IMPROVED QUALITY METRICS FOR LINGUISTIC RULE SELECTION

By

PREETICA KUMAR

Bachelor of Science in Chemical Engineering

Osmania University – College of Technology

Hyderabad, Andhra Pradesh, India

2003

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
December, 2005

IMPROVED QUALITY METRICS FOR LINGUISTIC

RULE SELECTION

Thesis Approved:

Dr.R.Russell Rhinehart

Dr Gary Yen

Dr Karen High

A. Gordon Emslie

ACKNOWLEDGEMENTS

I extend my sincere appreciation and gratitude to my research advisor Dr. R. Russell Rhinehart, without whom this master's thesis could not have been written. His continuous support and encouragement brought out the best in me. I deeply value his contribution to my academic and professional growth.

I thank my committee members Dr. Gary Yen and Dr. Karen High for their critical input towards the successful completion of this master's thesis.

I am thankful to my research partner Ming Su for helping me with the initial stages of this research.

I am also indebted to my dear friends Samuel Owusu and Mellicent Owusu for always being there for me.

Last but not the least; I thank my family and friends for having faith in me which helped me a great deal in accomplishing this goal.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

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
F1	Flow rate of hot water stream
F2	Flow rate of cold water stream
T1	Temperature of hot water stream
Ta	Truth of Antecedent
Tc	Truth of Consequent
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 Cause-and-effect Rules in The Chemical Process Industry

Cause-and-effect rules generated from raw process data are vital for successful plant operation; and especially so in the Chemical Process Industry (CPI), owing to the complexity of chemical processes. Plant and operator safety, the quality of the final product, environmental impact of the process, loss minimization, optimization of time and operational cost are important factors that define successful plant operation. Thus, knowledge gained from cause-and-effect rules can prove to be very valuable for design, operation and control purposes. However, mining of all the potentially useful information from the data available and its correct interpretation afterward is a challenge.

Cause-and-effect rules are linguistic, logical representations of the underlying behavior of the process at hand. An example of a rule for a chemical process is "IF (the reactor temperature has been high for an extended period AND the feed is in manual) THEN (in a short while the product will be slightly yellow) WITH (moderate certainty)". These rules are generally expressed as, If *Antecedent* THEN *Consequent.*

As seen in the example above, such rules are linguistic statements that can be easily comprehended by human beings as opposed to complex mathematical equations or statements. They describe process variables (like temperature, color) and express their values linguistically (high, low, and slightly yellow). They also incorporate temporal information like persistence (extended period) and delay (short while). The logic of the rule can be verified using existing logical understanding of the process. Thus, if interpreted correctly, these cause-and-effect rules have significant utility in the CPI. To name a few,

1. They can warn an operator of imminent events and can help predict future outcomes.
2. They can help the operator recognize antecedent elements that need to be sustained or eliminated to improve the process.
3. They can reveal unrecognized mechanisms.
4. They can guide feedforward and feedback control system strategies and thus help in their design.

1.2 The present research – The big picture

Many vendors offer software to the CPI with reasoning capabilities (e.g., Gensym's G2 or KnowledgeMiner by Script Software). But, there still seems to be a need to integrate learning of rules from data represented *linguistically*, using fitness measures to evaluate the rules, and the consideration of the temporal(changing with time) dynamics (transport delay, persistence) of a process. Working in this direction, the present work is

one of the four parts of a project funded by the Measurement and Control Engineering Center (MCEC), the aim of which is to "Develop a method to autonomously generate cause-and-effect rules (linguistic, logical, to include temporal dynamics) from natural process data."

In particular, the four parallel parts of the project are as follows:

1. Development of a rule-extraction mechanism that takes the temporal features of a process into consideration.
2. The management ad updating of the rule-set thus obtained.

These 2 parts of the project are being pursed by Dr. Gary Yen and Pedro De Lima. [1]. A preliminary algorithm and GUI has been developed and it produces desired results.

3. Generation of data which is to be used to extract knowledge.
4. Investigation of *metrics* (fitness criteria) to assess the quality of the rules extracted by steps 1 and 2.

These 2 parts are being pursued by Dr. R. Russell Rhinehart, Ming Su and Preetica Kumar. Their work inherits important concepts introduced by Nitin Sharma [2] (Refer to Section 1.4.1) and serve as *improvements* to previous work.

Ming Su [3] is presently working mainly on quantifying the concept of *persistence* and analyzing the right choice of *transport*.

1.2.1 The role of the present work

With this background, the *aim* of the present work can be stated as follows:

"To explore improved quality metrics to identify useful and logically correct cause-and-effect rules which can be believed to describe the dynamic and temporal behavior of the process at hand accurately and also to predict future behavior to some extent."

1.3 Background

1.3.1 Expert Systems and the Role of Operators

Modern process plants use latest data acquisition methods and storage technology as tools to gain useful insight into the process at hand (by generating cause-and-effect rules). *Expert Systems (ES)* are an approach to managing knowledge obtained from a process.

The extent of automation and application of computer technology (Expert Systems) in the CPI has increased multi-fold owing to the need for computer-aided operation. Stephanopoulos and Han [4] define ES as "computer programs that possess algorithms, which attempt to *model and emulate*, and thus automate engineering tasks that used to be carried out by a human". In other words, ES are just computer programs that work with large amounts of knowledge and attempt to identify structure (in the form of linguistic rules) in it; by no means do ES possess human-like intelligence.

It is important to note, though, that even today a lot of the chemical operations (Batch or Continuous) are carried out manually by operators depending on the process at

hand. The operators also bear the onus of having to be accurate in their predictions of operational states that might occur in the future from a safety point of view. The safe running of a plant is first on the list of priorities of the plant; and relies a lot on an operator's intuition, experience, diagnosis of the problem (if there is one), and on the operator's judgment as to what the corrective or preventive action should be. The ES by itself is completely incapable of predicting unexpected occurrences, but operators are not perfect either as human beings are prone to making errors in operation. They have to comprehend the vast complicated amounts of data they are flooded with, in possibly stressful situations due to lack of time.

Furthermore, an operator's knowledge is often outdated as a result of the continuous evolution of the operation of the process. Their knowledge is then relayed to a programmer, in spite of their incompleteness and with their misconceptions, often resulting in rules that are inefficient or ineffective. Expert systems that the operators are provided with aid them in situations like these to make more informed decisions. E.Oshima [5] describes in detail the problems that are faced by operators during plant operation, the use of Expert Systems (Computer-aided plant operation) and how they should be more self-consistent.

1.3.1.1 Applications of Expert Systems

Stephanopoulos and Han [4] give a detailed overview of the numerous areas of applications of expert systems, in their paper. A few of those applications are summarized below.

1. Fault Diagnosis: Expert systems have effectively and extensively been implemented industrially for on-line data monitoring and diagnosis. Some of these systems also determine the best course of action.
2. Analysis of Process Trends: Most related to the present work, this application relates to the use of data extracted from a process to create "a mental model of the process operations that fits the current facts about the process" as stated by Stephanopoulos and Han [4]. This helps the operator understand how the process behaves, what can be expected in the future under the same conditions and which control action will produce the desired results.
3. Process Control: Control systems in the CPI today employ expert systems, and concepts of fuzzy logic and neural networks. Systems like these are needed because the nonlinear control theory fails to deliver simple solutions to today's control problems. Within process control, the various sub-applications are as follows: knowledge-based expert control("use of logical inferences to confirm a given conclusion" [4]), supervisory control("used to monitor, evaluate, diagnose, adapt" [4]), controller tuning and adaptive control("to provide auto-tuning of PID controllers" [4]), controller-design("to make design decisions and for sequencing of design tasks" [4]), fuzzy logic controllers(FLCs)("used in the supervisory mode and in the loop" [4], uses *fuzzy* reasoning), neural controllers("use of a neural network in some function of a control system" [4]). Various other applications of expert systems (not mentioned here) are discussed in detail by Stephanopoulos and Han [4].

1.3.2 Knowledge Discovery in Databases (KDD) and Data Mining

Expert Systems are used to solve many difficult problems in the Chemical Process Industry (CPI) but definitely require significant operator intervention to make an in-depth analysis of huge amounts of data obtained from a process with the aid of state of the art data acquisition systems. This is where the field of KDD comes into play. As stated in [6] a common definition of KDD is "The non-trivial process of identifying valid, novel, potentially useful and ultimately understandable structure in data." From a Chemical Engineer's perspective, *structure* in data may refer to observed cause-and-effect mechanisms or relationships of a process.

"Data Mining" is the central component of the process of KDD. As defined in [6], it refers to the "application of computational techniques to the task of finding patterns and models in data." The process of KDD however is made up of many other components – Preparation and Pre-processing of data (raw data obtained from a plant cannot be used directly to observe patterns, as it contains noise), incorporation of prior knowledge and interpretation of the data-mining results (in order to make the right decisions the information obtained needs to be comprehended correctly.) Knowledge in the form of patterns and models are then used to learn tasks and make decisions.

1.3.3 The role of fuzzy systems in KDD

Zadeh [7] first introduced the concept of fuzzy reasoning in 1973. Since then *Fuzzy sets* have contributed extensively to all components of the KDD process, and especially so to data mining. Fuzzy sets help in making hard-to-understand patterns in

raw data very comprehensible by expressing the quantitative and qualitative information in the data, in terms of *fuzzy rules* that use human language terms. They thus serve as an excellent interface between the user and knowledge, because these fuzzy rules are very easily understood by the user. They are capable of handling complex, nonlinear, incomplete, extremely dynamic systems. They are also useful in data reduction and hence simplification. Although fuzzy sets have been applied relatively more to *data mining*, there lies great potential in their use in almost all the other areas of KDD; like in data processing.

1.3.4 Rules

The present work (described in Section 1.2) deals with the extraction of statements, termed as *rules*. Fuzzy rules represent the relationships between different variables (e.g., temperature, flow rate, color) or values (e.g., low, medium, high), that constitute a database, in linguistic terms. The process of extracting these rules from a database is termed as *Rule Mining*. Rules are a direct reflection of functional dependencies in the database and employ a simple *If-Then* structure. The following is an example of the structure of a simple rule.

IF <condition 1> AND (<condition 2> OR <condition 3>) THEN <effect 1>

The part of the rule between the *IF* and *THEN* keywords, is termed as the *antecedent* and represents a cause (certain process conditions) that bring about the effect stated in the part of the rule after the word *THEN*. The part of the rule after the word *THEN* is termed as the *consequent* and represents the effect caused by the cause stated in

the antecedent of the rule. These rules are thus termed as *Cause-and-effect* rules for purposes of the present work.

Any number of process variables can be included on either side of the rule (as conditions and/or effects). However, no variables should be in common between the antecedent and the consequent.

There is however a missing link. Fuzzy systems even with all their utility do not possess the capability to *learn*. They hence need to be used in conjunction with techniques that are capable of learning, to be used. Neural networks and Genetic Algorithms serve this purpose.

1.3.5 The 'learning tool': Genetic Algorithms (GAs)

Genetic Algorithms (GAs) are defined as *search algorithms* used to *evolve solutions* to problems. As the name suggests, they make use of the underlying principles of natural genetics. They use a randomly or heuristically generated initial database of possible solutions (candidate rules) to the problem (process) under consideration. These solutions (cause-and-effect rules) are termed as *chromosomes*. The GA then advances towards better solutions (cause-and-effect rules) in a series of steps. Each step is referred to as a *generation* and creates a new database of rules each time. The rules in the initial data-base are first evaluated according to a predefined quality criterion defined as the *fitness function*. To form a new database of rules, good rules from the initial data base are selected according to the evaluations of their fitness functions. The *selection* of good

rules from the initial rule data base however, will not introduce any new rules into the database. New rules are generated in each step (generation) by the use of genetic operators like *crossover* and *mutation*. These operators are basically ways to combine different combinations of antecedents picked from good rules, selected from the initial rule-base, to create a new rule-base containing rules which are also expected to be good or promising. This process is repeated for a certain number of times. The number of steps carried out depends on the *stopping criteria* employed – like an acceptable fitness level.

Genetic Fuzzy systems can now be defined as fuzzy systems that make use of an *evolutionary learning* process or search algorithm to generate *rules* that represent a process.

1.4 Literature Review

Rules can be extracted from numerical data using many approaches that have been proposed. However, rule number reduction (optimization of rule-bases) is imperative especially for complex processes with high dimensionality. In recent years, genetic techniques have been considered to address this problem with great success.

A genetic algorithm initiates and maintains a population of rules. It then evaluates the fitness (the strength or quality) of the rules based on their response to the training data, using objective or fitness functions. All good rules (‘parents’) are then allowed to reproduce new rules (‘offspring’), while the bad ones are removed from the rule set.

Thus, the appropriate selection of a fitness function is very critical to the success of the genetic algorithm.

Objective functions are *Metrics* or numerical measures that evaluate desirable (application dependant) characteristics of the rule and represent their quality or goodness, thus aiding the elimination of unnecessary rules from the initial rule-base.
Metrics can be classified as follows, depending on the rule-generation strategy employed.

1. Pittsburgh approach: It evolves a complete rule-set and thus uses metrics (*global* criteria) that evaluate the fitness of the entire rule-base.
2. Michigan approach: It evolves single rules individually and thus uses metrics that evaluate the fitness of individual rules (*local* criteria)

The choice of metrics or approach to be used is dependant on the nature of the problem**. The present work focuses on developing 'local' criteria which may be used, as objectives, by GA's that follow the Michigan approach.** In the Michigan approach, each iteration of the rule evolution adjusts the individual fuzzy rules in a collection.

What follows is a review of a few local fitness measures employed by some of the GA technologies, in chronological order.

Yuan and Zhuang [8] developed a Fuzzy Genetic Algorithm to generate fuzzy classification rules, with several techniques used to improve the efficiency and

effectiveness of the algorithm. In doing so, they developed a composite fitness function that consisted of three components: *Accuracy, Coverage* and *Contribution* of the rule. The coverage of a rule was the relative size of its condition (antecedent) set in the training space. So, the larger the coverage, the more general the rule. The accuracy in very simple terms was the truth that the condition (antecedent) implied the conclusion (consequent). Naturally, the higher the accuracy, the better the rule was. Both accuracy and coverage involved the use of the sum of membership functions (indicating the degree to which an object in the training set belonged to the antecedent or consequent set) in place of the number of data points. Contribution measured the uniqueness of a rule in the population. Higher contribution meant lesser overlap with other rules.

All the above-mentioned quality measures were integrated into a single fitness function. In doing so, accuracy was given more importance than coverage. However, it appears best not to combine multiple and competing objectives without a thorough understanding of the preference of all goodness metrics involved.

Chen and Black [9] used fuzzy systems and neural networks to sense tool breakage for end-milling operations. In their pursuit of generating fuzzy rules, they used the *degree* of a rule to resolve conflicting rules, i.e., rules with the same antecedent but different consequents. It was defined as the product of the individual membership values of the antecedent and consequent. The rule with the higher *degree* was deemed the winner. In cases where the deviation in the degrees of two rules was small, the number of fuzzy regions the input-space was increased until all the conflicts were resolved.

This approach is not recommended since the rule base complexity exponentially increases with the number of antecedents. Using the operator 'OR' to combine the conflicting antecedents into a single rule is a better solution because it clubs independent mechanisms that create the same effect thus making the rule more interpretable and the rule-base simpler. Besides, in the CPI it is rare to observe the same cause (antecedent) leading to different effects (consequence). In fact, different causes are found to lead to the same effect, more often.

Work by Herrera, *et al.* [10] focuses on the development of a genetic fuzzy system to extract fuzzy linguistic rules. They used the *completeness* property to eliminate redundant and unnecessary rules while maintaining a minimal completeness degree on the training set. Mathematically, it was defined to be a function of the *compatibility degree* between an individual rule and a training example. Other features such as high-frequency value of a fuzzy-rule through the example set, high average covering degree over positive examples (data-set matches) were also formulated as functions of the compatibility degree. They also defined the *covering value* of an example rule over the entire rule-base. Concepts of membership function width and symmetry were also used as part of the fitness function. The product operator was used to combine the criteria to yield the final fitness function.

Completeness and consistency have been used conventionally as the evaluation metrics. However, a complete rule covering the entire database is unrealistic in a real-life situation. So the support-confidence framework was employed by Ngan instead.

Ngan, *et al.* [11] used evolutionary computation as a search algorithm for discovering rules that capture patterns in real-life medical databases. Their learning approach was based on generic genetic programming (GGP). The following metrics were then used for rule-evaluation. *Confidence factor* which measures the consistency of a rule is defined as the ratio of the number of records matching both the consequents and the antecedents (number of both hits) to those matching only the antecedents (number of antecedent hits). The *consequent probability* on the other hand is defined as the ratio of consequent hits to the total number of records in the training set. This value represents the confidence for the consequents irrespective of the antecedent. Defined as the ratio of the number of records covered by the rule (the number of both antecedent and consequent hits) to the total number of records, *support* measures the coverage of the rule. The value of support should be above a user-defined minimum for the rule to be considered as adequate. The final fitness function also involved *count* and *ideal count* in addition to the above-mentioned factors. *Count* was the number of examples the rule actually seized and *ideal count* was the maximum number of examples it could have seized if there was no competition. A rule is said to seize a data example when the numerical data matches the antecedent of the rule statement. As a result, other weaker rules can no longer seize this same data example. They termed this concept as *token competition*. This was used to reduce redundancy, reduce rule conflict and hence the complexity of rule base. Redundant rules were replaced by new rules thus increasing diversity of the population and increasing the chances for generating good rules. The concept of *hits*, however, does not account for the degree of membership, or completeness that the antecedent or

consequent is true. Besides, the arbitrary support threshold makes it impractical to the users.

Castillo, *et al.* [12] proposed *simplicity* criteria to be included in the genetic fuzzy learning algorithm SLAVE (Structural Learning Algorithm in Vague Environment). As the main criteria used in SLAVE, they reformulated the concepts of completeness and consistency (used for crisp models) in order to adapt them to the special characteristics of linguistic terms (fuzzy models). These adaptations were called the *degree of completeness* and the *degree of soft consistenc*y respectively. The degree of completeness of a rule was defined as the ratio of the number of positive examples of the rule to the number of examples in the training set. In other words, the completeness degree determined the strength of the rule by measuring the number of examples of the class being learnt that support the validity of the antecedent of the rule. The degree of soft consistency made room for admitting some noise in the rules. It represented in the general case the set of rules having a number of negative examples strictly less than a percentage (k) of the positive examples and for completely consistent rules k =0. They then defined the degree to which a rule satisfied the soft consistency condition in terms of negative and positive examples of the rule. It determined the level to which the examples of the training set, which are covered by the antecedent of a rule, satisfied this rule. The degree of completeness and the degree of soft consistency were then combined using the product operator to yield the composite main criterion.

Their methodology consisted of including two simplicity measures: one with respect to the variables and another with respect to the values. These criteria would aid in discriminating between rules that had the same evaluation function value (involving completeness and consistency). In other words, they would choose the simpler and the more understandable rules among best rules in case of a tie situation. The first tie would be resolved by using simplicity of fewer variables and the second tie if any, would be resolved by using simplicity of values. The simplicity of variables determined the simplicity of a rule by counting the number of relevant variables that were involved in the antecedent of the rule and the simplicity of values was determined by evaluating the distribution of the values assigned to the relevant variables.

The final fitness function now involved three components. The lexicographical order was used as the optimizing criteria, i.e., the main criterion was maximized initially, and in case of a tie situation, simplicity of variables was maximized and in case of another tie, simplicity of values was maximized.

However, these criteria were defined only for the learning algorithm SLAVE and hence had a specific application. The optimization of these metrics was also specific to the learning algorithm SLAVE.

Kim and Lee [13] proposed a new design method of an FLC based on the Lamarckian co-adaptation mechanism of evolution and learning that used both global and local strategies : The evolution of many FLCs (global searching) involved use of GAs

and the learning of each individual FLC (local searching) involved use of Neural Networks (Backpropagation learning rule).

As opposed to other works mentioned so far in which the initial population comprised individual rules, in Kim and Lee's [13] scheme entire rule bases (corresponding to individual FLCs) formed the initial population and eventually competed in the evolution stage. Each rule base had its own set of input and output variables (antecedents and consequents). Both local and global fitness measures were then used to compare rule-bases on a global level and individual rules on a local level. The composite fitness function comprised of four such metrics.

To begin with, Kim and Lee [13] state that, a good fuzzy rule base should cover as many input-output data pairs in the training set as possible. The *covering value* of a rule-base over a training set was then calculated as the sum of the *compatibility degrees* over all fuzzy rules in the rule-base. The compatibility degree (earlier used by Herrera [10]) over a training example in turn was defined as the product of the membership functions of different parts of the antecedent and consequent of the rule. Compatibility degrees with respect to each training example were summed to yield the coverage of the individual rule. These individual coverage values were then combined to determine the coverage value of the entire rule base. A rule-base having a higher covering value was better.

The second metric was *the number of useful rules* in the rule-base. Rules were termed as useful if their covering values were higher than a certain threshold value. A smaller number of useful rules were desirable in a rule base.

The *system's approximation error* and the *tracing distance* over all training examples (with respect to the truck-backer upper control problem considered by Kim and Lee [13]) were used to measure the FLC's control performance. These two metrics were global in nature. Each of the four metrics were then normalized individually in the range of (0, 1) and integrated into one fitness function such that appropriate weight constants determined the importance of each metric.

However, the performance of the system depended a great deal on the right choice of these weight constants. The entire procedure is very computationally intense and time-taking. Also, using the product of the membership functions of the antecedent and consequent may lead to loss of important information, as the membership functions of the antecedent and consequent parts provide different information when considered individually, which could be critical in distinguishing between good and bad rules.

Ishibuchi and Yamatomo [14] proposed using rule evaluation measures (*support*, *confidence*) as rule selection criteria for *pre-screening* candidate fuzzy if-then rules, before using a multi-objective genetic algorithm to optimize the rule-base. *Confidence* (earlier used by Ngan [11]) indicated the grade of the validity of a rule A→ C. Qualitatively, if c represents the confidence, it implies that, c x 100% of the training

patterns that are compatible with the antecedent A are also compatible with the consequent C. Rules selected by the confidence criterion were very specific. *Support* indicated the coverage by A → C. If s represents the support, it implies that, s x 100% of the training patterns are compatible with the rule A → C (i.e., compatible with both the antecedent and the consequent). This criterion tends to select short if-then rules that lack in confidence. In addition to support and confidence, their product was also used as a pre-screening criterion because it balanced the specificity introduced by confidence and the generality introduced by support. These concepts of confidence and support were similar to what Ngan [11] used and thus shared the same drawbacks.

In many other works, objective measures such as support, confidence, interest factor, correlation and entropy were used to evaluate the interestingness of association rules. But in many situations, due to differences in some of their properties, these measures may provide conflicting information about the interestingness of a pattern.

Tan, *et al.* [15] describe several key properties that need to be examined in order to select the right measure for a given application. Depending on its properties, each measure is useful for some application, but not for others. They present an overview of 21 objective measures that were proposed in statistics, social science, machine learning and data mining literature. Several groups of consistent measures having similar properties are identified.

It has been shown that many well known measures are monotone functions of support and confidence, which explains the reason for the optimal rules to be located along the support-confidence border.

Wang, *et al.* [16] presented an approach to construct 1st order TS (Takagi-Sugeno) fuzzy models from data. In doing so, they employed a multi-objective hierarchical genetic algorithm to generate optimized fuzzy models with a high accuracy and good interpretability.

Interpretability represented the transparency of a rule or how easily it could be understood and interpreted for complex systems and comprised of the following components.

1. Completeness and Distinguishability
2. Non-redundancy
3. Compactness
4. Utility

Thus a total of four fitness functions with pre-defined preferences, as follows, were used for comparisons within the population.

1st priority: *Accuracy* – Objective: to be minimized

2nd priority: Completeness and Distinguishability Objective: to be maximized

3rd priority: Non-redundancy Objective: to be maximized

4th priority: Compactness Objective: to be minimized

However, since utility of the fuzzy system was guaranteed through chromosome formulation and genetic operators with constraints, before the fitness evaluation stage, it wasn't included as one of the interpretability considerations for fitness evaluation.

1. Accuracy was defined as the mean squared error between the true and model output vectors.
2. Completeness and Distinguishability (0-1): The completeness of fuzzy systems meant that for each input variable, at least one fuzzy set is fired. These qualities were measured by *Similarity*, which denoted the degree to which the fuzzy sets were equal. If similarity ~ 0 or too small, it implied that the fuzzy partitioning in this variable was incomplete/they did not have enough overlap. If similarity was too big, it implied that they overlapped too much, which in turn implied that distinguishability was poor.
3. Non-redundancy (0-1): Needed to be maximized to increase the interpretability of fuzzy rules. A rule was said to be redundant if it brought nothing new to the rule-base. It was calculated based on the similarity degree of rule antecedents. A high value of non-redundancy implied that the rules were very different. A value of 0 implied that the antecedents were the same.
4. Compactness: A compact fuzzy system meant that it had the minimal number of fuzzy sets and fuzzy rules.
5. Utility: If a fuzzy system was of sufficient utility then all of the fuzzy sets were utilized as consequents/antecedents by the fuzzy rules.

This approach suffered from a major drawback. An effective trade-off between interpretability and accuracy needed to be expressed. Thus it was a multi-objective optimization problem by nature. In other words, only a set of pareto-optimal solutions of which the improvement in one of the objectives will degrade other objectives, could be obtained.

To pursue the balance between necessary accuracy for modeling complex systems and interpretability degrees to provide expert knowledge remains an open issue in the development of future Genetic Fuzzy systems.

1.4.1 Sharma's work

Sharma's [2] work introduced a general strategy, based on use of the "Truth Space Diagram" generated for each rule, to evaluate multiple measures of goodness of linguistic rules. His work also recommended metrics for selecting good cause-and-effect rules from dynamic data obtained from processes in the Chemical Process Industry (CPI).
The present work inherits some parts of Sharma's work. The ensuing discussion restates concepts that were used.

A case study approach was used to determine the best (among many possible) metrics. To start with, data from a Hot and Cold water simulator was processed. Then an initial rule base containing all possible rules (all possible combinations of antecedent and consequent parts) was created using an exhaustive search method, designed metrics were calculated and the metrics were optimized to find the correct rule base. This final rule

base was inspected by the human operator and the best combination of metrics decided based on the quality and compactness of the rule base.

