rules in the group. This was not the understanding about the metrics in the earlier stages of this work because the geometric operator didn't support this relationship between the trips in QI and QIV of the rules. With minimum operator, the rule with the maximum value for the corroboration in its non-interacting group will also have the maximum value for the merit in the same group. Hence, it would be redundant to use both merit and number of good rules as metric to compare the validity of a rule. Thus, the merit is not required for the selection of rules.

It should be noted that now it is possible for a rule to get selected as a valid rule even after having a negative value for merit. This was prohibited in the previous research. An example of a scenario in which a rule would be selected even after having a negative value for the merit is shown in Figure 21. The TSD for three interacting rules is shown. The TSD1 registered 3 trips in the QI Quadrant. This resulted in three trips in the QIV for the TSD2 and TSD3 (as they are interacting rules). Similarly, the TSD2 registered 2 trips in QI which resulted in 2 trips in QIV for the TSD1 and TSD3. Finally, the TSD3 also registered 2 trips in the QI Quadrant which resulted in 2 trips in QIV for





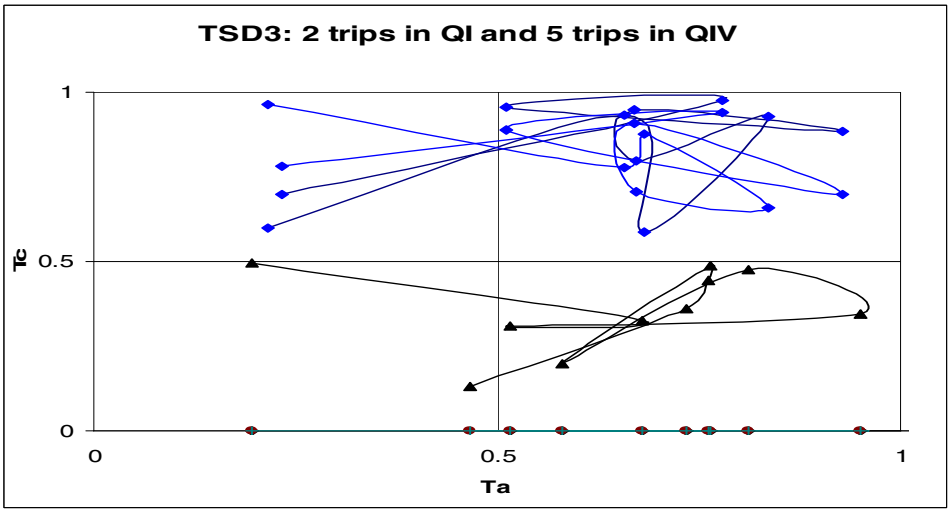


Figure 21: TSD diagrams of three interacting rules. Rule with TSD1 has negative value for merit but still gets selected because it has the maximum number of trips in QI.

the TSD1 and TSD2. The calculations for the merit for these three rules are shown in the Table 3.

Table 3: The merit values the three interacting rules indicate that a rule can be selected as a valid rule even if the value for merit is negative. In this case, TSD1 has a -1 value for the merit but even then it will be selected as a valid rule as it performs better than the other interacting rules.

	QI Trips	QIV Trips	Merit
TSD1	3	4	-1
TSD2	2	5	-3
TSD3	2	5	-3

The rule represented by TSD1 will be selected as the single valid rule from this rule interacting set (because it has the maximum value for corroboration) even though its merit value is negative. This is because the interacting rules are being compared to each other to find the best possible rule. In the work by Kumar, none of these rules would have been selected as valid rules as all of them have negative value for merit.

3.9 Universal applicability of the technique

As discussed above, the best combination of the parameters involved in the selection of a valid rule would be:

Mathematical Operator to calculate truth: Minimum operator

Quadrant size in the TSD: $T_a = 0.5$ and $T_c = 0.5$

Corroboration (Minimum number of trips in QI): 1

Merit: Not required

These values are universal as they are based on the combination of the fundamentals of fuzzy logic and TSD. No assumptions have been made during the selection of these values. Thus, these values are universal and do not depend on the process or the number of variables in the process. Thus, if this technique needs to be applied to a new process then the parameters listed above should be used. The accuracy of prediction, obtained from the rules, can then be improved by optimizing the membership functions of the variables.

3.10 Belief in a rule

In general, belief is the mental acceptance and conviction in the truth or validity of something. In this work, the belief in a rule R will be defined as the ratio of the number of trips in QI to that of sum of the trips in QI and QIV in the given rule R. It should be noted that the number of trips in QIV for a given rule R will be equal to the sum of trips in QI in the remaining interacting rules of the given rule R. Hence, the belief in a rule gives the relative validness of the given rule in comparison to its interacting rules. For a given rule r, if N_{QI} and N_{QIV} represent the number of trips in QI and QIV for the rule then the belief in a rule will be defined as:

$$Belief = \frac{N_{QI}}{N_{QIV} + N_{QI}} \tag{38}$$

Thus, belief in a rule indicates our level of confidence in the validity of that rule based on its past and in comparison to the other rules. This value can provide useful information to the user while making decisions based on that rule. A higher value of belief would increase the confidence of the user in the given rule and vice versa.

3.11 Curse of Dimensionality

It would be extremely complicated to perform an exhaustive search over all the possible rules in processes consisting of a large number of relevant variables. Thus, a genetic algorithm based technique is being developed by Dr. Gary Yen in which valid rules would evolve from the data with time. The TSD would be used inside this technique for the selection of the valid rules.

A major limitation of the data-driven techniques is the "curse of dimensionality". It refers to the problem caused by the rapid increase in the computational power with the addition of extra dimensions (variables) to the search space. In genetic algorithms, an initial population of rules is generated and then it is continuously optimized to generate a new improved populations consisting of valid rules. However, it would have to find the optimal solution (valid rules) in a very large search space. For example: For a process with 15 inputs, 3 outputs and 3 linguistic categories for each variable, the search space would consist of 3^{18} rules. Thus, the search space increases exponentially as the process variables involved increase. The fundamentals of the TSD, developed in this work would help the genetic algorithm in overcoming the "curse of dimensionality". Flowcharts for the conventional GA and the GA using TSD are shown in Figures 22 and 23 respectively.

It should be observed that the initial population was generated in the entire space in both algorithms. However, the computational power required for the GA combined with TSD would be lesser. This is because when an antecedent with $T_a > 0.5$ will be found, then a valid rule could be found just by searching in the consequent space (as a

cause has to produce an affect). Once the antecedent with Ta > 0.5 is found the exhaustive search can be performed over the consequents (3^3) to find the valid rules. Thus, the total search space would reduce from 3^{18} rules to 3^{15} antecedents + 3^3 consequents. In contrast, the search space in the conventional GA would have remained the same.

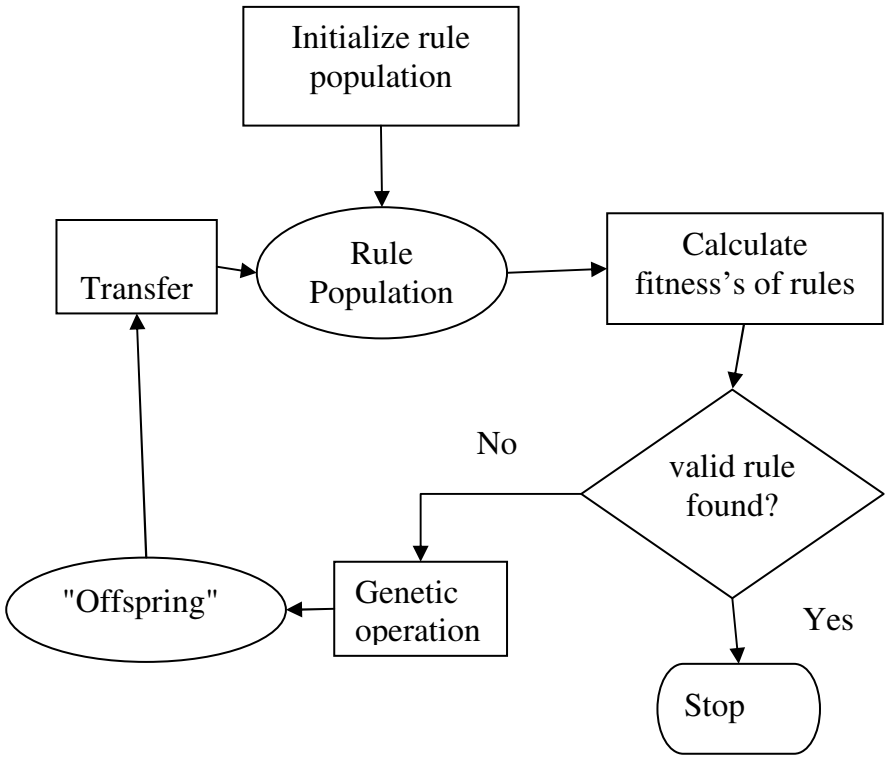


Figure 22: Flowchart of conventional genetic algorithm approach to extract rules. Reproduced from [15].

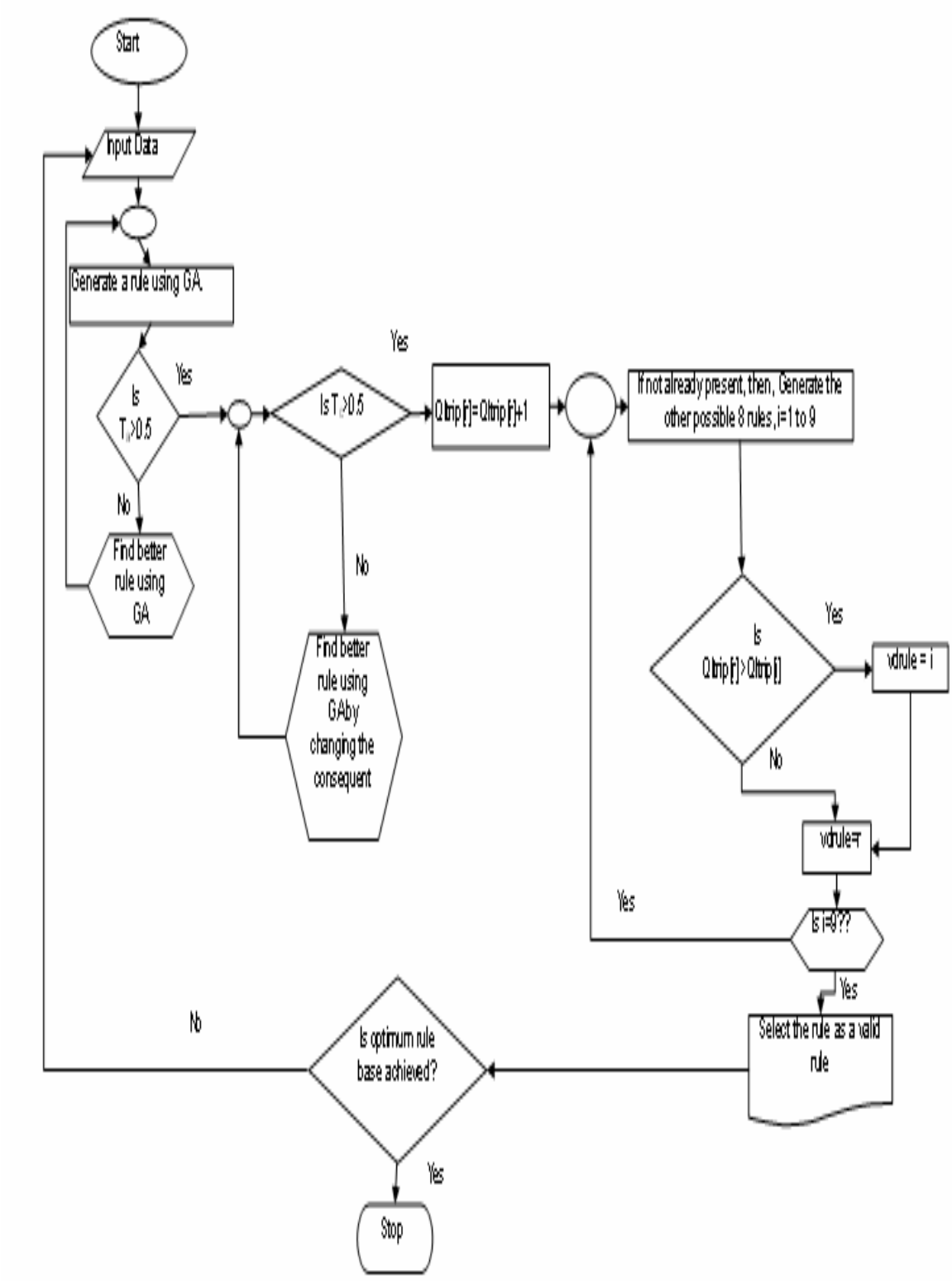


Figure 23: Flowchart for extracting rules using GA and TSD

3.12 Predicting the consequent

In this Section, two methods for predicting the value of the consequent will be discussed. The first method is a refinement of Kumar's [3] prediction technique and the second method is based on the conventional approach used to predict the output in the fuzzy logic. These methods have been discussed below:

3.12.1 Refinement of Kumar's technique

As discussed in Section 2.3.4, Kumar developed a novel technique for predicting T_c of the output, based on the historical information obtained by dividing the TSD into five T_a and 10 T_c zones. However, this predicted output was not too useful since T_c consists of the combined information of the delay and the T3. The desired result is to obtain separate values for these two outputs. Hence, it was important to decouple the consequents into two separate terms. The following section discusses a modification of Kumar's technique applied separately to the variables, Delay and T3. Again, the first step will use the historical data which will be processed and used in the next step for making predictions in the new data.

3.12.1.1 From Historical Data

The information from the historical data will be separated into the information about T3 and the Delay as discussed below:

3.12.1.1.1 Processing historical data for T3

Instead of dividing the TSD of the rule, the plot of μ_{T3} and T_a was divided into grids as shown in the Figure 24.

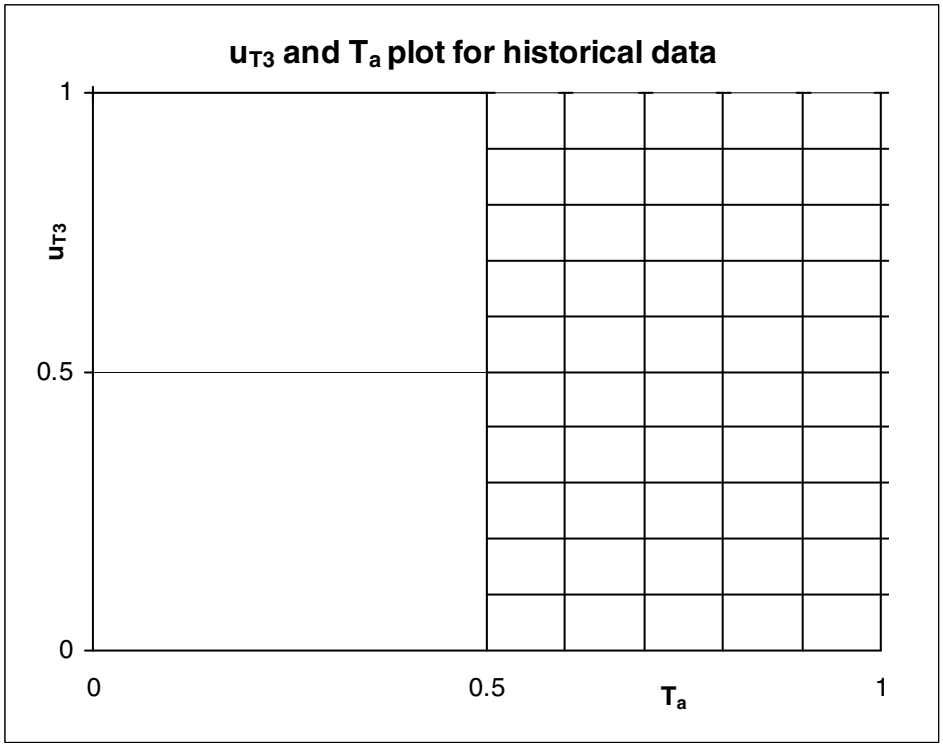


Figure 24: The plot of μ_{T3} and T_a showing the partitioning of the QI and QIV into a total of 50 zones of size 0.1 x 0.1 each. The T_a is divided into 5 T_a zones and the μ_{T3} is divided into 10 zones each.

Thus, the plot is divided into 5 T_a zones and 10 μ_{T3} Zones of size 0.1 x 0.1 each. Now, the data is processes in the following steps:

1. A column vector, *Hits,* is used to record the number of points or hits made in each of the five zones of the antecedent.
2. Then, a 10 x 5 matrix, *Numpoints,* was used to store the number of hits in each of the ten μ_{T3} zones for each of the five T_a zones.

3. Then, each column of the matrix *Numpoints* was separated into five column vectors represented by $\vec{V}_i$ and then normalized in the range 0-1 as shown below:

$$\vec{V}_{i_{norm}} = \left[\frac{\vec{V}_i}{\vec{1}^{\,T} \cdot \vec{V}_i} \right] \tag{39}$$

Here, i = 1 to 5 and $\vec{V}_{i_{norm}}$ is a 10 x 1 matrix. $\vec{1}$ is defined as $\vec{1} = [1,1,1,1,1,1,1,1,1,1]$.

3.12.1.1.2 Processing historical data for Delay

Similarly, the plot of μ_{delay} and T_a is shown in the Figure 25.

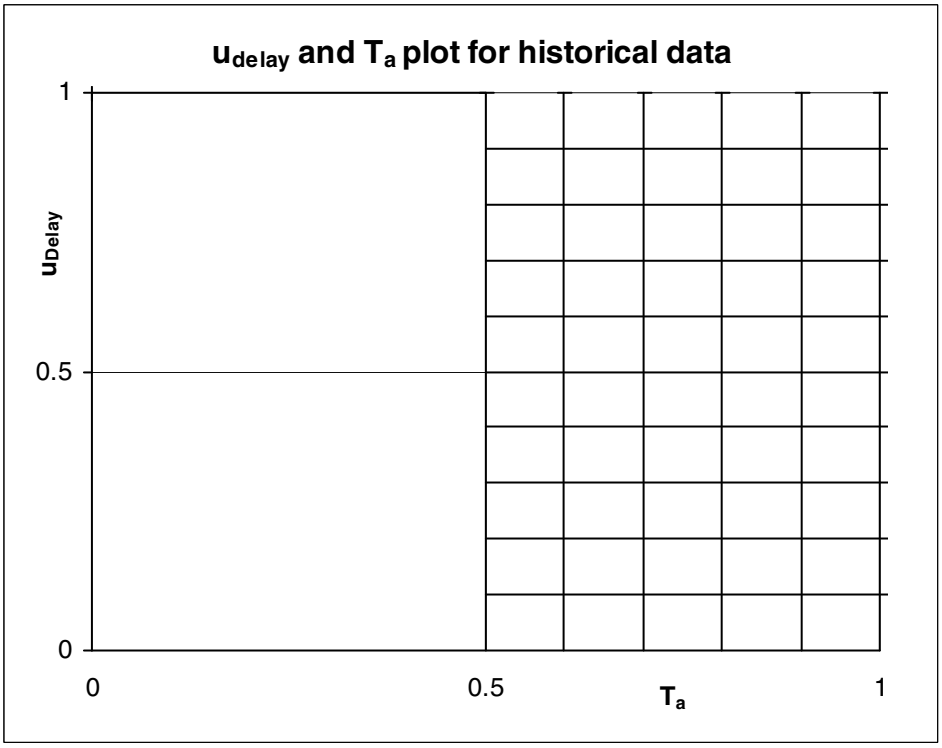


Figure 25: The plot of μ_{delay} and T_a showing the partitioning of the QI and QIV into a total of 50 zones of size 0.1 x 0.1 each. The T_a is divided into 5 T_a zones and the μ_{Delay} is divided into 10 zones each.

Thus, the plot is divided into 5 T_a zones and 10 μ_{Delay} Zones of size 0.1 x 0.1 each. Now, the data is processes in the following steps:

1. A column vector, *Hits,* is used to record the number of points or hits made in each of the five zones of the antecedent. This would be the same as the one defined during the processing of historical information for T3.
2. Then, a 10 x 5 matrix, *DelayNumpoints,* was used to store the number of hits in each of the ten μ_{Delay} zones for each of the five T_a zones.
3. Then, each column of the matrix *DelayNumpoints* was separated into five column vectors represented by $\vec{DV}_i$ and then normalized in the range 0-1 as shown below:

$$\vec{DV}_{i_{norm}} = \left[\frac{\vec{DV}_i}{\vec{1}^{\,T} \cdot \vec{DV}_i} \right] \tag{40}$$

Here, i = 1 to 5 and $\vec{DV}_{i_{norm}}$ is a 10 x 1 matrix.

3.12.1.2 From New Data

In Kumar's technique, the entire new data was used to register hits first and then those hits were used to predict the consequent values by using it in conjunction with the historical data. However, this will not necessarily give one output value for every input value. This is because several inputs in the new data will be combined together to form hits for a rule and then the effects of all those hits was combined together to give a single (per rule) predicted value of the output. However, in real processes, it would make more sense to have a one-to-one correspondence between the inputs and the outputs. Thus, in this work each input value was required to give one value for each output (delay and T3).

3.12.1.2.1 Calculating the membership value for T3 for the new data

1. For a given rule r (which is a valid rule), the T_a value for the first data point is calculated. If $T_a > 0.5$ then this will give a hit in one of the five antecedent zones. Let that zone be represented by j. Here, j can vary from 1 to 5.
2. The membership value of T3, for the given rule r, was calculated as:

$$\mu_{T3} = \frac{(0.05x\vec{V}_{i_{norm}}(1,j) + 0.15x\vec{V}_{i_{norm}}(2,j) + ... + 0.95x\vec{V}_{i_{norm}}(10,j))}{(TotSum)} \quad (41)$$

Where,

$$Totsum = \sum_{k=1}^{10} \vec{V}_{i_{norm}}(k,j) \quad (42)$$

3. If $T_a < 0.5$ in Step 1 then repeat the process with the next valid rule. Continue this process till $T_a > 0.5$ for a valid rule is found.
4. Continue this process for all the data points in the new data.

3.12.1.2.2 Calculating the membership value of delay for the new data

1. For a given rule r, the T_a value for the first data point is calculated. This will give a hit in one of the five antecedent zones. Let that zone be represented by j. Here, j can vary from 1 to 5.
2. The value of μ_{delay}, for the given rule r, was calculated as:

$$\mu_{Delay} = \frac{(0.05xD\vec{V}_{i_{norm}}(1,j) + 0.15xD\vec{V}_{i_{norm}}(2,j) + ... + 0.95xD\vec{V}_{i_{norm}}(10,j))}{(DTotSum)} \quad (43)$$

Where,

$$DTotsum = \sum_{k=1}^{10} \vec{DV}_{inorm}(k, j) \tag{44}$$

3. If $T_a < 0.5$ in Step 1 then repeat the process with the next valid rule. Continue this process till $T_a > 0.5$ for a valid rule is found.
4. Continue this process for all the data points in the new data.

3.12.2 Prediction based on conventional fuzzy logic approach

In the literature, it is a common practice to predict the output, based on the assumption that the truth of the antecedent is equal to the membership values of the consequents. Thus, it is assumed that $\mu_{T3} = \mu_{Delay} = T_a$.

3.12.3 Defuzzification of the membership values

Once the membership functions, μ_{T3} and μ_{delay}, are calculated (using any of the techniques discussed above) then defuzzification is performed to obtain the crisp value of the outputs. In this work, the defuzzification was performed to calculate the values of T3 and Delay as given by Equations (45) and (46). Here, assume that R is the total number of valid rules selected using the historical data.

$$T3 = \frac{\sum_{r=1}^{R} \mu_{T3,r} C_{T3,r}}{\sum_{r=1}^{R} \mu_{T3,r}} \tag{45}$$

$$Delay = \frac{\sum_{r=1}^{R} \mu_{Delay,r} C_{Delay,r}}{\sum_{r=1}^{R} \mu_{Delay,r}} \tag{46}$$

$C_{T3,r}$ represents the crisp T3 output for a given rule r and is equal to the center of gravity of the linguistic category of T3 (as given by the rule r). Similarly, $C_{Delay,r}$ represents the crisp Delay output for a given rule r and is equal to the center of gravity of the linguistic category of Delay (as given by the rule r).

3.12.4 Blending the delay values

The final step in making predictions is to blend the output value coming from different delay values. Assume that at time t = 2 seconds the predicted value for T3 is 75^{o} C after a delay of 5 seconds. However, imagine that after 2 seconds the predicted value for T3 is 85^{o} C after a delay of 3 seconds. Now, the issue is that which information should be used in making the prediction. In this work, the predictions would be governed by the most recent information. In fact, it is a common practice in forecasting the weather condition to base the forecast on the latest possible information of the process. Thus, in the given example, the value for the T3 will be 85^{o} C.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Results

The results, for the hot-cold water simulator, were obtained for the selection phase and the prediction phase. In the selection phase, the data generated from the simulator was used to select the valid rules for the process. In the prediction phase, the rules selected during the selection phase were used to predict the value for the outlet temperature, T3.

In the selection phase, the valid rules were selected several times by changing the mathematical operator (used in calculating the T_a and T_c), merit, and corroboration. This was done to study the affect of these factors on the selection of valid rules. In the prediction phase, the selected rules were used to predict the value of the output, T3. The predictions were made based on the conventional technique as well as Kumar's modified prediction technique to compare these two techniques. The root mean square (RMS) error was used to compare both these techniques and was calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N,pred} (T3_i^{Actual} - T3_i^{Calculated})^2}{N,pred}} \tag{47}$$

Here, $N, pred$ refers the total number of data points for which prediction results were obtained. It should be noted that $N, pred$ does not represent the total number of data points, $N, predtotal$, used in the prediction phase. $N, pred$ just represents the number of data points for which a value for the output, T3, could be obtained using the rules. The *prediction capability* of the rules would be calculated as:

$$predcapability(\%) = \frac{N, pred}{N, predtotal} \times 100 \tag{48}$$

The affect of the factors discussed above (mathematical operator, prediction technique, *merit* and *corroboration*) is discussed in the Section 4.2. No restriction was applied on the selection of interacting rules while extracting the results discussed in this Section. Thus, it was possible for more than one interacting rules to be selected as valid rules. Section 4.3 discusses the affect of allowing only one interacting rule in the rule base, on the accuracy of the prediction.

4.2 No restriction on the selection of interacting rules

4.2.1 Affect of corroboration on the selection of valid rules

The affect of *corroboration* on the selection of valid rules and on the accuracy of the prediction, was studied for both geometric and minimum operator. The results are shown in the Figures 25 and 26 respectively. The metric *merit* was not used in the selection of these rules. The error shown was calculated based on the conventional prediction technique ($T_a = \mu_{T3} = \mu_{delay}$) used in fuzzy logic.

4.2.1.1 Geometric Operator: Affect of corroboration

Consider the affect of corroboration on the selection of valid rules, when the geometric operator was used to perform the calculations of the T_a and T_c. It is observed from Figure 26, that the number of valid rules selected, decreased sharply with the increase in value of corroboration. This trend was expected since as corroboration increased, more trips in the Quadrant I of the rules were required, for the rule to be selected as a valid rule. The total numbers of selected rules dropped from 1115 to 157 as the value of corroboration increased from 1 to 10.

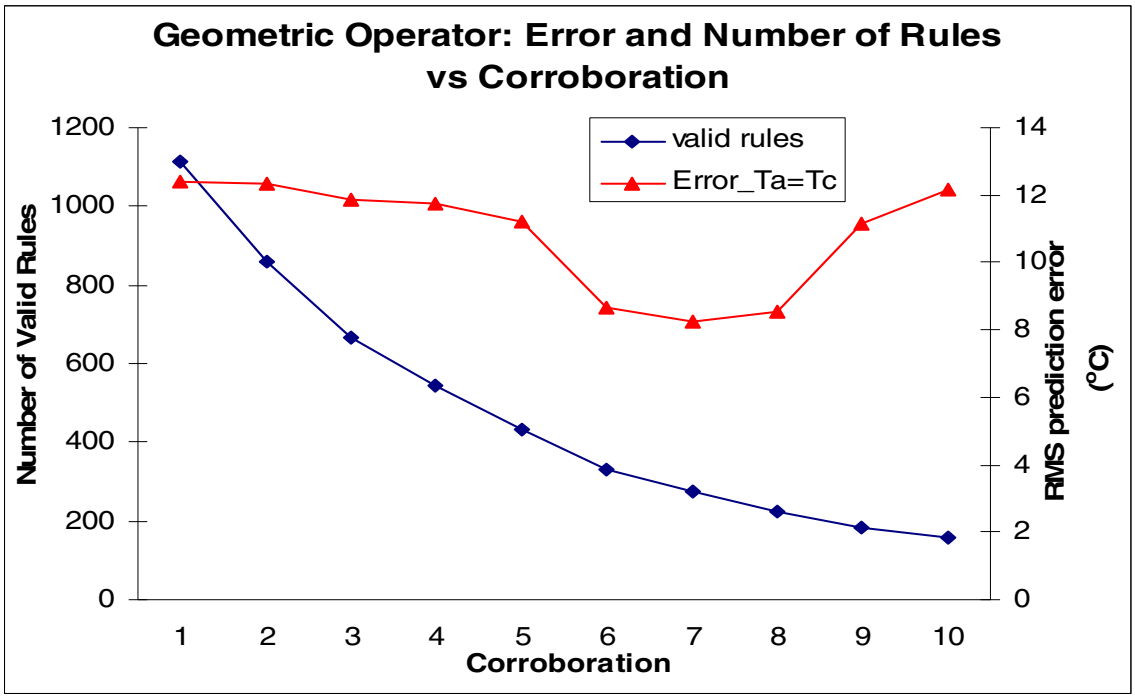


Figure 26: For the geometric operator, the affect of corroboration on the selection of valid rules and on the RMS prediction error has been shown. The number of valid rules falls sharply as the corroboration increases but the RMS prediction error first decreases and then increases as the corroboration increases.

The affect of corroboration on the prediction error is complicated than that on number of valid rules. The RMS prediction error first dropped with the increasing value of corroboration, but, after a given value of corroboration, the error started to increase

again. This behavior can be explained on the basis of the number of valid rules selected, and the characteristics of the geometric operator. As discussed in the Section 3.3, the geometric operator allows several semi-valid rules to register trips in QI. As a result, a large number of semi-valid rules form a part of the final rule base, and are used in making predictions. A low value for the corroboration makes the rule selection process even more lenient. For instance: 1115 out of the total 2187 rules were selected when a corroboration value of 1 was chosen. Due to the selection of a large number of rules, with limited validity, the ability to accurately predict the output is low and, thus, large errors are associated with small values of corroboration for geometric operator. As corroboration increased, the rule selection process became stricter. The error reduced gradually as the value of corroboration went from 1 to 5 indicating that several semi-valid rules were still present in the rule base. However, as the value of corroboration went from 5 to 6 the number of valid rules selected decreased from 433 to 333. During this transition, the RMS prediction error also dropped sharply from 11.22^{o}C to 8.66^{o}C which suggested that only few semi-valid rules were left in the rule base. The corroboration value of 7 gave the lowest prediction error of 8.22^{o}C. However, as the corroboration value was further increased, the RMS prediction error started to increase sharply. This trend can be explained on the basis of the removal of rules of high validity from the rule base. Thus, for high values of corroboration (9 and 10) several rules with high validity were removed along with the semi-valid rules due to the strictness of the rule selection process. This left very few valid rules in the rule base to predict the output. Hence, the RMS prediction error increased.

4.2.1.2 Minimum Operator: Affect of corroboration

The affect of corroboration on the selection of valid rules, when the minimum operator was used to perform the calculations of the T_a and T_c, is shown in Figure 27. The number of valid rules selected, decreased sharply with the increase in value of the corroboration. This trend is similar to the one observed for the geometric operator in Section 4.2.1.1. However, the number of selected rules was much lesser when the minimum operator was used (for the same value of corroboration). This is because the minimum operative is a strict operator then the geometric operator as discussed in Section 3.3. The number of selected valid rules decreased from 248 to 17, as the corroboration increased from 1 to 6.

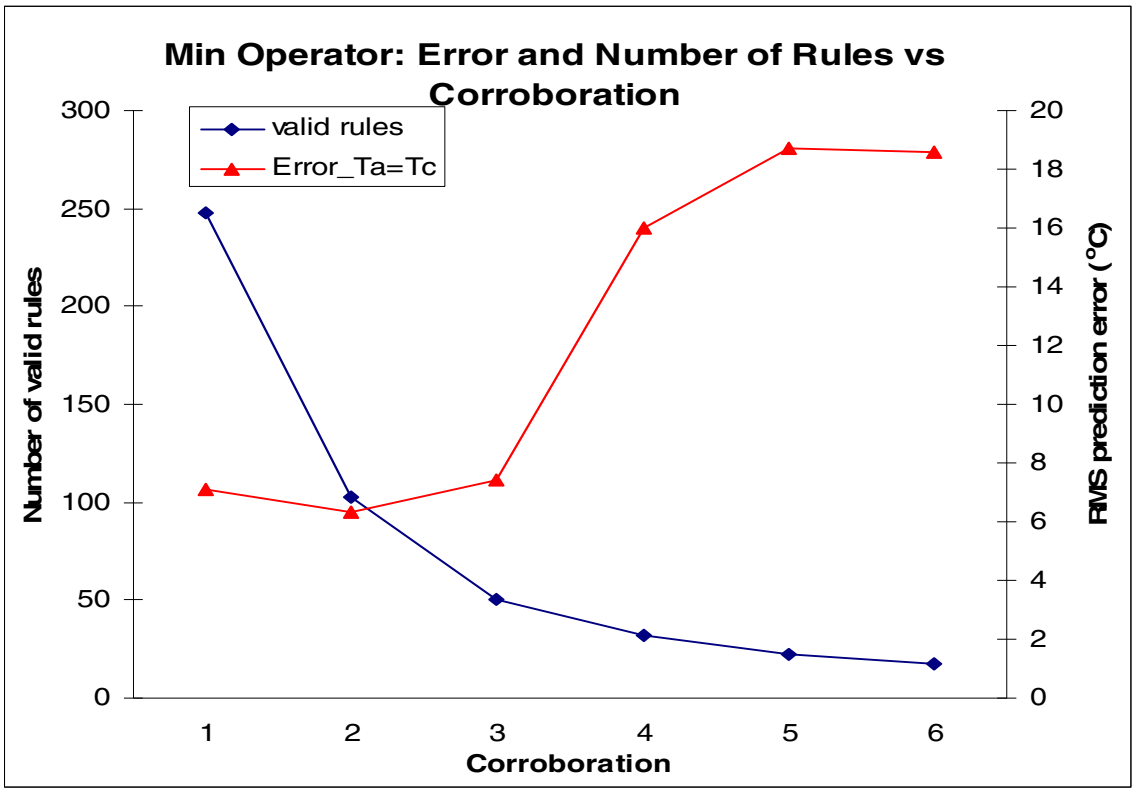


Figure 27: For the minimum operator, the affect of corroboration on the selection of valid rules, and on the RMS prediction error has been shown. The number of valid rules falls sharply as the corroboration increases but the RMS prediction tends to increase as the corroboration increases.

The trend in the plot between RMS error and corroboration, for the minimum operator, is similar (error first decreased and then increased) to the trend obtained for the geometric operator. However, with the minimum operator, the first phase of the trend (decrease in error with corroboration) was almost absent. This is because the plot between RMS error and corroboration reached its optimum (lowest) error at a low value of corroboration, for the minimum operator. Due to the lenient nature of the geometric operator, the optimum error was achieved at a higher value of corroboration. Thus, it seems that it is easier to locate the optimum point for the minimum operator, then for the geometric operator.

4.2.2 Affect of threshold value of merit on the prediction error

The affect of threshold value of *merit* on the RMS prediction error was studied for both geometric and minimum operator. The results are shown in the Figures 28 and 29 respectively. The value of corroboration was fixed at two while the threshold value of merit was changed from 0 to 2. The scenario in which the merit was not used has also been included in the Figures 28 and 29. The error shown was calculated based on the conventional prediction technique ($T_a = \mu_{T3} = \mu_{delay}$) used in fuzzy logic.

4.2.2.1 Geometric Operator: Affect of threshold value of merit

The affect of merit on the RMS prediction error, when the geometric operator was used to perform the calculations of the T_a and T_c is shown in Figure 28. The RMS error decreased slightly when the threshold value of merit, equal to zero, was used in the selection of valid rules. This is because the *corroboration* value of two consisted of a

large number of semi-valid rules (as discussed in Section 4.2.1.1) and the introduction of merit resulted in the removal of several semi-valid rules from the rule base. Hence, the prediction error decreased slightly.

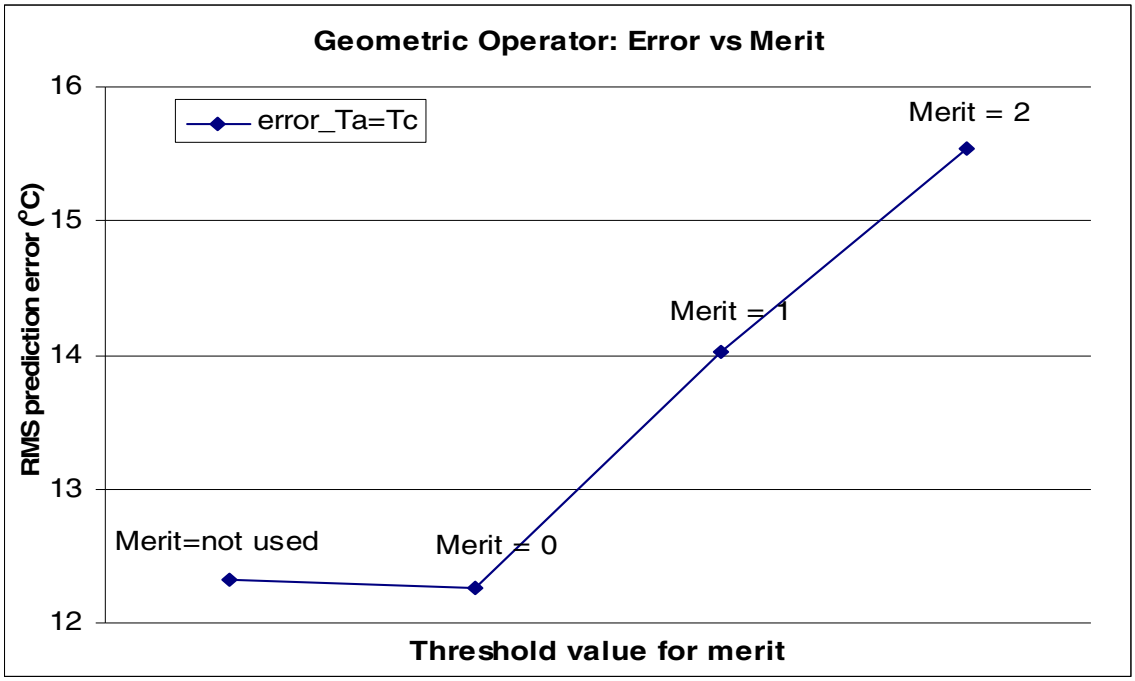


Figure 28: For the geometric operator, the affect of merit on RMS prediction error has been shown. The RMS prediction error tends to increase as the value of merit increases.

However, as the threshold value of merit increased, the RMS prediction error increased. This can be explained on the basis of removal of several valid rules from the rule base. Due to this, the RMS prediction error increased.

4.2.2.2 Minimum Operator: Affect of merit

The affect of merit on the RMS prediction error, when the minimum operator was used to perform the calculations of the T_a and T_c, is shown in Figure 29.

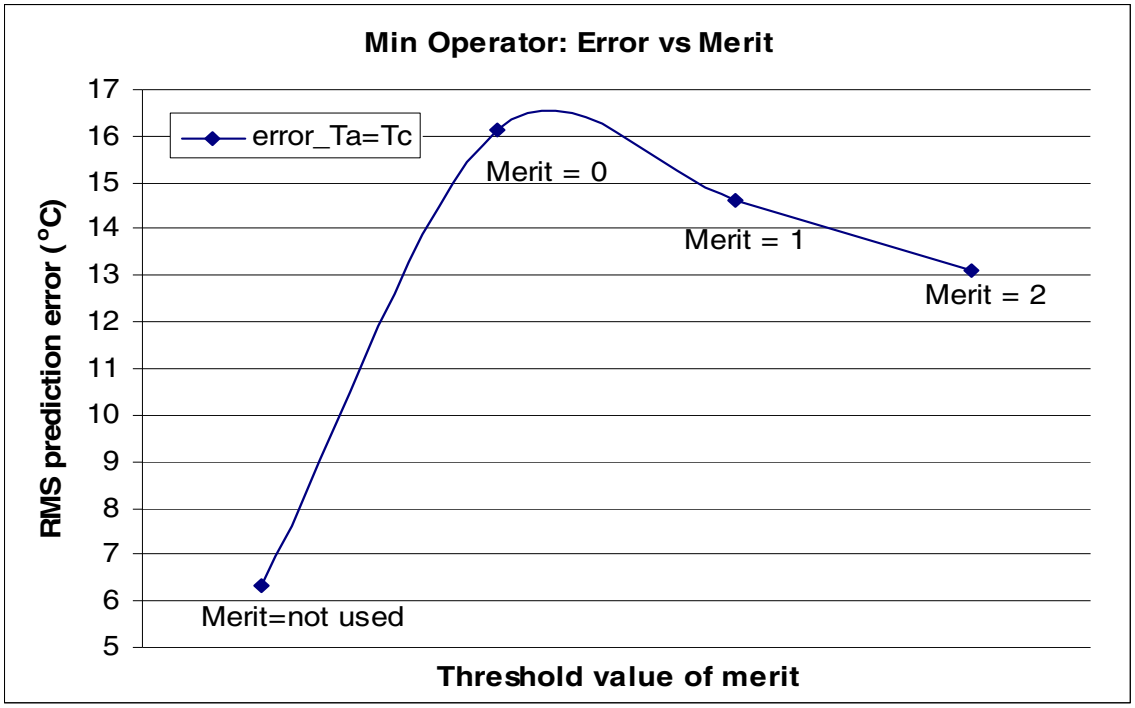


Figure 29: For the minimum operator, the affect of merit on RMS prediction error has been shown. The RMS prediction error tends to increase as the value of merit increases.

The error increased sharply as the merit was introduced, as a constraint, in the selection of valid rules. This is because the *corroboration* value of two provided the optimum set of valid rules (as discussed in Section 4.2.1.2) for this data set. The usage of merit resulted in the removal of several valid rules, from this optimum rule base. Hence, the sharp increase in the error was seen. The number of valid rules selected decreased from 248 (when merit was not used) to 43 (threshold value of merit = 0). As the merit value was increased from 0 to 2, the number of valid rules became very less. The *prediction capabilities* of the rules were 54.8% and 36.9% for the values of merit equal to 1 and 2 respectively. Hence, the decrease in error was of little importance because the prediction capability of the rule base was very low.

4.2.3 Comparison between Kumar's (modified) and conventional prediction technique

The comparison between the Kumar's modified prediction technique and the conventional technique was studied for both the geometric and minimum operator. The rules were generated by choosing different values of corroboration. Then, Kumar's modified technique was used to predict the output, T3. Similarly, prediction for T3 were made using the conventional technique ($T_a = T_c$). The results are shown in the Figures 30 and 31 respectively.