Thus the process used was three-fold and can be summarized as follows.

1. Data generation and processing
2. Initial Rule base generation (by Exhaustive search)
3. Calculation of Numerical Metrics and Rule base optimization

Steps 1 and 2 of Sharma's process, as mentioned above, were inherited in the present work and are re-explained below.

1.4.1.1 Data generation [2]

Data was acquired from a Hot and Cold water simulator. See Figure 1. Although simple, the simulation incorporated real-world dynamics like transport and measurement delays and was also capable of incorporating measurement bias, process drifts, noise and valve 'stick-tion'. These behaviors are observed in most of the unit operations within the CPI. Also, the simulation was nonlinear, had multiple inputs and the delay time of the output temperature depended upon the operating conditions.

For data generation, three input variables were manipulated and the effect on one output variable was monitored. Figure 2 depicts the transient response. The input (manipulated) variables 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).

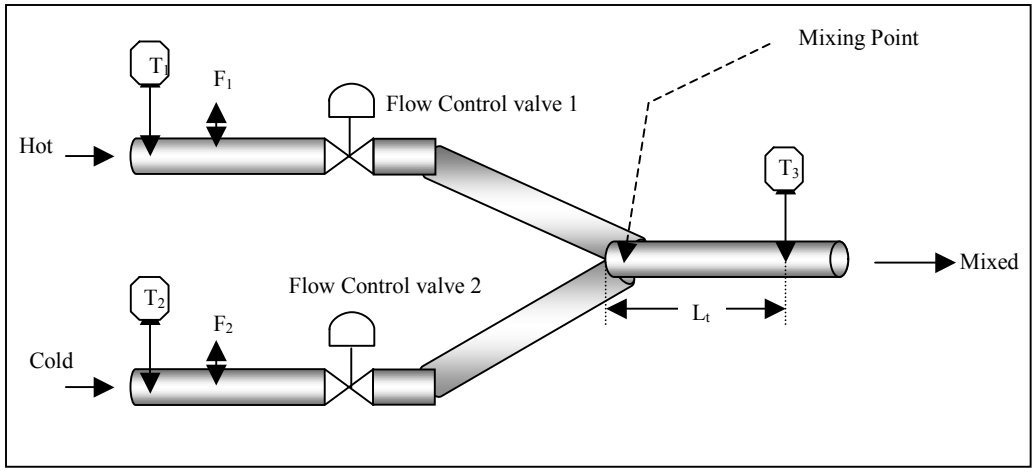


Figure 1: Hot and Cold water Simulator (reproduced from [2])

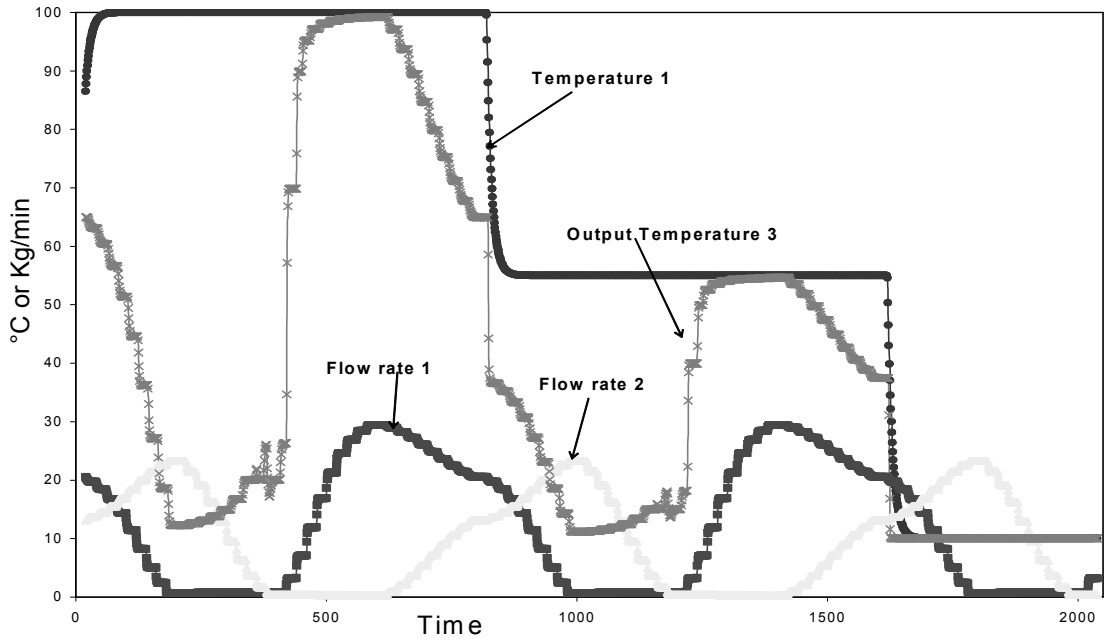


Figure 2: Transient Input-Output Data (reproduced from [2])

The output variable was the temperature of the mixed stream ($0 \leq T3 \leq 100$ ^{o}C).

The algorithm simulates the mixing of two process streams - one carrying hot water and the other carrying cold water. It calculates the resultant temperature and delays the measurement of this output temperature based on the mixing length L_t and the input flow rates. Data was sampled at an interval of one second.

The Simulator code is included in Appendix A.

1.4.1.2 Data processing [2]

The raw data was processed in two steps. First the output-data was *un-delayed* by shifting data backwards. This was done by deleting a number of data-points from the top of a column and shifting the rest of the column upwards (backwards) in time. The number of data points deleted was equal to the delay (measured in time units) in a certain category (short/medium/high). Figure 3 depicts this procedure schematically.

Secondly, the crisp input-output data was *fuzzified* using linguistic membership functions. Each variable was classified into three fuzzy categories – low, medium and high, using triangular membership functions defined by Equation 1.1

$$\mu^{i,j} = \frac{a_j - x_i}{a_j - b_j} \ , \tag{1.1}$$

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 x_i in the j^{th} fuzzy category

a_j and b_j = fuzzy set break points for category j

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	61.54251	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 3: Example of Backward Shifting of Output variable T3 with Short Delay = 2 sec; Medium Delay =6 sec; Long Delay = 22 sec. Reproduced from [2].

Figure 4 illustrates an example of 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. 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 .The values for fuzzy limits(a_j , b_j)were decided based on the range (lowest, mid and highest values) of numerical values of

the variable under classification. 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.

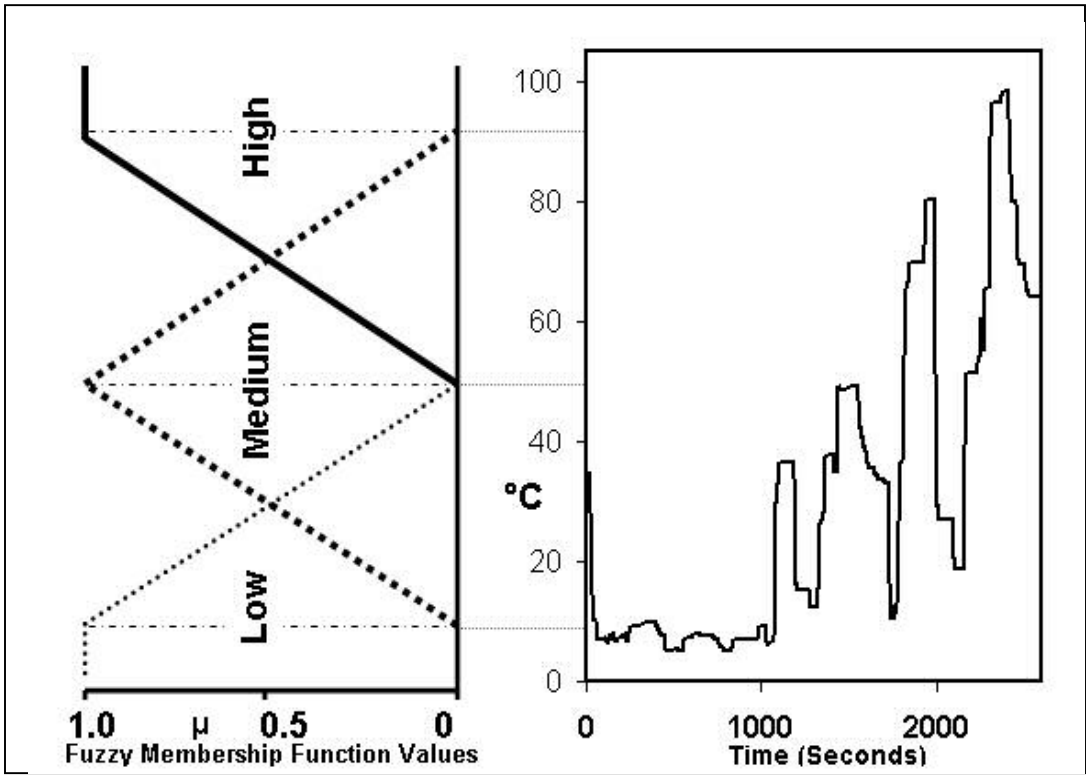


Figure 4: Fuzzy Classification of Output T3 (Reproduced from [2]).

Dynamic information was included by incorporating the *persistence* of an event in the antecedent of a rule and the resulting delay in the consequent of the rule. Thus the process of *fuzzification* converts crisp numerical values to membership values which measure the degree to which a certain variable belongs to a linguistic label.

1.4.1.3 Initial Rule-base and Truth Space Calculations [2]

The antecedent of each rule involved four linguistic labels (T1, F1, F2, persistence) while the consequent involved two linguistic labels (delay, T3). Each of these variables was classified into three fuzzy categories as explained in Section 1.2 – low, medium and high. The initial rule base consisted of all possible combinations of the linguistic categories of each variable in the antecedent and consequent parts and each rule had the following general structure:

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 is Low, Medium or High.

Thus the total number of possible rules was = (no. of fuzzy classes) $^{(\text{no. of variables})}$ = $3^6 = 729$. Hence the initial rule base had a size of 729 rules containing all possible antecedents and consequents. The algorithm generated the statements of each of the 729 rules programmatically and used the fuzzy data from the previous section to perform calculations depending on the statement of the rule.

Firstly, persistence (the length of time an event persists) of the antecedent for each data point was calculated as the minimum persistence of the three other parts of the rule antecedent (T1, F1, F2). The persistence of each linguistic label (T1 is High, F2 is Medium etc.) is measured by the number of time units the membership value of variable has persisted in the fuzzy category. Once calculated, the persistence is fuzzified.

Secondly, the *truth of the antecedent* and the *truth of the consequent* for each rule were calculated for each data point. 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 any antecedent or consequent was the measure of the match between the stated event (hypothesis) and the numerical data (reality). The truth of the antecedent and the consequent respectively, were calculated as follows:

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

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

After the above calculations were made the Truth Space Diagram (TSD) was constructed for each rule.

1.4.1.4 Truth Space Diagrams (TSD) [2]

A *truth space diagram* (TSD) 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". It is bounded by the region {T: $0 \leq T \leq 1$, where T = truth of antecedent or consequent}, where a truth of 0 means *zero truth* (it did not happen to even a slight degree) and a truth of 1 meant absolute truth. As stated by Sharma, the truth space diagram also represents a one-to-one mapping of the data-set from the real (crisp numerical values) space to a new truth (fuzzy membership values) space defined by the linguistic statement of the rule. As shown in a hypothetical TSD in Figure 5 the space was divided into four equal quadrants by Sharma and each quadrant provides different information about the linguistic rule in question.

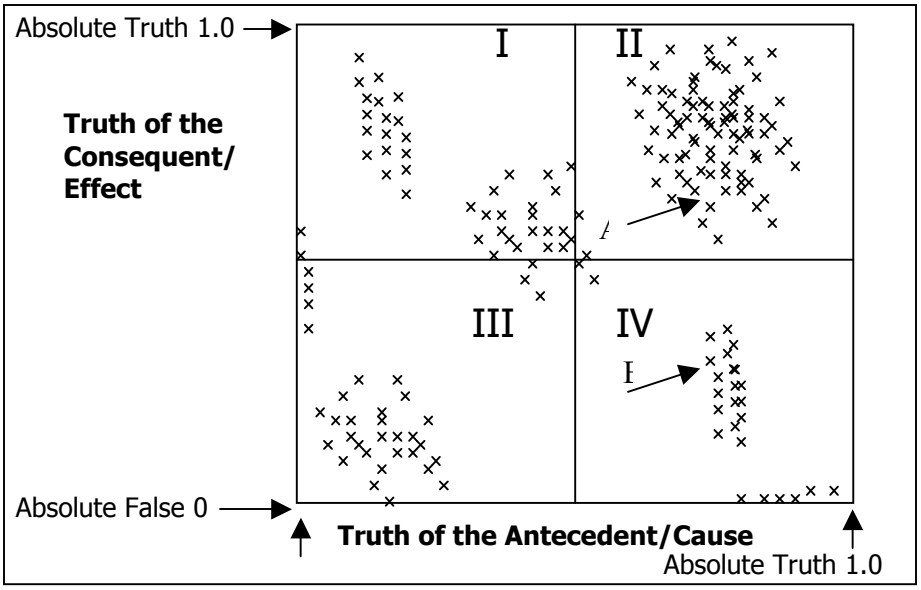


Figure 5: Hypothetical Truth Space Diagram (Reproduced from [2]).

1.4.1.5 Points on the TSD [2]

Each data-set is represented by a point on the TSD, and the location of the point is a measure of the membership of the data-set to the linguistic statement of the rule. For example, consider point A in Figure 5. It occurs in Quadrant II which suggests that the values for Ta_i and Tc_i are high for this data-set, i.e. the cause and effect match according to the rule statement. This reveals that the information expressed in the linguistic rule is contained in the numerical data from the simulator. Hence many points in Quadrant II of the TSD would reflect the validity of the rule.

Consequently, points in Quadrant IV would show that the rule statement was false, i.e. the process data expressed the rule antecedent but did not express the rule consequent. For example point B in Figure 5 has a high Ta_i which means that the event stated in the antecedent matches the numerical values however the Tc_i for this point is

low, which shows that the event expressed in the consequent does not match the real numerical values.

Similarly points in Quadrant I have a high Tc_i but low Ta_i which shows the incompleteness of the rule, i.e. the consequent was observed but was due to event other than the event expressed in the antecedent of the rule.

Quadrant III points show the possibility of the rule, but because the value of both the truths is low, it is not possible to confidently use this information to test a rule since it neither proves nor disproves the hypothesis stated in the rule statement. The points that lie on the axis ($Ta = 0$) show that either the event stated in the antecedent never occurred in the data, a large number of these points will indicate that the rule was insufficiently expressed within the data and cannot be judged as good or bad using the available numerical information.

Based on the above concept of the TSD one can define various numerical metrics which measure the desired qualities of linguistic rules and can be used to optimize a rule base or search a rule population so as to prescribe only Good, Complete and Sufficiently expressed rules. Many different metrics were designed by Sharma.

1.4.1.6 Sharma's Numerical Metrics:

Sharma proposed the quantitative measures of *strength of goodness, probability of goodness, combined goodness* (based on the concept of goodness), *strength of badness,*

probability of badness, quantity of badness (based on the concept of badness) *incompleteness* of a rule and *insufficiency of data.*

A rule was defined to be *good* if its consequent was in fact the actual effect of the cause expressed by the antecedent of the rule. Subsequently, points in Quadrant II of the TSD show that the rule in question is good. Based on this concept, *strength of goodness* was defined as the RMS perpendicular distance of points in Quadrant II from the Good Diagonal (line from the Point (0, 0) to Point (1, 1)) on the TSD. The closer the points were to the diagonal, the better the rule was. *Probability of goodness* as the name suggests was used to determine the expected probability of points to occur in Quadrant II rather than in any other quadrant. If a rule was good this probability increased. *Combined goodness* was a measure of the standard deviation of points in Quadrant II considering the mean to be the corresponding truth of the antecedent. This metric assumed that the data was normally distributed and that ideally for each point in the data set, the truth of antecedent should be equal to the truth of consequent. Subsequent investigation reveals both assumptions to be invalid.

On the same lines as strength of goodness, strength of badness was based on the concept of badness. A rule was defined to be *bad* if its consequent did not conform to what was expected of the cause stated in the antecedent of the rule. Thus points in Quadrant IV show that the rule is bad. *Strength of badness* was defined as the RMS perpendicular distance of points in Quadrant IV from the Bad Diagonal (line from Point (0, 1) to Point (1, 0)) on the TSD. Closer the points were to this diagonal, the worse the

rule was. *Probability of badness* (similar to probability of goodness) was the expected probability of points to occur in Quadrant II rather than in any other quadrant. If a rule was bad this probability increased. *Quantity of badness* indicated a scaled value of the amount of information present in Quadrant IV of the TSD.

A rule was said to be *incomplete* if the antecedent stated in the rule was not completely responsible for the effect stated in the consequent of the rule. Points in Quadrant I reflected this property. Thus *incompleteness* was defined as the RMS distance of points in Quadrant I from the Point (0.5, 0.5) on the TSD or the center of the TSD.

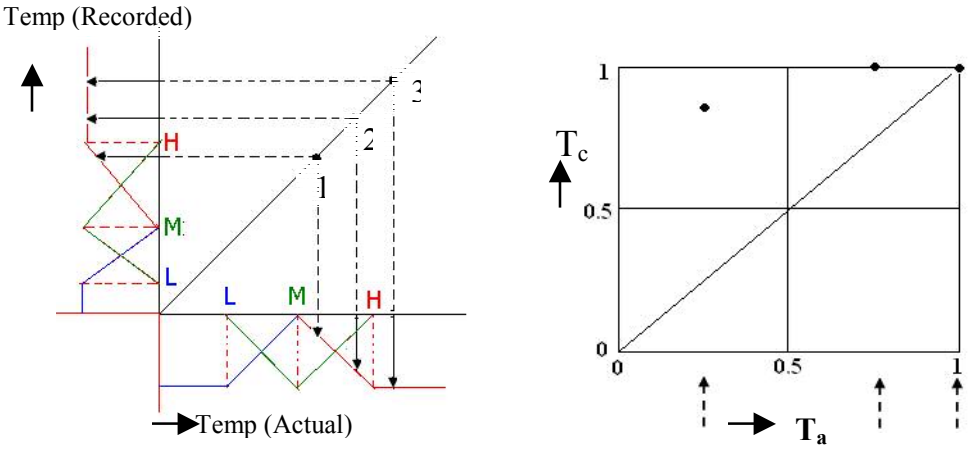
The metrics stated above reflected the quality of each individual rule. However, one final metric *Insufficiency* reflected the quality of the data-set in its entirety. This metric was used to determine if the antecedent of a rule was expressed in the data sufficient number of times to confidently state the goodness, badness or incompleteness of the rule.

Since four metrics were involved, a multi-objective optimization scheme was used. Two two-dimensional Pareto ranking schemes [2] were used and the best rules from each scheme were retained. Badness and goodness were used in the first ranking scheme while incompleteness and insufficiency were used in the second.

However, the metrics proposed by Sharma suffered from many drawbacks as discussed below and his analysis revealed the need for improved quality metrics.

1.4.1.7 Drawbacks of Sharma's metrics

1. Most of Sharma's metrics were based on the RMS distance of points from the upper-left to lower-right and upper-right to lower-left diagonals on the TSD. However, there is no reason that these 1:1 and 1: -1 diagonals should be used as ideal situations for goodness and badness respectively.

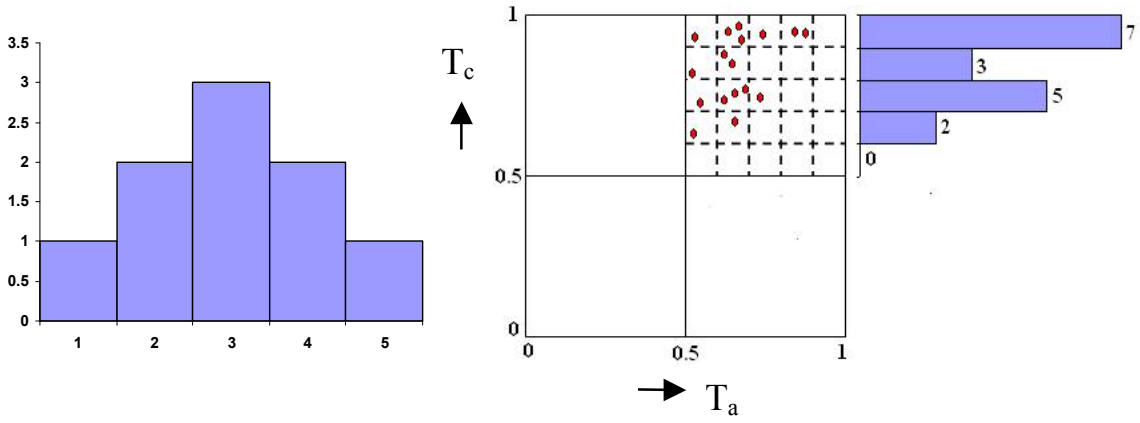


(a) (b)

Figure 6(a): Non-identical 'Low', 'Medium' and 'High' ranges for the antecedent (T actual) and the consequent (T Recorded). The 3 arbitrary points on the antecedent axis correspond to 3 points on the Consequent axis all of which in this case lie in the 'High' category. [μ_H (Temp Actual,1)=0.25, μ_H (Temp Actual,2)=0.75, μ_H (Temp Actual,3)=1.0 and μ_H (Temp Recorded,1)=0.8, μ_H (Temp Recordedl,2)=1.0, μ_H (TempRecorded,3)=1.0]

Figure 6(b): The 3 points when translated to a TSD. They don't lie on the 1:1 diagonal or close to it, but they still represent the good rule, "If 'actual' T (antecedent) is high then 'recorded' T (consequent) is high"

For example consider a variable T denoting temperature. An undeniably true rule would be, "If *actual* T (antecedent) is high then *recorded* T (consequent) is high". But for data belonging to this rule, to lay on the 1:1 TSD diagonal, the ranges of the fuzzy categories (*High, Medium, Low*) of the antecedent and consequent should be identical. This is illustrated in Figure 6 shown above.



7(a) 7(b)

Figure 7(a): An example Histogram depicting Normally- distributed data.

Figure 7(b): Sample data in Quadrant II of the TSD showing deviation from normal Gaussian distribution.

2. Some metrics assumed normal distribution of data. This is not a universally valid assumption since the data distribution could be varied.

   A histogram is a useful graph for exploring the shape of the distribution of the values of a variable. It was constructed [for each rule] for the distribution of points in each of the consequent zones of the TSD. Figure 7(a) shows a general example of normal data distribution in the form of a histogram with the x-axis (horizontal) denoting the number of the item, and the y-axis (vertical) denoting the value of the item. Figure 7(b) shows a sample of distribution of data in Quadrant II of the TSD. The adjacent inverted histogram is constructed by summing the number of points in each of the five depicted consequent zones. The x-axis (vertical) denotes the consequent zone and the y-axis (horizontal) denotes the number of points in each respective

consequent zone. The histogram thus shows that the distribution is far from normal distribution.

3. Insufficiency represented the quality of the data and was not a direct indication of the goodness/badness of a rule and hence should not have been a metric used during optimization.
4. Incompleteness evaluated the quality of the rule-set in whole and should not have been used as a metric for individual rules. Also, it did not measure the goodness of a rule.
5. The metrics had to be normalized.
6. They were not very simple to compute.
7. Since 4 metrics were used to determine the quality of a rule, it became a multi-objective problem and hence required the use of Pareto ranking schemes, making it complicated.
8. The metrics did not yield measure of confidence in the rule in prediction mode.

In this context, the present work aims at exploring improved quality metrics to identify useful If/Then rules, which can be believed to describe the process a hand accurately and to predict future behavior.

CHAPTER II

METHODOLOGY

2.1 The Concept of *Trips*

2.1.1 The problem

Sharma [3] developed a Truth Space Diagram (TSD) (refer to Section 1.4.1.4) which symbolically represents the quality of a rule. All the metrics hence calculated were based on the distribution or number of points in each of the four quadrants of the diagram. Figure 8 depicts a typical TSD and explains the significance of data in each of the four quadrants, as explained earlier in Section 1.4.1.5.

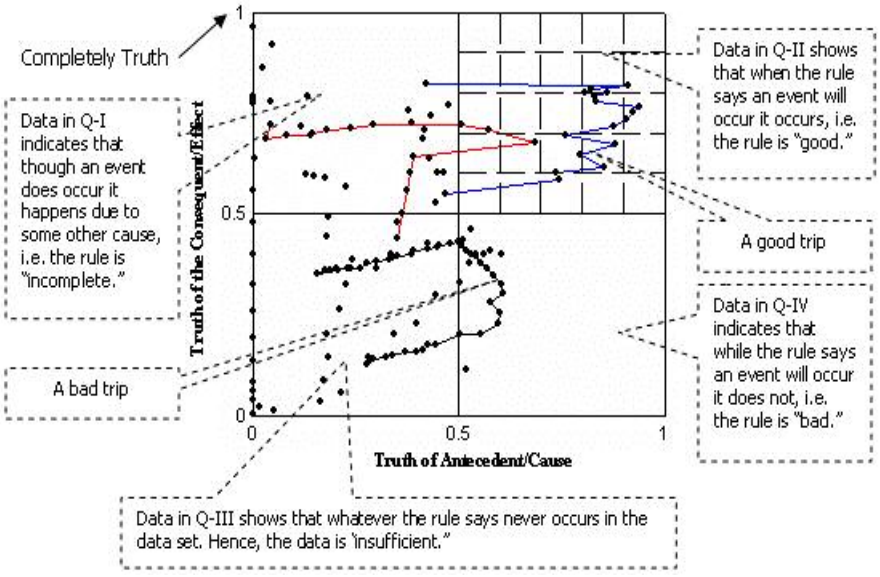


Figure 8: The Truth Space Diagram (TSD)

However, it is worthy to note that a few multiple points could be generated due to random data in the database and that there could be no actual process trends involved with their placement. Such random data may be generated due to noise in the process and spurious events, that shouldn't be allowed to contribute to the quality of a rule. Also, if a certain event persisted for a long time, it would cause many points to be placed in the TSD of the rule depicting the event. Most of the metrics proposed by Sharma, were functions of the *number of points* in a quadrant in the TSD. Thus the number of points in the TSD significantly affected the verdict about the *quality* of a rule. Eventually, the rule connected to that TSD would be over or under rated during the ranking process. Furthermore, there is no reason that closeness to the 1:1 diagonal on the TSD should be a measure of goodness of a rule, since this is true only in cases where the fuzzy categories are identical for both the antecedent and the consequent. This is illustrated in detail in Section 1.4.1.7.