4.2.3.1 Geometric operator: Choice of prediction technique

The trends, in the plot between RMS error and number of rules, obtained using Kumar's prediction technique and conventional technique, using the geometric operator, were quite similar as shown in Figure 30. The RMS error first decreased and, then increased, as the number of rules increased (due to decrease in the value of corroboration). Finally, as the corroboration value decreased further, very little change was noticed in the error.

The initial decrease in the error can be attributed to the relaxation in the strictness criteria of the corroboration. This allowed several valid rules to be included in the rule base. However, as the number of rules increased (due to decrease in corroboration value) several semi-valid rules were included in the rule base. This explains the increase in error as the number of rules increased from 220 to 550. Finally, the error became almost constant with increasing values of corroboration. This could have been because of the presence of large number of rules in the rule base (consisting of both valid and semi-valid rule base). This large number of rules resulted in an averaging of the error. Furthermore,

it seems that, after a large number of rules have been selected, this averaging does not change too much with further increase in the number of selected rules.

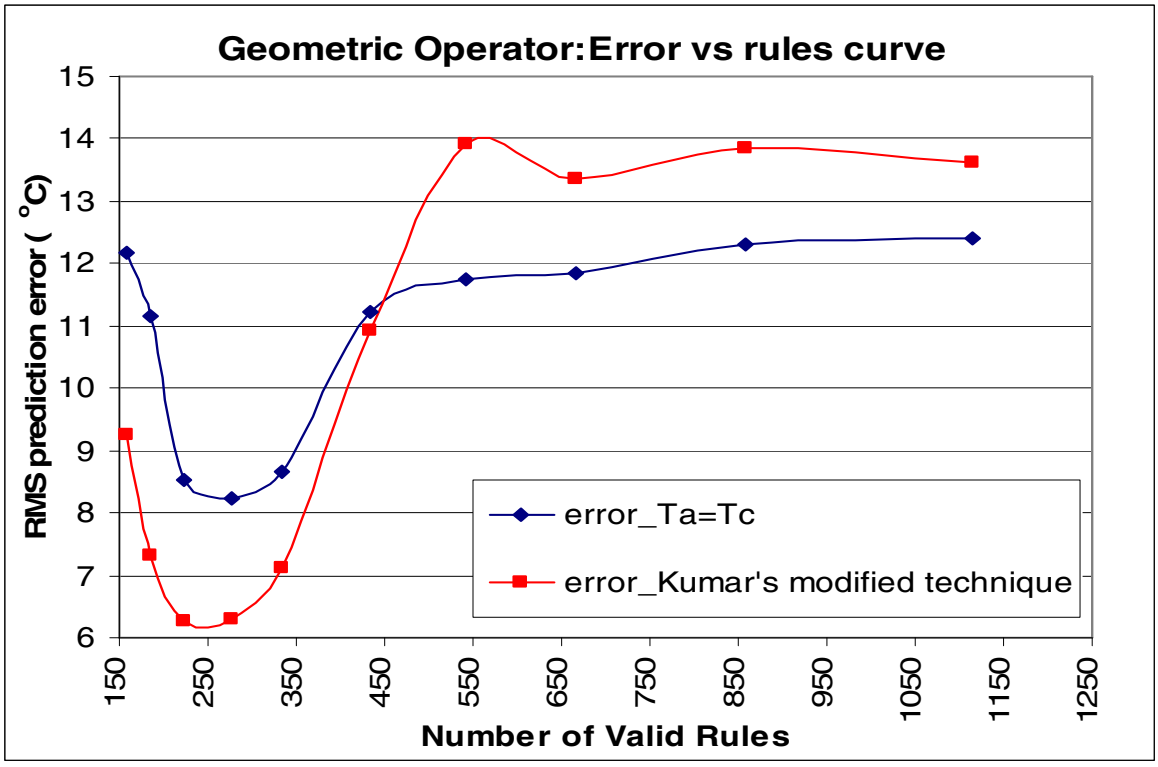


Figure 30: For geometric operator, comparison between the Kumar's modified prediction technique and the conventional technique has been shown. The error first decreases, then increases and, then becomes almost constant as the number of selected rules increase. Kumar's technique was more accurate for large values of corroboration (small number of rules) and conventional prediction technique was more accurate for smaller values of corroboration (large number of rules).

Figure 30 shows that Kumar's technique was more accurate for large values of corroboration (small number of rules) and conventional prediction technique was more accurate for smaller values of corroboration (large number of rules). Kumar's technique relies on using the historical information to predict the output. In the prediction phase, only the rules with the value of $T_{a,}$ greater than 0.5 were allowed to predict the output. On the other hand, all the selected rules were allowed to make predictions for the

conventional prediction technique; and the different predicted values were blended (averaged) to give the final output. For high values of corroboration, the rule base consisted, primarily, of valid rules. Thus, the accuracy of the Kumar's modified technique was better than conventional techniques because the valid rules predicted the output based on the historical information and the conventional techniques predicted the output based on the assumption of $T_a = T_c$. For low values of corroboration, several semi-valid rules were included in the rule base along with the valid rule. Hence, Kumar's modified technique predicted the output based on historical information of both the valid and semi-valid rules. This explains the better accuracy of conventional prediction technique, over Kumar's modified prediction technique for large number of selected rules.

4.2.3.2 Minimum operator: Choice of prediction technique

The trends, in the plot between RMS error and number of rules, obtained using Kumar's prediction technique and conventional technique are shown in Figure 31. The RMS error first decreased and, then increased, as the number of rules increased (due to decrease in the value of corroboration). The increased much rapidly for Kumar's prediction technique, then the conventional prediction technique.

Figure 31 shows that Kumar's technique was more accurate for large values of corroboration (small number of rules) and conventional prediction technique was more accurate for smaller values of corroboration (large number of rules). This trend is similar to the trend obtained using the geometric operator. Again, the decrease in error, with decrease in corroboration, can be attributed to the inclusion of valid rules in the rule base

and, the increase in error, with further decrease in corroboration, was because of inclusion of several semi-valid rules in the rule base.

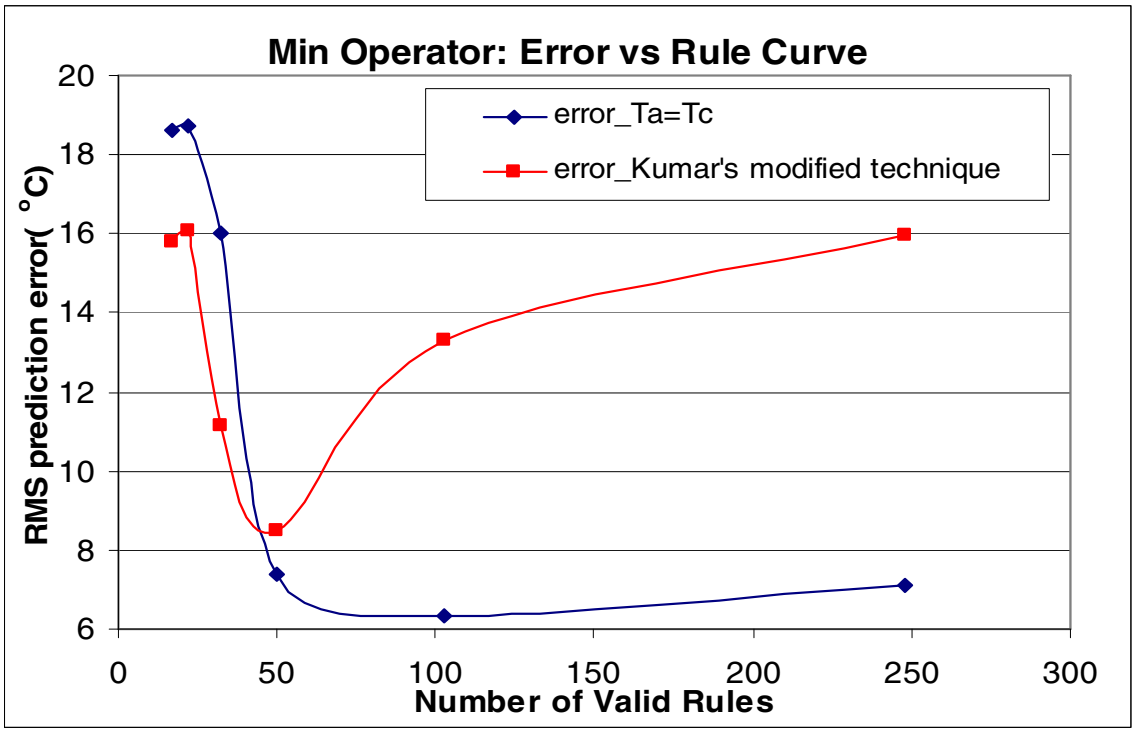


Figure 31: For minimum operator, comparison between the Kumar's modified prediction technique and conventional technique has been shown. The error first decreased, then increased and, then became almost constant as the number of selected rules increased. Kumar's technique was more accurate for large values of corroboration (small number of rules) and conventional prediction technique was more accurate for smaller values of corroboration (large number of rules).

A rapid increase in error, with increase in number of rules, was observed with Kumar's prediction technique as compared to the conventional prediction technique. This is because Kumar's prediction technique relies on much lesser rules (with $T_a > 0.5$) to make predictions as compared to conventional technique (which utilizes all the selected rules to predict). Thus, Kumar's prediction technique is more sensitive to the inclusion of semi-valid rules than the conventional technique.

It should be noted that the prediction capability of Kumar's prediction technique becomes very low as the number of selected rules decrease. The prediction capability (%) became less than 20% when the number of rules became less than 50 (corroboration value greater than 3). However, the prediction capability was much higher with the conventional prediction technique. Hence, Kumar's prediction technique does not appear to be too useful, when used with the minimum operator. It resulted in high errors, for low values of corroboration and its prediction capability became very low, for higher values of corroboration.

4.2.4 Comparison between geometric and minimum operator

The plot between the RMS prediction error and the number of valid rules, obtained for the geometric and minimum operator, is shown in Figure 32. The number of valid rules was obtained by choosing different values of corroboration. The maximum possible rules selected (using corroboration =1) for geometric and minimum operator were 1115 and 248 respectively. The RMS prediction errors, for the given number of rules, were 12.4 and 7.1 respectively. Thus, the minimum operator was 1.75 times more accurate than the geometric operator; and the size of rule base, for the minimum operator, was only 22.2% of the size of rule base for geometric operator (using corroboration =1).

The minimum operator gave high errors as the number of rules decreased (corroboration value increased). The number of rules decreased rapidly with corroboration. Even with a moderate value of corroboration equal to three, only 50 rules were selected, for the minimum operator. On the other hand, geometric operator selected a large number of rules, even for high values of corroboration. For the given prediction

data set, the geometric operator gave the least error with 276 rules (corroboration = 7). However, it is difficult to predict beforehand, which value of corroboration would give the minimum error, for the geometric operator. On the other hand, it seems that a low value of corroboration (1 or 2) would give the optimum (least error) result, for the minimum operator.

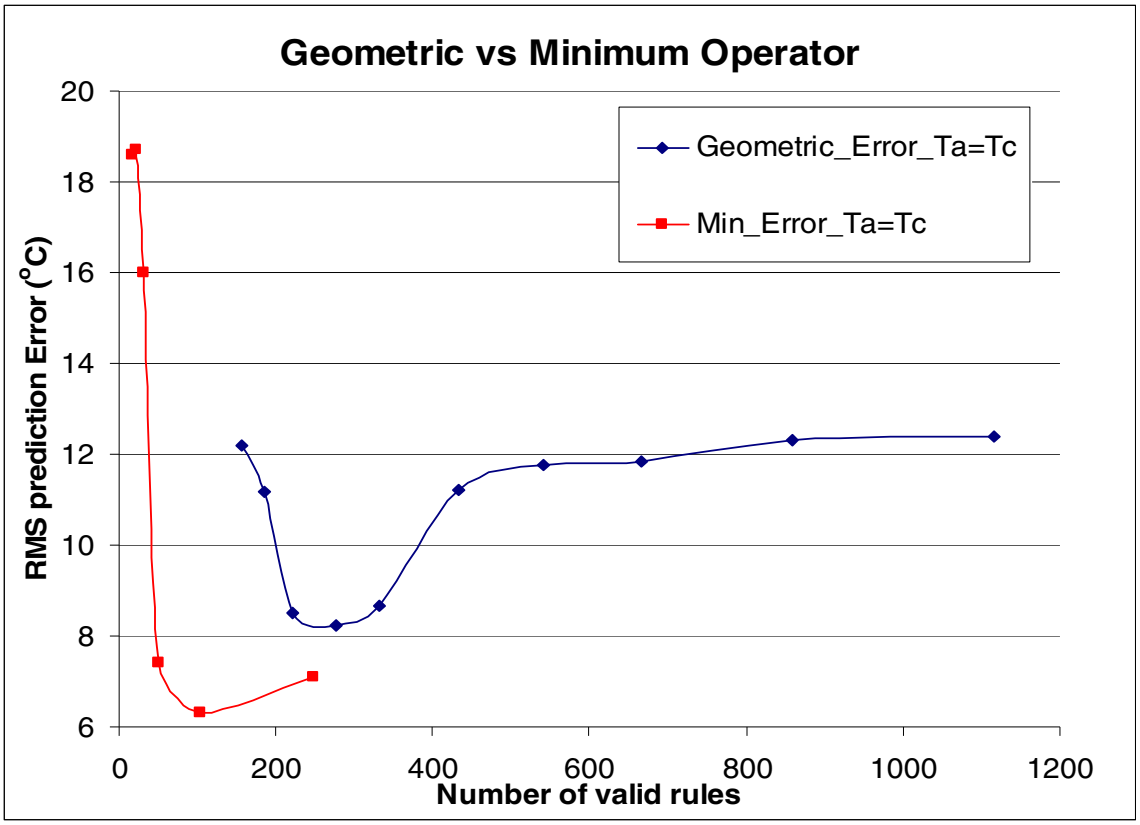


Figure 32: Comparison of geometric and minimum operator has been shown. The minimum error was obtained with minimum operator, using a corroboration value of two.

Furthermore, it is observed that for both the operators, the least error was observed when the number of rules was close to the *least number of rules required to describe the system completely (243).* This may or may not be the universal behavior. This technique should be applied to other processes to confirm or reject the results of this observation.

4.3 Allowing only one interacting rule in the rule base

The affect of allowing only one interacting rule, from a given non-interacting group, to form a part of the valid rule base, has been discussed in this Section. Out of all the selected rules with the same antecedent, only one rule (with maximum number of trips in QI) was allowed to remain in the valid rule base. The other interacting rules were removed from the rule base. One rule was chosen randomly, in situations, where more than one interacting rules had the same number of trips in QI. This new valid rule base (obtained using restriction of allowing only one interacting rule to be selected as a valid rule) was then used to make predictions. These prediction results were compared to those obtained from the old valid rule base, which had no restriction, during the selection of interacting rules. These results are shown in Figure 33. The RMS errors were calculated using the conventional prediction technique.

The number of valid rules and RMS prediction error change gradually, with the increase in corroboration, for the rule base obtained by allowing only one interacting rule to be selected as a valid rule. On the other hand, the number of valid rules and RMS error change much rapidly, with the increase in corroboration, for the rule base obtained with no restriction on the selection of interacting rules. The errors obtained from the rule base, with restriction on the selection of interacting rules, were higher, than those obtained from the rule base obtained with no restriction on the selection of interacting rules. However, for a given value of corroboration, the number of valid rules was much lesser for the rule base, with restriction on the selection of interacting rules, then that for the rule base obtained with no restriction on the selection of valid rules.

High value of errors, for the rule base with restriction on the selection of interacting rules, was because a small number of rules were left in the rule base to make predictions. This explains the gradual increase in error, from the rule base obtained with restriction on the selection of interacting rules, with increase in corroboration. The error curve obtained from the rule base, with no restriction on the selection of interacting rules, has already been discussed in Section 4.2.1.1.

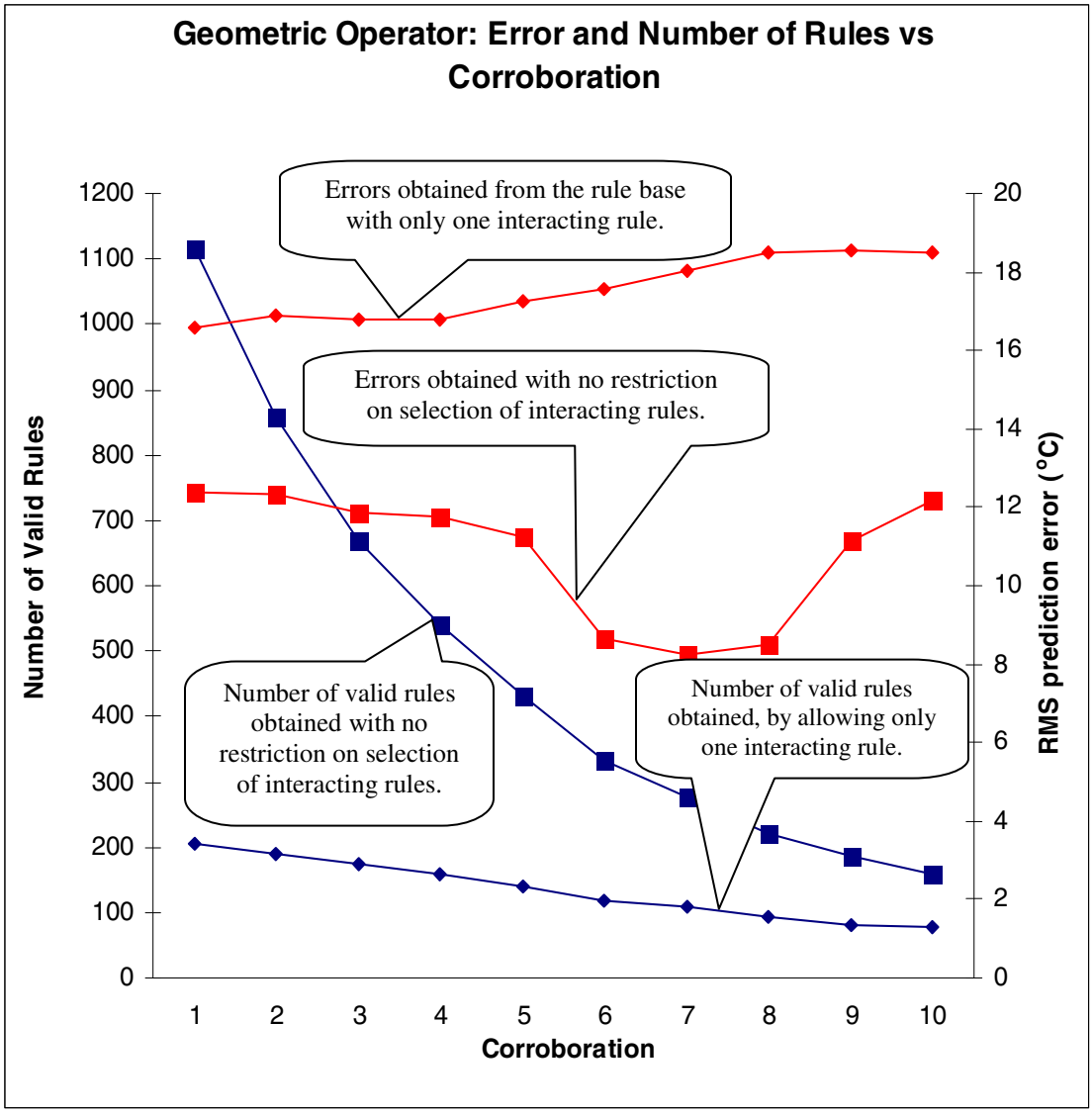


Figure 33: Comparison of the rule base obtained by restricting the selection of only one interacting rule to the rule base obtained with no restriction on the selection of interacting rules.

The rule base, with no restriction on the selection of valid rules, can be generated from the rule base, with restriction on the selection of interacting rules, by allowing interacting rules to be selected. Thus, the inclusion of interacting rules resulted in a decrease in error but lead to rapid changes in the plots of number of rules vs. corroboration and error vs. corroboration.

The plot for the minimum operator has not been discussed because after the removal of interacting rules, very few rules very left in the rule base which would have been of little value to study. However, one particular value of interest for minimum operator, obtained by allowing only one interacting rule to be selected as a valid rule and using a corroboration value of 1, has been discussed in Section 4.4.

4.4 Comparison of results from this work with Kumar's work

Kumar proposed the following values, for the selection of valid rules:

- Mathematical operator to calculate T_a and T_c: Geometric operator
- Corroboration = 2
- Threshold value for Merit = 1
- Threshold = 5
- No restriction on the selection of interacting rules

Using these values, 182 rules were selected. These rules gave a RMS error of 17.6^{o}C for the given prediction data set. However, when only one interacting rule was

allowed to be form a part of the valid rule base, then the total number of rule selected dropped down to 147. These rules gave a RMS prediction error of 17.03^{o}C.

The following values have been proposed in this work for the selection of valid rules:

- Mathematical operator to calculate T_a and T_c: Minimum operator
- Corroboration = 1
- Threshold value for Merit = Not used
- Threshold = 5
- Allow only one out of all the possible interacting rules

Using these values, 109 rules were selected. These rules gave a RMS error of 15^{o}C for the given prediction data set. However, when no restriction was imposed on the selection of interacting rules, then the total number of rule selected were 248. These rules gave a RMS prediction error of 8.6^{o}C.

These results show that the restriction on the selection of interacting rules decrease the size of the rule base but increases the RMS prediction error too. In addition, the rule base gave the minimum error for the value of corroboration, at which the total number of valid rules selected was close to the *minimum number of rules required to describe the system completely*. Thus, it is proposed that interacting rules should not be allowed to form a part of the rule base when the total number of valid rules selected (without including the interacting rules) is close to the *minimum number of rules required to describe the system completely*. However, if the number of valid rules selected is much

less then the *minimum number of rules required to describe the system completely*, then the interacting rules should be allowed to form a part of the rule base to compensate for the missing rules in the rule base.

CHAPTER V

CONCLUSIONS AND SCOPE FOR FUTURE WORK

In this work, the TSD was reinvestigated to analyze the affect of over specification and under specification of the variables, delay quantification was performed, efforts were made to impart universal applicability to this technique and new methods were developed to predict the values of the consequent. The conclusions obtained from this work and the scope of future work in this field has been described below:

5.1 Conclusions

1) TSD can assist in indicating the over specification and under specification in the variables. Firstly, the problem of under specification of the antecedent should be solved. Once this problem is solved then the problem of over specification in the antecedent or "hidden" mechanisms should be solved.
2) Only two consequents (delay and the output variable) should be allowed to form a part of the rule base. For more consequents, separate rule sets should be used simultaneously.

3) The minimum operator should be used to perform the calculations for the truth of the antecedent (T_a) and the truth of the consequent (T_c).
4) The TSD should consist of four Quadrants of size 0.5 x 0.5 each.
5) Only the interacting rules should be compared to each other and perhaps, only one out of all the possible interacting rules should be selected as a valid rule.
6) The metric, corroboration, should be used in the selection of the valid rules. The threshold value to be used for this metric is unity. Merit is not required as a metric for the selection of valid rules.
7) In general, the conventional technique for predicting the output performed slightly well than the modified Kumar's technique for making predictions. However, in few instances, the modified Kumar's technique proved to be more accurate that the conventional techniques.

5.2 Scope for future work

1. Qualitatively, the TSD can indicate the over specification or the under specification in the antecedent or the presence of a hidden mechanism. However, there is a need to develop a quantitative measure to establish the presence of over specification and under specification in the rule base. This quantification should be based on the number of trips in the Quadrants which would ascertain the presence of over specification, under specification or the hidden mechanism.
2. The delay quantification has been performed on the basis of the value of the Pearson's correlation coefficient. This coefficient provides only the strength of

the linear relationship between the T_a and T_c. However, the T_a and T_c might possess a non-linear relationship. In such case, the non-parametric coefficients (such as Spearman R, Kendall Tau and Gamma coefficents) should be calculated and compared to find the best delay value. However, the robustness and sensitivity of these methods is a concern and needs to be taken into consideration. In addition, a single value of the delay was assigned to each trip in QI and QIV. However, in a trip, as the flow rates and the persistence varies, the delay would change. Thus, would be useful to find a value of delay for each data point in a trip rather then finding a single value a delay for the entire trip.

3. The absolute relative average error in prediction has been calculated by using the valid rules selected using the metrics. However, no optimization has been performed to improve the predicted values. This optimization can be done by tuning the membership values of the variables. This tuning of membership value should also ensure minimum shifting of trips from one Quadrant to another. In addition, the minimum operator is a discontinuous function and hard to differentiate. Thus, it is imperative to analyze its affect on the optimization process.
4. Kumar's modified prediction technique is fundamentally sounder than the conventional prediction techniques since it utilizes the historical information to predict the future. However, the computational effort required for Kumar's modified prediction technique is much higher than that for conventional techniques. In addition, the results obtained in this study, for the hot-cold water simulator, have been inconclusive in deciding the best technique based on the

accuracy of the predictions. Thus, a deeper analysis of Kumar's modified prediction technique needs to be done and tested on other systems to judge its ability to predict as compared to the conventional techniques.

5. As suggested in this work, only one out of all the possible interacting rules should be allowed to form a part of the valid rule base to reduce the computational effort required for maintaining the rule set. However, it has also been found that the prediction capability of the minimum operator reduces sharply as the number of valid rules decreases. Hence, the computational effort and prediction capability are two opposing factors involved in the selection of interacting rules. Thus, the possibility of including more than one interacting rules, when very small number of valid rules are discovered by the data, needs to considered.
6. If two interacting rules had the same number of trips in QI then one out of these two rules was selected randomly. However, to improve the prediction capability, it would be better to have a criterion for this selection. For this purpose, a quantitative measure for the *confidence* in a rule should be obtained. The *confidence* in a rule should increase as the number of trips in QI increases with time.

REFERENCES

1) Personal communication, Fluor Corporation, March 2007.
2) Christopher R., "Use it or Lose it: Chemical Industry Energy consumption," Chemical Processing Magazine. August 2006.
3) Sharma N., "Metrics for evaluation of the goodness of linguistic rules," Unpublished Master's Thesis, Oklahoma State University, Oklahoma, USA, 2003.
4) Kumar P., "Improved Quality Metrics for Linguistic Rule Selection," Unpublished Master's Thesis, Oklahoma State University, Oklahoma, USA, 2005.
5) Su M., "Autonomous generation of cause-and-effect rules in dynamics processes", *Unpublished PhD Proposal*, Oklahoma State University, Oklahoma, USA, 2005.
6) Johnson A., "Expert system ensures pipeline efficiency and safety through real-time monitoring and diagnosis," Gensym application paper, 2005
7) Campos M., Satuf E. and Mesquita M., "Intelligent system for start-up of a petroleum offshore platform," Gensym application paper, 2005.
8) Wang X., Mendel M., "Generating Fuzzy Rules by Learning from Examples," IEEE, pp.1414-1427, 1992.
9) Hong T., "Mining coverage-based fuzzy rules by evolutional computation," IEEE, pp.218-224, 2001.

10) Wang J., Yen G., Sun H., “A method of Self-Generating Fuzzy Rule Base via Genetic Algorithm,” Technical Report, National Central University, Taiwan, 2003.

11) Umano M., Okado T., Hatono I. and Tamura H., “Extraction of quantified fuzzy rules form numerical data”, IEEE, pp.1062-1067, 2000.

12) Juuso E., “Knowledge and data integration with linguistic equation”, ESIT, pp.407-415, 2000.

13) Juuso E., “Modeling and control with linguistic equations”, Technical Report, University of Oulu, Finland, 2002.

14) Kermani, “Fuzzy logic system with evolutionary variable rules”, Patent No. US 6,853,991 B1, 2005.

15) Jakobsen T., “Genetic Algorithms”, http://subsimple.com/genealgo.asp, May 2006.