2.1.2 The solution

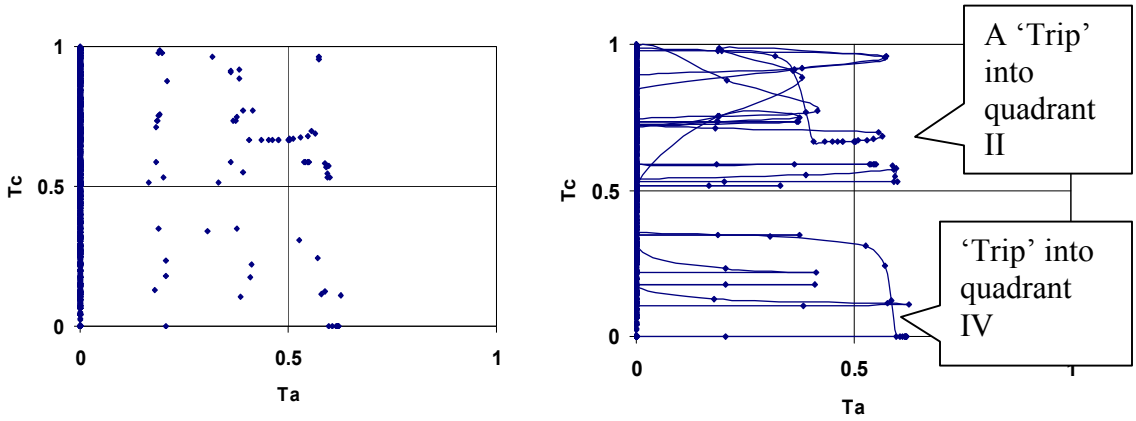
Discovery and isolation of trends in the process data is a possible solution to the problem of misjudging the quality of a rule. The temporal behavior of measured variables in a chemical process is the result of the interplay of many underlying phenomena and process conditions. Thus rules expressing this temporal behavior, if identified, reflect hidden mechanisms in the processes such as, process dynamics, external noise or operator-induced effects.

Thus the extraction of these temporal features or trends contained in measured data and their correct interpretation afterward provides stronger corroboration for good or

bad rules than analyzing the number of points in the respective quadrants, and their distance from the two principal diagonals defined by Sharma, as discussed in Section 1.4.1.6.

2.1.3 The tool – *Trips*

Trips isolate underlying trends in seemingly scattered data, which in turn help us make a better judgment on whether or not a rule in question is good. A *trip* within a quadrant, can be defined as the locus of a path traced by points into and out of the quadrant. It is a combination of monotonic increasing and decreasing behavior of the Ta and Tc values of points. In other words, it makes visible trends, in what appears to be a random scattering of points.



8(a) 8(b)

Figure 9(a): A TSD when trends are not identified in the points – seemingly scattered data

Figure 9(b): The same TSD from figure 9(a) when it denotes the loci of paths traced by points into and out of a quadrant – termed as 'trips'.

Figure 9 illustrates this concept. For this example 15 points in Quadrant II of the TSD should not imply that 15 separate events occurred. Instead, the four loci should be counted as four independent trips that corroborate the statement of the rule.

If multiple events or chain of events are identified from the patterns of points in the TSD then this would corroborate or support the goodness or badness of the rule statement more strongly than simple placement of multiple points in different quadrants of TSD, by the rule.

2.1.3.1 Threshold condition

To exclude spurious events from being called a trip the *Minimum Time* that a path into a quadrant needs to stay in the quadrant, is chosen arbitrarily. Minimum Time is a user-input value and determines how strict or lenient one wishes to be with respect to what qualifies as a trip. The Sampling Time is the time interval between two consecutive data samples, assumed to remain constant throughout the process. It is also a user-input value.

Then, the *Threshold* is defined as,

$$Threshold = \frac{Minimum\,Time}{Sampling\,Time} \quad , \tag{2.1}$$

where, Threshold is dimensionless(sec/sec).

Therefore, the Threshold is defined as the least number of successive points within a quadrant that can be termed as a *trip* into that quadrant. For the purpose of this project a Threshold value of 5 was chosen intuitively assuming the data sampling time was 1 time unit.

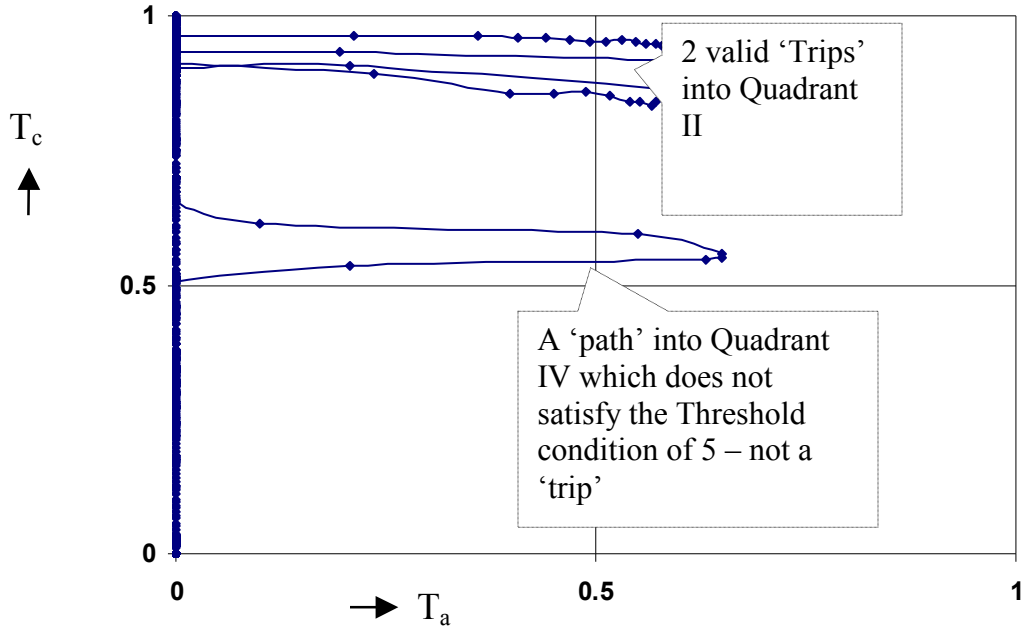


Figure 10: A Truth Space Diagram depicting 3 paths traced into Quadrant II, of which only 2 are 'Trips'

For example, in Figure 10 shown above, three paths can be traced into Quadrant II but only two of them are valid trips. One of the paths has only three successive data points in Quadrant II, which is lesser than the Threshold value of five points. It therefore does not qualify as a trip.

2.2 Corroboration

In the present work, the value 2 was chosen as the minimum number of trips a rule had to make into a quadrant for it to provide sufficient evidence of 'corroboration'.

2.2.1 Trips into Quadrant II

According to Sharma's work (Refer to Section 1.4.1.4), a point is said to be in Quadrant II if $0.5 \leq Tc \leq 1.0$ and $0.5 \leq Ta \leq 1.0$.The second quadrant consists of points

that have a high Tc value for a corresponding high Ta value. This implies that the consequent of the rule is actually caused by the antecedent expressed in the rule. More trips into this quadrant therefore corroborate the validity of a rule suggesting it was *good*.

2.2.2 Trips into Quadrant IV

Similarly, a point is said to be in Quadrant IV if $0.5 \leq Ta \leq 1.0$ and $0 \leq Tc < 0.5$. These points have a low Tc value for a corresponding high Ta value. Subsequently, this implies that the consequent of the rule does not comply with the effect expected from the antecedent of the rule, which in turn implies that the rule is bad or wrongly stated. More trips into this quadrant provide stronger evidence that the rule is *bad*.

2.2.3 Trips into both Quadrants (II, IV)

Some trips make their way into Quadrant II through Quadrant IV or vice versa. In such cases, the quadrant in which the threshold condition is satisfied is *dominant* and the trip is said to be made into this dominant quadrant. Figure 11 illustrates one such example.

2.2.4 Other cases

In the event of the threshold condition being satisfied in both the quadrants, the quadrant in which more time was spent is dominant.

Furthermore, in addition to satisfying the threshold condition, if equal time is spent in both the quadrants, a trip is said to be made into both.

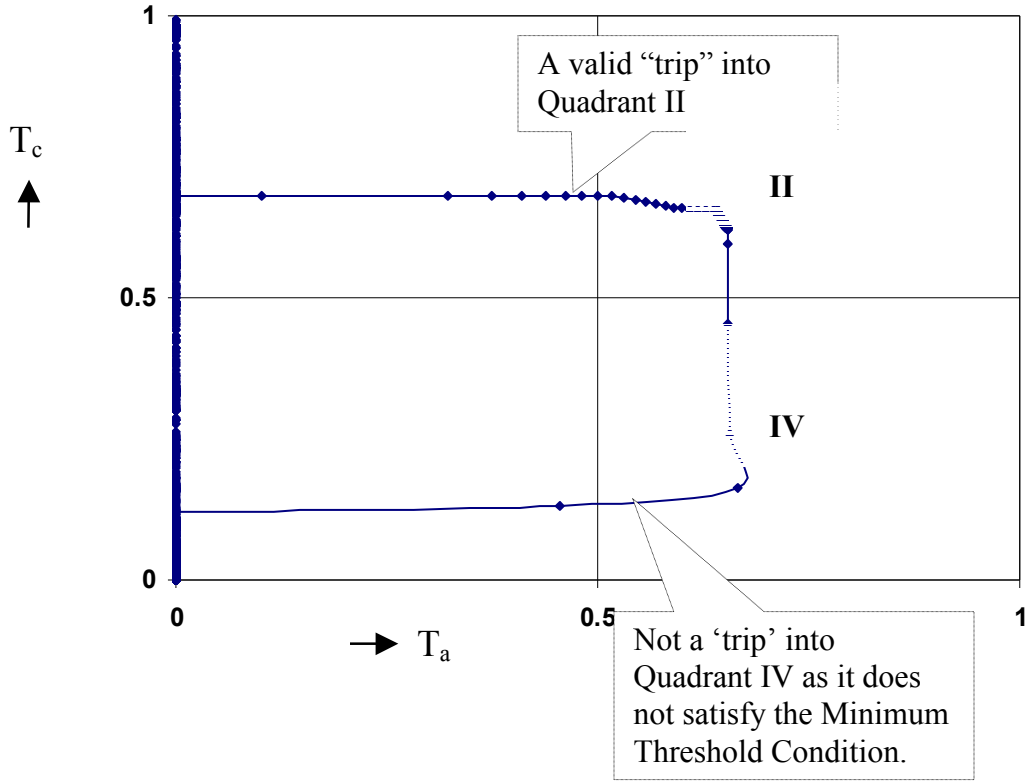


Figure 11: A Truth Space Diagram depicting a path that can be traced into both Quadrant II and Quadrant IV, of which only the path into Quadrant II is a trip

2.3 Numerical Metrics

It is the objective of this work to propose new and improved metrics for rule evaluation. The following is expected of them.

1. A metric should be *independent* of the size of the data-set. A rule that makes two good trips, zero bad trips and has 40 data points in the good quadrant, and another rule that also makes two good trips, zero bad trips but has 100 data points in the good quadrant, should essentially be equivalent, since the number of *events* that occurred are the same.
2. A metric should be robust to noise and anomalies in the data.

3. A metric should be fundamentally easy to compute.

4. A metric should directly reflect the quality of the rule (good or bad).

Truth Space Diagrams which depict the number of trips made into the II and the IV Quadrants are generated for all the possible rules, following which, the numerical metrics are calculated for each.

2.3.1 The Selection Metric

2.3.1.1 Merit

Merit is a metric proposed for the selection of good rules from the initial rule data base. Although it is the single objective for selecting rules, it is used in combination with the minimum corroboration condition for the selection of good rules (explained in Section 2.5.1.1). It is defined as the difference between the number of good and bad trips.

$$Merit = No.\,of\ \ Good\ Trips - No.\,Of\ \ Bad\ Trips \tag{2.2}$$

Qualitative Meaning: A positive value of Merit implies the presence of more good trips than bad. The higher the value, the higher the evidence for the rule being observed often. It does not require normalization since it is the only metric used in the optimization process. It consists of both qualitative and quantitative information about the rule. Also, it uses the concept of *trips* which makes it independent of the number of data points in the II and IV Quadrants. It is also independent of the total number of data points in the TSD.

2.3.2 The Prediction Metric

Until this point, all calculations and analysis involved use of *Historical Data* where both T_a and T_c were known. The *Prediction Mode* is a new mode of data analysis. In this mode, only T_a of *New Data* are analyzed to determine the rules being activated and to predict future outcomes of those T_a.

2.3.2.1 Expectation

This is a metric proposed to predict the likelihood of the truth of consequent occurrences given the truth of antecedent for each of the rules in the initial rule-base. It is calculated based on information gathered from historical data. For the calculation of *Expectations* the II and the IV Quadrants in the TSD are divided into grids as shown in Figure 12 below.

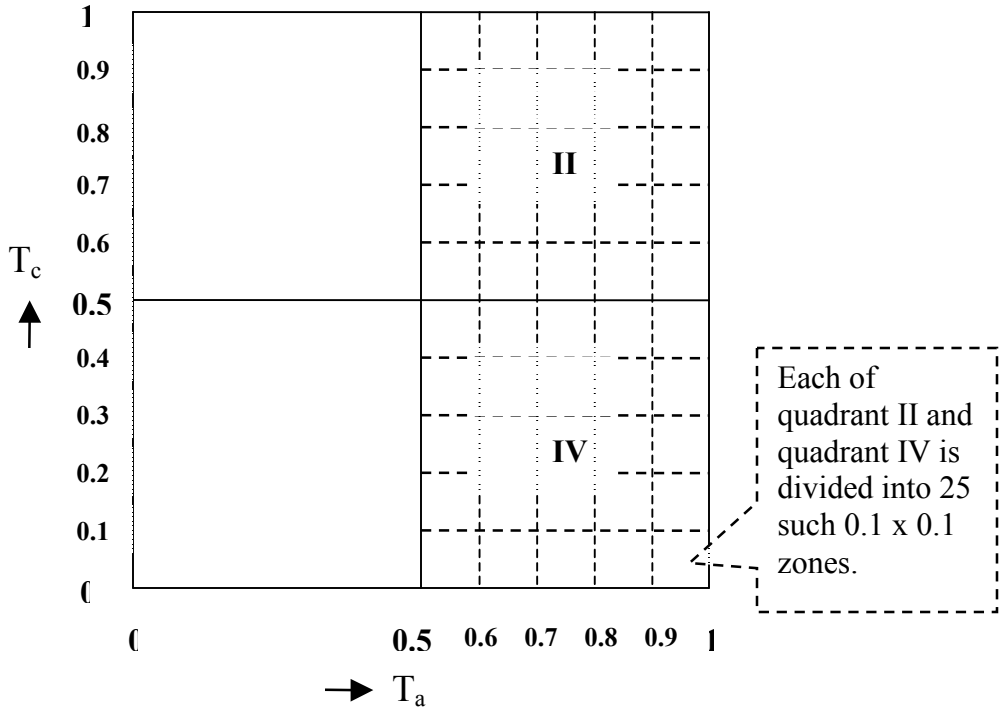


Figure 12: A TSD showing the division of Quadrants II, IV into a total of 50 zones of size 0.1 x 0.1. The Ta axis is divided into five zones of size 0.1 each (0.5-1), and the Tc axis is divided into ten zones of size 0.1 each (0-1).

Thus the zones can be represented as follows

Note: $(A, B] = \{x \mid (x > A) \bigcap (x \leq B)\}$

$[A, B] = \{x \mid (x >= A) \bigcap (x \leq B)\}$

Five T_a zones: (0.5, 0.6], (0.6, 0.7], (0.7, 0.8], (0.8, 0.9], (0.9, 1]

Ten T_c zones: [0, 0.1], (0.1, 0.2], (0.2, 0.3], (0.3, 0.4], (0.4, 0.5], (0.5, 0.6], (0.6, 0.7], (0.7, 0.8], (0.8, 0.9], (0.9, 1]

Specifically, data distribution in each of the ten consequent zones for the five antecedent zones for each rule in the initial data-base is analyzed using a histogram and normalized. Figure 13 illustrates the distribution of points in Quadrants II, IV based on historical data, as an example.

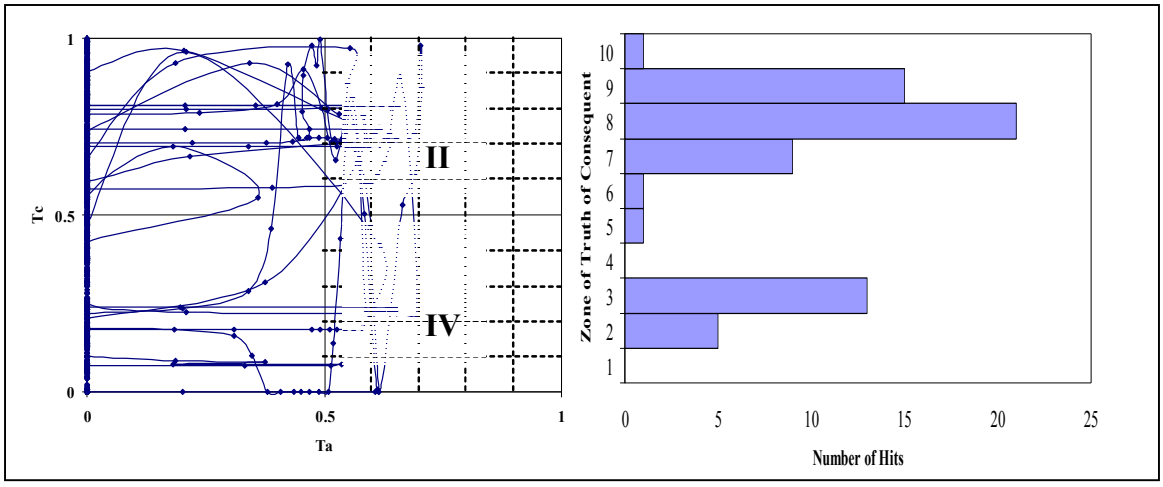


Figure 13: Example of distribution of points in the II, IV quadrants based on 'Historical Data'. The adjoining histogram represents cumulative historical hits in each of the ten consequent zones. Data in individual Ta zones is to be normalized and used in conjunction with the antecedent hits in the 'New Data' to calculate the 'Expectations' as shown in Figure 14.Note: Only points that contribute to making trips(good or bad) are considered for all calculation purposes.

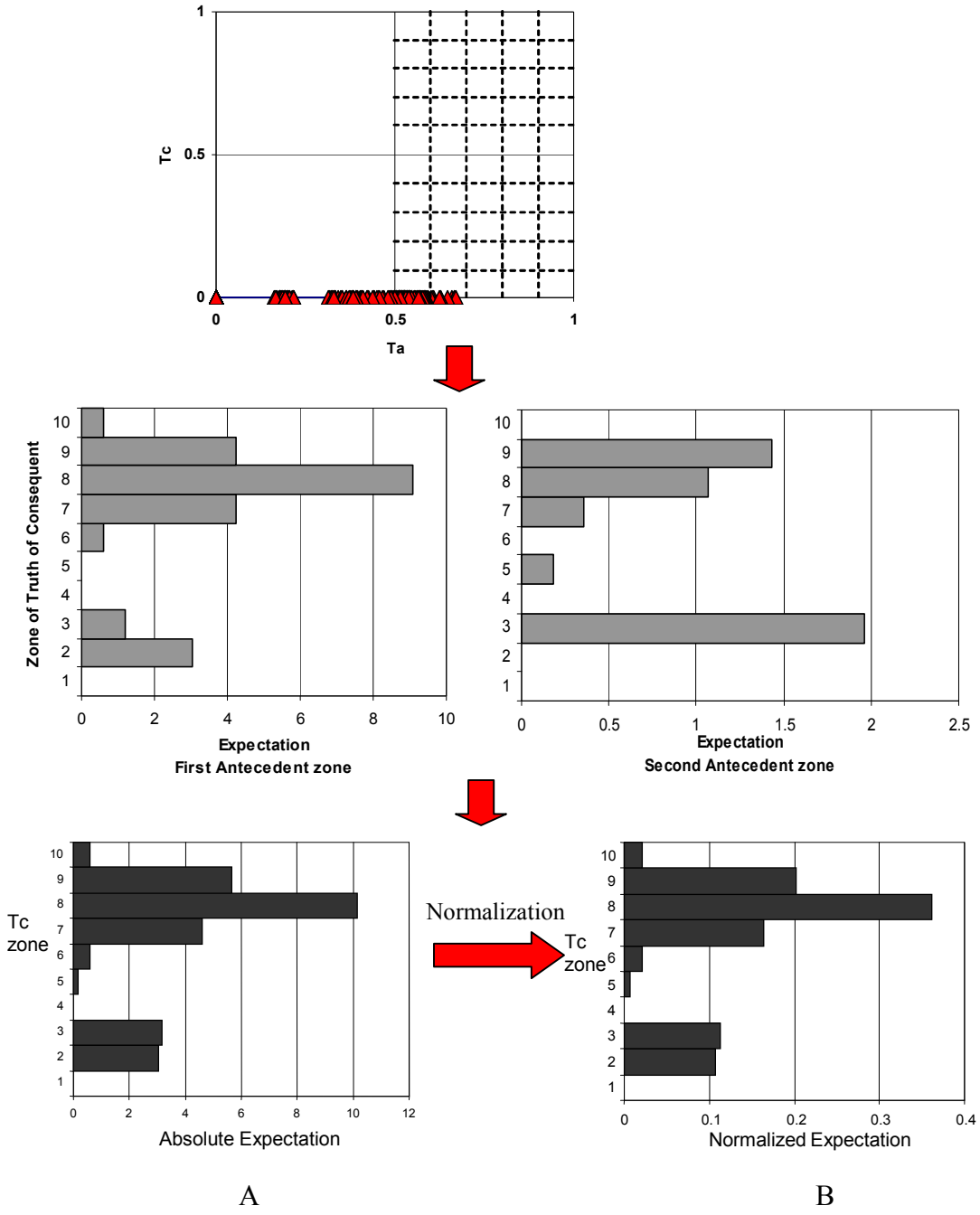


Figure 14: Based on the antecedent hits in the two Ta zones, expectations are calculated for each of the two Ta zones to yield the cumulative absolute 'Expectation' of Occurrences of Truth of Consequent in the ten zones, based on the Antecedent Hits of 'New Data'. The values of these 'Expectations' are then normalized in the range (0-1) to yield the cumulative normalized 'Expectations' for each of the ten consequent zones.

Once the historical data is analyzed, new data is collected. This data is put through the data-processing procedure described in Section 1.4.1.2. The number of *new* antecedent hits in each of the five antecedent zones is then calculated based on the truth of antecedents. This information is then used in conjunction with the historical normalized data, to yield the normalized expected occurrences of the truth of consequent. As seen in Figure 14, antecedent hits were made in two Ta zones. 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.

2.4 Calculations

Based on the zones (as defined in Figure 12) the following values are calculated.

2.4.1 From Historical Data

The first historical data, that was used to select rules is processed.

1. The number of points or hits in each of the five zones of the antecedent, irrespective of the T_c value, is recorded in a column vector *Hits*.
2. For each of the five Ta zones, the number of hits in each of the ten T_c zones is calculated. This results in a 10 x 5 matrix *NumPoints (number of points*) with values of the number of hits in each of the 50 zones.

3. Each column of the above matrix *NumPoints* is considered to be a column vector $\vec{V}_i$**,** where the subscript *i* denotes the column number, and ranges from 1 to 5. These vectors are normalized, to yield another matrix *NormNumPts*, as follows.

   a. **Normalization :** Each column of the *NumPoints* matrix is comprised of a column vector consisting of ten elements and is normalized as follows,

$$\vec{V}_{inorm} = \left[\frac{\vec{V}_i}{\vec{1}^{T} \cdot \vec{V}_i} \right], \tag{2.3}$$

   where i = 1 to 5 and, $\vec{1}$ is 10-element vector,

$$\vec{1} = [1,1,1,1,1,1,1,1,1,1]$$

   b. **Concatenation**: The complete normalized matrix *NormNumPts*(10 x 5**)** is formed by concatenating each normalized column vector obtained in Equation 2.3.

$$NormNumPts_{10\times5} = \left[\left(\vec{V}_{1norm}\right), \left(\vec{V}_{2norm}\right), \left(\vec{V}_{3norm}\right), \left(\vec{V}_{4norm}\right), \left(\vec{V}_{5norm}\right) \right]_{10\times5} \tag{2.4}$$

2.4.2 From New Data

After the historical data defines Equation 2.4, begin collecting new data.

1. The number of Antecedent Hits in each of the five zones is recorded in a column vector $\vec{predHits}$**.** This column vector is converted into a 5 x 5 diagonal matrix, *PredDiagHits*.