APPENDIX A CODE LISTING

The following Q- Basic code listing is from the *Hot and Cold water mixing* simulator for the case of *With Noise*.

```
DECLARE SUB CLEAN ()
DECLARE SUB ATV (a$, time!, mode1!, mode2!, mdot3sp!, mdot3filt!, t3sp!, t3meas!,
o1!, o2!)
DECLARE SUB FILTINI ()
DECLARE SUB FILTER (mdot1meas!, mdot2meas!, mdot3meas!, mdot1filt!,
mdot2filt!, mdot3filt!)
DECLARE SUB DISPLAY (mode1, mode2, o1, o2, mdot1filt, mdot2filt, mdot3filt,
t1meas, t2meas, t3meas, mdot3sp, t3sp, theta)
DECLARE SUB OPERATOR (a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
DECLARE SUB EVAL (mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
DECLARE SUB CTLINI ()
DECLARE SUB process (o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas, t1meas,
t2meas, t3meas)
DECLARE SUB PLOTINI ()
DECLARE SUB CTL (mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
DECLARE SUB PLOT (o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas,
mdot3sp, t3sp)
DECLARE SUB PROCINI ()
   '
   '                    CONTROL.BAS
   '                 Spring 1998 CHENG-5xxx
   '         Dr. R. Russell Rhinehart, School of Chem. Engr. Oklahoma State U.
   '                    25 Dec 97
   '
   ' This program is a basis for CHENG-5xxx students to test their controllers.
   '
   ' The program models control valves, fluid flow, mixing of a hot and cold
   ' water in a pipe system, and flow and temperature measurement. It also
   ' contains a control subrouting for primitive PID T and F controllers.
   ' The students will write the code for various control strategys,
   ' filters, and goodness of control evaluations; tune their controllers;
   ' and explore the solutions for a variety of process events that cause
   ' control difficulty.
   '
   ' The program is structured so that each stage in the controller-process-
   ' evaluation system are written as subroutines. This MAIN program links and
   ' orders the execution of each subroutine.
   '
   ' The MAIN program calls subroutine PROCESS to dynamically simulate the
```

```
' fluid mixing process for a time interval, t, of 0.1 seconds. PROCESS
' simulates the final element dynamics, as well as the ChEs view of the
' process behavior (fluid dynamics and mixing). It also adds measurement
' bias and process beavior drifts that have an ARMA stochastic behavior.
' It also adds measurement noise and valve "stick-tion".
'
' MAIN then calls subroutine FILT to filter noise from the measurements.
'
' MAIN then calls subroutine CTL, where, eventually students will write
' the code for the various controllers and control strategies. Presently
' CTL contains two independent PID controllers, one for T control (manipulating
' O1) and one for F control (manipulating O2).
'
' MAIN then calls subroutine EVAL, where, eventually students will write
' the code for the various goodness of control measures. Presently EVAL
' calculates T and F NISE.
'
' MAIN then calls subroutine PLOT to generate a strip chart display
' of the controlled and manipulated variables.
'
' Finally MAIN calls DISPLAY to refresh data on the screen.
'
' On operator demand (by keyboard touches) MAIN will call subroutine
' OPERATOR to execute the operator-initiated (student-initiated) changes.
' See subroutine OPERATOR to see what INKEY touches start which commands.
' One of these commands is to initiate ATV tuning, an automatic tuning for
' PID controllers.
'
' This sequence is then repeated. However, first MAIN initializes the
' devices, sets up common variables, and calls PLOTINI, PROCINI, and
' CTLINI to initialize the PLOT, PROCESS, and CTL subroutine variables.
'

Dim plotvmax(10), plotvmin(10), plotvrng(10), plotvar(10), plotyo(10), tf(2000)
COMMON SHARED plotvmax(), plotvmin(), plotvrng(), plotvar(), plotyo(), tf()
COMMON SHARED numvar, plottime, reference, horizon, plotx, plotxo, ploty, time
COMMON SHARED ap1, bp1, cp11b, cp12b, dp1, tauvp1
COMMON SHARED ap2, bp2, cp21b, cp22b, dp2, tauvp2
COMMON SHARED m1biasb, m2biasb, m3biasb, t1biasb, t2biasb, t3biasb
COMMON SHARED taut1, taut2, taut3, t1inpb, t2inpb, tf1, tf2, tf3
COMMON SHARED t, dt, timedelta
COMMON SHARED dpp1b, hp1, power1
COMMON SHARED dpp2b, hp2, power2
COMMON SHARED enviro
COMMON SHARED lambda1, lambda2, lambda3
COMMON SHARED kc1, taui1, taud1, kc2, taui2, taud2, detune
```

```
   COMMON SHARED which$, tune, dataout
   COMMON SHARED iset3, isdo1, isemdot3, isdo2, isenumber
   COMMON SHARED o1, o2
Open "C:\data4.csv" For Output As #1
'PRINT #1, "time", "theta", "t3meas", "t1meas", "t2meas", "mdot3meas", "mdot1meas",
"mdot2meas"
Print #1, "time, t1meas, t2meas, mdot1meas, mdot2meas, t3meas"
Screen 12 'set-up screen for graphics, 640 X 350 x-y pixils, 82 X 25 x-y positions
Randomize ((Timer - 12300) / 3)   'randomize the seed for the random number generator
Cls
enviro = 1
tune = -1              'do not start with ATV tuning
dataout = 1            '**now start without data logging
   Call FILTINI
   Call CTLINI
   Call PROCINI
   Call PLOTINI

   For Interval = 1 To 60000
         time = Interval * t
         If time = 20 Then
         dataout = 1
         End If

         'Adding noise
         If 20 * Int(time / 20) = time Then
            o1 = Rnd * 100
            o2 = Rnd * 100
            t1inpb = Rnd * 100
            t2inpb = Rnd * 100
         End If

         Call process(o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas, t1meas, t2meas,
t3meas)
         a$ = INKEY$
         If a$ <> "" Then
            Call OPERATOR(a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
         End If
         Call FILTER(mdot1meas, mdot2meas, mdot3meas, mdot1filt, mdot2filt,
mdot3filt)
         If tune = 1 Then
           Call ATV(a$, time, mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
         Else
           Call CLEAN
         End If
         Call CTL(mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
```

```
        Call PLOT(o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas,
mdot3sp, t3sp)
        Call EVAL(mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
        Call DISPLAY(mode1, mode2, o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas,
t2meas, t3meas, mdot3sp, t3sp, theta)
        If dataout = 1 Then
          If timedelta * Int(time / timedelta) = time Then  '****log on every (timedelta)
second
            Print #1, time; ","; t1meas; ","; t2meas; ","; mdot1meas; ","; mdot2meas; ",";
t3meas; ","; theta
          End If
        End If
   Next Interval
Close #1
   '   Variable definitions
   ' plotvmax(10) maximum values of the plotted variables
   ' plotvmin(10) minimum values of the plotted variables
   ' plotvrng(10) calculated maximum minus minimum values, range of plotted variables
   ' plotvar(10)  values of the plotted variables
   ' plotyo(10)   pixel positions for the previous strip chart ordinate
   ' tf(200)      array that holds the values for the fictitious temperature
   ' numvar       number of variables plotted
   ' plottime     time argument for the plotting routine, same as time
   ' reference    time at the beginning of each strip chart sweep
   ' horizon      time window of the strip chart
   ' plotx        pixel position for the strip chart abscissa
   ' plotxo       value of the previous plotx pixel position
   ' ploty        pixel position for the strip chart ordinate
   ' time         simulated time, seconds
   ' ap1          "a" coefficient value for process #1, kg/s^2/kPa
   ' bp1          "b" coefficient value for process #1, kg/s^2/m
   ' cp11b        "c11" coefficient base value for process #1, kg/s^2/kg^2/min^2
   ' cp12b        "c12" coefficient base value for process #1, kg/s^2/kg^2/min^2
   ' dp1          "d" coefficient value for process #1, kg/s^2/kg^2/min^2
   ' tauvp1       time constant for process valve #1, seconds
   ' ap2          "a" coefficient value for process #2, kg/s^2/kPa
   ' bp2          "d" coefficient value for process #2, kg/s^2/m
   ' cp21b        "c21" coefficient base value for process #2, kg/s^2/kg^2/min^2
   ' cp22b        "c22" coefficient base value for process #2, kg/s^2/kg^2/min^2
   ' dp2          "d" coefficient value for process #2, kg/s^2/kg^2/min^2
   ' tauvp2       time constant for process valve #2, seconds
   ' taut1        time constant for first temperature lag, seconds
   ' taut2        time constant for second temperature lag, seconds
   ' taut3        time constant for third temperature lag, seconds
   ' t1inpb       process stream #1 inlet temperature base value, centigrade
   ' t2inpb       process stream #2 inlet temperature base value, centigrade
```

```
' tf1          first lagged temperature at the fictitious sensor, centigrade
' tf2          second lagged temperature at the fictitious sensor, centigrade
' tf3          third lagged temperature at the fictitious sensor, centigrade
' t            process sampling time and control period, seconds
' dt           process integration time step, seconds
' dpp1b        driving pressure drop base case for stream #1, kPa
' hp1          elevation head for stream #1, m
' power1       power coefficient for valve #1 characteristic
' dpp2b        driving pressure drop base case for stream #2, kPa
' hp2          elevation head for stream #2, m
' power2       power coefficient for valve #2 characteristic
' enviro       coefficient to toggle environmental effects on/off, 1 if on, 0 if off
' time         simulated time, seconds
' interval     controller sampling period and process integration time step, seconds
' o1           output of controller #1, % of full scale
' o2           output of controller #2, % of full scale
' s1           valve #1 stem position, fraction open
' s2           valve #2 stem position, fraction open
' mdot1meas    measured value of flow rate of stream #1, kg/min
' mdot2meas    measured value of flow rate of stream #2, kg/min
' mdot3meas    measured value of combined flow rate, kg/min
' t3meas       measured value of mixed temperature, centigrade
' a$           variable to store the value of INKEY$, alpha-numeric string
' INKEY$       BASIC function that inputs a keyboard hit, alpha-numeric string
' mode1        mode of controller #1, 1 if AUTO, 0 if MAN
' mode2        mode of controller #2, 1 if AUTO, 0 if MAN
' mdot3sp      set point for total flow rate, kg/min
' t3sp         set point for mixed temperature, centigrade
' lambda1      filter factor for the first-order noise filter on mdot1meas
' lambda2      filter factor for the first-order noise filter on mdot2meas
' lambda3      filter factor for the first-order noise filter on mdot3meas
' kc1          controller 1 gain, %output / kg/min
' taui1        controller 1 integral time, seconds
' taud1        controller 1 derivative time, seconds
' kc2          controller 2 gain, %output / centigrade
' taui2        controller 2 integral time, seconds
' taud2        controller 2 derivative time, seconds
' which$       variable that defines which controller is being ATV tested
' tune         variable to indicate whether ATV tuning is desired
' dataout      variable to indicate whether data is to be recorded in the output file
' iset3        integral of the squared error for t3meas
' isdo1        integral of the squared change in output of controller 1
' isemdot3     integral of the squared error for mdot3filt
' isdo2        integral of the squared change in output of controller 2
' isenumber    count to normalize the ise and isdo
' m*bias       bias on flow rate * measurement
```

```
   ' m*biasb      base level for the bias on flow rate * measurement
   ' t*bias       bias on temperature * measurement
   ' t*biasb      base level for the bias on temperature * measurement

Static Sub ATV(a$, time, mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
    '
    '   ATV tuning
    '   NOTE 1 - I think that I used the ZN Ultimate rules for interacting for non-
interacting PID control
    '   NOTE 2 - need a better way to detect zero crossing in the presence of noise
    '
    If a$ = "a" Or a$ = "A" Then   'you just got here, initialize the factors
      start = 0                 'start time for the ATV test
      e = 0                    'deviation from atvtarg
      eold = 0                  'old deviation
      emax = 0                  'maximum CV deviation from atvtarg in a cycle
      emin = 0                  'minimum CV deviation from atvtarg in a cycle
      LOCATE 15, 1
      INPUT "Do you wish to implement ATV tuning on the O1-T3 loop (1) or O2-F3
(2)"; which$
      LOCATE 15, 1
      Print "                                                              "
      '
      '   initialize the atvtarg and set the controller to manual
      '
      If which$ = "1" Then        'O1-T3 loop was chosen
        atvtarg = t3meas         'initialize the atvtarg with the first CV value
        mode1 = 0               'set the controller to MAN
      LOCATE 14, 1
      Print USING; "atvtarg = ###.# C"; atvtarg
      Else                      'O2-F3 loop was chosen
        atvtarg = mdot3filt
        mode2 = 0
      LOCATE 14, 1
      Print USING; "atvtarg = ###.# kg/min"; atvtarg
      End If
    End If
    '
    '   ATV test controller #1
    '
    If which$ = "1" Then
      If start = 0 Then          'if this is the first time initialize
        start = time            'start time for test
        Switch = time           'time when output was switched
        relay = 20              'output step size (high - low)
        o1 = o1 + relay / 2      'make the first output step, up, by 1/2 of the relay
```

```
        LOCATE 15, 1
        Print "ATV initiated on O1-T3 loop, T3 controller is overridden"
      End If
      If time - start > 15 Then       'hold the first bump for 15 seconds
        e = atvtarg - t3meas         'then calculate the deviation
        If e > emax Then emax = e    'set emax
        If e < emin Then emin = e    'set emin
        LOCATE 14, 1
        Print USING; "atvtarg = ###.# C   emax = ###.### C    emin = ###.### C     ";
atvtarg; emax; emin
        If e * eold <= 0 Then        'if the error changed sign, the atvtarg was crossed
          If e < 0 Then              'if the error is negative
            o1 = o1 - relay          'then step the output down by 1/1 relay
          End If
          If e > 0 Then              'if the error is positive, then a cycle had finished
            o1 = o1 + relay          'then step the output up by 1/1 relay
            pu = time - Switch       'calculate the ultimate period
            ku = 4 * relay / (emax - emin) / 3.14159    'and the ultimate gain
            LOCATE 15, 1
            Print USING; "ATV O1-T3 in cycling mode. Ult. P. = ###.## sec    Ult. Kc =
###.## %/C"; time - Switch; 4 * relay / (emax - emin) / 3.14159
            LOCATE 16, 1
            Print USING; "(Kc=###.#)  (Kc=###.#  taui=###.#)  (Kc=###.#  taui=###.#
taud=###.#)"; 0.5 * ku; 0.45 * ku; 0.83 * pu; 0.59 * ku; 0.5 * pu; 0.125 * pu
            o1 = o1 + 0.25 * relay * (emax + emin) / (emax - emin) 'shift o1 for symmetry
            emax = 0                 'reset emax for the next cycle
            emin = 0                 'reset emin for the next cycle
            Switch = time            'reset switch for the next cycle
          End If
        End If
        eold = e
      End If
    Else                             'which = 2, ATV the flow loop
      If start = 0 Then
        start = time
        Switch = time
        relay = 30
        o2 = o2 + relay / 2
        LOCATE 15, 1
        Print "ATV initiated on O2-F3 loop, F3 controller is overridden"
      End If
      If time - start > 5 Then
        e = atvtarg - mdot3filt
        If e > emax Then emax = e
        If e < emin Then emin = e
        LOCATE 14, 1
```

```
        Print USING; "atvtarg = ###.# kg/min    emax = ###.### kg/min    emin = ###.###
kg/min"; atvtarg; emax; emin
        If e * eold <= 0 Then
          If e < 0 Then
            o2 = o2 - relay
          End If
          If e > 0 Then
            o2 = o2 + relay
            pu = time - Switch
            ku = 4 * relay / (emax - emin) / 3.14159
            LOCATE 15, 1
            Print USING; "ATV O2-F3 in cycling mode. Ult. P. = ###.## sec    Ult. Kc =
###.## %/kg/min"; pu; ku
            LOCATE 16, 1
            Print USING; "(Kc=###.#)  (Kc=###.#  taui=###.#)  (Kc=###.#  taui=###.#
taud=###.#)"; 0.5 * ku; 0.45 * ku; 0.83 * pu; 0.59 * ku; 0.5 * pu; 0.125 * pu
            o2 = o2 + 0.25 * relay * (emax + emin) / (emax - emin) 'shift o2 for symmetry
            emax = 0
            emin = 0
            Switch = time
          End If
        End If
        eold = e
      End If
    End If
End Sub

   Sub CLEAN()
   '
   '   clean the ATV messages from the screen
   '
   LOCATE 14, 1
   Print "                                                    "
   LOCATE 15, 1
   Print "                                                    "
   LOCATE 16, 1
   Print "                                                      "
End Sub

   Static Sub CTL(mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
   '
   '  Presently there are two independent, standard PID controllers here.
   '  One controls T3 by manipulating O1, the output to valve 1, the hot water
   '  valve. The other controls F3 by manipulating O2, the output to valve 2,
   '  the cold water valve. Because the process is interactive (O1 affects both
   '  T3 and F3), the controllers use the "BLT" method of detuning them jointly,
```

```
  '  after they were independently tuned by "ATV" for "QAD" process behavior.
  '
  '
  '  Temperature controller
     If mode1 = 1 Then                          'temperature controller in AUTO
       e1 = t3sp - t3meas                       'reverse acting
       bias1 = bias1 + t * kc1 * e1 / taui1 / detune ^ 2 'adjustable bias, rectangle rule
       eant1 = e1 - taud1 * (t3meas - t3old) / t   'anticipated error, D-on-X
       t3old = t3meas
       o1 = kc1 * eant1 / detune + bias1          'proportional plus bias
       If o1 > 110 Then                         'anti-windup provision
         o1 = 110
         bias1 = o1 - kc1 * eant1 / detune
       End If
       If o1 < -10 Then                         'anti-windup provision
         o1 = -10
         bias1 = o1 - kc1 * eant1 / detune
       End If
     Else                                     'temperature controller in MAN
       t3sp = t3meas                            'setpoint tracking, bumpless transfer
       t3old = t3meas                           'no D spike, bumpless transfer
       bias1 = o1                             'bias tracking, bumpless transfer
     End If
  '
  '  Flow controller
  '
     If mode2 = 1 Then                          'flow controller in AUTO
       e2 = mdot3sp - mdot3filt                   'reverse acting
       bias2 = bias2 + t * kc2 * e2 / taui2 / detune ^ 2      'adjustable bias, rectangle rule
       eant2 = e2 - taud2 * (mdot3filt - mdot3old) / t   'anticipated error, D-on-X
       mdot3old = mdotfilt
       o2 = kc2 * eant2 / detune + bias2          'proportional plus bias
       If o2 > 110 Then                         'anti-windup provision
         o2 = 110
         bias2 = o2 - kc2 * eant2 / detune
       End If
       If o2 < -10 Then                         'anti-windup provision
         o2 = -10
         bias2 = o2 - kc2 * eant2 / detune
       End If
     Else                                     'flow controller in MAN
       mdot3sp = mdot3filt                        'setpoint tracking, bumpless transfer
       mdot3old = mdot3filt
       bias2 = o2                             'bias tracking, bumpless transfer
     End If
End Sub
```

```
    Static Sub CTLINI()
    '
    '  Initial controller settings go here  static makes them constant
    '
    t = 0.1
    timedelta = 1  'log every timedelta seconds
    mode1 = 0      'controller 1 is in manual
    mode2 = 0      'controller 2 is in manual
    kc1 = 2        '% / centigrade
    taui1 = 12     'seconds
    taud1 = 3      'seconds
    kc2 = 8        '% / kg/min
    taui2 = 2.5    'seconds
    taud2 = 0      'seconds
    detune = 1     'dimensionless
End Sub

   Sub DISPLAY(mdot1, mdot2, o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas,
t3meas, mdot3sp, t3sp, theta)
   '
   '   subroutine to display variables and status on the screen
   '
   LOCATE 17, 1
   Print USING; " theta = ###.#####  time =   ####"; theta; time
   Print USING; "    o1 = ###.#       o2 = ###.#"; o1; o2
   Print USING; "F1filt = ###.#   F2filt = ###.#"; mdot1filt; mdot2filt
   Print USING; "T1meas = ###.###  T2meas = ###.#"; t1meas; t2meas
   Print USING; "T3meas = ###.#   F3filt = ###.#"; t3meas; mdot3filt
   Print USING; "T3sp   = ###.#   F3sp   = ###.#"; t3sp; mdot3sp
   Print USING; "kc1=##.# taui1=##.# taud1=##.#    kc2=##.# taui2=##.# taud2=##.#
detune=#.#"; kc1; taui1; taud1; kc2; taui2; taud2; detune
End Sub

   Static Sub EVAL(mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
   '
   '   measures of control goodness are calculated here
   '
   isenumber = isenumber + 1
   iset3 = iset3 + t * (t3sp - t3meas) ^ 2
   isdo1 = isdo1 + t * (o1 - o1old) ^ 2
   o1old = o1
   niset3 = iset3 / (isenumber * t)
   nisdo1 = isdo1 / (isenumber * t)
   isemdot3 = isemdot3 + t * (mdot3sp - mdot3filt) ^ 2
   isdo2 = isdo2 + t * (o2 - o2old) ^ 2
```

```
   o2old = o2
   nisemdot3 = isemdot3 / (isenumber * t)
   nisdo2 = isdo2 / (isenumber * t)
   '
   '  LOCATE Y,X locates the beginning of the subsequent print statement
   '  at Y text rows down from the top of the screen and X text columns to
   '  the right from the left of the screen. The screen is 22 rows by 75
   '  columns.
   '  PRINT USING "    ";   is a formatted print statement. # marks locations
   '  for numerical values.
   '
   LOCATE 21, 35
   Print USING; " rmset = #.####^^^^    rmsef = #.####^^^^"; Sqr(niset3);
Sqr(nisemdot3)
   LOCATE 22, 35
   Print USING; "rmsdo1 = #.####^^^^   rmsdo2 = #.####^^^^"; Sqr(nisdo1);
Sqr(nisdo2)
End Sub

Static Sub FILTER(mdot1meas, mdot2meas, mdot3meas, mdot1filt, mdot2filt, mdot3filt)
   '
   '  subroutine to first-order filter the noisy process measurements
   '  lambda = 1-exp(T/taufilt)
   '
    mdot1filt = lambda1 * mdot1meas + (1 - lambda1) * mdot1filt
    mdot2filt = lambda2 * mdot2meas + (1 - lambda2) * mdot2filt
    mdot3filt = lambda3 * mdot3meas + (1 - lambda3) * mdot3filt
End Sub

   Static Sub FILTINI()
   '
   '  subroutine to initialize the filter coefficients
   '
    lambda1 = 0.2
    lambda2 = 0.2
    lambda3 = 0.2
End Sub

Sub OPERATOR(a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
    '
    '  operator initiated action is made here
    '
    iset3 = 0          'Reset the goodness of control measures
    isdo1 = 0          '  "
    isemdot3 = 0       '  "
    isdo2 = 0          '  "
```

```
isenumber = 0        '  "
If a$ = "q" Or a$ = "Q" Then
  Close #1
  Stop   'key in "q" to stop the program
End If
If a$ = "a" Or a$ = "A" Then tune = -tune
If a$ = "-" Then t1inpb = t1inpb - 5 '***add or subtract input temperature
If a$ = "+" Then t1inpb = t1inpb + 5
If a$ = "9" Or a$ = "L" Then dataout = -dataout
If a$ = "n" Or a$ = "N" Then        'key in "n" to toggle enviro and disturbances
  If enviro = 1 Then
    enviro = 0
  Else
    enviro = 1
  End If
End If
If a$ = "1" Then                'key in "1" to toggle controller 1 MAN-AUTO
  If mode1 = 1 Then
    mode1 = 0
  Else
    mode1 = 1
  End If
End If
If a$ = "2" Then                'key in "2" to toggle controller 2 MAN-AUTO
  If mode2 = 1 Then
    mode2 = 0
  Else
    mode2 = 1
  End If
End If
'
'  change output if in manual
'
If a$ = "3" And mode1 = 0 Then o1 = o1 - 5  'key in "3" lower o1 in MAN
If a$ = "#" And mode1 = 0 Then o1 = o1 + 5  'key in "#" raise o1 in MAN
If a$ = "4" And mode2 = 0 Then o2 = o2 - 5  'key in "4" lower o2 in MAN
If a$ = "$" And mode2 = 0 Then o2 = o2 + 5  'key in "$" raise o2 in MAN
'
'  limit output to between -10 and 110 %
'
If o1 > 110 Then o1 = 110
If o1 < -10 Then o1 = -10
If o2 > 110 Then o2 = 110
If o2 < -10 Then o2 = -10
'
'  change setpoint if in automatic - method 1:
```

```
    '
    If a$ = "5" And mode1 = 1 Then t3sp = t3sp - 2  'key in "5" lower tsp in AUTO
    If a$ = "%" And mode1 = 1 Then t3sp = t3sp + 2  'key in "%" raise tsp in AUTO
    If a$ = "6" And mode2 = 1 Then mdot3sp = mdot3sp - 2  'key in "6" lower mdotsp in
AUTO
    If a$ = "^" And mode2 = 1 Then mdot3sp = mdot3sp + 2  'key in "^" raise mdotsp in
AUTO
    '
    '  change setpoint if in automatic - method 2:
    '
    If a$ = "s" Or a$ = "S" Then
      LOCATE 16, 35
      Print "Enter one of these setpoints:"
      LOCATE 17, 35
      Print "t3, f3"
      LOCATE 18, 35
      INPUT "Which value do you wish to change"; b$
      If b$ = "t3" And mode1 = 1 Then
        LOCATE 19, 35
        INPUT "Enter t3sp value, C"; t3sp
      End If
      If b$ = "f3" And mode2 = 1 Then
        LOCATE 19, 35
        INPUT "Enter mdot3sp value, kg/min"; mdot3sp
      End If
    '
    '  erase on-screen trash
    '
      LOCATE 16, 35
      Print "                                  "
      LOCATE 17, 35
      Print "                                  "
      LOCATE 18, 35
      Print "                                  "
      LOCATE 19, 35
      Print "                                  "
    End If
    '
    '  if tuning is desired
    '
    If a$ = "t" Or a$ = "T" Then
      LOCATE 16, 35
      Print "Enter one of these parameters:"
      LOCATE 17, 35
      Print "kc1, taui1, taud1, kc2, taui2, taud2, detune"
      LOCATE 18, 35
```

```
      INPUT "Which value do you wish to change"; b$
      If b$ = "kc1" Then
        LOCATE 19, 35
        INPUT "Enter kc1 value, %/C"; kc1
      End If
      If b$ = "taui1" Then
        LOCATE 19, 35
        INPUT "Enter taui1 value, s"; taui1
      End If
      If b$ = "taud1" Then
        LOCATE 19, 35
        INPUT "Enter taud1 value, s"; taud1
      End If
      If b$ = "kc2" Then
        LOCATE 19, 35
        INPUT "Enter kc2 value, %/kg/min"; kc2
      End If
      If b$ = "taui2" Then
        LOCATE 19, 35
        INPUT "Enter taui2 value, s"; taui2
      End If
      If b$ = "taud2" Then
        LOCATE 19, 35
        INPUT "Enter taud2 value, s"; taud2
      End If
      If b$ = "detune" Then
        LOCATE 19, 35
        INPUT "Enter detune value"; detune
      End If
    '
    '  erase on-screen trash
    '
      LOCATE 16, 35
      Print "                                  "
      LOCATE 17, 35
      Print "                                  "
      LOCATE 18, 35
      Print "                                  "
      LOCATE 19, 35
      Print "                                  "
    End If
End Sub


 Static Sub PLOT(o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas,
mdot3sp, t3sp)
```

```
'
'   This routine plots the scaled variables on a strip chart display
'
'                       PLOT.BAS
'              R. Russell Rhinehart Company
'                   10 October 1994
'
'   After calculating the variable values assign them to the plot variables
'
plottime = time             'simulated time, seconds
plotvar(1) = o1             'output of controller 1, %
plotvar(2) = o2             'output of controller 2, %
plotvar(3) = mdot1filt      'filtered flow rate 1, kg/min
plotvar(4) = mdot2filt      'filtered flow rate 2, kg/min
plotvar(5) = mdot3filt      'filtered total flow rate, kg/min
plotvar(6) = t1meas         'measured temperature, centigrade
plotvar(7) = t2meas         'measured temperature, centigrade
plotvar(8) = t3meas         'measured temperature, centigrade
plotvar(9) = mdot3sp        'flow 3 setpoint, kg/min
plotvar(10) = t3sp          'temperature 3 setpoint, centigrade
'
'  Plot routine
'
If plottime - reference >= horizon Then       ' locate the x position
   reference = reference + horizon
   plotxo = 50
   Line (plotxo, 20)-(plotxo, 160), 15
   Line (plotx, 20)-(plotx, 160), 15
   Line (plotx, 161)-(plotx, 168), 14
End If
plotx = 50 + Int(0.5 + 580 * (plottime - reference) / horizon)
If 50 + 58 * Int((plotx - 50) / 58) = plotx Then Line (plotx, 20)-(plotx, 160), 15
Line (plotx + 1, 20)-(plotx + 1, 160), 14
Line (plotx, 161)-(plotx, 168), 0
Line (plotx - 1, 161)-(plotx - 1, 168), 14
For plotyy = 20 To 160 Step 14
Line (plotx, plotyy)-(plotx + 1, plotyy), 15
Next plotyy
For ploti = 1 To numvar
ploty = 160 - 140 * (plotvar(ploti) - plotvmin(ploti)) / plotvrng(ploti)
If ploty < 20 Then ploty = 20
If ploty > 160 Then ploty = 160
Line (plotxo, plotyo(ploti))-(plotx, ploty), ploti
plotyo(ploti) = ploty
Next ploti
plotxo = plotx
```

```
End Sub

   Static Sub PLOTINI()
     '   This routine initializes the strip chart display plot subroutine
     '
     '                      PLOT.BAS
     '              R. Russell Rhinehart Company
     '                   10 October 1994
     '
     ' initialize the plotting variables
     '
     plotxo = 50             ' time = 0  position on the screen
     numvar = 10              ' number of variables to plot, maximum = 10
     horizon = 60            ' strip chart horizon, seconds
     plotvmax(1) = 100       ' maximum value for controller #1 output, %
     plotvmin(1) = 0         ' minimum value for controller #1 output, %
     plotvmax(2) = 100       ' maximum value for controller #2 output, %
     plotvmin(2) = 0         ' minimum value for controller #2 output, %
     plotvmax(3) = 30        ' maximum value for flow rate #1, kg/min
     plotvmin(3) = 0         ' minimum value for flow rate #1, kg/min
     plotvmax(4) = 30        ' maximum value for flow rate #2, kg/min
     plotvmin(4) = 0         ' minimum value for flow rate #2, kg/min
     plotvmax(5) = 60        ' maximum value for total flow rate, kg/min
     plotvmin(5) = 0         ' minimum value for total flow rate, kg/min
     plotvmax(6) = 100       ' maximum value for mixed temperature, C
     plotvmin(6) = 0         ' minimum value for mixed temperature, C
     plotvmax(7) = 100       ' maximum value for temperature 1, C
     plotvmin(7) = 0         ' minimum value for temperature 1, C
     plotvmax(8) = 100       ' maximum value for temperature 2, C
     plotvmin(8) = 0         ' minimum value for temperature 2, C
     plotvmax(9) = 60        ' maximum value for flow3 setpoint, kg/min
     plotvmin(9) = 0         ' minimum value for flow3 setpoint, kg/min
     plotvmax(10) = 100      ' maximum value for temperature 3 setpoint, C
     plotvmin(10) = 0        ' minimum value for temperature 3 setpoint, C
                     ' repeat for all plotted variables
     reference = 0           ' time of the beginning of each strip chart
     '
     '   Initialize the graph
     '   (setup lables, background, grid lines, and initial points)
     '
       LOCATE 1, 1
       Print USING; "PV's (fraction of full scale) VERSUS TIME (fraction of window =
####.# seconds)"; horizon
       For plotj = 0 To 1 Step 0.5           ' lable the y axis
       ploty = 2 + 10 * plotj
       LOCATE ploty, 1
```

```
        Print USING; "#.##"; 1 - plotj
        Next plotj
        For ploti = 0 To 1.01 Step 0.1          ' lable the x axis
        plotx = 6 + 71 * ploti
        LOCATE 13, plotx
        Print USING; "#.##"; ploti;
        Next ploti
        Line (40, 13)-(640, 168), 14, BF        ' fill in the background
        For plotyy = 20 To 160 Step 14          ' draw the horizontal grid
        Line (50, plotyy)-(630, plotyy), 15
        Next plotyy
        For plotxx = 50 To 630 Step 58          ' draw the vertical grid
        Line (plotxx, 20)-(plotxx, 160), 15
        Next plotxx
        For ploti = 1 To numvar                 ' calculate the plot variable
                                    ' ranges and initial locations
        plotvrng(ploti) = plotvmax(ploti) - plotvmin(ploti)
        ploty = 160 - 140 * (plotvar(ploti) - plotvmin(ploti)) / plotvrng(ploti)
        If ploty < 20 Then ploty = 20
        If ploty > 160 Then ploty = 160
        plotyo(ploti) = ploty
      Next ploti
End Sub

   Static Sub process(o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas, t1meas,
t2meas, t3meas)
   '
   '  Subroutine to model the flow rates and temperatures. There are several
   '  sections to this routine. First, if enviro is active, stochastic models
   '  are used to change the flow rate driving pressures, flow pressure loss
   '  coefficients, and inlet stream temperatures. Also, if enviro is active,
   '  control valve action is subject to "sticktion." Next, the ODEs that
   '  dynamically model the valve stem positions, and the coupled ODEs that
   '  dynamically model the flow rates and mixture temperature are solved
   '  using the second order Runge-Kutta method. Since the ODE-modeled
   '  temperature is the mixing point temperature, the temperature values are
   '  placed in an array so that the transport-delayed value can be used for
   '  the fluid temperature at the sensor. Since the transport delay is
   '  variable, the how-far-back-in-the-array index, nt, is calculated from
   '  the transport delay, theta. The "clock" concept is used for efficient
   '  array management. The temperature sensor is modeled as a third order ODE.
   '  Finally, noise is added to the flow rate measurement to simulate orifice
   '  turbulence noise.
   '
   '
   '  if enviro is active then add drift and spikes to the pressure drops
```

```
'
ddpp1 = 0.999 * ddpp1 + 0.015 * dpp1b * (Rnd - 0.5) * enviro 'drift
If Rnd < 0.01 Then spike1 = 50 * (Rnd - 0.5) * enviro        'spike
spike1 = 0.9 * spike1                                        'fade the spike
dpp1 = dpp1b '+ ddpp1 + spike1 <<<<<<*****making sure no spikes
ddpp2 = 0.999 * ddpp2 + 0.015 * dpp2b * (Rnd - 0.5) * enviro 'drift
If Rnd < 0.01 Then spike2 = 50 * (Rnd - 0.5) * enviro        'spike
spike2 = 0.9 * spike2                                        'fade the spike
dpp2 = dpp2b '+ ddpp2 + spike2 <<<<<<<<******ditto
'
'   if enviro is active then add drift to the flow pressure loss factors
'****here i made sure again that no drift is there
dcp11 = 0.999 * dcp11 + 0.015 * cp11b * (Rnd - 0.5) * enviro 'drift
cp11 = cp11b '+ dcp11
dcp12 = 0.999 * dcp12 + 0.015 * cp12b * (Rnd - 0.5) * enviro 'drift
cp12 = cp12b '+ dcp12
dcp21 = 0.999 * dcp21 + 0.015 * cp21b * (Rnd - 0.5) * enviro 'drift
cp21 = cp21b '+ dcp21
dcp22 = 0.999 * dcp22 + 0.015 * cp22b * (Rnd - 0.5) * enviro 'drift
cp22 = cp22b '+ dcp22
'
'   if enviro is active then add drift to the inlet temperatures
' ***ditto
dt1in = 0.999 * dt1in + 0.015 * t1inpb * (Rnd - 0.5) * enviro 'drift
t1inp = t1inpb '+ dt1in
dt2in = 0.999 * dt2in + 0.015 * t2inpb * (Rnd - 0.5) * enviro 'drift
t2inp = t2inpb '+ dt2in
'
'  If enviro is active then add "sticktion" hysteresis to the valves.
'  Deadband is the amount of change in valve position the controller must
'  call for before the valve stem will move. Here, deadband is either 0 %
'  or 2.5 %. Dels1 and dels2 are the valve stem position changes that the
'  controller wants. Note: If sticktion is present, and the valve position
'  is 2 % open, and the controller wants it closed (o = 0 %), then the valve
'  will stay at 2 % open! This is real. To fix it, controllers are designed
'  so that their output goes from -10 % to 110 %, or so. Ideally the 0-100 %
'  controller output is converted to a 4-20 mA d.c. current "signal" then to
'  a 3-15 psig pneumatic "signal" which operates the valve. Ideally the stem
'  position goes from 0 to 1 as the pressure goes from 3 to 15 psig. Allowing
'  the controller output to range from -10 to 110 %, ideally causes the
'  pneumatic signal to range from 1.8 to 16.2 psig which, hopefully, will
'  overcome both sticktion and calibration errors in the D/A and i/p devices,
'  and, thereby, allow the valve to fully close and to fully open.
'
deadband = 0.025 * enviro * 0              'deadband<<****making sure it is 0
current1 = 4 + o1 * 16 / 100               'i1 from A/D conversion of o1
```

```
    current2 = 4 + o2 * 16 / 100              'i2 from A/D conversion of o2
    p1targ = 3 + (current1 - 4) * 12 / 16     'p1 target from i/p conversion of i1
    p2targ = 3 + (current2 - 4) * 12 / 16     'p2 target from i/p conversion of i2
    '
    '  In the following segment of code, the ODEs are solved using a
    '  second-order Rung-Kutta method with an integration time step that
    '  is one tenth of the control interval (dt = t/10).
    '
    '  Calculate the R-K k1s for p1, p2, mdot1, mdot2, tf1, tf2, and tf3.
    '  The IF statements either allow for sticktion or prevent numerical
    '  overflow. If the valves are nearly closed, then f1 or f2 are extremely
    '  small, and their contributions to the Ks are large negative. The -20
    '  is a relatively large negative value.
    '
    For i = 1 To 10
    '
    '  Calculate the transport delay from the mixing point to the temperature
    '  sensor 1.06 meters down stream. Then, nt, the nearest integer number of
    '  sample intervals backward in the clock array. Then, ifind, the array
    '  location of that transport-delayed temperature. Note, this deadtime
    '  delayed temperature is the influence for the third-order lagged sensor
    '  temperature.
    '
      SHARED theta
      If (mdot1 + mdot2) > 0.1 Then          'if mdot total is greater than the minimum
        theta = 80 / (mdot1 + mdot2)         '****calculate transport delay doubled the value
of Lt from 20 to 80
      Else
        theta = 800                          'limit delay to maximum allowed by tf(200)
      End If
      'OPEN "c:theta.dat" FOR OUTPUT AS #2
      'PRINT #2, theta
      'CLOSE #2
      nt = Int(theta / t + 0.5)              'Number of Time intervals in delay
      If nt > ntold + 1 Then nt = ntold + 1   'can't sample fluid past the sensor
      If nt > 1999 Then nt = 1999            'can't sample around the tf(200) "clock"
      ntold = nt
      ifind = iput - nt                      'calculate the find location
      If ifind < 0 Then ifind = ifind + 2001  'increment it if it passes 12 O'clock
      '
      '  calculate the R-K k1s
      '
      k1p1 = (p1targ - p1) / tauvp1  'rate of change of p1, now, due to p1targ
      k1p2 = (p2targ - p2) / tauvp2  'rate of change of p2, now, due to p2targ
      f1 = s1 ^ power1               'inherrent valve characteristic from stem
      If f1 > 0.0001 Then
```

```
      k1mdot1 = ap1 * dpp1 + bp1 * hp1 - cp11 * mdot1 ^ 2 - cp12 * (mdot1 + mdot2) ^
2 - dp1 * mdot1 ^ 2 / f1 ^ 2
    Else
      k1mdot1 = -20
    End If
    If k1mdot1 < -20 Then k1mdot1 = -20
    f2 = s2 ^ power2              'inherent valve characteristic from stem
    If f2 > 0.0001 Then
      k1mdot2 = ap2 * dpp2 + bp2 * hp2 - cp21 * mdot2 ^ 2 - cp22 * (mdot1 + mdot2) ^
2 - dp2 * mdot2 ^ 2 / f2 ^ 2
    Else
      k1mdot2 = -20
    End If
    If k1mdot2 < -20 Then k1mdot2 = -20
    k1tf1 = (tf(ifind) - tf1) / taut1
    k1tf2 = (tf1 - tf2) / taut2
    k1tf3 = (tf2 - tf3) / taut3
    k1tt1 = (t1inp - tt1) / 10
    k1tt2 = (t2inp - tt2) / 10
  '
  ' Use the k1s to estimate where the state variables might go.
  ' The h addended to the state variable indicates Hypothesized.
  ' The limits are for physical reality.
  '
    p1h = p1 + dt * k1p1
    p2h = p2 + dt * k1p2
    dels1h = (p1h - 3) / 12 - s1   'change in s1 that the p1h would make w/o sticktion
    dels2h = (p2h - 3) / 12 - s2   'change in s2 that the p2h would make w/o sticktion
    If Abs(dels1h) > deadband Then s1h = s1 + dels1h   's1 only changes if p1
overcomes sticktion
    If Abs(dels2h) > deadband Then s2h = s2 + dels2h   's2 only changes if p2
overcomes sticktion
    mdot1h = mdot1 + dt * k1mdot1
    mdot2h = mdot2 + dt * k1mdot2
    tf1h = tf1 + dt * k1tf1
    tf2h = tf2 + dt * k1tf2
    tf3h = tf3 + dt * k1tf3
    tt1h = tt1 + dt * k1tt1
    tt2h = tt2 + dt * k1tt2
    If s1h < 0 Then s1h = 0
    If s1h > 1 Then s1h = 1
    If s2h < 0 Then s2h = 0
    If s2h > 1 Then s2h = 1
    If mdot1h < 0 Then mdot1h = 0
    If mdot2h < 0 Then mdot2h = 0
  '
```

```
  ' Calculate the R-K k2s for s1, s2, mdot1, mdot2, tf1, tf2, and tf3.
  ' The IF statements either allow for sticktion or prevent numerical overflow.
  '
    k2p1 = (p1targ - p1h) / tauvp1
    k2p2 = (p2targ - p2h) / tauvp2
    f1h = s1h ^ power1
    If f1h > 0.0001 Then
      k2mdot1 = ap1 * dpp1 + bp1 * hp1 - cp11 * mdot1h ^ 2 - cp12 * (mdot1h +
mdot2h) ^ 2 - dp1 * mdot1h ^ 2 / f1h ^ 2
    Else
      k2mdot1 = -20
    End If
    If k2mdot1 < -20 Then k2mdot1 = -20
    f2h = s2h ^ power2
    If f2h > 0.0001 Then
      k2mdot2 = ap2 * dpp2 + bp2 * hp2 - cp21 * mdot2h ^ 2 - cp22 * (mdot1h +
mdot2h) ^ 2 - dp2 * mdot2h ^ 2 / f2h ^ 2
    Else
      k2mdot2 = -20
    End If
    If k2mdot2 < -20 Then k2mdot2 = -20
    k2tf1 = (tf(ifind) - tf1h) / taut1
    k2tf2 = (tf1h - tf2h) / taut2
    k2tf3 = (tf2h - tf3h) / taut3
    k2tt1 = (t1inp - tt1h) / 10
    k2tt2 = (t2inp - tt2h) / 10
  '
  ' Use the k1s and k2s to estimate where the state variables will go.
  ' The limits are for physical reality.
  '
    p1 = p1 + dt * (k1p1 + k2p1) / 2
    p2 = p2 + dt * (k1p2 + k2p2) / 2
    dels1 = (p1 - 3) / 12 - s1
    dels2 = (p2 - 3) / 12 - s2
    If Abs(dels1) > deadband Then s1 = s1 + dels1
    If Abs(dels2) > deadband Then s2 = s2 + dels2
    mdot1 = mdot1 + dt * (k1mdot1 + k2mdot1) / 2
    mdot2 = mdot2 + dt * (k1mdot2 + k2mdot2) / 2
    tf1 = tf1 + dt * (k1tf1 + k2tf1) / 2
    tf2 = tf2 + dt * (k1tf2 + k2tf2) / 2
    tf3 = tf3 + dt * (k1tf3 + k2tf3) / 2
    tt1 = tt1 + dt * (k1tt1 + k2tt1) / 2
    tt2 = tt2 + dt * (k1tt2 + k2tt2) / 2
    If s1 < 0 Then s1 = 0
    If s1 > 1 Then s1 = 1
    If s2 < 0 Then s2 = 0
```

```
    If s2 > 1 Then s2 = 1
    If mdot1 < 0 Then mdot1 = 0
    If mdot2 < 0 Then mdot2 = 0
  Next i
  '
  '  Place tf3 into the array for delayed retrieval. "iput," the put index,
  '  has to be updated for the next sampling interval.
  '
  If (mdot1 + mdot2) > 0.01 Then
    tf(iput) = (mdot1 * t1inp + mdot2 * t2inp) / (mdot1 + mdot2)
  End If
  iput = iput + 1
  If iput = 2001 Then iput = 0        're start iput values at 12 O'clock
  '
  '  If enviro is active, then add noise and bias to the flow measurements
  '  and bias to the temperature measurement.
  '  here noise is removed completely with bias also neautralised
  m1bias = 0.95 * m1bias + 0.05 * m1biasb * enviro
  m2bias = 0.95 * m2bias + 0.05 * m2biasb * enviro
  m3bias = 0.95 * m3bias + 0.05 * m3biasb * enviro
  t1bias = 0.95 * t1bias + 0.05 * t1biasb * enviro
  t2bias = 0.95 * t2bias + 0.05 * t2biasb * enviro
  t3bias = 0.95 * t3bias + 0.05 * t3biasb * enviro
  mdot1meas = mdot1 * (1 + m1bias + (Sqr(-0.002 * Log(Rnd)) * Sin(2 * 3.14159 *
Rnd)) * enviro)
  mdot2meas = mdot2 * (1 + m2bias + (Sqr(-0.002 * Log(Rnd)) * Sin(2 * 3.14159 *
Rnd)) * enviro)
  mdot3meas = (mdot1 + mdot2) '* (1 + m3bias + 0 * (SQR(-.002 * LOG(RND)) *
SIN(2 * 3.14159 * RND)) * enviro)
  t1meas = tt1 '+ t1bias
  t2meas = tt2 '+ t2bias
  t3meas = tf3 '+ t3bias
End Sub

   Sub PROCINI()
   '
   '    Routine to initialize the process parameter values
   '
   enviro = 1             'environmental effects are on
   dt = t / 10           'integration and control periods, sec
   ap1 = 0.3016          'A for Process #1
   bp1 = 2.9576          'B for Process #1
   cp11b = 0.003979      'C #1 for Process #1, Base value
   cp12b = 0.01082       'C #2 for Process #1, Base value
   dp1 = 0.002327        'D for Process #1
   dpp1b = 30            'Differential Pressure for Process #1
```

```
    hp1 = 2              'Height of hydrostatic head Process #1
    tauvp1 = 1           'Valve TAU for Process #1
    ddpp1 = 0            'Deviation of Differential Pressure for Process #1
    dcp11 = 0            'Deviation of C #1 for Process #1
    dcp12 = 0            'Deviation of C #2 for Process #1
    power1 = 2           'value of power for valve #1 characteristic
    t1inpb = 100         'INlet Temperature Base value for Process #1
    ap2 = 0.3427
    bp2 = 3.3609
    cp21b = 0.008139
    cp22b = 0.0123
    dp2 = 0.01058
    dpp2b = 60
    hp2 = -1
    tauvp2 = 1.5
    ddpp2 = 0
    dcp21 = 0
    dcp22 = 0
    power2 = 2
    t2inpb = 20
    taut1 = 0.6          'Temperature sensor TAU for 1st lag***values changed
    taut2 = 0.4          'Temperature sensor TAU for 2nd lag
    taut3 = 0.3          'Temperature sensor TAU for 3rd lag
    tf1 = t2inpb         'Fictitious Temperature #1
    tf2 = t2inpb         'Fictitious Temperature #2
    tf3 = t2inpb         'Fictitious Temperature #3
    For i = 0 To 2000
      tf(i) = t2inpb     'array that holds the Fictitious Temperatures for delay
    Next i
    m1biasb = 0.1 - 0.2 * Rnd
    m2biasb = 0.1 - 0.2 * Rnd
    m3biasb = 0.1 - 0.2 * Rnd
    t1biasb = 2 - 4 * Rnd
    t2biasb = 2 - 4 * Rnd
    t3biasb = 2 - 4 * Rnd
    'mdot1 = 5  '<***here start the mdot's
    'mdot2 = 5
     o1 = 100
     o2 = 100
End Sub
```