2.4.3 Calculation of the 'Expectations' Matrix

1. The normalized number of hits in each of the ten consequent zones, obtained from the *Historical Data* is then multiplied by the number of *New* antecedent hits to yield *product* which is a 10x5 matrix.

$$product_{10x5} = NormNumPts_{10x5} \cdot PredDiagHits_{5x5}\,, \tag{2.5}$$

2. The elements of *product* are totaled along the row to yield the absolute *Expectation* [0-1] of Truth of Consequents in each of the ten T_c zones, based on the *New a*ntecedent hits and *Historical* information of T_c distribution.

$$Expectation_{10x1} = \sum_{row-wise} product_{10x5} \tag{2.6}$$

3. These absolute Expectations are then normalized as described by Equation 2.7 to yield the normalized Expectations, *NormExpectation* (10x1).

$$NormExpectation_{10x1} = \frac{Expectation_{10x1}}{\sum_{column-wise} Expectation_{10x1}} \tag{2.7}$$

2.4.4 Weighted Mean Average

The weighted mean average (*PredMean***)** of the Expectations is then calculated as follows, for each rule, within 95% confidence limits. Although the distributions were not assumed to be Gaussian, the weighted mean average was used as an approximation. Late breaking research suggests however, that the Median would be a better representation owing to the fact that the distribution need not be Gaussian (Refer to Section 5.2).

$$Totsum = \sum_{Column-wise} Expectation_{10x1} \tag{2.8}$$

$$PredMean = \frac{(0.05 xExpectation(1,1) + 0.15 xExpectation(2,1) + ... + 0.95 xExpectation(10,1))}{(TotSum)} \quad (2.9)$$

2.4.4.1 95% Confidence Limits

Statistical t-values were not used to determine the confidence limits as there were many occurrences where the distribution of data in the zones deviated from normal behavior. This was observed by plotting cumulative histograms for rules, depicting the number of points in each of the ten consequent zones of Quadrants II, IV, with the zone of Tc represented by the x-axis and the number of points represented by the y-axis. The shape of the histogram depicted the non-Gaussian distribution of points in the TSD. In the present work, a simple interpolation procedure was used to determine the 95% confidence instead.

2.4.4.2 The interpolation procedure

1. The Expectation values for each Tc zone are cumulatively added, yielding 10 *CumulExp* values
2. The *Cumulative Distribution Function (CDF*) for each of the 10 values of the Expectation matrix is determined as follows:

$$CDF_i = \frac{CumulExp_i}{Max(CumulExp)}, \quad (2.10)$$

where, i = 1 to 10.

Figure 15 depicts the conversion of Expectation values (represented in the form of a histogram) into a cumulative plot which depicts CDF on the y-axis and Tc on the x-axis. The cumulative graph also depicts the lower and upper

95% confidence limits. A CDF of 0.025 is the lower confidence limit and A CDF of 0.975 is the upper confidence limit.

3. Figure 16 denotes the zoomed in version of the lower part of the cumulative graph denoted in Figure 15. It depicts the Tc ($T_{c\ 0.025}$) corresponding to a CDF of 0.025 positioned between the Tcs (T_i, T_{i+1}) corresponding to the outer limits i, i+1 of the Tc zone in which $T_{c\ 0.025}$ occurs. Similarly, the Tc corresponding to a CDF of 0.975 ($T_{c\ 0.975}$) will be positioned between the Tcs (Ti, T_{i+1}) corresponding to the outer limits i, i+1 of the zone of in which $T_{c0.975}$ occurs. The values of $T_{c\ 0.025}$ and $T_{c\ 0.975}$ need to be determined. The properties of similar triangles are used to achieve this. For the lower confidence limit for example, as seen in the Figure 16, triangles ABC, ADE are similar and by principles of geometry the value of the only unknown variable $T_{c\ 0.025}$ can be determined.

   Tc corresponding to the Lower Confidence Limit of 0.025 is calculated as follows.

$$\text{If } CDF_{i+1} > 0.025 \text{ and } CDF_i < 0.025 \text{ Then,}$$
$$T_{c_{0.025}} = T_i + T_{i+1}\left(\frac{0.025 - CDF_i}{CDF_{i+1} - CDF_i}\right) \tag{2.11}$$

   Similarly, Tc corresponding to the Upper Confidence Limit of 0.975 is calculated as follows.

$$\text{If } CDF_{i+1} > 0.975 \text{ and } CDF_i < 0.975 \text{ Then,}$$
$$T_{c_{0.975}} = T_i + T_{i+1}\left(\frac{0.975 - CDF_i}{CDF_{i+1} - CDF_i}\right) \tag{2.12}$$

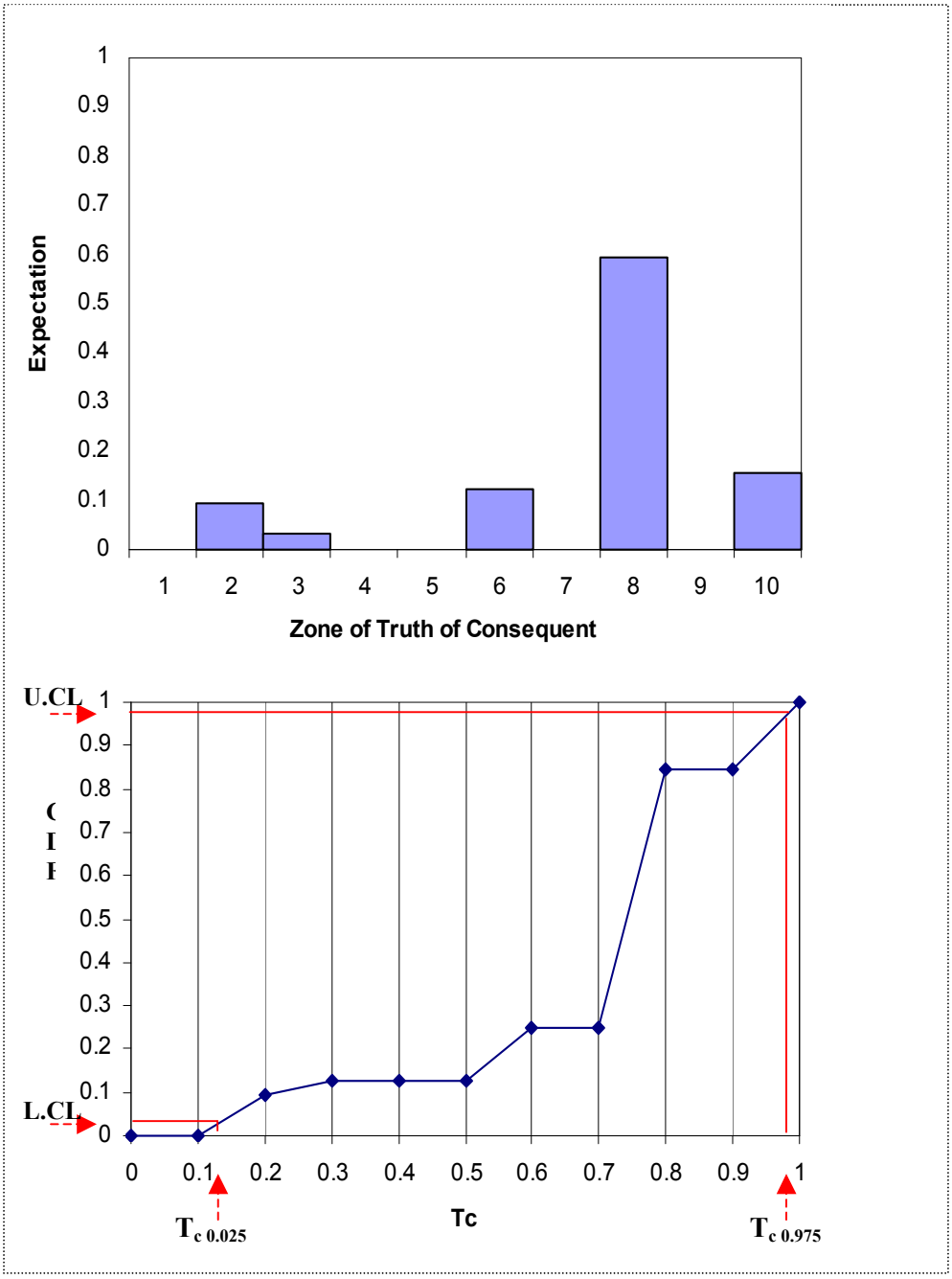


Figure 15: A Cumulative graph representing the Histogram. The graph uses the Upper Confidence Limit (U.CL) = 0.975 and the Lower Confidence Limit (L.CL) = 0.025 to determine $T_{c\ 0.025}$ *and* $T_{c\ 0.975}$ *respectively.*

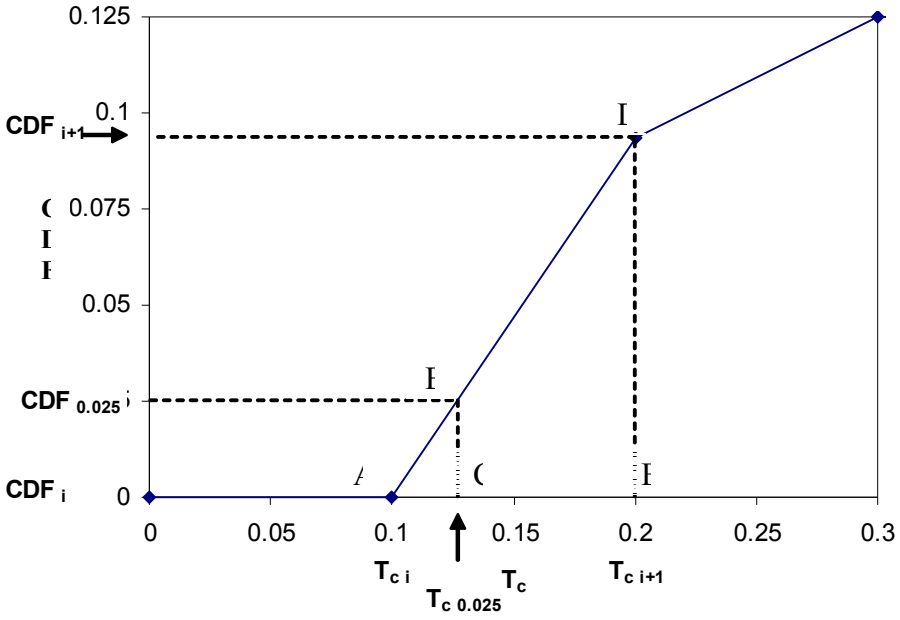


Figure 16: The Interpolation procedure. The Lower Confidence Limit from Figure 15 is zoomed in, with the axes denoting Cumulative Distribution Factors on the y-axis and their corresponding Tc on the x-axis. By Geometry, the similar triangle property is used to estimate $T_{c\ 0.025}$

2.4.5 Window Length for Prediction

New Data was collected for 65 seconds at a time, where a 60 second (1 minute) window was allowed for the calculation of maximum persistence. New antecedent hits however were calculated only for the last 5 seconds labeled as *NOW* in Figure 17 below. At this point all prediction calculations (as described in Sections 2.4.2, 2.4.3, 2.4.4) were made to yield the weighted mean 'expectation' after say *t* seconds, where *t* was defined by the *delay* associated with the rule in question(short or medium or long). The entire 65 second window then moved by a time interval, defined by the shortest delay, along the timeline and the above procedure was repeated. The bold lines in Figure 17 denote the prediction window to start with. The dotted lines in Figure 17 denote the movement of the prediction window along the timeline.

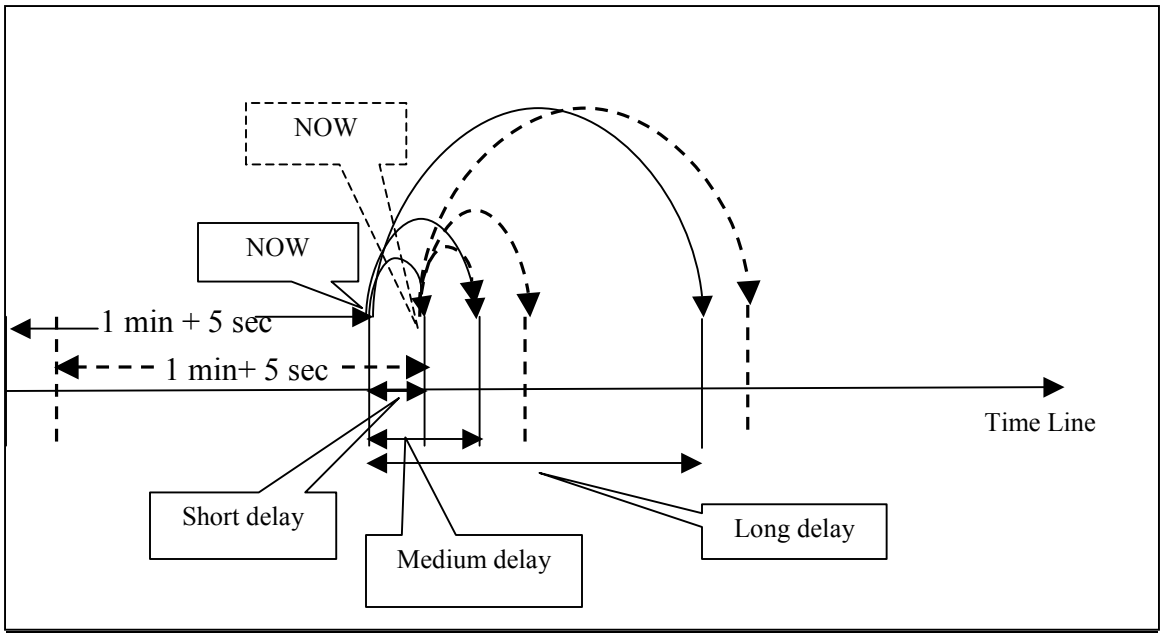


Figure 17: Window length for prediction.

2.5 Overview

The entire selection and prediction procedure can be described as follows.

2.5.1 Rule-base Optimization/Selection

An 'exhaustive-search' procedure was used to generate all possible rules (combinations of antecedents and consequents).This was done to check the robustness of the suggested metrics and to check if the metrics produce desirable results.

The rule-base thus generated needs to be optimized to choose the best rules among them. The Numerical Metric *Merit* served the purpose of evaluation of all the rules.

Two criteria are all that were used to optimize the initial rule-base. Any rule satisfying these conditions was termed as *Good* and any rule with evidence of not

satisfying the condition was termed as *Bad*. Rules that showed no evidence of an occurrence of an event were not commented upon.

2.5.1.1 Criteria of Acceptance of Rules

The Metric *Merit* was to aid this selection process.

1. The rule in question should show proof of *Corroboration* by making at least two good trips.
    a. *Two* was chosen as a reasonable number for corroboration of the goodness of a rule because it proved that a good trip wasn't made just by chance, as there was more than just 1 instance of a good trip.
2. The Merit of the rule should be >= 1
    b. This also is not being claimed as a universal number. It seemed reasonable because one good trip more than the bad proves that the rule is inclined more towards goodness than badness. Rules that had equal good trips and bad trips showed no proof of being categorically good and so were deemed *bad*.

Thus the initial rule-base was condensed to a smaller one, in which each rule could be believed to represent the process at hand, accurately.

It needs to be clarified here though, that more instances of corroboration does not mean the rule is better than another good rule with lesser instances of corroboration (Refer to Section 5.2).

2.5.2 Prediction

Historical information in the form of Normalized data distribution for all rules was used to predict the expected occurrences (Expectation) of the Truth of Consequent, given the antecedent hits captured from New data.

The weighted mean average was also calculated within 95% confidence limits.

CHAPTER III

PROGRAMMING METHODOLOGY

A program was written in VB 6.0 (listed in Appendix A) to carry out all the operations discussed in the Methodology in Chapter II. It consists of 'forms' (sub-programs) which are interconnected to each other, and each form in turn is made up of many successive subroutines, each with a specific function. These various 'forms' serve as the Graphic User Interface (GUI).

Figure 18 depicts the over all algorithm and the functions of some of the most important subroutines. Each process block corresponds either to a subroutine that was used to perform the function it denotes or a smaller part of the code which may be important. The code for the truth-space calculations and the creation of all possible rules were inherited from Sharma's work [3].

To start with, the user is asked to make a choice between the selection mode and the prediction mode. Figure 19 represents this GUI (choose.frm).

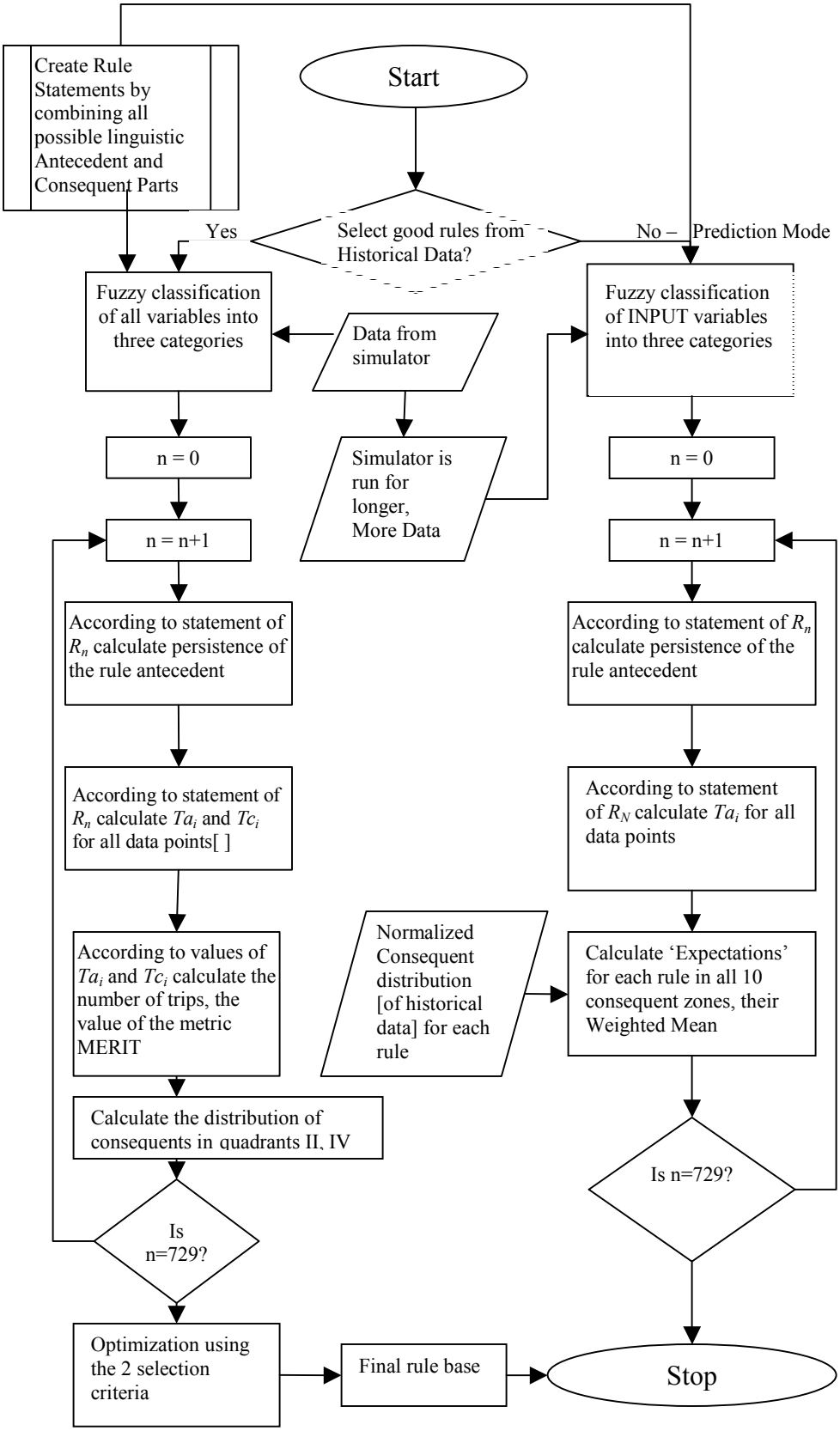


Figure 18: The algorithm used for the program.

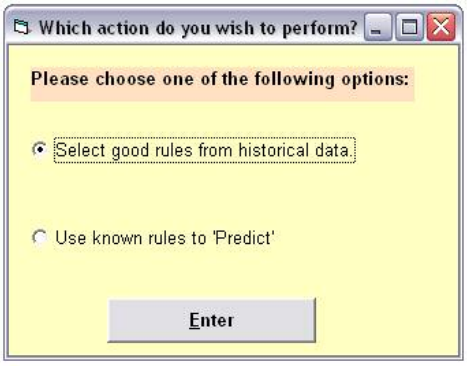


Figure 19: GUI for choosing between the selection and the prediction mode. The name of this form is choose.frm

The *selection* mode uses an input file (.csv format) consisting of initial data (Historical Data) collected from the simulator. It is used to determine the best rules that represent the process. The *prediction* mode uses another input file (.csv format) consisting of data obtained by running the simulator for longer (New Data). This is where the consequents are predicted for the new antecedents. Each of the above-mentioned modes uses a different GUI.

3.1 Selection Mode

The blocks under *Yes* in Figure 18, depict the selection part of the algorithm. This mode works with the initial batch of data obtained from the simulator, considered as historical data.

Figure 20 depicts the GUI (SelMode.frm) for this mode, and as can be seen, it consists of three parts that perform three important operations. The following discussion describes important aspects of the above-mentioned parts.

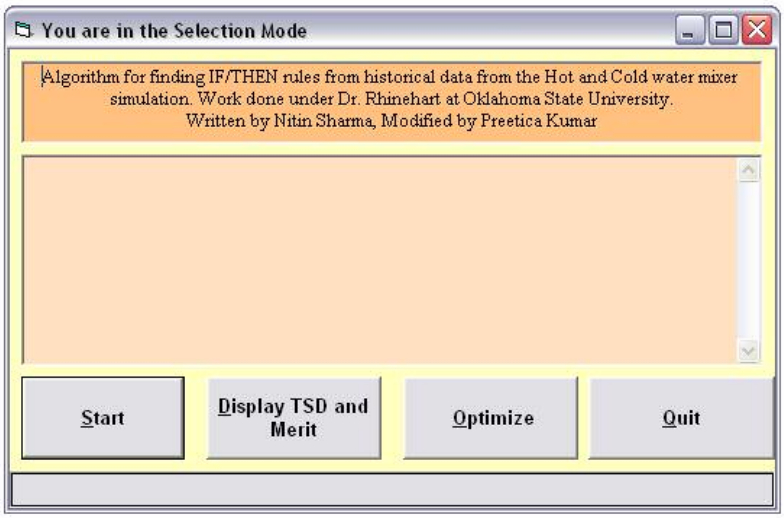


Figure 20: GUI for the selection mode. The name of this form is SelMode.frm.

3.1.1 The S*tart* button

The button labeled *Start* when hit, performs Truth Space calculations (inherited from Sharma), calculates the number of trips into Quadrants II (good trips), Quadrant IV (bad trips) and the metric *Merit*, and the distribution of points in each of the 50 0.1 x 0. 1 grids in Quadrants II, IV together.

The reader is referred to [3] for details about the Truth Space calculations and the loop structure.

Once Ta, Tc were calculated for every rule, the numbers of good and bad trips made by each rule were calculated, within the same loop structure used by Sharma. Figure 21 depicts a code snippet calculating the number of trips into Quadrant II.

```
If ta(i) > 0.5 And ta(i) <= 1 And tc(i) > 0.5 And tc(i) <= 1 Then
        counter(i) is used to calculate the number of consecutive points
        into the Quadrant
   If i = 1 Then
      counter(i) = 1
   Else: counter(i) = counter(i - 1) + 1
   End If
        tot is the threshold value input by the user
   If counter(i) = tot Then
         goodtrip(r) = goodtrip(r) + 1
   End If
ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) >= 0 And tc(i) <= 0.5 Then
   counter(i) = 0
ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) > 0.5 And tc(i) <= 1 Then
   counter(i) = 0
ElseIf ta(i) > 0.5 And ta(i) <= 1 And tc(i) >= 0 And tc(i) <= 0.5 Then
   counter(i) = 0
End If
```

Figure 21: Calculation of trips into quadrant II (good trips) based on the threshold value input by the user.

Once the Merit is calculated, the numbers of points in each of the ten consequent zones (for each of the five antecedent zones) are calculated and so is the number of antecedent hits in each of the five antecedent zones. These calculations are made in accordance with those in Chapter II.

3.1.2 The *Display TSD* and *Merit* button

The button labeled *Display TSD and Merit* as the name suggests displays the TSD of each of the 729 rules (DispTSD.frm), and depicts the number of trips made into Quadrants II, IV. The value of the metric *Merit* is also displayed. Figure 22 depicts the GUI for this display mode. Trips into Quadrant II were denoted by green and trips into Quadrant IV were denoted by red. There is also an option for zooming into the II or the IV Quadrants in the event that a lot of trips are made.

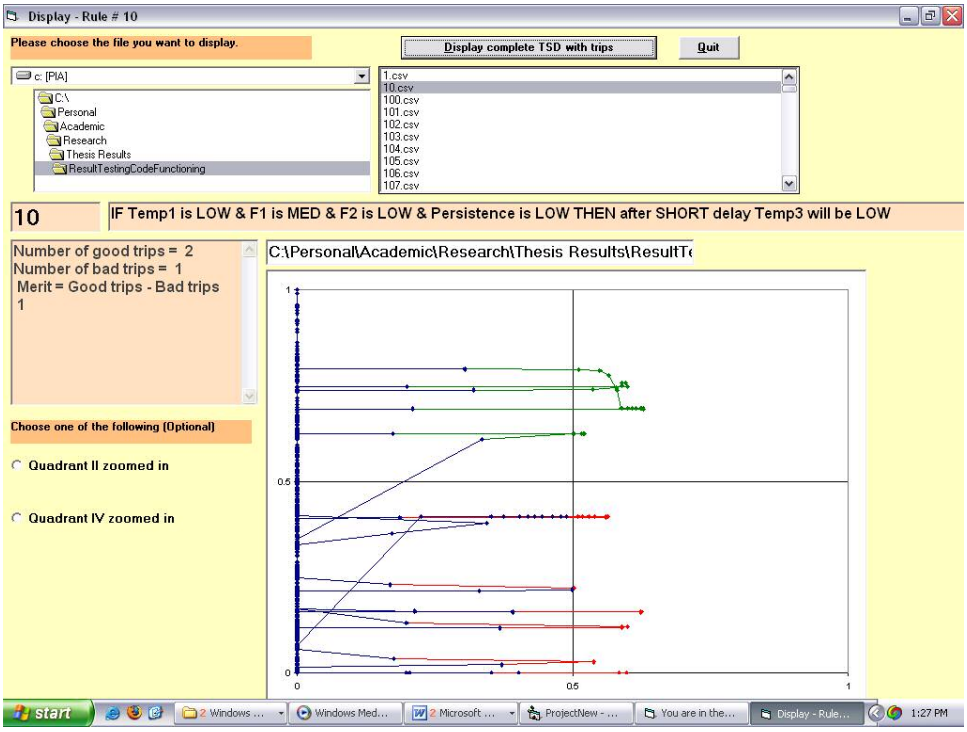


Figure 22: GUI for Displaying the TSD. The name of this form is DispTSD.frm

3.1.3 The *Optimize* button

This button when clicked optimizes the initial rule-base using the two selection criteria (minimum corroboration and minimum *Merit*) described in Chapter II. Figure 23 depicts the algorithm used for optimization.

```
For i = 1 to 729
        'Minimum corroboration and Minimum Merit – Two selection criteria
        If (number of good trips) > 1 and Merit > =1 then
                Select Rule
        End if
Next i
```

Figure 23: Algorithm for Optimization using the two selection criteria.