The following Q- Basic code listing is from the *Hot and Cold water mixing* simulator for the case of *Without Noise*.

```
DECLARE SUB CLEAN ()
DECLARE SUB ATV (a$, time!, mode1!, mode2!, mdot3sp!, mdot3filt!, t3sp!, t3meas!,
o1!, o2!)
DECLARE SUB FILTINI ()
DECLARE SUB FILTER (mdot1meas!, mdot2meas!, mdot3meas!, mdot1filt!,
mdot2filt!, mdot3filt!)
DECLARE SUB DISPLAY (mode1, mode2, o1, o2, mdot1filt, mdot2filt, mdot3filt,
t1meas, t2meas, t3meas, mdot3sp, t3sp, theta)
DECLARE SUB OPERATOR (a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
DECLARE SUB EVAL (mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
DECLARE SUB CTLINI ()
DECLARE SUB PROCESS (o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas,
t1meas, t2meas, t3meas)
DECLARE SUB PLOTINI ()
DECLARE SUB CTL (mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
DECLARE SUB PLOT (o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas,
mdot3sp, t3sp)
DECLARE SUB PROCINI ()
   '
   '                      CONTROL.BAS
   '                  Spring 1998 CHENG-5xxx
   '         Dr.R.Russell Rhinehart, School of Chem. Engr. Oklahoma State U.
   '                       25 Dec 97
   '
   ' This program is a basis for CHENG-5xxx students to test their controllers.
   '
   ' The program models control valves, fluid flow, mixing of a hot and cold
   ' water in a pipe system, and flow and temperature measurement. It also
   ' contains a control subrouting for primitive PID T and F controllers.
   ' The students will write the code for various control strategys,
   ' filters, and goodness of control evaluations; tune their controllers;
   ' and explore the solutions for a variety of process events that cause
   ' control difficulty.
   '
   ' The program is structured so that each stage in the controller-process-
   ' evaluation system are written as subroutines. This MAIN program links and
   ' orders the execution of each subroutine.
```

```
'
' The MAIN program calls subroutine PROCESS to dynamically simulate the
' fluid mixing process for a time interval, t, of 0.1 seconds. PROCESS
' simulates the final element dynamics, as well as the ChEs view of the
' process behavior (fluid dynamics and mixing). It also adds measurement
' bias and process beavior drifts that have an ARMA stochastic behavior.
' It also adds measurement noise and valve "stick-tion".
'
' MAIN then calls subroutine FILT to filter noise from the measurements.
'
' MAIN then calls subroutine CTL, where, eventually students will write
' the code for the various controllers and control strategies. Presently
' CTL contains two independent PID controllers, one for T control (manipulating
' O1) and one for F control (manipulating O2).
'
' MAIN then calls subroutine EVAL, where, eventually students will write
' the code for the various goodness of control measures. Presently EVAL
' calculates T and F NISE.
'
' MAIN then calls subroutine PLOT to generate a strip chart display
' of the controlled and manipulated variables.
'
' Finally MAIN calls DISPLAY to refresh data on the screen.
'
' On operator demand (by keyboard touches) MAIN will call subroutine
' OPERATOR to execute the operator-initiated (student-initiated) changes.
' See subroutine OPERATOR to see what INKEY touches start which commands.
' One of these commands is to initiate ATV tuning, an automatic tuning for
' PID controllers.
'
' This sequence is then repeated. However, first MAIN initializes the
' devices, sets up common variables, and calls PLOTINI, PROCINI, and
' CTLINI to initialize the PLOT, PROCESS, and CTL subroutine variables.
'

DIM plotvmax(10), plotvmin(10), plotvrng(10), plotvar(10), plotyo(10), tf(2000)
COMMON SHARED plotvmax(), plotvmin(), plotvrng(), plotvar(), plotyo(), tf()
COMMON SHARED numvar, plottime, reference, horizon, plotx, plotxo, ploty, time
COMMON SHARED ap1, bp1, cp11b, cp12b, dp1, tauvp1
COMMON SHARED ap2, bp2, cp21b, cp22b, dp2, tauvp2
COMMON SHARED m1biasb, m2biasb, m3biasb, t1biasb, t2biasb, t3biasb
COMMON SHARED taut1, taut2, taut3, t1inpb, t2inpb, tf1, tf2, tf3
COMMON SHARED t, dt, timedelta
COMMON SHARED dpp1b, hp1, power1
COMMON SHARED dpp2b, hp2, power2
COMMON SHARED enviro
```

```
   COMMON SHARED lambda1, lambda2, lambda3
   COMMON SHARED kc1, taui1, taud1, kc2, taui2, taud2, detune
   COMMON SHARED which$, tune, dataout
   COMMON SHARED iset3, isdo1, isemdot3, isdo2, isenumber
   COMMON SHARED o1, o2
OPEN "F:\VBPROGS\rules2\Newruns\testdata.csv" FOR OUTPUT AS #1
'PRINT #1, "time", "theta", "t3meas", "t1meas", "t2meas", "mdot3meas", "mdot1meas",
"mdot2meas"
PRINT #1, "time, t1meas, mdot1meas, mdot2meas, t3meas"
SCREEN 12 'set-up screen for graphics, 640 X 350 x-y pixils, 82 X 25 x-y positions
RANDOMIZE ((TIMER - 12300) / 3)   'randomize the seed for the random number
generator
CLS
enviro = 1
tune = -1              'do not start with ATV tuning
dataout = -1           '**now start without data logging
   CALL FILTINI
   CALL CTLINI
   CALL PROCINI
   CALL PLOTINI

   FOR interval = 1 TO 600000
         time = interval * t
         IF time = 20 THEN
         dataout = 1
         END IF

         IF 20 * INT(time / 20) = time THEN

            IF time > 0 AND time < 200 THEN o1 = o1 - 10
            IF time > 200 AND time < 400 THEN o2 = o2 - 10
            IF time > 400 AND time < 600 THEN o1 = o1 + 10
            IF time > 600 AND time < 800 THEN o2 = o2 + 10

            IF time > 800 AND time < 1000 THEN o1 = o1 - 10
            IF time > 1000 AND time < 1200 THEN o2 = o2 - 10
            IF time > 1200 AND time < 1400 THEN o1 = o1 + 10
            IF time > 1400 AND time < 1600 THEN o2 = o2 + 10

            IF time > 1600 AND time < 1800 THEN o1 = o1 - 10
            IF time > 1800 AND time < 2000 THEN o2 = o2 - 10
            IF time > 2000 AND time < 2200 THEN o1 = o1 + 10
            IF time > 2200 AND time < 2400 THEN o2 = o2 + 10

            IF time = 820 THEN t1inpb = t1inpb + 40
            IF time = 1620 THEN t1inpb = t1inpb + 40
```

```
            END IF

            CALL PROCESS(o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas, t1meas,
t2meas, t3meas)
            a$ = INKEY$
            IF a$ <> "" THEN
                  CALL OPERATOR(a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
            END IF
            CALL FILTER(mdot1meas, mdot2meas, mdot3meas, mdot1filt, mdot2filt,
mdot3filt)
            IF tune = 1 THEN
              CALL ATV(a$, time, mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1,
o2)
            ELSE
              CALL CLEAN
            END IF
            CALL CTL(mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
            CALL PLOT(o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas,
mdot3sp, t3sp)
            CALL EVAL(mdot3sp, mdot3filt, t3sp, t3meas, o1, o2)
            CALL DISPLAY(mode1, mode2, o1, o2, mdot1filt, mdot2filt, mdot3filt,
t1meas, t2meas, t3meas, mdot3sp, t3sp, theta)
            IF dataout = 1 THEN
              IF timedelta * INT(time / timedelta) = time THEN  '****log on every
(timedelta) second
                  PRINT #1, time; ","; t1meas; ","; mdot1meas; ","; mdot2meas; ",";
t3meas; ","; theta
              END IF
            END IF
   NEXT interval
CLOSE #1
   '
   '   Variable definitions
   '
   ' plotvmax(10) maximum values of the plotted variables
   ' plotvmin(10) minimum values of the plotted variables
   ' plotvrng(10) calculated maximum minus minimum values, range of plotted variables
   ' plotvar(10)  values of the plotted variables
   ' plotyo(10)   pixel positions for the previous strip chart ordinate
   ' tf(200)      array that holds the values for the fictitious temperature
   ' numvar       number of variables plotted
   ' plottime     time argument for the plotting routine, same as time
   ' reference    time at the beginning of each strip chart sweep
   ' horizon      time window of the strip chart
   ' plotx        pixel position for the strip chart abscissa
```

```
' plotxo       value of the previous plotx pixel position
' ploty        pixel position for the strip chart ordinate
' time         simulated time, seconds
' ap1          "a" coefficient value for process #1, kg/s^2/kPa
' bp1          "b" coefficient value for process #1, kg/s^2/m
' cp11b         "c11" coefficient base value for process #1, kg/s^2/kg^2/min^2
' cp12b         "c12" coefficient base value for process #1, kg/s^2/kg^2/min^2
' dp1          "d" coefficient value for process #1, kg/s^2/kg^2/min^2
' tauvp1        time constant for process valve #1, seconds
' ap2          "a" coefficient value for process #2, kg/s^2/kPa
' bp2          "d" coefficient value for process #2, kg/s^2/m
' cp21b         "c21" coefficient base value for process #2, kg/s^2/kg^2/min^2
' cp22b         "c22" coefficient base value for process #2, kg/s^2/kg^2/min^2
' dp2          "d" coefficient value for process #2, kg/s^2/kg^2/min^2
' tauvp2        time constant for process valve #2, seconds
' taut1        time constant for first temperature lag, seconds
' taut2        time constant for second temperature lag, seconds
' taut3        time constant for third temperature lag, seconds
' t1inpb        process stream #1 inlet temperature base value, centigrade
' t2inpb        process stream #2 inlet temperature base value, centigrade
' tf1          first lagged temperature at the fictitious sensor, centigrade
' tf2          second lagged temperature at the fictitious sensor, centigrade
' tf3          third lagged temperature at the fictitious sensor, centigrade
' t            process sampling time and control period, seconds
' dt           process integration time step, seconds
' dpp1b          driving pressure drop base case for stream #1, kPa
' hp1          elevation head for stream #1, m
' power1         power coefficient for valve #1 characteristic
' dpp2b          driving pressure drop base case for stream #2, kPa
' hp2          elevation head for stream #2, m
' power2         power coefficient for valve #2 characteristic
' enviro        coefficient to toggle environmental effects on/off, 1 if on, 0 if off
' time         simulated time, seconds
' interval     controller sampling period and process integration time step, seconds
' o1           output of controller #1, % of full scale
' o2           output of controller #2, % of full scale
' s1           valve #1 stem position, fraction open
' s2           valve #2 stem position, fraction open
' mdot1meas    measured value of flow rate of stream #1, kg/min
' mdot2meas    measured value of flow rate of stream #2, kg/min
' mdot3meas    measured value of combined flow rate, kg/min
' t3meas       measured value of mixed temperature, centigrade
' a$           variable to store the value of INKEY$, alpha-numeric string
' INKEY$       BASIC function that inputs a keyboard hit, alpha-numeric string
' mode1        mode of controller #1, 1 if AUTO, 0 if MAN
' mode2        mode of controller #2, 1 if AUTO, 0 if MAN
```

```
   ' mdot3sp      set point for total flow rate, kg/min
   ' t3sp         set point for mixed temperature, centigrade
   ' lambda1      filter factor for the first-order noise filter on mdot1meas
   ' lambda2      filter factor for the first-order noise filter on mdot2meas
   ' lambda3      filter factor for the first-order noise filter on mdot3meas
   ' kc1          controller 1 gain, %output / kg/min
   ' taui1        controller 1 integral time, seconds
   ' taud1         controller 1 derivative time, seconds
   ' kc2          controller 2 gain, %output / centigrade
   ' taui2        controller 2 integral time, seconds
   ' taud2         controller 2 derivative time, seconds
   ' which$        variable that defines which controller is being ATV tested
   ' tune          variable to indicate whether ATV tuning is desired
   ' dataout       variable to indicate whether data is to be recorded in the output file
   ' iset3         integral of the squared error for t3meas
   ' isdo1         integral of the squared change in output of controller 1
   ' isemdot3      integral of the squared error for mdot3filt
   ' isdo2         integral of the squared change in output of controller 2
   ' isenumber     count to normalize the ise and isdo
   ' m*bias        bias on flow rate * measurement
   ' m*biasb       base level for the bias on flow rate * measurement
   ' t*bias        bias on temperature * measurement
   ' t*biasb       base level for the bias on temperature * measurement

SUB ATV (a$, time, mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2) STATIC
    '
    '   ATV tuning
    '   NOTE 1 - I think that I used the ZN Ultimate rules for interacting for non-
interacting PID control
    '   NOTE 2 - need a better way to detect zero crossing in the presence of noise
    '
    IF a$ = "a" OR a$ = "A" THEN   'you just got here, initialize the factors
        start = 0                 'start time for the ATV test
        e = 0                    'deviation from atvtarg
        eold = 0                  'old deviation
        emax = 0                   'maximum CV deviation from atvtarg in a cycle
        emin = 0                   'minimum CV deviation from atvtarg in a cycle
        LOCATE 15, 1
        INPUT "Do you wish to implement ATV tuning on the O1-T3 loop (1) or O2-F3
(2)"; which$
        LOCATE 15, 1
        PRINT "                                                          "
        '
        '   initialize the atvtarg and set the controller to manual
        '
        IF which$ = "1" THEN        'O1-T3 loop was chosen
```

```
            atvtarg = t3meas         'initialize the atvtarg with the first CV value
            mode1 = 0                'set the controller to MAN
          LOCATE 14, 1
          PRINT USING "atvtarg = ###.# C"; atvtarg
          ELSE                       'O2-F3 loop was chosen
            atvtarg = mdot3filt
            mode2 = 0
          LOCATE 14, 1
          PRINT USING "atvtarg = ###.# kg/min"; atvtarg
          END IF
      END IF
      '
      '   ATV test controller #1
      '
      IF which$ = "1" THEN
          IF start = 0 THEN          'if this is the first time initialize
            start = time             'start time for test
            switch = time            'time when output was switched
            relay = 20               'output step size (high - low)
            o1 = o1 + relay / 2      'make the first output step, up, by 1/2 of the relay
            LOCATE 15, 1
            PRINT "ATV initiated on O1-T3 loop, T3 controller is overridden"
          END IF
          IF time - start > 15 THEN       'hold the first bump for 15 seconds
            e = atvtarg - t3meas         'then calculate the deviation
            IF e > emax THEN emax = e    'set emax
            IF e < emin THEN emin = e    'set emin
            LOCATE 14, 1
            PRINT USING "atvtarg = ###.# C   emax = ###.### C    emin = ###.### C     ";
atvtarg; emax; emin
            IF e * eold <= 0 THEN        'if the error changed sign, the atvtarg was crossed
              IF e < 0 THEN              'if the error is negative
                  o1 = o1 - relay        'then step the output down by 1/1 relay
              END IF
              IF e > 0 THEN              'if the error is positive, then a cycle had finished
                  o1 = o1 + relay        'then step the output up by 1/1 relay
                  pu = time - switch     'calculate the ultimate period
                  ku = 4 * relay / (emax - emin) / 3.14159    'and the ultimate gain
                  LOCATE 15, 1
                  PRINT USING "ATV O1-T3 in cycling mode. Ult. P. = ###.## sec    Ult.
Kc = ###.## %/C"; time - switch; 4 * relay / (emax - emin) / 3.14159
                  LOCATE 16, 1
                  PRINT USING "(Kc=###.#)  (Kc=###.#  taui=###.#)  (Kc=###.#
taui=###.#  taud=###.#)"; .5 * ku; .45 * ku; .83 * pu; .59 * ku; .5 * pu; .125 * pu
                  o1 = o1 + .25 * relay * (emax + emin) / (emax - emin) 'shift o1 for
symmetry
```

```
                emax = 0              'reset emax for the next cycle
                emin = 0              'reset emin for the next cycle
                switch = time         'reset switch for the next cycle
            END IF
          END IF
          eold = e
        END IF
    ELSE                              'which = 2, ATV the flow loop
        IF start = 0 THEN
          start = time
          switch = time
          relay = 30
          o2 = o2 + relay / 2
          LOCATE 15, 1
          PRINT "ATV initiated on O2-F3 loop, F3 controller is overridden"
        END IF
        IF time - start > 5 THEN
          e = atvtarg - mdot3filt
          IF e > emax THEN emax = e
          IF e < emin THEN emin = e
          LOCATE 14, 1
          PRINT USING "atvtarg = ###.# kg/min    emax = ###.### kg/min    emin =
###.### kg/min"; atvtarg; emax; emin
          IF e * eold <= 0 THEN
            IF e < 0 THEN
                o2 = o2 - relay
            END IF
            IF e > 0 THEN
                o2 = o2 + relay
                pu = time - switch
                ku = 4 * relay / (emax - emin) / 3.14159
                LOCATE 15, 1
                PRINT USING "ATV O2-F3 in cycling mode. Ult. P. = ###.## sec    Ult.
Kc = ###.## %/kg/min"; pu; ku
                LOCATE 16, 1
                PRINT USING "(Kc=###.#)  (Kc=###.#  taui=###.#)  (Kc=###.#
taui=###.#  taud=###.#)"; .5 * ku; .45 * ku; .83 * pu; .59 * ku; .5 * pu; .125 * pu
                o2 = o2 + .25 * relay * (emax + emin) / (emax - emin)'shift o2 for
symmetry
                emax = 0
                emin = 0
                switch = time
            END IF
          END IF
          eold = e
        END IF
```

```
    END IF
END SUB

   SUB CLEAN
   '
   '   clean the ATV messages from the screen
   '
   LOCATE 14, 1
   PRINT "                                                    "
   LOCATE 15, 1
   PRINT "                                                    "
   LOCATE 16, 1
   PRINT "                                                      "
END SUB

   SUB CTL (mode1, mode2, mdot3sp, mdot3filt, t3sp, t3meas, o1, o2) STATIC
   '
   '  Presently there are two independent, standard PID controllers here.
   '  One controls T3 by manipulating O1, the output to valve 1, the hot water
   '  valve. The other controls F3 by manipulating O2, the output to valve 2,
   '  the cold water valve. Because the process is interactive (O1 affects both
   '  T3 and F3), the controllers use the "BLT" method of detuning them jointly,
   '  after they were independently tuned by "ATV" for "QAD" process behavior.
   '
   '
   '  Temperature controller
   '
      IF mode1 = 1 THEN                        'temperature controller in AUTO
         e1 = t3sp - t3meas                    'reverse acting
         bias1 = bias1 + t * kc1 * e1 / taui1 / detune ^ 2 'adjustable bias, rectangle rule
         eant1 = e1 - taud1 * (t3meas - t3old) / t   'anticipated error, D-on-X
         t3old = t3meas
         o1 = kc1 * eant1 / detune + bias1          'proportional plus bias
         IF o1 > 110 THEN                         'anti-windup provision
           o1 = 110
           bias1 = o1 - kc1 * eant1 / detune
         END IF
         IF o1 < -10 THEN                         'anti-windup provision
           o1 = -10
           bias1 = o1 - kc1 * eant1 / detune
         END IF
      ELSE                                  'temperature controller in MAN
         t3sp = t3meas                          'setpoint tracking, bumpless transfer
         t3old = t3meas                         'no D spike, bumpless transfer
         bias1 = o1                           'bias tracking, bumpless transfer
      END IF
```

```
    '
    '  Flow controller
    '
      IF mode2 = 1 THEN                         'flow controller in AUTO
          e2 = mdot3sp - mdot3filt                'reverse acting
          bias2 = bias2 + t * kc2 * e2 / taui2 / detune ^ 2     'adjustable bias, rectangle rule
          eant2 = e2 - taud2 * (mdot3filt - mdot3old) / t   'anticipated error, D-on-X
          mdot3old = mdotfilt
          o2 = kc2 * eant2 / detune + bias2         'proportional plus bias
          IF o2 > 110 THEN                        'anti-windup provision
            o2 = 110
            bias2 = o2 - kc2 * eant2 / detune
          END IF
          IF o2 < -10 THEN                        'anti-windup provision
            o2 = -10
            bias2 = o2 - kc2 * eant2 / detune
          END IF
      ELSE                                  'flow controller in MAN
          mdot3sp = mdot3filt                     'setpoint tracking, bumpless transfer
          mdot3old = mdot3filt
          bias2 = o2                            'bias tracking, bumpless transfer
      END IF
END SUB

    SUB CTLINI STATIC
    '
    '  Initial controller settings go here  static makes them constant
    '
    t = .1
    timedelta = 1  'log every timedelta seconds
    mode1 = 0      'controller 1 is in manual
    mode2 = 0      'controller 2 is in manual
    kc1 = 2        '% / centigrade
    taui1 = 12     'seconds
    taud1 = 3      'seconds
    kc2 = 8        '% / kg/min
    taui2 = 2.5    'seconds
    taud2 = 0      'seconds
    detune = 1     'dimensionless
END SUB

   SUB DISPLAY (mdot1, mdot2, o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas,
t2meas, t3meas, mdot3sp, t3sp, theta)
    '
    '   subroutine to display variables and status on the screen
    '
```

```
   LOCATE 17, 1
   PRINT USING " theta = ###.#####  time =   ####"; theta; time
   PRINT USING "    o1 = ###.#       o2 = ###.#"; o1; o2
   PRINT USING "F1filt = ###.#   F2filt = ###.#"; mdot1filt; mdot2filt
   PRINT USING "T1meas = ###.###  T2meas = ###.#"; t1meas; t2meas
   PRINT USING "T3meas = ###.#   F3filt = ###.#"; t3meas; mdot3filt
   PRINT USING "T3sp   = ###.#   F3sp   = ###.#"; t3sp; mdot3sp
   PRINT USING "kc1=##.# taui1=##.# taud1=##.#    kc2=##.# taui2=##.# taud2=##.#
detune=#.#"; kc1; taui1; taud1; kc2; taui2; taud2; detune
END SUB

   SUB EVAL (mdot3sp, mdot3filt, t3sp, t3meas, o1, o2) STATIC
   '
   '   measures of control goodness are calculated here
   '
   isenumber = isenumber + 1
   iset3 = iset3 + t * (t3sp - t3meas) ^ 2
   isdo1 = isdo1 + t * (o1 - o1old) ^ 2
   o1old = o1
   niset3 = iset3 / (isenumber * t)
   nisdo1 = isdo1 / (isenumber * t)
   isemdot3 = isemdot3 + t * (mdot3sp - mdot3filt) ^ 2
   isdo2 = isdo2 + t * (o2 - o2old) ^ 2
   o2old = o2
   nisemdot3 = isemdot3 / (isenumber * t)
   nisdo2 = isdo2 / (isenumber * t)
   '
   '  LOCATE Y,X locates the beginning of the subsequent print statement
   '  at Y text rows down from the top of the screen and X text columns to
   '  the right from the left of the screen. The screen is 22 rows by 75
   '  columns.
   '  PRINT USING "    ";   is a formatted print statement. # marks locations
   '  for numerical values.
   '
   LOCATE 21, 35
   PRINT USING " rmset = #.####^^^^    rmsef = #.####^^^^"; SQR(niset3);
SQR(nisemdot3)
   LOCATE 22, 35
   PRINT USING "rmsdo1 = #.####^^^^   rmsdo2 = #.####^^^^"; SQR(nisdo1);
SQR(nisdo2)
END SUB

SUB FILTER (mdot1meas, mdot2meas, mdot3meas, mdot1filt, mdot2filt, mdot3filt)
STATIC
   '