Figure 24 displays the GUI (opti.frm) for optimization. All the selected rules are displayed with the number of good and bad trips they make.

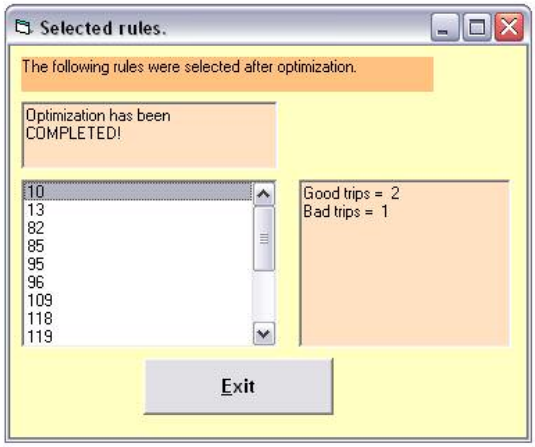


Figure 24: GUI for Optimization, depicting the selected rules. The name of this form is Opti.frm

The present work recommends adding the *ranking* phase in this stage of the program. This brings us to the end of the selection mode.

3.2 Prediction Mode

The blocks under *No* in Figure 18, depict the prediction part of the algorithm. This mode works with new data (data obtained by running the simulator for longer), for which only the antecedents are known.

Figure 25 depicts the GUI (PredMode.frm) for this mode and it consists of two parts that perform two important operations. The following discussion describes important aspects of the above-mentioned parts.

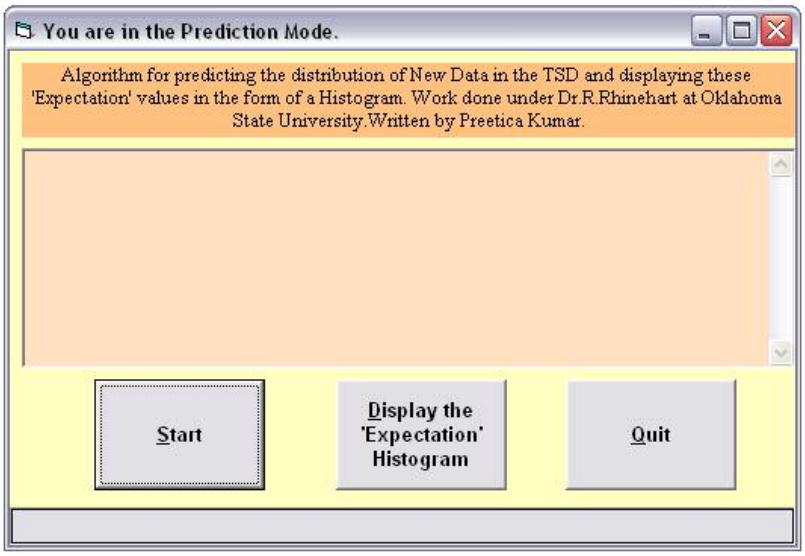


Figure 25: GUI for the prediction mode. The name of this form is PredMode.frm.

3.2.1 The *start* button

This button when clicked in the prediction mode performs truth space calculations only for the antecedents of new data. The total number of antecedent hits in each of the five antecedent zones is determined.

Note: At a certain point in time when an event is long past, its corresponding long-past antecedent hits have no more influence on the future. The *Time* aspect of prediction is not handled in the present work.

The consequent distribution of historical data obtained in the selection mode, is normalized here and used in conjunction with the new antecedent hits to determine the value of the metric Expectation for each of the 729 rules. The weighted mean average of these *Expectations* in each of the ten consequent zones, is also reported for each rule within 95% confidence limits. These calculations are explained in detail in Chapter II.

3.2.2 The *Display the 'Expectations' Histogram* button

This button when clicked, displays the *Expectations* calculated by the *Start* button of the prediction mode, in the form of a histogram for each rule. It also displays the weighted mean average of the *expectations* (in each of the ten consequent zones) within 95% confidence limits. For comparison purposes, the weighted mean average obtained from Historical Data in the selection mode for that rule is also displayed. Figure 26 displays the GUI for this display. (DispExpect.frm). On the upper right corner of the form, this GUI also gives a 'verdict' as to whether the rule was good, bad, insufficiently expressed in data or showed insufficient corroboration, according to historical data used in the selection mode. Figure 27 depicts a code snippet that decides what this verdict should be.

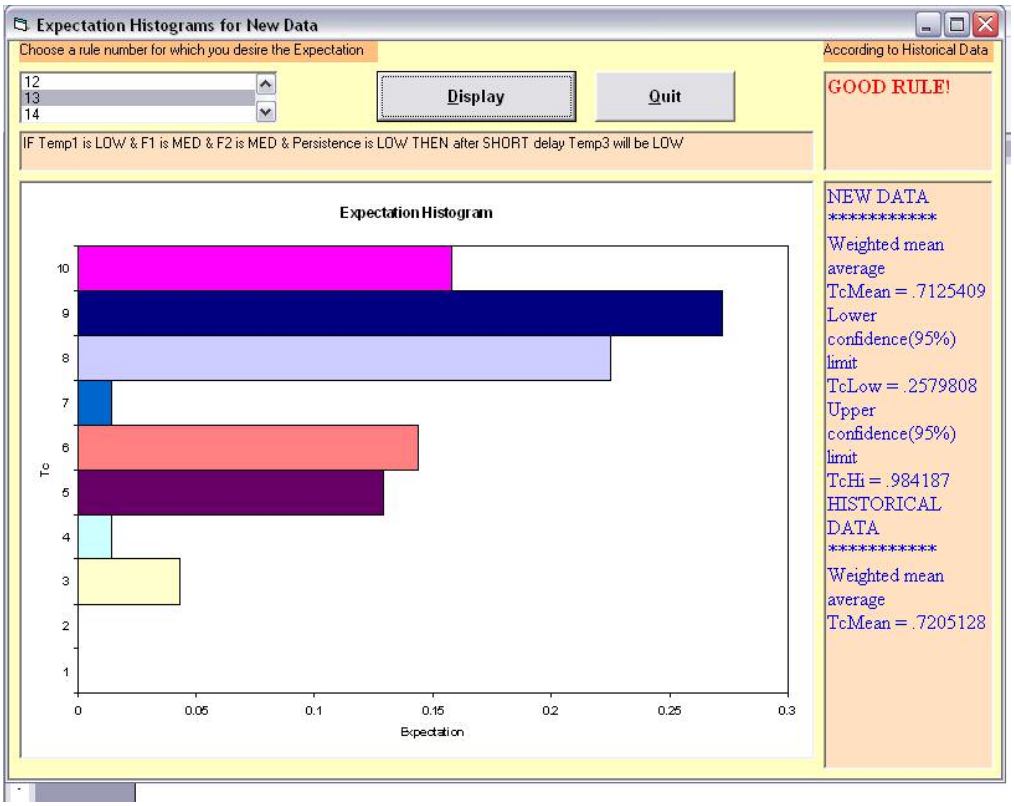


Figure 26: GUI for the displaying the 'expectations' of each rule. The name of this form is DispExpect.frm.

```
For i = 1 to number of rules selected
'If a rule is selected by the optimization process, it is a good rule.
   If Val(List1.Text) = number(i) Then
   Text3.Text = "GOOD RULE!"
   Exit For
   End If
Next i
'If the number of good and bad trips are 0
If good(rownum) = 0 And bad(rownum) = 0 Then
        Text3.Text = "Not Expressed Sufficiently In Data"
'If the number of good trips is 1 and the number of bad trips is 0
ElseIf (good(rownum) = 1 And bad(rownum) = 0) Then
        Text3.Text = "Insufficient Corroboration"
'If the number of good trips is less than the number of bad trips
ElseIf good(rownum) <= bad(rownum) Then
        Text3.Text = "BAD RULE"
End If
```

Figure 27: Making a 'verdict' about a rule, depending on the number of good and bad trips made by it.

3.3 Known Issues

There are a few problems that the user of this program needs to be aware of. These problems arise because of glitches in interaction of the program with the Windows Operating System or the user. Although most of these errors were trapped, there are a few more that need to be dealt with.

1. When the *Start* button or the *Display* button is hit either in the selection or the prediction mode, the computer hangs. This happens because a large part of the computer's memory is used up for this process. This effectively means that while this application is running, it is advisable that the user

run no other application. The user should wait till the program ends. The time this takes will depend on the computer's speed.

2. All instances of the EXCEL process have to be ended manually by hitting 'End Task' on the task manager (pressing ctr+alt+del) before hitting the display button the second time consecutively. This needs to be done because sometimes the embedded EXCEL object that the program uses remains open. Also when too many instances of EXCEL are loaded, Microsoft EXCEL stops responding to the program. Refer to Issue 6 in [3]. This is the only major issue in this code.
3. The user is referred to [3] for more issues in this code. Those issues apply to the present program also as those parts of Sharma's code were inherited in the present work.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Definition of Good and Bad Rules

The exhaustive-search method (as explained in Section 1.4.1.3) was used in order to determine the best metrics to select only good rules and reject all bad rules. This relies on the definition of *good* and *bad* rules.

A rule is said to be *good* if the logical relationship between its antecedent and consequent translates into an observed and corroborated cause-and-effect relationship in the data.

Consider Rule 148 for example. Its Antecedent states that Hot water at a high temperature of T1 flowing at a medium rate of F1, mixes with cold water also flowing at a medium rate of F2 and that this mixing persists for a low time interval. Logically this should result in a short delay and a medium final temperature of T3, as is in fact stated by the rule. Therefore, Rule 148 is a good rule.

As seen in the TSD shown in Figure 28, both selection criteria that were defined in Section 2.5.1.1 to select rules, are satisfied by this rule.

1. The rule makes four good trips (trips into Quadrant II) which are more than the minimum criterion of two, thus providing ample corroboration.
2. The Merit of the rule is 4, which is greater than the minimum criterion of 1 (no trips were made into Quadrant IV).

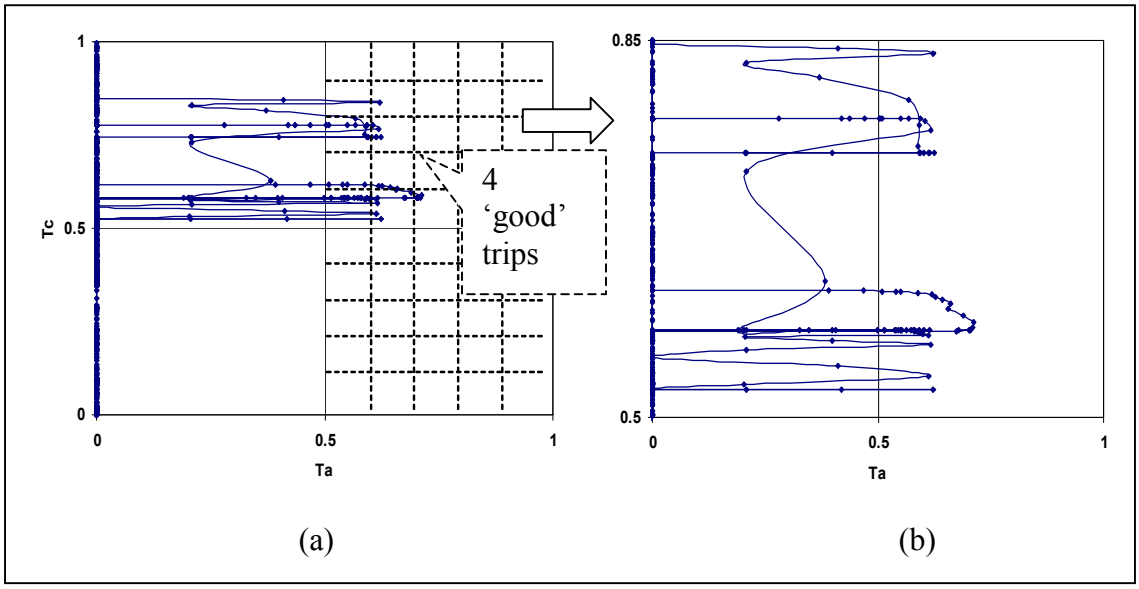


Figure 28(a): TSD for Rule148 showing 4 'good' trips and 0 'bad' trips - IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED

Figure 28(b): Quadrant II zoomed in to depict the 'good' trips. Four trips satisfy the threshold condition of five.

Consequently, a rule is said to be *bad* if it is inconsistent with the actual process phenomena. Rule 229 is one such example. It shares the same antecedent as Rule 148 but states that after a short delay, the final temperature T3 will be high. This is not a logical consequence and therefore, Rule 229 is a bad rule.

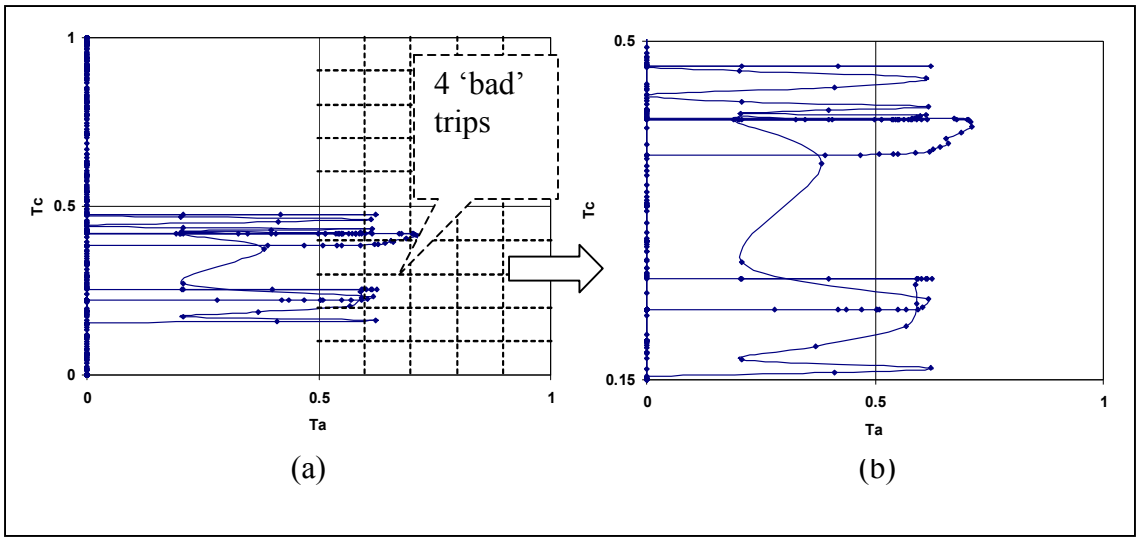


Figure 29(a): TSD for Rule 229 showing 4 'bad' trips and 0 'good' trips - IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH

Figure 29(b): Quadrant IV zoomed in to depict the 'bad' trips. Four trips satisfy the threshold condition of five.

The TSD shown in Figure 29 corroborates this fact, as the rule is seen to make four trips into Quadrant IV (four bad trips) and none into Quadrant II (zero good trips). Thus neither selection criteria are satisfied, which leads to the rule being eliminated from the final rule data-base as desired.

Some true and invalid rules may not be expressed in the data and so are never deemed as *good* or *bad*.

4.2 Results

The results were generated for the selection phase and the prediction phase.

Note for the selection phase: The phenomena expressed by the simulator, used for generating the data, are known. Hence it was known which rules are good and which

rules are bad. The selection criteria were tested to determine if they selected good rules and reject bad rules. This was a test on validation of the choice of *Merit*.

4.2.1 Selection - Without Noise

The simulator was used to generate data (for 2403 seconds). The code for the simulator is listed in Appendix A. The ideal situation of no noise was first considered. This data was then processed as described in Section 1.4.1.2. The TSDs for each of the 729 rules (generated by exhaustive search) were then generated, which were used to determine if a rule was to be selected or not. Only rules satisfying both the selection criteria (described in Section 2.5.1.1) were included in the final rule base. It was observed that all the rules that were thus selected were good.

The selection criteria depended on the metric *Merit* which in turn depended on the number of good or bad trips made by the rule. This metric alone was used to select good rules from the data base.

4.2.1.1 Choice of Threshold Criteria

Whether or not a path qualified to be termed as a *trip* depended on the threshold criteria (minimum number of consecutive points into or out of a quadrant). Initially, four consecutive points was used as a threshold to term a path as a trip. But using four points as a threshold caused one bad rule (Rule 169) to be selected and one good rule (Rule 138) to be eliminated.

For example Rule 169 which states that “IF Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH”, is completely illogical and hence is a bad rule and should not be selected. Figure 30 depicts its TSD.

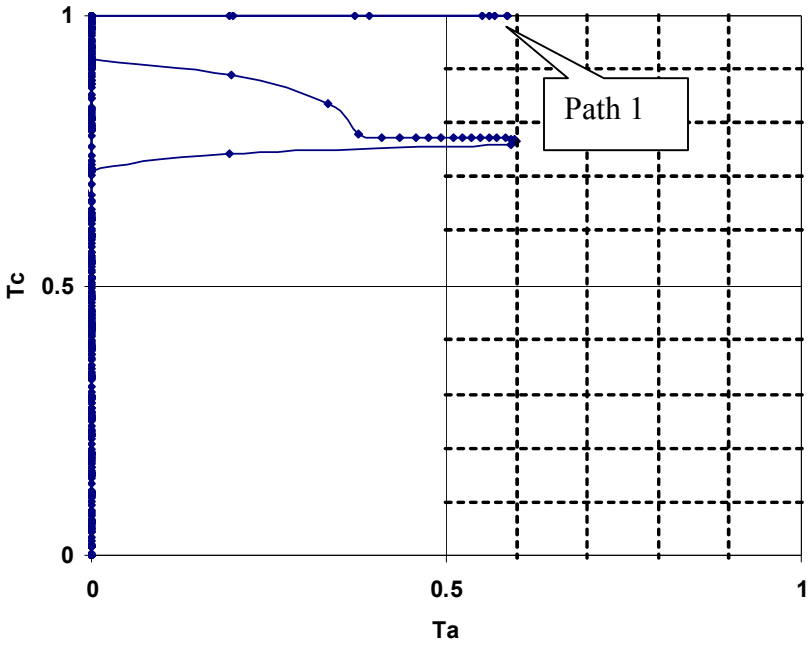


Figure 30: TSD for Rule 169 (Bad Rule) - IF Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH.
Path 1 consisting of four points is considered as a good trip only if the threshold is set at four points, making the total number of good trips=2. If the threshold is increased to five points, the total number of good trips=1.

As seen in Figure 30, the threshold criterion determines whether or not *Path 1* is considered as a good trip. A threshold of four points makes *Path 1* a good trip, thus making the total number of good trips two. Since the number of good trips is now two, and there are no bad trips, Merit equals 2 and both selection criteria are satisfied, thus causing the rule to be selected, when in fact it should be eliminated.

On the other hand, a threshold of five points causes *Path 1* to not be termed as a trip. Hence, the total number of good trips reduces to one. The 1st selection criterion of having a minimum of two good trips is now not satisfied and this rule is eliminated as is desired. On the same lines, Rule 138 which states that "IF Temp1 is HIGH & F1 is LOW & F2 is LOW & Persistence is HIGH THEN after SHORT delay Temp3 will be MED", is consistent with process phenomena and hence is a good rule and should be selected.

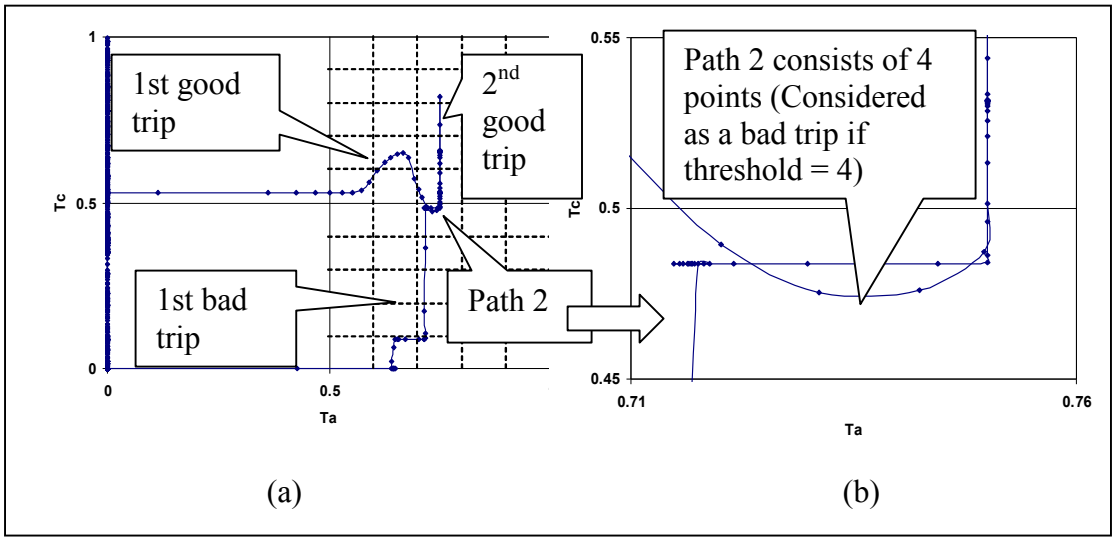


Figure 31a: TSD for rule 138 (Good Rule) - IF Temp1 is HIGH & F1 is LOW & F2 is LOW & Persistence is HIGH THEN after SHORT delay Temp3 will be MED. Path 2 determines whether or not the rule is good.

Figure 31b: Path 2 consisting of four points is considered as a bad trip only if threshold is set at four points, making the total number of bad trips=2. If the threshold is increased to five points the total number of bad trips = 1.

As shown in the TSD Rule 138 in Figure 31, the threshold criterion determines whether or not *Path 2* is considered as a bad trip. A threshold of four points makes *Path 2* a bad trip, thus making the total number of bad trips two. Since the number of good trips is two and the number of bad trips is also two, Merit equals 0 which is less than the

selection criterion of one, thus causing the rule to be eliminated, when it should be selected instead.

As a solution to avoid rejecting Rule 138, a threshold of five points causes *Path 2* to not be termed as a trip thus reduces the number of bad trips to one. Now, the total number of good trips is two and the number of bad trips is one, causing the Merit to equal 1. This causes both the selection criteria to be satisfied and this rule is hence selected into the final rule base.

Note: A method to determine the optimum threshold for a process needs to be formulated (Refer to Section 5.2)

4.2.1.2 Evaluation of Rules

The simulator was run several times and it was observed that a threshold of five points selected only good rules that were sufficiently represented in the data each time. Using these criteria nine good rules and no bad rules were selected from an initial rule-base of 729. The final list of rules is listed in Appendix B1.

Figures 28, 31, 32, 33 depict some of the rules selected to be a part of the final rule-base. Further, it was also observed that increasing the threshold to seven points selected only six good rules as it demanded more number of successive points in a path for the path to qualify as a trip. Thus increasing the threshold determined how “strict” one wished to be in defining good rules.

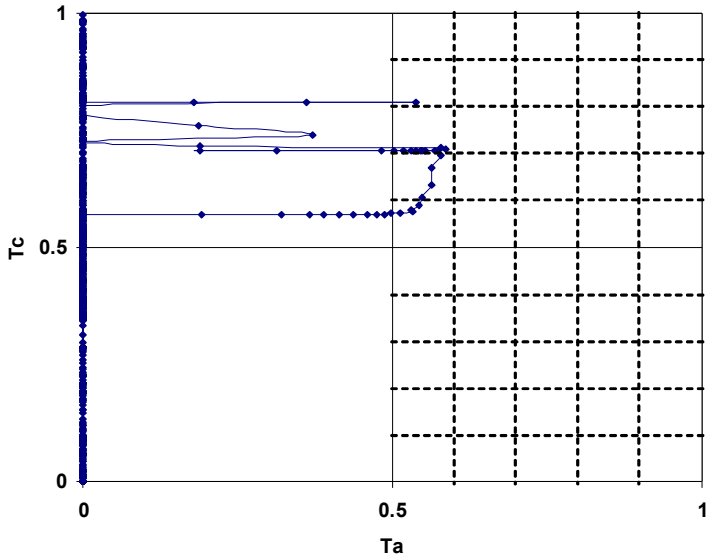


Figure 32: TSD for rule 97 - IF Temp1 is LOW & F1 is MED & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED. Two good trips denote sufficient corroboration. The Merit of the rule is 2. This rule is selected.

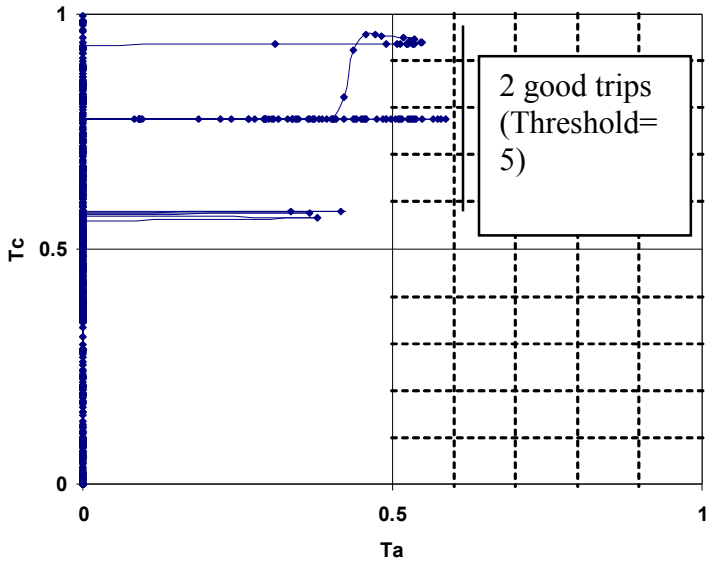


Figure 33: TSD for rule 122 - IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED. Two good trips provide sufficient corroboration. The Merit of the rule is 2. This rule is selected.

4.2.1.3 Comparison with Sharma's Work

Using the same simulator data under the same ideal (noise-less) conditions, Sharma's metrics and optimization procedure selected 306 rules. All rules with rank 0 or 1 were selected because they were either dominated by no other rule or just one more rule respectively.

Using the same data used in the present work, it was observed that 76 rules were termed as the *best* (76 rules had a rank of 0) after Sharma's optimization procedure. But 74 of the 76 rules selected by Sharma, made zero good trips and zero bad trips, and the other two made just one good trip and zero bad trips. Using the two selection criteria proposed in the present work, none of the 76 rules were accepted as good rules in the final data base. For rules that made zero good and zero bad trips, there were no instances of the rule in Quadrants II and IV to decide if it was bad or not. Such rules were eliminated. Rule 325 was one such example. Figure 34 depicts the TSD of Rule 325.

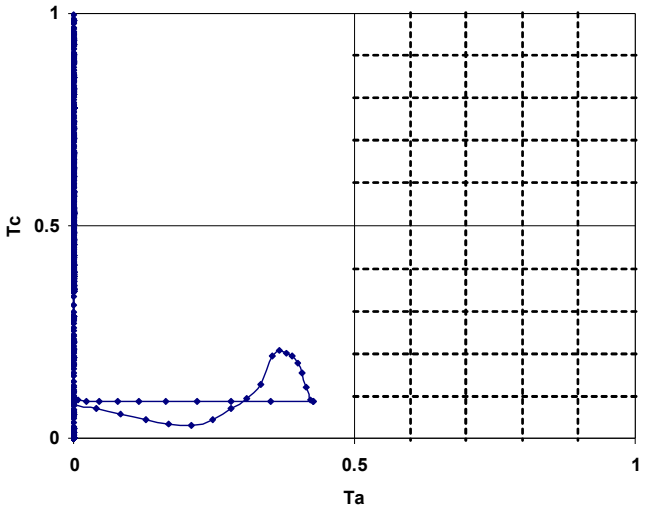


Figure 34: TSD for Rule 325 - IF Temp1 is LOW & F1 is LOW & F2 is LOW & Persistence is LOW THEN after MED delay Temp3 will be MED. This rule was selected as a good rule using Nitin's metrics but was eliminated in the present work as it made zero good trips and zero bad trips.

Although some of these eliminated rules were good, they were not accepted into the final rule base because the event they described did not exist within the data from which the knowledge was being extracted.

Rules that made just one good trip and zero bad trips were also eliminated because they did not satisfy either of the selection criteria. Rules 723, 635 were two such examples. Their TSDs are shown in Figures 35, 36 respectively.

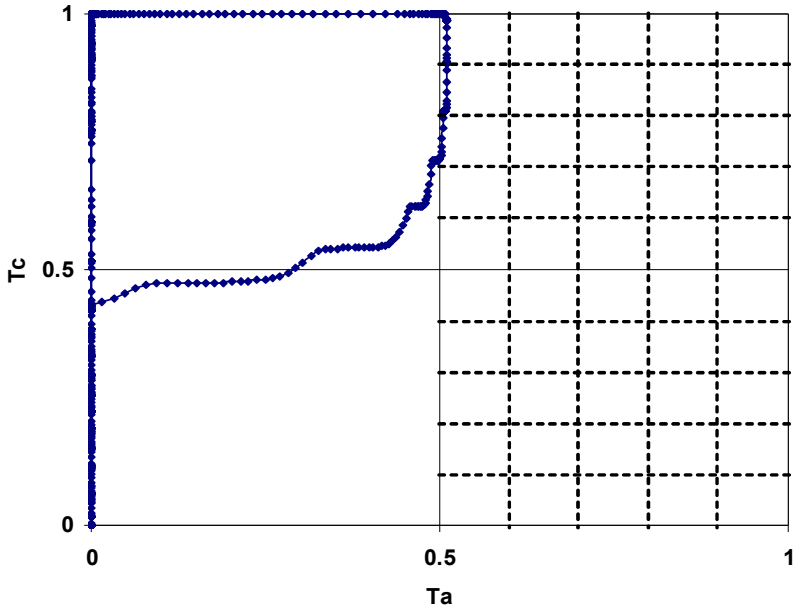


Figure 35: TSD for rule 723 - IF Temp1 is HIGH & F1 is HIGH & F2 is LOW & Persistence is HIGH THEN after LONG delay Temp3 will be HIGH. This rule is not selected in the present work because it makes just one good trip (threshold=5) which is less than the minimum criterion of two. There is not enough corroboration.

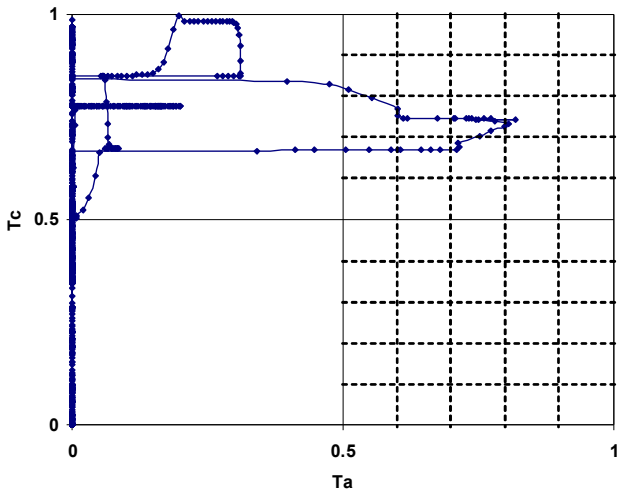


Figure 36: TSD for rule 635 - IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is MED THEN after LONG delay Temp3 will be MED. This rule is not selected in the present work because it makes just one good trip (threshold=5) which is less than the minimum criterion of two. There is not enough corroboration.

Unlike in Sharma's work, no bad rules were accepted as good because of scarce data points in Quadrant II and no good rules were rejected as bad because of scarce data points in Quadrant IV. This was achieved because the threshold condition made sure that vagaries of numerical data were not considered as a trip. Also the corroboration criterion made sure that sufficient evidence of the rule being good or bad existed. Figures 29, 37, 38 depict the TSDs of a few bad rules that were eliminated from the final rule base.

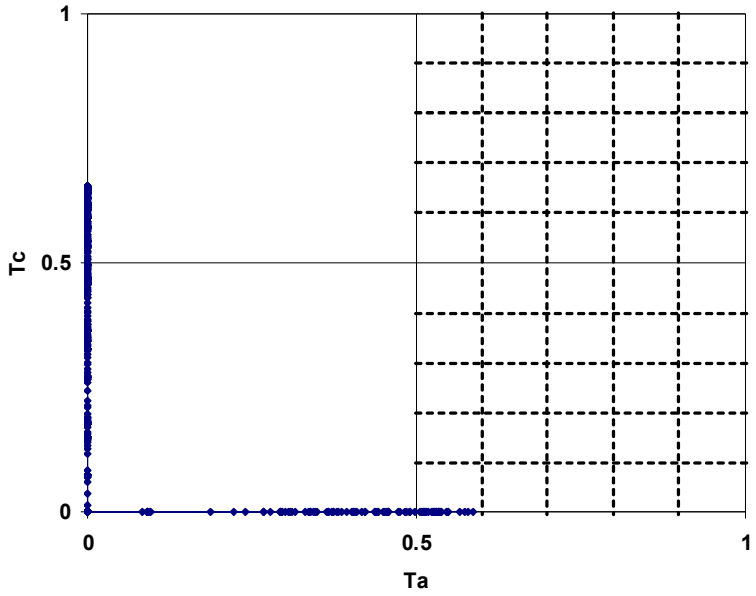


Figure 37: TSD for rule 41 - IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be LOW. This is a bad rule and hence was eliminated. It made two bad trips (Threshold=5).

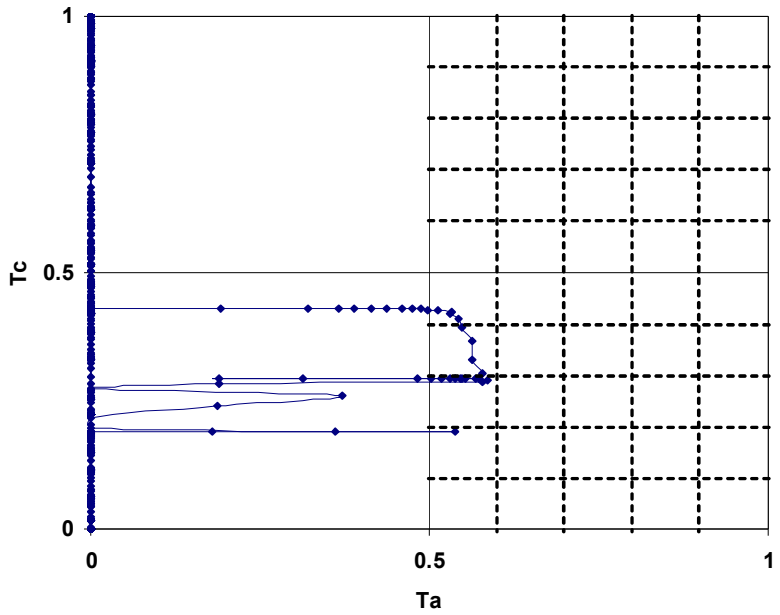


Figure 38: TSD for rule 178 - IF Temp1 is LOW & F1 is MED & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH. This is a bad rule and hence was eliminated. It made two bad trips (Threshold=5).

4.2.2 Selection - With Noise

The simulator was then run several times by adding noise to the input variables. The code is included in Appendix A. This was done to check the robustness of the rule selection method. In one of the instances, the simulator was run for 3000 seconds. As with the ideal case, the data was processed and an exhaustive search was done to determine all possible rules (729). The TSDs for each rule were then examined to determine the *Merit* for each rule and check for both the selection criteria. Again, it was observed that even for noisy data only good rules were selected.

4.2.2.1 Choice of Threshold Criteria

The rules that were selected were found to be sensitive to the choice of the threshold that determined whether a path could be termed as a trip.

A threshold of five points, selected a total of 25 rules. But, a few bad rules were selected. As the threshold was increased to six points a few bad rules were eliminated. But a threshold of seven points selected a total of 22 good rules. So a threshold of seven points seemed like the optimum in order to generate good rules and was used to determine the number of trips, when the process data was very noisy.

4.2.2.2 Evaluation of Rules

Rule 121 states that "IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED", which is logically right. This implies that Rule 121 is not a bad rule. Figure 39 depicts the TSD of rule 121.

When its TSD was examined, this rule was found to make 19 good trips and eight bad trips with a threshold of seven points. Hence, the goodness of the rule was very well corroborated. Also, it had a merit of 8 which was well above the minimum requirement of 1. Thus, this rule was included in the final rule base. TSDs of some other rules that were selected (112, 115) are shown in Figures 40, 41. The final rule-base is listed in Appendix B2.

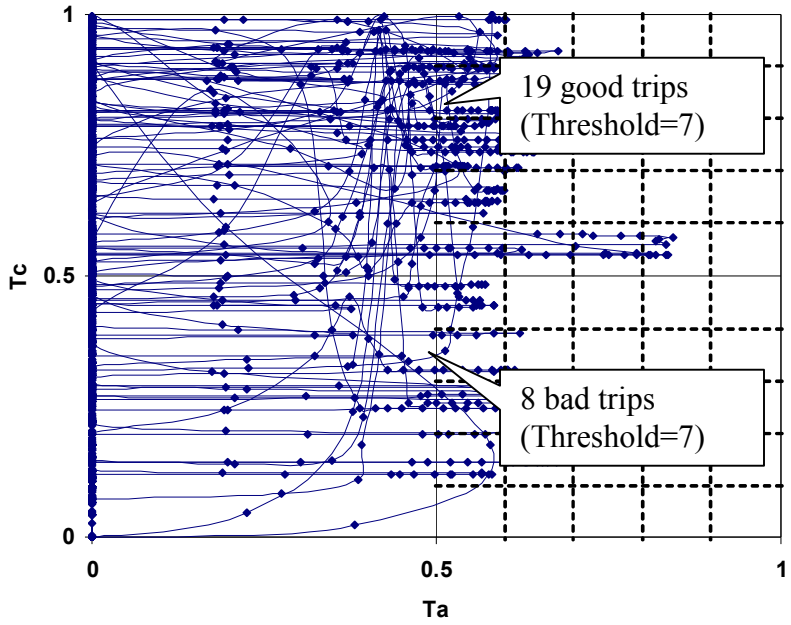


Figure 39: TSD for rule 121 - IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED. It made 19 good trips and eight bad trips. This rule was selected.

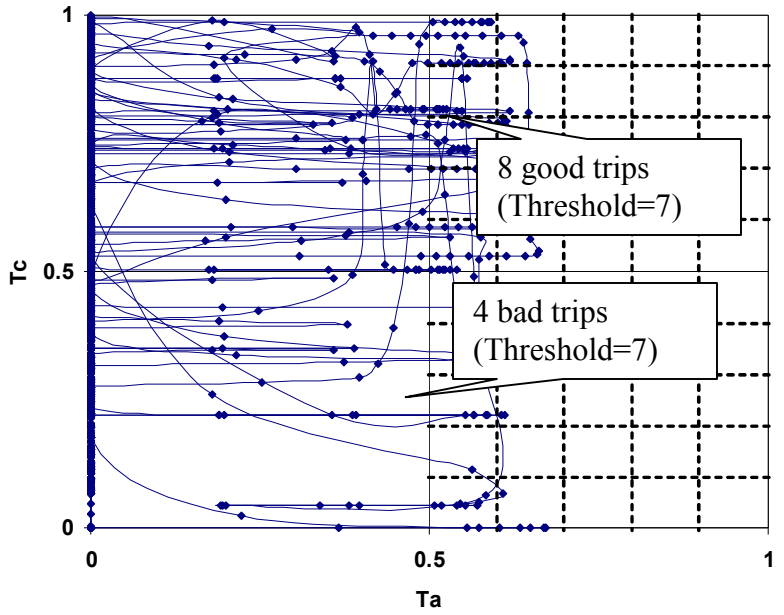


Figure 40: TSD for rule 112- IF Temp1 is MED & F1 is LOW & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED. It made eight good trips and four bad trips. This rule was selected.

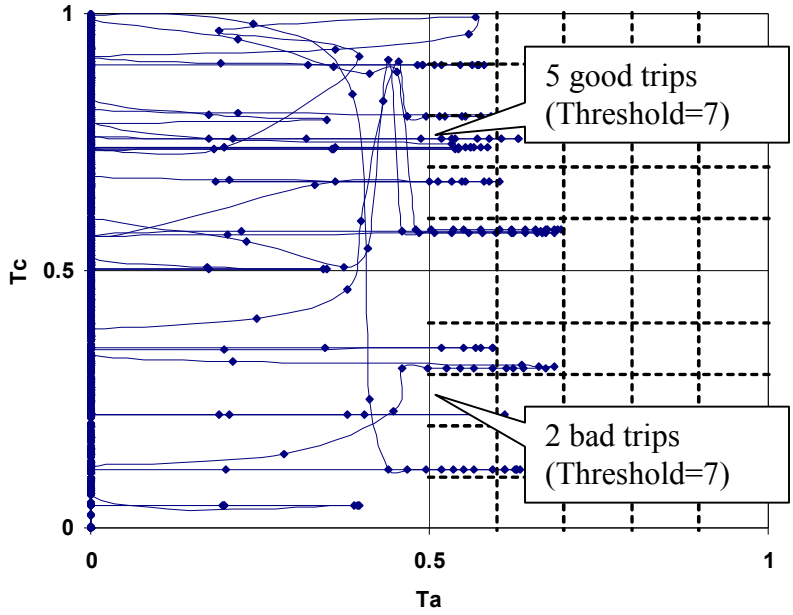


Figure 41: TSD for rule 115 - IF Temp1 is MED & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED. It made five good trips and two bad trips. This rule was selected.

Rules 115, 121,122, 127, 148 were found to be selected as good rules for both the ideal and the noisy process data, thus reinforcing the robustness of these five rules.
It was observed however, that a threshold of seven points performed better with very noisy data, and a threshold of five points performed well with data that was not too noisy.

4.2.2.3 Comparison with Sharma's Work

Sharma's metrics were then used to select good rules using the same noisy process data as discussed above. His optimization procedure selected a total of 240 rules. But none of the rules termed as the *best* by Sharma's metrics were accepted as good rules in the present work, using the two proposed selection criteria. A common trait among all the rules selected by Sharma were that they all made zero good trips and zero bad trips, which means there was no corroboration of the fact that the rule was good. They either had too scarce points in Quadrant II or no points at all.

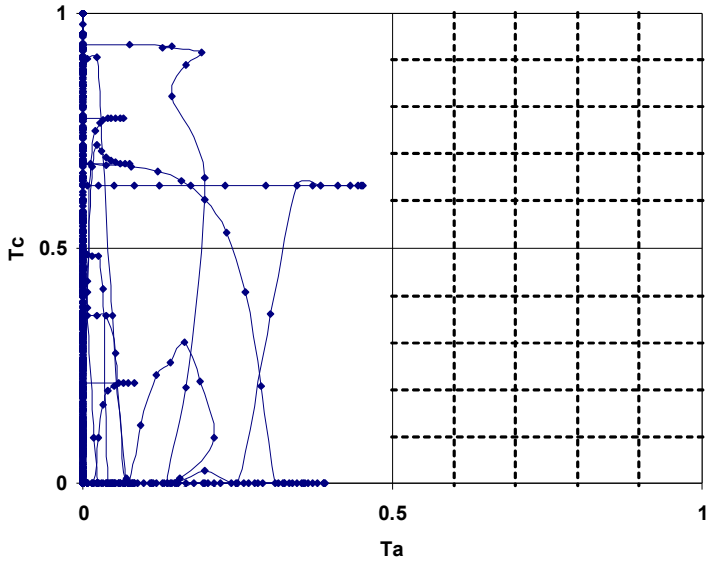


Figure 42: TSD for rule 247 - IF Temp1 is LOW & F1 is LOW & F2 is MED & Persistence is LOW THEN after MED delay Temp3 will be LOW. It made zero good trips and zero bad trips. This rule was eliminated.

An example of a rule selected by Sharma is Rule 247.Figure 42 above depicts its TSD. It states that "IF Temp1 is LOW & F1 is LOW & F2 is MED & Persistence is LOW THEN after MED delay Temp3 will be LOW", which is consistent with the process phenomena, making it a theoretically good rule. However, this rule was not selected into the final rule database in the present work because there were no instances of this rule expressed in the data used to extract knowledge.

Figure 43 depicts the TSD of Rule 130 which is another example of a rule selected by Sharma that was eliminated in the present work.

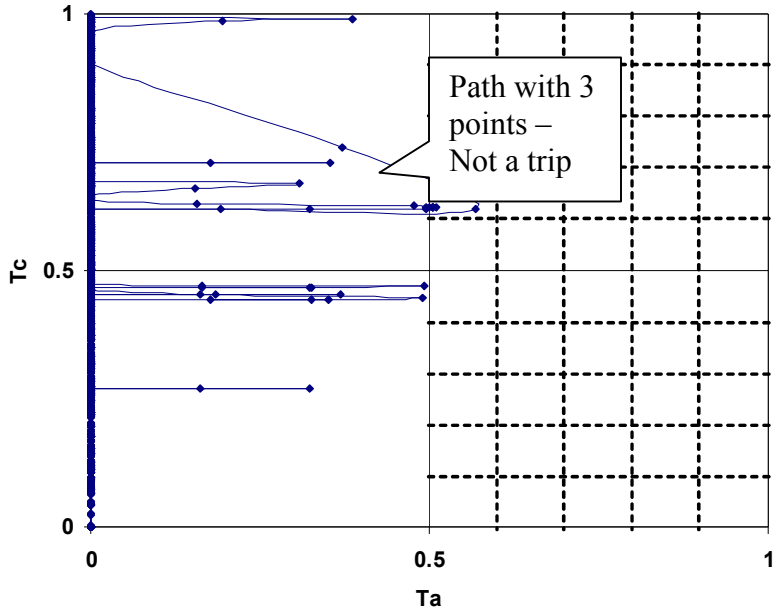


Figure 43: TSD for rule 130 - IF Temp1 is MED & F1 is HIGH & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED. It made zero good trips and zero bad trips (threshold=7). This rule was eliminated.

Figure 44 depicts the TSD of Rule 202 which states that "IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH". This is clearly a bad rule because the consequent stated in the rule is not the

effect that is expected from the cause stated by the antecedent of the rule. This rule made three good trips and 22 bad trips. Thus it was eliminated, as desired.

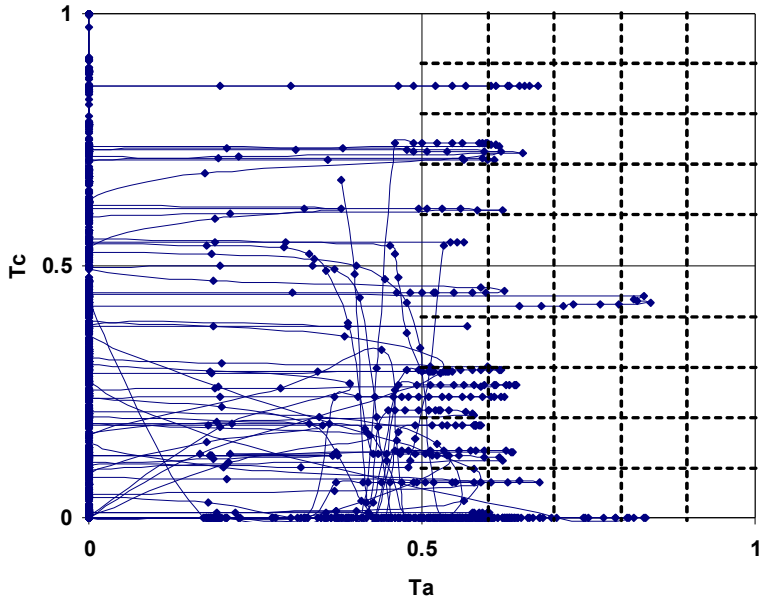


Figure 44: TSD for rule 202 - IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH. It made three good trips and 22 bad trips (threshold=7). This is a bad rule and was eliminated

4.2.3 Prediction

This phase is independent of the rule selection phase and helps predict future outcomes based on historical data.

Process data was collected from the simulator (with noise) for 3541 seconds, of which 2500 data points were considered to be 'historical' data and were used in the selection mode. The remaining 1041 data points were considered as *new* data, for which the distribution of consequents in the 10 zones were predicted using the information gathered from the previous 2500 data points. New data was used in batches of 65 seconds which was set as the window length for prediction, as described in Section 2.4.5.

The historical data was put through the selection process mentioned in Section 2.5.1. Once the TSDs were constructed for all rules, the distribution of the consequents in each of the 10 consequent zones was determined for each of the individual five antecedent zones.

For example, Figure 45 depicts the TSD of Rule 118. Based on historical data, this rule made six good trips and two bad trips. Since it satisfied both the selection criteria, it was selected into the final rule-base.

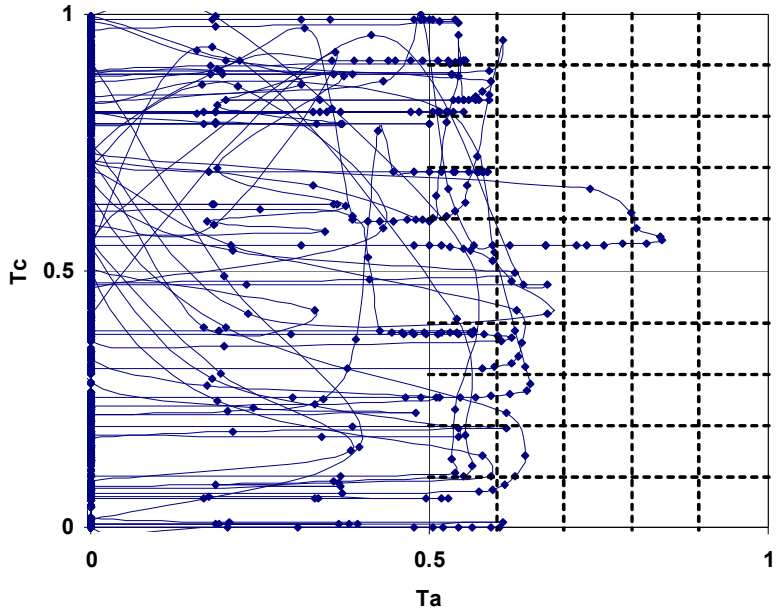


Figure 45: TSD for rule 118 based on 'Historical' Data - IF Temp1 is LOW & F1 is MED & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be LOW. It made six good trips and two bad trips (threshold=7). This was one of the good rules that were selected.

The consequent distribution in each antecedent zone was determined for Rule 118 based on historical data and then normalized to aid the calculation of *Expectations*. In calculating the number of points in each zone, noise was eliminated by only considering points which contributed to either a good or a bad trip based on the threshold criteria.

New data was collected and processed; it was un-delayed and only the antecedents were fuzzified [As described in Section 1.4.1.3]. The number of antecedent hits in each of the five antecedent zones was calculated. For example in the new data, Rule 118 made two hits in the first antecedent zone, one hit in the second antecedent zone and one hit in the third antecedent zone. Figure 46 depicts the antecedent hits made by Rule 118 based on *new* data.