   '  subroutine to first-order filter the noisy process measurements
```

```
    '  lambda = 1-exp(T/taufilt)
    '
     mdot1filt = lambda1 * mdot1meas + (1 - lambda1) * mdot1filt
     mdot2filt = lambda2 * mdot2meas + (1 - lambda2) * mdot2filt
     mdot3filt = lambda3 * mdot3meas + (1 - lambda3) * mdot3filt
END SUB

    SUB FILTINI STATIC
    '
    '  subroutine to initialize the filter coefficients
    '
     lambda1 = .2
     lambda2 = .2
     lambda3 = .2
END SUB

SUB OPERATOR (a$, mode1, mode2, o1, o2, mdot3sp, t3sp)
     '
     '  operator initiated action is made here
     '
     iset3 = 0           'Reset the goodness of control measures
     isdo1 = 0           '  "
     isemdot3 = 0        '  "
     isdo2 = 0           '  "
     isenumber = 0       '  "
     IF a$ = "q" OR a$ = "Q" THEN
          CLOSE #1
          STOP   'key in "q" to stop the program
     END IF
     IF a$ = "a" OR a$ = "A" THEN tune = -tune
     IF a$ = "-" THEN t1inpb = t1inpb - 5 '***add or subtract input temperature
     IF a$ = "+" THEN t1inpb = t1inpb + 5
     IF a$ = "9" OR a$ = "L" THEN dataout = -dataout
     IF a$ = "n" OR a$ = "N" THEN        'key in "n" to toggle enviro and disturbances
          IF enviro = 1 THEN
            enviro = 0
          ELSE
            enviro = 1
          END IF
     END IF
     IF a$ = "1" THEN                    'key in "1" to toggle controller 1 MAN-AUTO
          IF mode1 = 1 THEN
            mode1 = 0
          ELSE
            mode1 = 1
          END IF
```

```
    END IF
    IF a$ = "2" THEN                'key in "2" to toggle controller 2 MAN-AUTO
        IF mode2 = 1 THEN
          mode2 = 0
        ELSE
          mode2 = 1
        END IF
    END IF
    '
    '  change output if in manual
    '
    IF a$ = "3" AND mode1 = 0 THEN o1 = o1 - 5  'key in "3" lower o1 in MAN
    IF a$ = "#" AND mode1 = 0 THEN o1 = o1 + 5  'key in "#" raise o1 in MAN
    IF a$ = "4" AND mode2 = 0 THEN o2 = o2 - 5  'key in "4" lower o2 in MAN
    IF a$ = "$" AND mode2 = 0 THEN o2 = o2 + 5  'key in "$" raise o2 in MAN
    '
    '  limit output to between -10 and 110 %
    '
    IF o1 > 110 THEN o1 = 110
    IF o1 < -10 THEN o1 = -10
    IF o2 > 110 THEN o2 = 110
    IF o2 < -10 THEN o2 = -10
    '
    '  change setpoint if in automatic - method 1:
    '
    IF a$ = "5" AND mode1 = 1 THEN t3sp = t3sp - 2  'key in "5" lower tsp in AUTO
    IF a$ = "%" AND mode1 = 1 THEN t3sp = t3sp + 2  'key in "%" raise tsp in AUTO
    IF a$ = "6" AND mode2 = 1 THEN mdot3sp = mdot3sp - 2  'key in "6" lower mdotsp
in AUTO
    IF a$ = "^" AND mode2 = 1 THEN mdot3sp = mdot3sp + 2  'key in "^" raise mdotsp
in AUTO
    '  change setpoint if in automatic - method 2:
    '
    IF a$ = "s" OR a$ = "S" THEN
        LOCATE 16, 35
        PRINT "Enter one of these setpoints:"
        LOCATE 17, 35
        PRINT "t3, f3"
        LOCATE 18, 35
        INPUT "Which value do you wish to change"; b$
        IF b$ = "t3" AND mode1 = 1 THEN
          LOCATE 19, 35
          INPUT "Enter t3sp value, C"; t3sp
        END IF
        IF b$ = "f3" AND mode2 = 1 THEN
          LOCATE 19, 35
```

```
        INPUT "Enter mdot3sp value, kg/min"; mdot3sp
      END IF
'  erase on-screen trash
      LOCATE 16, 35
      PRINT "                                  "
      LOCATE 17, 35
      PRINT "                                  "
      LOCATE 18, 35
      PRINT "                                  "
      LOCATE 19, 35
      PRINT "                                  "
END IF
'
'  if tuning is desired
'
IF a$ = "t" OR a$ = "T" THEN
      LOCATE 16, 35
      PRINT "Enter one of these parameters:"
      LOCATE 17, 35
      PRINT "kc1, taui1, taud1, kc2, taui2, taud2, detune"
      LOCATE 18, 35
      INPUT "Which value do you wish to change"; b$
      IF b$ = "kc1" THEN
        LOCATE 19, 35
        INPUT "Enter kc1 value, %/C"; kc1
      END IF
      IF b$ = "taui1" THEN
        LOCATE 19, 35
        INPUT "Enter taui1 value, s"; taui1
      END IF
      IF b$ = "taud1" THEN
        LOCATE 19, 35
        INPUT "Enter taud1 value, s"; taud1
      END IF
      IF b$ = "kc2" THEN
        LOCATE 19, 35
        INPUT "Enter kc2 value, %/kg/min"; kc2
      END IF
      IF b$ = "taui2" THEN
        LOCATE 19, 35
        INPUT "Enter taui2 value, s"; taui2
      END IF
      IF b$ = "taud2" THEN
        LOCATE 19, 35
        INPUT "Enter taud2 value, s"; taud2
      END IF
```

```
        IF b$ = "detune" THEN
          LOCATE 19, 35
          INPUT "Enter detune value"; detune
        END IF
    '
    '  erase on-screen trash
    '
        LOCATE 16, 35
        PRINT "                                    "
        LOCATE 17, 35
        PRINT "                                    "
        LOCATE 18, 35
        PRINT "                                    "
        LOCATE 19, 35
        PRINT "                                    "
    END IF
END SUB

   SUB PLOT (o1, o2, mdot1filt, mdot2filt, mdot3filt, t1meas, t2meas, t3meas, mdot3sp,
t3sp) STATIC
      '
      '   This routine plots the scaled variables on a strip chart display
      '
      '                    PLOT.BAS
      '             R. Russell Rhinehart Company
      '                 10 October 1994
      '
      '   After calculating the variable values assign them to the plot variables
      '
      plottime = time            'simulated time, seconds
      plotvar(1) = o1            'output of controller 1, %
      plotvar(2) = o2            'output of controller 2, %
      plotvar(3) = mdot1filt     'filtered flow rate 1, kg/min
      plotvar(4) = mdot2filt     'filtered flow rate 2, kg/min
      plotvar(5) = mdot3filt     'filtered total flow rate, kg/min
      plotvar(6) = t1meas        'measured temperature, centigrade
      plotvar(7) = t2meas        'measured temperature, centigrade
      plotvar(8) = t3meas        'measured temperature, centigrade
      plotvar(9) = mdot3sp       'flow 3 setpoint, kg/min
      plotvar(10) = t3sp         'temperature 3 setpoint, centigrade
      '
      '  Plot routine
      '
      IF plottime - reference >= horizon THEN       ' locate the x position
          reference = reference + horizon
          plotxo = 50
```

```
        LINE (plotxo, 20)-(plotxo, 160), 15
        LINE (plotx, 20)-(plotx, 160), 15
        LINE (plotx, 161)-(plotx, 168), 14
    END IF
    plotx = 50 + INT(.5 + 580 * (plottime - reference) / horizon)
    IF 50 + 58 * INT((plotx - 50) / 58) = plotx THEN LINE (plotx, 20)-(plotx, 160), 15
    LINE (plotx + 1, 20)-(plotx + 1, 160), 14
    LINE (plotx, 161)-(plotx, 168), 0
    LINE (plotx - 1, 161)-(plotx - 1, 168), 14
    FOR plotyy = 20 TO 160 STEP 14
    LINE (plotx, plotyy)-(plotx + 1, plotyy), 15
    NEXT plotyy
    FOR ploti = 1 TO numvar
    ploty = 160 - 140 * (plotvar(ploti) - plotvmin(ploti)) / plotvrng(ploti)
    IF ploty < 20 THEN ploty = 20
    IF ploty > 160 THEN ploty = 160
    LINE (plotxo, plotyo(ploti))-(plotx, ploty), ploti
    plotyo(ploti) = ploty
    NEXT ploti
    plotxo = plotx
END SUB

  SUB PLOTINI STATIC
    '
    '   This routine initializes the strip chart display plot subroutine
    '
    '                       PLOT.BAS
    '               R. Russell Rhinehart Company
    '                    10 October 1994
    '
    ' initialize the plotting variables
    '
    plotxo = 50             ' time = 0  position on the screen
    numvar = 10               ' number of variables to plot, maximum = 10
    horizon = 60            ' strip chart horizon, seconds
    plotvmax(1) = 100       ' maximum value for controller #1 output, %
    plotvmin(1) = 0         ' minimum value for controller #1 output, %
    plotvmax(2) = 100       ' maximum value for controller #2 output, %
    plotvmin(2) = 0         ' minimum value for controller #2 output, %
    plotvmax(3) = 30        ' maximum value for flow rate #1, kg/min
    plotvmin(3) = 0         ' minimum value for flow rate #1, kg/min
    plotvmax(4) = 30        ' maximum value for flow rate #2, kg/min
    plotvmin(4) = 0         ' minimum value for flow rate #2, kg/min
    plotvmax(5) = 60        ' maximum value for total flow rate, kg/min
    plotvmin(5) = 0         ' minimum value for total flow rate, kg/min
    plotvmax(6) = 100       ' maximum value for mixed temperature, C
```

```
    plotvmin(6) = 0         ' minimum value for mixed temperature, C
    plotvmax(7) = 100       ' maximum value for temperature 1, C
    plotvmin(7) = 0         ' minimum value for temperature 1, C
    plotvmax(8) = 100       ' maximum value for temperature 2, C
    plotvmin(8) = 0         ' minimum value for temperature 2, C
    plotvmax(9) = 60        ' maximum value for flow3 setpoint, kg/min
    plotvmin(9) = 0         ' minimum value for flow3 setpoint, kg/min
    plotvmax(10) = 100      ' maximum value for temperature 3 setpoint, C
    plotvmin(10) = 0        ' minimum value for temperature 3 setpoint, C
                            ' repeat for all plotted variables
    reference = 0           ' time of the beginning of each strip chart
    '
    '   Initialize the graph
    '   (setup lables, background, grid lines, and initial points)
        LOCATE 1, 1
        PRINT USING "PV's (fraction of full scale) VERSUS TIME (fraction of
window = ####.# seconds)"; horizon
        FOR plotj = 0 TO 1 STEP .5          ' lable the y axis
        ploty = 2 + 10 * plotj
        LOCATE ploty, 1
        PRINT USING "#.##"; 1 - plotj
        NEXT plotj
        FOR ploti = 0 TO 1.01 STEP .1       ' lable the x axis
        plotx = 6 + 71 * ploti
        LOCATE 13, plotx
        PRINT USING "#.##"; ploti;
        NEXT ploti
        LINE (40, 13)-(640, 168), 14, BF    ' fill in the background
        FOR plotyy = 20 TO 160 STEP 14      ' draw the horizontal grid
        LINE (50, plotyy)-(630, plotyy), 15
        NEXT plotyy
        FOR plotxx = 50 TO 630 STEP 58      ' draw the vertical grid
        LINE (plotxx, 20)-(plotxx, 160), 15
        NEXT plotxx
        FOR ploti = 1 TO numvar             ' calculate the plot variable
                                            ' ranges and initial locations
        plotvrng(ploti) = plotvmax(ploti) - plotvmin(ploti)
        ploty = 160 - 140 * (plotvar(ploti) - plotvmin(ploti)) / plotvrng(ploti)
        IF ploty < 20 THEN ploty = 20
        IF ploty > 160 THEN ploty = 160
        plotyo(ploti) = ploty
    NEXT ploti
END SUB

  SUB PROCESS (o1, o2, s1, s2, mdot1meas, mdot2meas, mdot3meas, t1meas, t2meas,
t3meas) STATIC
```

```
'  Subroutine to model the flow rates and temperatures. There are several
'  sections to this routine. First, if enviro is active, stochastic models
'  are used to change the flow rate driving pressures, flow pressure loss
'  coefficients, and inlet stream temperatures. Also, if enviro is active,
'  control valve action is subject to "sticktion." Next, the ODEs that
'  dynamically model the valve stem positions, and the coupled ODEs that
'  dynamically model the flow rates and mixture temperature are solved
'  using the second order Runge-Kutta method. Since the ODE-modeled
'  temperature is the mixing point temperature, the temperature values are
'  placed in an array so that the transport-delayed value can be used for
'  the fluid temperature at the sensor. Since the transport delay is
'  variable, the how-far-back-in-the-array index, nt, is calculated from
'  the transport delay, theta. The "clock" concept is used for efficient
'  array management. The temperature sensor is modeled as a third order ODE.
'  Finally, noise is added to the flow rate measurement to simulate orifice
'  turbulence noise.
'  if enviro is active then add drift and spikes to the pressure drops
ddpp1 = .999 * ddpp1 + .015 * dpp1b * (RND - .5) * enviro    'drift
IF RND < .01 THEN spike1 = 50 * (RND - .5) * enviro          'spike
spike1 = .9 * spike1                                         'fade the spike
dpp1 = dpp1b '+ ddpp1 + spike1 <<<<<<*****making sure no spikes
ddpp2 = .999 * ddpp2 + .015 * dpp2b * (RND - .5) * enviro    'drift
IF RND < .01 THEN spike2 = 50 * (RND - .5) * enviro          'spike
spike2 = .9 * spike2                                         'fade the spike
dpp2 = dpp2b '+ ddpp2 + spike2 <<<<<<<<******ditto
'
'   if enviro is active then add drift to the flow pressure loss factors
'****here i made sure again that no drift is there
dcp11 = .999 * dcp11 + .015 * cp11b * (RND - .5) * enviro    'drift
cp11 = cp11b '+ dcp11
dcp12 = .999 * dcp12 + .015 * cp12b * (RND - .5) * enviro    'drift
cp12 = cp12b '+ dcp12
dcp21 = .999 * dcp21 + .015 * cp21b * (RND - .5) * enviro    'drift
cp21 = cp21b '+ dcp21
dcp22 = .999 * dcp22 + .015 * cp22b * (RND - .5) * enviro    'drift
cp22 = cp22b '+ dcp22
'   if enviro is active then add drift to the inlet temperatures
' ***ditto
dt1in = .999 * dt1in + .015 * t1inpb * (RND - .5) * enviro   'drift
t1inp = t1inpb '+ dt1in
dt2in = .999 * dt2in + .015 * t2inpb * (RND - .5) * enviro   'drift
t2inp = t2inpb '+ dt2in
'  If enviro is active then add "sticktion" hysteresis to the valves.
'  Deadband is the amount of change in valve position the controller must
'  call for before the valve stem will move. Here, deadband is either 0 %
'  or 2.5 %. Dels1 and dels2 are the valve stem position changes that the
```

```
  '  controller wants. Note: If sticktion is present, and the valve position
  '  is 2 % open, and the controller wants it closed (o = 0 %), then the valve
  '  will stay at 2 % open! This is real. To fix it, controllers are designed
  '  so that their output goes from -10 % to 110 %, or so. Ideally the 0-100 %
  '  controller output is converted to a 4-20 mA d.c. current "signal" then to
  '  a 3-15 psig pneumatic "signal" which operates the valve. Ideally the stem
  '  position goes from 0 to 1 as the pressure goes from 3 to 15 psig. Allowing
  '  the controller output to range from -10 to 110 %, ideally causes the
  '  pneumatic signal to range from 1.8 to 16.2 psig which, hopefully, will
  '  overcome both sticktion and calibration errors in the D/A and i/p devices,
  '  and, thereby, allow the valve to fully close and to fully open.
  '
    deadband = .025 * enviro * 0              'deadband<<****making sure it is 0
    current1 = 4 + o1 * 16 / 100             'i1 from A/D conversion of o1
    current2 = 4 + o2 * 16 / 100             'i2 from A/D conversion of o2
    p1targ = 3 + (current1 - 4) * 12 / 16     'p1 target from i/p conversion of i1
    p2targ = 3 + (current2 - 4) * 12 / 16     'p2 target from i/p conversion of i2
  '
  '  In the following segment of code, the ODEs are solved using a
  '  second-order Rung-Kutta method with an integration time step that
  '  is one tenth of the control interval (dt = t/10).
  '
  '  Calculate the R-K k1s for p1, p2, mdot1, mdot2, tf1, tf2, and tf3.
  '  The IF statements either allow for sticktion or prevent numerical
  '  overflow. If the valves are nearly closed, then f1 or f2 are extremely
  '  small, and their contributions to the Ks are large negative. The -20
  '  is a relatively large negative value.
  '
    FOR i = 1 TO 10
  '
  '  Calculate the transport delay from the mixing point to the temperature
  '  sensor 1.06 meters down stream. Then, nt, the nearest integer number of
  '  sample intervals backward in the clock array. Then, ifind, the array
  '  location of that transport-delayed temperature. Note, this deadtime
  '  delayed temperature is the influence for the third-order lagged sensor
  '  temperature.
  '
      SHARED theta
      IF (mdot1 + mdot2) > .1 THEN          'if mdot total is greater than the minimum
          theta = 80 / (mdot1 + mdot2)      '****calculate transport delay doubled the
value of Lt from 20 to 80
      ELSE
          theta = 800                   'limit delay to maximum allowed by tf(200)
      END IF
      'OPEN "c:theta.dat" FOR OUTPUT AS #2
      'PRINT #2, theta
```

```
    'CLOSE #2
    nt = INT(theta / t + .5)              'Number of Time intervals in delay
    IF nt > ntold + 1 THEN nt = ntold + 1   'can't sample fluid past the sensor
    IF nt > 1999 THEN nt = 1999           'can't sample around the tf(200) "clock"
    ntold = nt
    ifind = iput - nt                     'calculate the find location
    IF ifind < 0 THEN ifind = ifind + 2001  'increment it if it passes 12 O'clock
    '
    '  calculate the R-K k1s
    '
    k1p1 = (p1targ - p1) / tauvp1  'rate of change of p1, now, due to p1targ
    k1p2 = (p2targ - p2) / tauvp2  'rate of change of p2, now, due to p2targ
    f1 = s1 ^ power1               'inherrent valve characteristic from stem
    IF f1 > .0001 THEN
        k1mdot1 = ap1 * dpp1 + bp1 * hp1 - cp11 * mdot1 ^ 2 - cp12 * (mdot1 +
mdot2) ^ 2 - dp1 * mdot1 ^ 2 / f1 ^ 2
    ELSE
        k1mdot1 = -20
    END IF
    IF k1mdot1 < -20 THEN k1mdot1 = -20
    f2 = s2 ^ power2               'inherent valve characteristic from stem
    IF f2 > .0001 THEN
        k1mdot2 = ap2 * dpp2 + bp2 * hp2 - cp21 * mdot2 ^ 2 - cp22 * (mdot1 +
mdot2) ^ 2 - dp2 * mdot2 ^ 2 / f2 ^ 2
    ELSE
        k1mdot2 = -20
    END IF
    IF k1mdot2 < -20 THEN k1mdot2 = -20
    k1tf1 = (tf(ifind) - tf1) / taut1
    k1tf2 = (tf1 - tf2) / taut2
    k1tf3 = (tf2 - tf3) / taut3
    k1tt1 = (t1inp - tt1) / 10
    k1tt2 = (t2inp - tt2) / 10

  '  Use the k1s to estimate where the state variables might go.
  '  The h addended to the state variable indicates Hypothesized.
  '  The limits are for physical reality.

    p1h = p1 + dt * k1p1
    p2h = p2 + dt * k1p2
    dels1h = (p1h - 3) / 12 - s1   'change in s1 that the p1h would make w/o sticktion
    dels2h = (p2h - 3) / 12 - s2   'change in s2 that the p2h would make w/o sticktion
    IF ABS(dels1h) > deadband THEN s1h = s1 + dels1h   's1 only changes if p1
overcomes sticktion
    IF ABS(dels2h) > deadband THEN s2h = s2 + dels2h   's2 only changes if p2
overcomes sticktion
```

```
      mdot1h = mdot1 + dt * k1mdot1
      mdot2h = mdot2 + dt * k1mdot2
      tf1h = tf1 + dt * k1tf1
      tf2h = tf2 + dt * k1tf2
      tf3h = tf3 + dt * k1tf3
      tt1h = tt1 + dt * k1tt1
      tt2h = tt2 + dt * k1tt2
      IF s1h < 0 THEN s1h = 0
      IF s1h > 1 THEN s1h = 1
      IF s2h < 0 THEN s2h = 0
      IF s2h > 1 THEN s2h = 1
      IF mdot1h < 0 THEN mdot1h = 0
      IF mdot2h < 0 THEN mdot2h = 0

   ' Calculate the R-K k2s for s1, s2, mdot1, mdot2, tf1, tf2, and tf3.
   ' The IF statements either allow for sticktion or prevent numerical overflow.

      k2p1 = (p1targ - p1h) / tauvp1
      k2p2 = (p2targ - p2h) / tauvp2
      f1h = s1h ^ power1
      IF f1h > .0001 THEN
          k2mdot1 = ap1 * dpp1 + bp1 * hp1 - cp11 * mdot1h ^ 2 - cp12 * (mdot1h +
mdot2h) ^ 2 - dp1 * mdot1h ^ 2 / f1h ^ 2
      ELSE
           k2mdot1 = -20
      END IF
      IF k2mdot1 < -20 THEN k2mdot1 = -20
      f2h = s2h ^ power2
      IF f2h > .0001 THEN
           k2mdot2 = ap2 * dpp2 + bp2 * hp2 - cp21 * mdot2h ^ 2 - cp22 * (mdot1h +
mdot2h) ^ 2 - dp2 * mdot2h ^ 2 / f2h ^ 2
      ELSE
           k2mdot2 = -20
      END IF
      IF k2mdot2 < -20 THEN k2mdot2 = -20
      k2tf1 = (tf(ifind) - tf1h) / taut1
      k2tf2 = (tf1h - tf2h) / taut2
      k2tf3 = (tf2h - tf3h) / taut3
      k2tt1 = (t1inp - tt1h) / 10
      k2tt2 = (t2inp - tt2h) / 10
   '
   ' Use the k1s and k2s to estimate where the state variables will go.
   ' The limits are for physical reality.
   '
      p1 = p1 + dt * (k1p1 + k2p1) / 2
      p2 = p2 + dt * (k1p2 + k2p2) / 2
```

```
    dels1 = (p1 - 3) / 12 - s1
    dels2 = (p2 - 3) / 12 - s2
    IF ABS(dels1) > deadband THEN s1 = s1 + dels1
    IF ABS(dels2) > deadband THEN s2 = s2 + dels2
    mdot1 = mdot1 + dt * (k1mdot1 + k2mdot1) / 2
    mdot2 = mdot2 + dt * (k1mdot2 + k2mdot2) / 2
    tf1 = tf1 + dt * (k1tf1 + k2tf1) / 2
    tf2 = tf2 + dt * (k1tf2 + k2tf2) / 2
    tf3 = tf3 + dt * (k1tf3 + k2tf3) / 2
    tt1 = tt1 + dt * (k1tt1 + k2tt1) / 2
    tt2 = tt2 + dt * (k1tt2 + k2tt2) / 2
    IF s1 < 0 THEN s1 = 0
    IF s1 > 1 THEN s1 = 1
    IF s2 < 0 THEN s2 = 0
    IF s2 > 1 THEN s2 = 1
    IF mdot1 < 0 THEN mdot1 = 0
    IF mdot2 < 0 THEN mdot2 = 0
  NEXT i
  '
  ' Place tf3 into the array for delayed retrieval. "iput," the put index,
  ' has to be updated for the next sampling interval.
  '
  IF (mdot1 + mdot2) > .01 THEN
    tf(iput) = (mdot1 * t1inp + mdot2 * t2inp) / (mdot1 + mdot2)
  END IF
  iput = iput + 1
  IF iput = 2001 THEN iput = 0        're start iput values at 12 O'clock
  '
  ' If enviro is active, then add noise and bias to the flow measurements
  ' and bias to the temperature measurement.
  ' here noise is removed completely with bias also neautralised
  m1bias = .95 * m1bias + .05 * m1biasb * enviro
  m2bias = .95 * m2bias + .05 * m2biasb * enviro
  m3bias = .95 * m3bias + .05 * m3biasb * enviro
  t1bias = .95 * t1bias + .05 * t1biasb * enviro
  t2bias = .95 * t2bias + .05 * t2biasb * enviro
  t3bias = .95 * t3bias + .05 * t3biasb * enviro
  mdot1meas = mdot1 * (1 + m1bias + (SQR(-.002 * LOG(RND)) * SIN(2 * 3.14159 *
RND)) * enviro)
  mdot2meas = mdot2 * (1 + m2bias + (SQR(-.002 * LOG(RND)) * SIN(2 * 3.14159 *
RND)) * enviro)
  mdot3meas = (mdot1 + mdot2) '* (1 + m3bias + 0 * (SQR(-.002 * LOG(RND)) *
SIN(2 * 3.14159 * RND)) * enviro)
  t1meas = tt1 '+ t1bias
  t2meas = tt2 '+ t2bias
  t3meas = tf3 '+ t3bias
```

```
END SUB

SUB PROCINI
    '    Routine to initialize the process parameter values
    enviro = 1            'environmental effects are off
    dt = t / 10           'integration and control periods, sec
    ap1 = .3016           'A for Process #1
    bp1 = 2.9576          'B for Process #1
    cp11b = .003979       'C #1 for Process #1, Base value
    cp12b = .01082        'C #2 for Process #1, Base value
    dp1 = .002327         'D for Process #1
    dpp1b = 30            'Differential Pressure for Process #1
    hp1 = 2               'Height of hydrostatic head Process #1
    tauvp1 = 1            'Valve TAU for Process #1
    ddpp1 = 0             'Deviation of Differential Pressure for Process #1
    dcp11 = 0             'Deviation of C #1 for Process #1
    dcp12 = 0             'Deviation of C #2 for Process #1
    power1 = 2            'value of power for valve #1 characteristic
    t1inpb = 20           'INlet Temperature Base value for Process #1
    ap2 = .3427
    bp2 = 3.3609
    cp21b = .008139
    cp22b = .0123
    dp2 = .01058
    dpp2b = 60
    hp2 = -1
    tauvp2 = 1.5
    ddpp2 = 0
    dcp21 = 0
    dcp22 = 0
    power2 = 2
    t2inpb = 10
    taut1 = .6            'Temperature sensor TAU for 1st lag***values changed
    taut2 = .4            'Temperature sensor TAU for 2nd lag
    taut3 = .3            'Temperature sensor TAU for 3rd lag
    tf1 = t2inpb          'Fictitious Temperature #1
    tf2 = t2inpb          'Fictitious Temperature #2
    tf3 = t2inpb          'Fictitious Temperature #3
    FOR i = 0 TO 2000
        tf(i) = t2inpb     'array that holds the Fictitious Temperatures for delay
    NEXT i
    m1biasb = .1 - .2 * RND
    m2biasb = .1 - .2 * RND
    m3biasb = .1 - .2 * RND
    t1biasb = 2 - 4 * RND
    t2biasb = 2 - 4 * RND
```

```
    t3biasb = 2 - 4 * RND
    'mdot1 = 5  '<***here start the mdot's
    'mdot2 = 5
     o1 = 100
     o2 = 100
END SUB
```