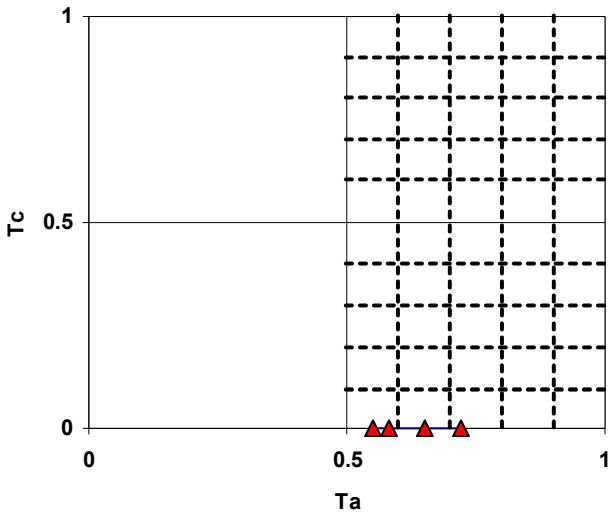


Figure 46: TSD for rule 118 based on 'new' data, depicting the antecedent hits. Two hits were made in the first antecedent zone, one hit in the second antecedent zone and one hit in the third antecedent zone.

All the information gathered was then used to calculate the Expectations of the consequent hits of this rule based on the antecedent hits in the new data. Section 2.4 details this calculation procedure. These expectations were first calculated individually for each antecedent zone and then summed to yield the *absolute* expectations for the rule in question. Finally these absolute expectations were normalized to yield *normalized* expectations. Histograms were used to depict these Expectations. Figure 47 depicts the

individual expectations for Rule 118, for the first and the second antecedent zones, respectively, based on the historical data of Rule 118 and the new hits the rule made.

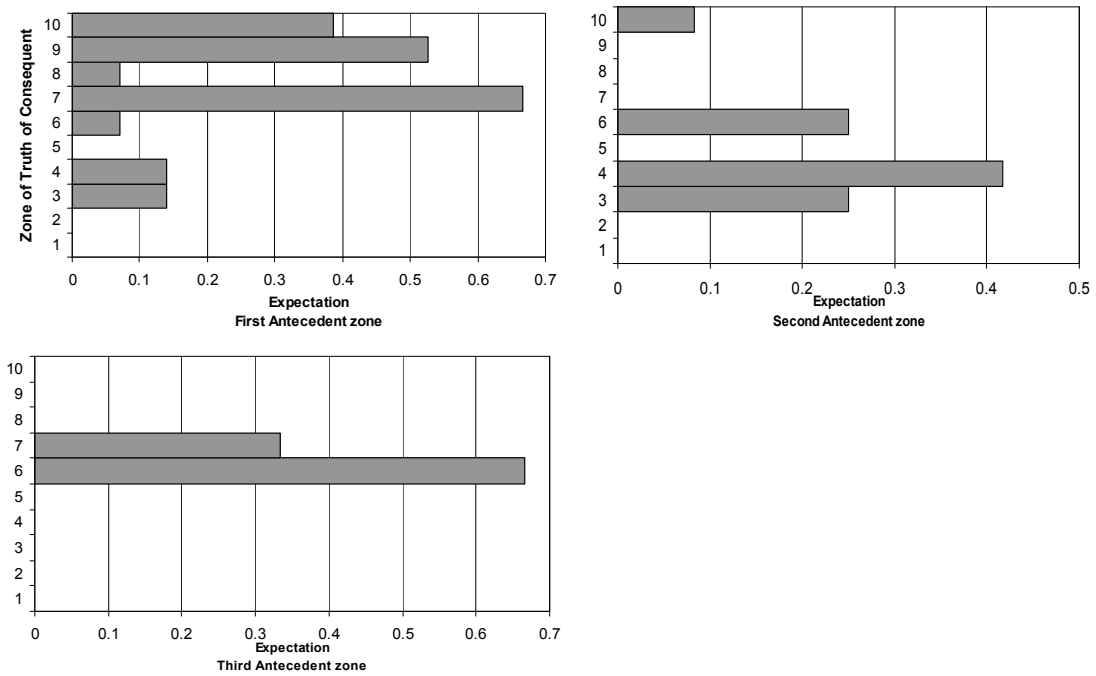


Figure 47: Individual Absolute 'Expectations' for each of the three antecedent zones in which Rule 118 made hits.

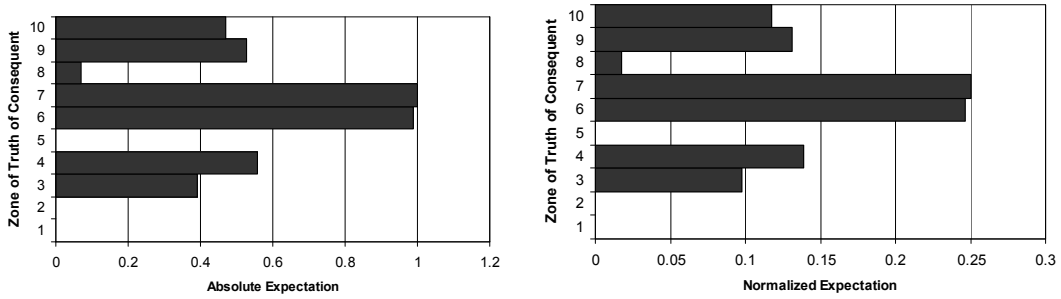


Figure 48: Cumulative Absolute and Normalized 'Expectations' for Rule 118.

Figure 48 depicts the cumulative absolute 'expectations' histogram for Rule 118, obtained by summing the individual expectations for each zone and the normalized expectations histogram obtained by normalizing the absolute expectations.

By inspecting the expectation histogram we can infer that for the antecedent hits made by Rule 118 based on new data, most of the consequents are predicted to lie in Quadrant II, which in turn may mean that the rule may make another good trip in the future. Similarly expectations were successfully generated for all rules in the initial rule data-base.

The weighted mean average of the expectations was also calculated within 95% confidence limits as described in Section 2.4.4. But since the data was not assumed to be normally distributed, the weighted mean average provides as just an approximation of the central tendency of the expected consequents. Late breaking research suggests the calculation of median instead. This is discussed in Section 5.2.

For example, the weighted mean average of the expectations of Rule 10 was found to be 0.6 within 95% confidence limits of 0.41, 0.79.But the median of the expectations was 0.65. The difference in the mean and the median implies that the distribution is definitely far from normal Gaussian distribution. Table I depicts the means, medians for a few good rules.

Table II depicts predictions made at every five second interval over a time period of 45 seconds, at a certain point in time. Table III depicts values of T3 based on what actually happened (the simulator data). A comparison of the two tables denotes the accuracy of the prediction.

Rule number	Weighted Mean Average(Lower Confidence Limit, Upper Confidence Limit)	Median
10	0.599 (0.41, 0.79)	0.65
13	0.71 (0.26, 0.98)	0.75
82	0.74 (0.51, 0.98)	0.75
85	0.71 (0.51, 0.96)	0.75
95	0.52 (0.2, 0.79)	0.55
109	0.62 (0.13, 0.9)	0.75
118	0.61 (0.22, 0.98)	0.65
119	0.61 (0.22, 0.98)	0.55
121	0.71 (0.28, 0.99)	0.75
122	0.75 (0.35, 0.98)	0.75
130	0.75 (0.7, 0.8)	0.75
226	0.65 (0.12, 1)	0.6
229	0.66 (0.13, 0.9)	0.75
235	0.89 (0.71, 1)	0.95

Table I: This table denotes the comparison of weighted mean average and median of 'expectations' for a few good rules.

As denoted by Table II, Rule 118, 119 predict that T3 will be MED after SHORT delay, at 10, 15, 20 seconds from the point in time where prediction begun. Table III (which denotes the actual values and membership functions of T3 for all three fuzzy categories at the same above-mentioned points in time) depicts that the actual T3 has μ_M values of about 0.2 and μ_L values of about 0.8. This implies that the actual T3 was more LOW than MED, making the predictions only partially correct.

Similarly, Rules 121, 122 predict that T3 will be MED after SHORT delay, at 35, 40, 45 seconds from the point in time from where prediction begun. Table III depicts that the actual T3 has μ_M values of about 0.1 and μ_L values of about 0.9. This implies that the actual T3 was more HIGH than MED, making the predictions only partially correct.

Good Rule #	118			230			121		
Good Rule Consequent	after SHORT delay, Temp3 will be MED			after SHORT delay, Temp3 will be HIGH			after SHORT delay, Temp3 will be MED		
Time (Seconds)	Expectation	L.C.L	U.C.L	Expectation	L.C.L	U.C.L	Expectation	L.C.L	U.C.L
5									
10	0.71	0.24	0.99						
15	0.61	0.23	0.98						
20	0.57	0.22	0.98						
25									
30									
35							0.74	0.36	0.97
40							0.69	0.26	0.99
45							0.75	0.34	0.98
50				0.69	0.31	0.98			
Good Rule #	10			119			122		
Good Rule Consequent	after SHORT delay, Temp3 will be LOW			after SHORT delay, Temp3 will be MED			after SHORT delay, Temp3 will be MED		
Time (Seconds)	Expectation	L.C.L	U.C.L	Expectation	L.C.L	U.C.L	Expectation	L.C.L	U.C.L
5									
10									
15				0.63	0.21	0.99			
20				0.58	0.5	0.78			
25	0.6	0.41	0.79						
30	0.6	0.41	0.79						
35							0.75	0.34	0.98
40							0.74	0.36	0.98
45							0.75	0.34	0.98
50									

Table II: This table denotes the predictions made for six good rules at a certain point in time.

As seen in Table II Rules 10, 230 predicted LOW and HIGH T3 values at 25 and 50 seconds from the point in time where prediction begun respectively. Table III depicts that at 25 seconds the actual T3 has a μ_L of about 0.8 and a μ_M of about 0.2 making T3 more LOW than MED. Also, at 50 seconds the actual T3 has a μ_M of about 0.1 and a μ_H of about 0.9 making T3 more HIGH than MED. Thus the predictions made by Rules 10, 230 were observed to be very accurate as can be seen in Table III.

Thus, two of the six good rules made accurate predictions.The anamolies in prediction can be owed to the fact that Table II lists predictions made by only six good rules. If predictions made by all the selected good rules is analysed, more accurate results are obtained.

Time (Sec)	Actual T3	Prediction	μ_L	μ_M	μ_H
10	13.7051	MED	0.8066	0.1934	0
15	13.75272	MED	0.8055	0.1945	0
20	13.75964	MED	0.8053	0.1947	0
25	14.71551	LOW	0.784	0.216	0
30	89.35258	LOW	0	0.1255	0.8745
35	89.58073	MED	0	0.1204	0.8796
40	89.58817	MED	0	0.1203	0.8797
45	89.58825	MED	0	0.1203	0.8797
50	88.99691	HIGH	0	0.1111	0.8888

Table III: This table denotes the actual values of the variable T3 at the same point in time for which predictions were made, as depicted in Table II. The Low (μ_L), Medium (μ_M) and High (μ_H) membership functions were calculated based on the fuzzy break points of 5, 50, and 95.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

The results presented in Chapter IV provide proof of concept for *trips* (independent events) to be used as the basis of calculating metrics (*Merit*) used to select good rules from an initial rule data-base, using truth-space evaluation as suggested by Sharma. They also present the possibility of predicting future consequents based on historical data using the merit *Expectations*.

5.1 Conclusions

These metrics are based on the Truth Space Diagram proposed by Sharma. Efficient technologies like genetic algorithms can be used to generate rules and the TSD approach can be used in conjunction, to optimize rule bases and evaluate rules using the metrics proposed.

1. The two selection criteria proposed (involving corroboration and the metric Merit) were able to select good and only good rules that were sufficiently expressed in the data used for extracting knowledge. For rules that were selected, sufficient evidence existed of the fact that they were good.

2. The proposed selection metric *Merit* in combination with minimum corroboration (based on the concept of trips) alone is capable of selecting good, and only good, rules from process data. There was no need to implement Pareto-optimization techniques.
3. The *threshold condition* used to define paths that were *trips*, ensured that vagaries in numerical data were not considered as an event. This removed the inclusion of incorrect or bad rules in the final rule database.
4. The metric *expectation,* proposed for predicting consequents given the antecedents, was used to predict future behavior of every rule. The predictions were reasonably accurate.
5. Both proposed metrics were simple to calculate. The metric *Merit* did not require normalization.

5.2 Issues and Recommendations

1. The most important issue with the metrics that are recommended in this work is that there is no mathematical proof for them. They are purely intuitive. Proof of concept however does exist as shown in Chapter IV.
2. In the present work the threshold that defined a trip was set at five data points for ideal data and at seven data points for noisy data. These numbers though efficient, are not claimed to be universal. A method to determine the threshold depending on the application needs to be formulated.

3. The two selection criteria that were used in this work could also be subject to change depending on the application. In the present study, the selection criteria were set up such that, only rules that made at least two good trips and had a merit of at least 1 were eligible to be included in the final rule data-base. While a minimum of two good trips were required to corroborate the goodness of a rule, a single bad trip was enough to corroborate the badness of a rule. Therefore, a rule that made zero good trips and one bad trip was deemed as a bad rule, but a rule that made one good trip and zero bad trips was not termed as good rule. This made the selection criteria "strict", but there needs to be a way to determine how strict one should be in selecting good rules.
4. As discussed in Chapter II, there is enough evidence of the fact that the distribution of points in the TSD cannot be assumed as Gaussian distribution. In the prediction mode however, the weighted mean average was used as a measure to denote the central tendency of the expected consequents, for the given antecedents. This can be misleading in skewed distributions since it is greatly influenced by extreme values. The Median may be more informative for skewed distributions like those observed in the present work, since it is less sensitive to extreme values. Hence, use of the median to report the central tendency is recommended.
5. The boundaries of the TSD were fixed at certain values. Specifically, the TSD was split into 4 quadrants of size 0.5 x 0.5 each. The sizes of these quadrants could be customized depending on the process at hand. These boundaries form the foundation of the present work. A path into a certain quadrant was qualified as a

trip if it satisfied the threshold condition. For example, if the Ta = 0.5 boundary is moved (backwards) to Ta = 0.4, many paths that did not qualify as trips with the Ta = 0.5 boundary, may now qualify as trips. This will affect the number of good and bad trips made by a rule which in turn, may affect the rule's conformation to the selection criteria and hence its selection into the final rule-base. In the prediction mode, more number of antecedent zones would be required to depict the distribution of the consequents. Similarly, shifting any of the boundaries (Ta = 0.5, Tc = 0.5) backward or forward will affect the performance of both metrics.

6. Once good rules are selected into the final rule-base they need to 'Ranked' in order to determine which rules are better than a certain rule. It is recommended that the following possibilities be considered.

Rules of distinctly different antecedents and consequents should not be compared. For example, Rule 7 states that "IF Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be LOW", and rule 121 states that "IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED". Both rules are 'good'. The data shows that Rule 7 makes 2 good trips and 0 bad trips, which implies that it has a merit of 2. On the other hand, Rule 121 makes 18 good trips and 6 bad trips, implying that it has a merit of 12, which is much larger than the merit of Rule 7. However, this does not mean that Rule 121 is better or that Rule 7 needs to be rejected. It just means that Rule 121 is 'seen more' in data. Thus more instances of corroboration need not mean the rule is better.

Rules that have the same antecedent can be compared, assuming each antecedent has a unique outcome. For example consider Rule 121 which states that "IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED" and rule 607 which states that "IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after LONG delay Temp3 will be MED". Both rules share the same antecedent, but predict different consequents. Both rules are 'good', and both are selected. But Rule 607 makes only two good trips and zero bad trips implying that its merit is 2, which is much lesser than the merit (12) of rule 121. In this case, in can be said that Rule 121 is "better" than Rule 607. Thus use of "better" is a distinction only for rules of like antecedent.

If there are two valid mechanisms (antecedents of rules) that cause the same effect (consequent), then both rules can be selected. This could imply that both rules are incomplete and can be combined by the 'OR' operator. For example, consider Rule 7 which states that "IF Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be LOW" and Rule 10 which states that "IF Temp1 is LOW & F1 is MED & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be LOW". Both rules predict the same outcome but are caused by different causes. Both rules are good and both are selected. Both rules make two good trips and zero bad trips. They can be combined into one rule using the "OR" operator as follows: "If

(Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW) OR (IF Temp1 is LOW & F1 is MED & F2 is LOW & Persistence is LOW) THEN after SHORT delay Temp3 will be LOW".

7. In the present work, since the phenomena expressed by the simulator are known. It is also known whether a given rule is good or bad. This was used as an advantage in determining the optimum threshold criteria. A trial-and-error method was used to determine the optimum threshold criterion that selected good and only good rules. Hence, a method to determine the optimum threshold for a process, when all the phenomena are not known, needs to be formulated.

REFERENCES

(In order of appearance)

1. De Lima P. and Yen G.G., "A Truth Space Diagram temporal linguistic rule extraction procedure using multiple objective Genetic Algorithm", *ISA Transactions*, **40**(2), pp. 315-327, 2005.
2. Sharma N., "Metrics for evaluation of the goodness of linguistic rules", *Unpublished Master's Thesis,* Oklahoma State University, Oklahoma, USA, 2003.
3. Su M., "Autonomous generation of cause-and-effect rules in dynamics processes", *Unpublished PhD Proposal*, Oklahoma State University, Oklahoma, USA, 2005.
4. Stephanopoulos G. and Han C., "Intelligent systems in process engineering: A review", *Computers & Chemical Engineering*, **20**(6), pp. 743-791, 1996.
5. O'Shima E., "Computer aided plant operation," *Computers & Chemical Engineering*, **7**(4), pp. 311-329, 1983.
6. Hüllermeier E., "Special issue on fuzzy sets in knowledge discovery", *Fuzzy Sets and Systems, Editorial*, **149**(1), pp. 1-3, 2005.
7. Zadeh L.A., "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-3**, 28-44, 1973.
8. Yuan Y. and Zhuang H., "A genetic algorithm for generating fuzzy classification rules", *Fuzzy Sets and Systems*, **84**(1), pp. 1-19, 1996.

9. Chen J.C. and Black JT., "A fuzzy-nets in-process (FNIP) system for tool-breakage monitoring in end-milling operations", *International Journal of Machine Tools and Manufacture,* **37**(6), pp. 783-800, 1997.

10. Herrera F., Lozano M. and Verdegay J.L., "A learning process for fuzzy control rules using genetic algorithms," *Fuzzy Sets and Systems*, **100**(1), pp. 143-158, 1998.

11. Ngan P.S., Wong M.L., Lam W., Leung K.S. and Cheng C.Y., "Medical data mining using evolutionary computation," *Artificial Intelligence in Medicine*, **16**(1), pp. 73-96, 1999.

12. Castillo L., Gonzalez A. and Perez R., "Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm", *Fuzzy Sets and Systems*, **120**(2), pp. 309-321, 2001.

13. Kim D. and Lee H.P., "An optimal design of neuro-FLC by Lamarckian co-adaptation of learning and evolution," *Fuzzy Sets and Systems*, **118**(2), pp. 319-337, 2001.

14. Ishibuchi H. and Yamamoto T., "Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining," *Fuzzy Sets and Systems*, **141**(1), pp. 59-88, 2004.

15. Tan PN., Kumar V. and Srivastava J., "Selecting the right objective measure for association analysis", *Information Systems*, **29**(4), pp. 293-313, 2004.

16. Wang H., Kwong S., Jin Y., Wei W. and Man K.F., "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction", *Fuzzy Sets and Systems*, **149**(1), pp. 149-186, 2005.

APPENDIX A CODE LISTING

The following is the listing of the codes for every form used in the program.

The VBA code for the form *Choose.frm* is listed below.

```
'Algorithm for finding good rules from historical data using
'linguistic classification of input output variables using exhaustive search
'and for evaluation of rules using novel metrics.

'Uses .CSV files for input and output of data

'Preetica Kumar   12th June 2004
'NOTE: Some parts of this algorithm were inherited from
'previous code written by Nitin Sharma ( Calculation of Ta, Tc,Persistence,linguistic
classification
'of input and output variables, exhaustive search of rules)

'Work done under Dr Rhinehart at Oklahoma State University

'This program asks the user whether he wishes to enter the selection mode or the
prediction mode

Private Sub Command1_Click()

Choose.Hide
Set Choose = Nothing

If Option1.Value = True Then
   SelMode.Show
Else
   PredMode.Show
End If

End Sub

Private Sub Option1_Click()
Option2.Value = False
End Sub

Private Sub Option2_Click()
Option1.Value = False
End Sub
```