VITA

Gaurav Arora

Candidate for the Degree of

Master of Science

Thesis: ON THE USE OF A TRUTH-SPACE DIAGRAM FOR ASSESSING LINGUISTIC RULES

Major Field: Chemical Engineering

Biographical:

Personal Data: Born in New Delhi, India on January 10th, 1984, the son of Mr. Ravinder Kumar Arora and Ms. Kamal Arora.

Education: Received Bachelor of Science degree in Chemical Engineering from the GGSIP University, Delhi, India in May 2005. Completed the requirements for the Master of Science Degree with a major in Chemical Engineering at the Oklahoma State University in May 2007.

Experience: Employed by Oklahoma State University, Department of Chemical Engineering, as a research and teaching assistant from August 2005 to present.

Professional Memberships: Phi Kappa Phi Honor Society.

Name: Gaurav Arora Date of Degree: May, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: ON THE USE OF A TRUTH-SPACE DIAGRAM FOR ASSESSING LINGUISTIC RULES

Pages in Study: 161 Candidate for the Degree of Master of Science

Major Field: Chemical Engineering

Scope and Method of Study: The main objectives of this study were to improve the process of selecting the valid rules and to accurately predict the value of the output. A hot and cold water mixer simulator was used to generate the data. The For the purpose of comparison, the truth calculations for the antecedent and consequent were performed using both the geometric operator and minimum operator. For each operator, different rule sets were obtained by using different threshold values for *corroboration* and *merit*. A statistical technique was developed to quantify the delay. In the prediction phase, Kumar's modified prediction technique was compared to the conventional prediction technique used in fuzzy logic. The RMS prediction error obtained from these techniques were calculated and compared.

Findings and Conclusions: The minimum operator was found to be more suitable, for performing the T_a and T_c calculations, as compared to the geometric operator. C*orroboration* was sufficient to assess the validity of the rules and *merit* was not required. The threshold value of *corroboration* equal to unity has been proposed in this work. In addition, it has beem proposed that only one out of all the possible interacting rules should be included in the final rule set. In general, conventional prediction technique was found to more accurate and computationally efficient then Kumar's modified prediction technique. In addition, minimum RMS errors were obtained when the value of the number of selected (valid) rules was close to the minimum number of rules required to describe the system completely.

ADVISER'S APPROVAL: Dr. Russell Rhinehart