The VBA code for the form *SelMode.frm* is listed below.

```
'Algorithm for the selection mode
'This program uses HISTORICAL DATA to create the initial rule-base(729 rules)
'by the exhaustive search method.
'It also calculates the value of the selection metric MERIT for each rule and
'the distribution of consequents in quadrants II,IV.

Public p As Variant, q As Variant 'counters

Public n As Variant 'rule number

Dim time()     'time
Dim t1()       'temperature 1
Dim f1()       'flow rate 1
Dim f2()       'flow rate 2
Dim t3s()      'temperature 3 after short delay
Dim t3m()      'temperature 3 after medium delay
Dim t3l()      'temperature 3 after long delay
Dim cp()       'combined persistance

'mua()=(T1(L,M,H),F1(L,M,H),F2(L,M,H),P(L,M,H))
Dim mua() As Double 'membership of antecedent
Dim muc() As Double 'membership of concequent
Dim filepath$
Dim filepersist$
Dim i As Integer    'counter of data point
Dim pa() As Double  'individual persistence of every variable in antecedent
Public runi As Integer

Dim tot As Integer
Dim samptime As Integer
Dim totaltime As Integer

Public prog As Integer

Sub lingpersclassify(i, tone, fone, ftwo)
'This subroutine calculates the combined persistence of the antecedent and
'classifies it into three fuzzy categories(L,M,H)
'written by Nitin Sharma

mua(10, i) = 0
mua(11, i) = 0
mua(12, i) = 0
```

```
'the combined persistence time is the minimum persistence time among all inputs
If pa(fone, i) >= pa(tone, i) Then
   If pa(ftwo, i) >= pa(tone, i) Then
      cp(i) = pa(tone, i)
   End If
End If

If pa(tone, i) >= pa(fone, i) Then
   If pa(ftwo, i) >= pa(fone, i) Then
      cp(i) = pa(fone, i)
   End If
End If

If pa(tone, i) >= pa(ftwo, i) Then
   If pa(fone, i) >= pa(ftwo, i) Then
      cp(i) = pa(ftwo, i)
   End If
End If

'Classification of persistance using 1-20-40 as limits
      If cp(i) = 1 Then mua(10, i) = 1
      If cp(i) >= 40 Then mua(12, i) = 1
      If cp(i) = 20 Then mua(11, i) = 1
      If cp(i) > 1 And cp(i) < 20 Then
      mua(10, i) = (20 - cp(i)) / (19)
      mua(11, i) = (cp(i) - 1) / (19)
      boogi = cp(i)
      End If
      If cp(i) > 20 And cp(i) < 40 Then
      mua(11, i) = (40 - cp(i)) / (20)
      mua(12, i) = (cp(i) - 20) / (20)
      End If
End Sub

Sub linguipersist(n, mua() As Double)
'This subroutine provides a linguistic label for the maximum persistence of each data
point
'written by Nitin Sharma

Dim X As Integer
Dim maxi() As Integer
ReDim maxi(1 To 3)
ReDim pa(1 To 9, 1 To n)

filepaths$ = filepath$ + "lingpers.csv"
Open filepaths$ For Output As #10
```

```
Print #10, "time,patl,patm,path,paf1l,paf1m,paf1h,paf2l,paf2m,paf2h"

For i = 2 To n
For k = 1 To 3 'k represents t1 f1 and f2 respectively
j = (3 * k - 2)

 If mua(j, i) > mua(j + 1, i) Then
    If mua(j, i) > mua(j + 2, i) Then
       maxi(k) = j
    Else: maxi(k) = (j + 2)
    End If
 ElseIf mua(j, i) < mua(j + 1, i) Then
    If mua(j + 1, i) > mua(j + 2, i) Then
       maxi(k) = (j + 1)
    Else: maxi(k) = (j + 2)
    End If
End If

   X = maxi(k)
   pa(X, i) = pa(X, i - 1) + 1

   For l = j To (j + 2)
   If l <> maxi(k) Then pa(l, i) = 0
   Next l

Next k

Print #10, time(i); ","; pa(1, i); ","; pa(2, i); ","; pa(3, i); ","; pa(4, i); ","; pa(5, i); ",";
pa(6, i); ","; pa(7, i); ","; pa(8, i); ","; pa(9, i)

Next i

Close #10

End Sub

Sub TATC(n, mua() As Double, muc() As Double, pa() As Double, tot As Integer)

'Subroutine for calculating the truth of the antecedent and the truth of the
'consequent parts of the rules, using the predefined values of linguistic variables
'in the previous SUB. The number of iterations are Nx3x3x3x3x3x3 i.e. 7 loops
'in total cover N points and 729 rules in 81 categories.
'written by Nitin Sharma

'This subroutine also calculates the number of trips made into quadrant II, quadrant IV,
```

```
'the number of consequent hits in each of the 10 consequent zones for each of the 5
antecedent zones,
'the number of antecedent hits in each of the 5 antecedent zones.
'written by Preetica Kumar

time1 = Timer 'for time calculation
SelMode.MousePointer = 13

Dim zz, zx As Integer
Dim max, val As Integer
Dim goodtrip() As Single
Dim badtrip() As Single
Dim accuracy() As Single 'MERIT of each rule
Dim counter2() As Single
Dim counter() As Single
Dim slope As Double
Dim Y As Double
Dim X As Double
Dim ta() As Double
Dim ta2() As Double
Dim tc() As Double
Dim tasum() As Double
Dim kmax() As Single
Dim numpts()          'number of points in each 0.1x0.1 grid of quadrant II and quadrant IV
Dim hits() As Double    'number of antecedent hits in each of the 5 antecedent zones
Dim row, column As Integer
Dim ii, jj, pp, qq As Integer
Dim num As Integer
Dim k, l As Single
Dim filepath2$
Dim filepath3$

'strings for formulation of the rules
Dim clt1$()
Dim clf1$()
Dim clf2$()
Dim clp$()
Dim cld$()
Dim clt3$()
Dim rule$()

'Redimensioning of variables and initialization of certain linguistic string constants

ReDim clt1$(1 To 3)
ReDim clf1$(4 To 6)
ReDim clf2$(7 To 9)
```

```
ReDim clp$(10 To 12)
ReDim cld$(1 To 3)
ReDim clt3$(1 To 3)
ReDim rule$(1 To 729)
ReDim ta(1 To n)
ReDim tc(1 To n)
ReDim kmax(1 To 729)
ReDim cp(1 To n)
ReDim goodtrip(1 To 729)
ReDim badtrip(1 To 729)
ReDim accuracy(1 To 729)
ReDim counter(1 To n)
ReDim counter2(1 To n)
ReDim numpts(1 To 729, 1 To 10, 1 To 5)
ReDim hits(1 To 729, 1 To 5)
ReDim ta2(1 To n)
ReDim tasum(1 To n)

clt1(1) = "IF Temp1 is LOW"
clt1(2) = "IF Temp1 is MED"
clt1(3) = "IF Temp1 is HIGH"

clf1(4) = " & F1 is LOW"
clf1(5) = " & F1 is MED"
clf1(6) = " & F1 is HIGH"

clf2(7) = " & F2 is LOW"
clf2(8) = " & F2 is MED"
clf2(9) = " & F2 is HIGH"

clp(10) = " & Persistence is LOW"
clp(11) = " & Persistence is MED"
clp(12) = " & Persistence is HIGH"

cld(1) = " THEN after SHORT delay"
cld(2) = " THEN after MED delay"
cld(3) = " THEN after LONG delay"

clt3(1) = " Temp3 will be LOW"
clt3(2) = " Temp3 will be MED"
clt3(3) = " Temp3 will be HIGH"

'asking the user where he/she wants to save the files
answer:
answer$ = InputBox("Continue saving output files in same directory? Please type y
(Yes)// n (No) // q (Quit)", "Enter Information")
```

```
If answer$ = "y" Then
zz = Len(filepersist$)
zx = Len(filepaths$)
filepath$ = Left$(filepaths$, zx - zz)
ElseIf answer$ = "n" Then
filepath$ = InputBox("Please Enter target directory for output files", "Enter Information",
"e:\vbprogs\output\")
ElseIf answer$ = "q" Then
End
Else
text1.Text = "Please input  y or n or q to proceed"
GoTo answer
End If

'Opening output files in the user-specified filepath

filepath2$ = filepath$ + "rulesheet.csv"
Open filepath2$ For Output As #12
filepath4$ = filepath$ + "combop.csv"
Open filepath4$ For Output As #14
filepath6$ = filepath$ + "tasums.csv"
Open filepath6$ For Output As #16
filepath100$ = filepath$ + "trips.csv"
Open filepath100$ For Output As #50
filepath101$ = filepath$ + "numpoints.csv"
Open filepath101$ For Output As #51

'Titles in the .CSV files
Print #51, "Number of Hits in each of the antecedent zones(HISTORICAL DATA)"
Print #51, "Rule,TaZone1, TaZone2,TaZone3,TaZone4,TaZone5"
Print #14, "r,i,cp,mua10,mua11,mua12"

r = 0
cat = 0

For theta = 1 To 3
      For tthree = 1 To 3
      cat = cat + 1
         For tone = 1 To 3
               For fone = 4 To 6
                     For ftwo = 7 To 9
                           For pers = 10 To 12

'this is indexing for the consequent part
a2 = tthree + 3
a3 = tthree + 6
```

```
'now calculate rule number
r = r + 1

'Visual bar for progress metering
prog = prog + 1
bar1.Value = prog

'Initialization of variables
goodtrip(r) = 0
badtrip(r) = 0

For i = 1 To 10
   For j = 1 To 5
      numpts(r, i, j) = 0
   Next j
Next i

For i = 1 To 5
   hits(r, i) = 0
Next i

'Create linguistic statement of rule and add to end of rulesheet file
rule(r) = Str(cat) + "," + Str(r) + "," + "  " + clt1(tone) + clf1(fone) + clf2(ftwo) +
clp(pers) + cld(theta) + clt3(tthree)
Print #12, rule(r)

'Now to find which file to put data into, required for getting the correct filename and
filepath in windows
d$ = Chr$(34)
c$ = filepath$
a$ = Str$(r)
g = Len(a$)
q = g - 1
s$ = Right$(a$, q)
b$ = ".csv"
FileName$ = c$ + s$ + b$
FileName2$ = c$ + s$ + "NonSpurious.csv"
Open FileName$ For Output As #9
'Open FileName2$ For Output As #500

'Print #500, "Ta,Tc,GoodCount,BadCount"
```

```
'Main iteration for all datasets begins here

For i = 1 To n - 1

Call lingpersclassify(i, tone, fone, ftwo)

Print #14, r; ","; i; ","; cp(i); ","; mua(10, i); ","; mua(11, i); ","; mua(12, i)

'Calculation of truth of antecedent

If theta = 1 Then
kmax(r) = 2
ElseIf theta = 2 Then
kmax(r) = 10
ElseIf theta = 3 Then
kmax(r) = 50
End If

tasum(i) = 0
If i <= 60 Then
tasum(i) = 0
ta2(i) = (mua(tone, i) * mua(fone, i) * mua(ftwo, i) * mua(pers, i)) ^ (1 / 4)
ta(i) = ta2(i)
Else:
ta2(i) = (mua(tone, i) * mua(fone, i) * mua(ftwo, i) * mua(pers, i)) ^ (1 / 4)
For ff = 0 To kmax(r)
mudelay = 1 - ((kmax(r) - ff) / (kmax(r)))
tasum(i) = tasum(i) + (mudelay * ta2(i - ff))
Next ff
ta(i) = tasum(i) / kmax(r)
End If

'Calculation of truth of consequent
If theta = 1 Then
  tc(i) = muc(tthree, i)
  ElseIf theta = 2 Then: tc(i) = muc(a2, i)
  ElseIf theta = 3 Then: tc(i) = muc(a3, i)
End If

If r = 20 * Int((r / 20)) Then
Print #16, r; ","; tasum(i); ","; kmax(r); ","; ta(i)
End If

'To calculate the number of trips made into quadrant II

If ta(i) > 0.5 And ta(i) <= 1 And tc(i) > 0.5 And tc(i) <= 1 Then
```

```
        If i = 1 Then
        counter(i) = 1
        ' counter(i) is used to calculate the number of consecutive points in the quadrant
        Else: counter(i) = counter(i - 1) + 1
        End If

        If counter(i) = tot Then
           'goodtrip(r) is used to calculate the number of good trips made by a rule
           goodtrip(r) = goodtrip(r) + 1
        End If

ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) >= 0 And tc(i) <= 0.5 Then
   counter(i) = 0
ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) > 0.5 And tc(i) <= 1 Then
   counter(i) = 0
ElseIf ta(i) > 0.5 And ta(i) <= 1 And tc(i) >= 0 And tc(i) <= 0.5 Then
   counter(i) = 0
End If

'To calculate the number of trips made into quadrant 4
If ta(i) > 0.5 And ta(i) <= 1 And tc(i) >= 0 And tc(i) <= 0.5 Then
      If i = 1 Then
         counter2(i) = 1
      Else: counter2(i) = counter2(i - 1) + 1
      End If

      If counter2(i) = tot Then
         badtrip(r) = badtrip(r) + 1
      End If
ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) >= 0 And tc(i) <= 0.5 Then counter2(i) = 0
ElseIf ta(i) >= 0 And ta(i) <= 0.5 And tc(i) > 0.5 And tc(i) <= 1 Then counter2(i) = 0
ElseIf ta(i) > 0.5 And ta(i) <= 1 And tc(i) > 0.5 And tc(i) <= 1 Then counter2(i) = 0
End If

' to identify only those Tas and Tcs that are non-spurious
If i <> 1 Then
   If (counter(i) = 0 And counter(i - 1) >= tot) Or (counter2(i) = 0 And counter2(i - 1) >=
tot) Then
        If (counter(i) = 0 And counter(i - 1) >= tot) Then
           max = counter(i - 1)

        ElseIf (counter2(i) = 0 And counter2(i - 1) >= tot) Then
           max = counter2(i - 1)
        End If
           val = i - 1 - (max - 1)
           For j = 1 To max
```

```
'Calculation of the number of (non-spurious) hits of consequents in all regions(of size 0.1
by 0.1) of quadrants II,IV
            column = 0
               For k = 0.6 To 1 Step 0.1
               column = column + 1
               row = 0
                  For l = 0.1 To 1.1 Step 0.1
                  row = row + 1
                  'Note:0 is included in the first consequent zone
                     If l = 0.1 Then
                        If ta(val) <= k And ta(val) > (k - 0.1) And tc(val) <= 0.1 And tc(val)
>= 0 Then
                           numpts(r, row, column) = numpts(r, row, column) + 1
                        End If
                     Else:
                        If ta(val) <= k And ta(val) > (k - 0.1) And tc(val) <= l And tc(val) >
(l - 0.1) Then
                           numpts(r, row, column) = numpts(r, row, column) + 1
                        End If
                     End If
                  Next l
               Next k

            'Calculation of number of hits of antecedents in each of the 5 antecedent zones

               If ta(val) <= 0.6 And ta(val) > 0.5 Then
               num = 1
               ElseIf ta(val) <= 0.7 And ta(val) > 0.6 Then
               num = 2
               ElseIf ta(val) <= 0.8 And ta(val) > 0.7 Then
               num = 3
               ElseIf ta(val) <= 0.9 And ta(val) > 0.8 Then
               num = 4
               ElseIf ta(val) <= 1 And ta(val) > 0.9 Then
               num = 5
               Else: GoTo continue
               End If

               hits(r, num) = hits(r, num) + 1

continue:
               'Print #500, ta(val); ","; tc(val); ","; counter(val); ","; counter2(val)
               val = val + 1
            Next j

   End If
```

```
End If

Print #9, cat; ","; r; ","; ta(i); ","; tc(i); ","; counter(i); ","; counter2(i)

Next i

'Calculating MERIT(metric 1) for each rule
'Note: The variable used for MERIT is 'accuracy'
accuracy(r) = goodtrip(r) - badtrip(r)


Close #9
'Close #500

For i = 1 To 5
   Print #51, ""; ","; ""; ","; ""; ","; ""; ","; ""; ","; ""; ","; hits(r, i); ","; "TotalHitsTaZone"
& i
Next i

For jjj = 1 To 10
   Print #51, r; ","; numpts(r, jjj, 1); ","; numpts(r, jjj, 2); ","; numpts(r, jjj, 3); ",";
   numpts(r, jjj, 4); ","; numpts(r, jjj, 5); ","; ""; ","; "TcZone" & jjj
Next jjj

                        Next pers
                    Next ftwo
                Next fone
            Next tone
        Next tthree
Next theta

Print #50, "Rule,Good Trips, Bad Trips,, Merit"

For r = 1 To 729
   Print #50, r; ","; goodtrip(r); ","; badtrip(r); ","; ""; ","; accuracy(r)
Next r

Close #500
Close #12
Close #14
Close #16
Close #50
Close #51

time2 = Timer
text1.Text = "First part of programme is over using time=" + Str$(time2 - time1)
```

```
SelMode.MousePointer = 0

End Sub

Public Sub classify(n As Variant, t1() As Variant, f1() As Variant, f2() As Variant, t3s()
As Variant, t3m() As Variant, t3l() As Variant)
'sub for linguistic variable classification of all  variables except persistence
'Written by Nitin Sharma

ReDim mua(1 To 12, 1 To n)
ReDim muc(1 To 9, 1 To n)

CommonDialog1.DialogTitle = "Please Choose location and name for classification file"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowSave
filepathr$ = CommonDialog1.FileName
fileclassify$ = CommonDialog1.FileTitle
zz = Len(fileclassify$)
zx = Len(filepathr$)
filepath$ = Left$(filepathr$, zx - zz)

Open filepathr$ For Output As #2
Print #2,
"time,mua1,mua2,mua3,mua4,mua5,mua6,mua7,mua8,mua9,muc1,muc2,muc3,muc4,mu
c5,muc6,muc7,muc8,muc9"

For i = 1 To n
'Classification of t1 using 5-50-100 limits
    If t1(i) <= 5 Then mua(1, i) = 1
    If t1(i) >= 100 Then mua(3, i) = 1
    If t1(i) = 50 Then mua(2, i) = 1
    If t1(i) > 5 And t1(i) < 50 Then
    mua(1, i) = (50 - t1(i)) / (50 - 5)
    mua(2, i) = (t1(i) - 5) / (50 - 5)
    End If
    If t1(i) > 50 And t1(i) < 100 Then
    mua(2, i) = (100 - t1(i)) / (100 - 50)
    mua(3, i) = (t1(i) - 50) / (100 - 50)
    End If

'Classification of f1 using 1-15-29 limits
    If f1(i) <= 1 Then mua(4, i) = 1
    If f1(i) >= 29 Then mua(6, i) = 1
    If f1(i) = 15 Then mua(5, i) = 1
    If f1(i) > 1 And f1(i) < 15 Then
```

```
        mua(4, i) = (15 - f1(i)) / (14)
        mua(5, i) = (f1(i) - 1) / (14)
        End If
        If f1(i) > 15 And f1(i) < 29 Then
        mua(5, i) = (29 - f1(i)) / (14)
        mua(6, i) = (f1(i) - 15) / (14)
        End If

'Classification of f2 using 1-12-23 limits
        If f2(i) <= 1 Then mua(7, i) = 1
        If f2(i) >= 23 Then mua(9, i) = 1
        If f2(i) = 12 Then mua(8, i) = 1
        If f2(i) > 1 And f2(i) < 12 Then
        mua(7, i) = (12 - f2(i)) / (11)
        mua(8, i) = (f2(i) - 1) / (11)
        End If
        If f2(i) > 12 And f2(i) < 23 Then
        mua(8, i) = (23 - f2(i)) / (11)
        mua(9, i) = (f2(i) - 12) / (11)
        End If

'Classification of t3_after_short_delay  using limits 5-50-95
        If t3s(i) <= 5 Then muc(1, i) = 1
        If t3s(i) >= 95 Then muc(3, i) = 1
        If t3s(i) = 50 Then muc(2, i) = 1
        If t3s(i) > 5 And t3s(i) < 50 Then
        muc(1, i) = (50 - t3s(i)) / (45)
        muc(2, i) = (t3s(i) - 5) / (45)
        End If
        If t3s(i) > 50 And t3s(i) < 95 Then
        muc(2, i) = (95 - t3s(i)) / (45)
        muc(3, i) = (t3s(i) - 50) / (45)
        End If

'Classification of t3_after_medium_delay  using limits 5-50-95
        If t3m(i) <= 5 Then muc(4, i) = 1
        If t3m(i) >= 95 Then muc(6, i) = 1
        If t3m(i) = 50 Then muc(5, i) = 1
        If t3m(i) > 5 And t3m(i) < 50 Then
        muc(4, i) = (50 - t3m(i)) / (45)
        muc(5, i) = (t3m(i) - 5) / (45)
        End If
        If t3m(i) > 50 And t3m(i) < 95 Then
        muc(5, i) = (95 - t3m(i)) / (45)
        muc(6, i) = (t3m(i) - 50) / (45)
        End If
```

```
'Classification of t3_after_long_delay  using limits 5-50-95
        If t3l(i) <= 5 Then muc(7, i) = 1
        If t3l(i) >= 95 Then muc(9, i) = 1
        If t3l(i) = 50 Then muc(8, i) = 1
        If t3l(i) > 5 And t3l(i) < 50 Then
        muc(7, i) = (50 - t3l(i)) / (45)
        muc(8, i) = (t3l(i) - 5) / (45)
        End If
        If t3l(i) > 50 And t3l(i) < 95 Then
        muc(8, i) = (95 - t3l(i)) / (45)
        muc(9, i) = (t3l(i) - 50) / (45)
        End If
Print #2, time(i); ","; mua(1, i); ","; mua(2, i); ","; mua(3, i); ","; mua(4, i); ","; mua(5, i);
","; mua(6, i); ","; mua(7, i); ","; mua(8, i); ","; mua(9, i); ","; muc(1, i); ","; muc(2, i); ","
_
; muc(3, i); ","; muc(4, i); ","; muc(5, i); ","; muc(6, i); ","; muc(7, i); ","; muc(8, i); ",";
muc(9, i)
Next i

Reset
Erase t1, f1, f2
text1.Text = text1.Text + vbCrLf + "Classify is done"

End Sub

Public Sub persist(n As Variant, t1() As Variant, f1() As Variant, f2() As Variant)

'This sub introduces the persistence term into the antecedent of the rules
'persistence is calculated by counting the number of minutes the current value
'of the 3 variables has persisted based on an error function and combines it
' to get the peristence product or weighting function that should be multiplied
'into the antecedent truth later in TATC, the counter is according to data point
'and not rule because persistence changes with time.

'Written by Nitin Sharma

Dim t1p(), f1p(), f2p()
ReDim t1p(1 To n)
ReDim f1p(1 To n)
ReDim f2p(1 To n)
ReDim cp(1 To n)

filepaths$ = filepath$ + "persist.csv"

Open filepaths$ For Output As #4
```

```
t1p(1) = 0
f1p(1) = 0
f2p(1) = 0
cp(1) = 0

Print #4, "time,t1p,f1p,f2p,cp"
Print #4, time(1); ","; t1p(1); ","; f1p(1); ","; f2p(1); ","; cp(1)

For i = 2 To n
If Abs(t1(i) - t1(i - 1)) <= 0.1 Then
        t1p(i) = t1p(i - 1) + 1
        Else: t1p(i) = 0
End If
If Abs(f1(i) - f1(i - 1)) <= 0.1 Then
        f1p(i) = f1p(i - 1) + 1
        Else: f1p(i) = 0
End If
If Abs(f2(i) - f2(i - 1)) <= 0.1 Then
        f2p(i) = f2p(i - 1) + 1
        Else: f2p(i) = 0
End If
cp(i) = (t1p(i) * f1p(i) * f2p(i)) ^ (1 / 3)
Print #4, time(i); ","; t1p(i); ","; f1p(i); ","; f2p(i); ","; cp(i)
Next i
text1.Text = text1.Text + vbCrLf + "Persist is done"

End Sub

Private Sub Command2_Click()
Opti.Show
End Sub

Private Sub dispmode_Click()
DispTSD.Show
End Sub

Private Sub Form_Unload(Cancel As Integer)
prog = 740
bar1.Value = prog
Unload Me
Set SelMode = Nothing
End
End Sub

Private Sub quit_Click()
Call Form_Unload(0)
```

```
End Sub

Private Sub start_Click()

q = InputBox("Please enter number of lines in input.csv including the header", "Enter
Information", "2605")
p = InputBox("Please enter the value of largest delay", "Enter Information", "60")

If p = "" Or q = "" Then
   text1.Text = "You did not enter value of variables"
   Else
   n = q - p
   Call afterstart((n))
End If

End Sub

Public Sub afterstart(n As Variant)

'Main subroutine that calls all other subs
'Written by Nitin Sharma
'Modified by Preetica Kumar

prog = prog + 1
bar1.Value = prog

ReDim time(1 To n)
ReDim t1(1 To n)
ReDim f1(1 To n)
ReDim f2(1 To n)
ReDim t3s(1 To n)
ReDim t3m(1 To n)
ReDim t3l(1 To n)
'CommonDialog1.FileTitle = "input"
CommonDialog1.DialogTitle = "Please Choose input file"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowOpen
filepatha$ = CommonDialog1.FileName

Open filepatha$ For Input As #1
Input #1, a$, b$, c$, d$, e$, f$, g$ 'dummy variables for first line to remove the header
containing name of variables
'Print a$, b$, c$, d$, e$, f$, g$

For i = 1 To n 'input of data
```

```
Input #1, time(i), t1(i), f1(i), f2(i), t3s(i), t3m(i), t3l(i)
Next i

Call classify(n, t1(), f1(), f2(), t3s(), t3m(), t3l())
prog = prog + 1
bar1.Value = prog

Call linguipersist(n, mua())
prog = prog + 1
bar1.Value = prog

'Threshold criteria
totaltime = InputBox("Please enter the minimum amount of time to be spent in the good
quadrant(in seconds)", "Enter Info")
samptime = InputBox("Please enter the sampling time(in seconds)", "Enter Info")
tot = totaltime / samptime

Call TATC(n, mua(), muc(), pa(), tot)
SelMode.start.Enabled = False
text1.Text = text1.Text + vbCrLf + "Press Display for results"

End Sub
```

The VBA code for the form *DispTSD.frm* is listed below.

```
'Program that displays the TSD of each rule in the selection mode(for historical data)
'The TSD denotes the 'trips' made by each rule

Public runi As Integer
Public fpath$
Public fpathold$
Dim xlApp As Excel.Application
Dim xlbook As Excel.Workbook
Dim xlsheet As Excel.Worksheet
Dim rul$()
Dim gu() 'number of good trips
Dim ba() 'number of bad trips
Dim ac() 'MERIT of all rules

Public Sub initializer()
Set xlApp = New Excel.Application
End Sub
```

```
Public Sub diplay_Click()
'sub that displays the TSD

a = Len(filer.FileName)
b = a - 4
c$ = Left(filer.FileName, b)
d = val(c$)
Text2.Text = d
text1.Text = fpath$
Text3.Text = rul(d)
Text6.Text = "Number of good trips = " + Str(gu(d)) + vbCrLf + "Number of bad trips =
" + Str(ba(d)) + vbCrLf + " Merit = Good trips - Bad trips" + vbCrLf + Str(ac(d))
runi = runi + 1

If runi = 1 Then
   Call initializer
   xlApp.Application.DisplayAlerts = False
'ElseIf runi > 1 Then
'   xlApp.Workbooks.Close
End If

'If fpath$ = fpathold$ Then
'text1.Text = "This file is already displayed, Choose another first to reopen"
'GoTo multipath
'End If

Set xlbook = xlApp.Workbooks.Open(fpath$)
Set xlsheet = xlbook.ActiveSheet

With xlApp
xlsheet.Range("C1:D2544").Select
Charts.Add
   ActiveChart.ChartType = xlXYScatterLines
   ActiveChart.SetSourceData Source:=xlsheet.Range("C1:D2544"), PlotBy _
      :=xlColumns

   ActiveChart.Axes(xlValue).MajorGridlines.Select
   Selection.Delete
   ActiveChart.PlotArea.Select
   With Selection.Border
      .ColorIndex = 16
      .Weight = xlThin
      .LineStyle = xlContinuous
   End With
   With Selection.Interior
```

```
    .ColorIndex = 2
    .PatternColorIndex = 1
    .Pattern = xlSolid
 End With

ActiveChart.Legend.Delete

If Option1.Value = False And Option2.Value = False Then
 ActiveChart.Axes(xlCategory).Select
 With ActiveChart.Axes(xlCategory)
    .MinimumScaleIsAuto = True
    .MaximumScaleIsAuto = True
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
 End With
 ActiveChart.Axes(xlValue).Select
 With ActiveChart.Axes(xlValue)
    .MinimumScaleIsAuto = True
    .MaximumScale = 1
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
 End With

 ElseIf Option1.Value = True Then
 ActiveChart.Axes(xlCategory).Select
 With ActiveChart.Axes(xlCategory)
    .MinimumScaleIsAuto = True
    .MaximumScaleIsAuto = True
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
 End With
 ActiveChart.Axes(xlValue).Select
 With ActiveChart.Axes(xlValue)
    .MinimumScale = 0.5
```

```
    .MaximumScale = 1
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
End With

ElseIf Option2.Value = True Then

ActiveChart.Axes(xlCategory).Select
With ActiveChart.Axes(xlCategory)
    .MinimumScaleIsAuto = True
    .MaximumScaleIsAuto = True
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
End With
ActiveChart.Axes(xlValue).Select
With ActiveChart.Axes(xlValue)
    .MinimumScale = 0
    .MaximumScale = 0.5
    .MinorUnit = 0.5
    .MajorUnit = 0.5
    .Crosses = xlAutomatic
    .ReversePlotOrder = False
    .ScaleType = xlLinear
    .DisplayUnit = xlNone
End With
End If
ActiveChart.PlotArea.Select
With ActiveChart.Axes(xlCategory)
    .HasMajorGridlines = True
    .HasMinorGridlines = False
End With
With ActiveChart.Axes(xlValue)
    .HasMajorGridlines = True
    .HasMinorGridlines = False
End With

For i = 1 To 2544
```

```
If i <> 1 Then
If xlsheet.Cells(i, 3) >= 0.5 And xlsheet.Cells(i, 4) > 0.5 Then

   With ActiveChart.SeriesCollection(1).Points(i)
      .MarkerBackgroundColorIndex = 10
      .MarkerForegroundColorIndex = 10
      .MarkerStyle = xlDiamond
      .MarkerSize = 4
      .Shadow = False
   End With

   With ActiveChart.SeriesCollection(1).Points(i).Border
      .ColorIndex = 10
      .Weight = xlThin
   End With

   With ActiveChart.SeriesCollection(1).Points(i + 1).Border
      .ColorIndex = 10
      .Weight = xlThin
   End With
End If
If xlsheet.Cells(i, 3) >= 0.5 And xlsheet.Cells(i, 4) <= 0.5 Then

  With ActiveChart.SeriesCollection(1).Points(i)
      .MarkerBackgroundColorIndex = 3
      .MarkerForegroundColorIndex = 3
      .MarkerStyle = xlDiamond
      '.Smooth = True
      .MarkerSize = 4
      .Shadow = False
   End With

   With ActiveChart.SeriesCollection(1).Points(i).Border
      .ColorIndex = 3
      .Weight = xlThin
   End With

   With ActiveChart.SeriesCollection(1).Points(i + 1).Border
      .ColorIndex = 3
      .Weight = xlThin
   End With

End If
End If

Next i
```

```
End With

Set Chart1 = xlbook.ActiveChart
OLE2.CreateLink fpath$
OLE2.SizeMode = 1

fpathold$ = fpath$
DispTSD.Caption = "Display - " + "Rule #" + Str(d)
multipath:
End Sub

Private Sub direr_Change()

ChDir direr.path
End Sub

Private Sub driver_Change()
'Written by Nitin Sharma
On Error GoTo booga

direr.path = driver.Drive
ChDrive driver.Drive

booga:
text1.Text = "The disk is not ready"
driver.Drive = direr.path

End Sub

Private Sub filer_Click()
If Right(filer.path, 1) <> "\" Then
   fpath$ = filer.path + "\" + filer.FileName
   text1.Text = filer.path + "\" + filer.FileName
Else
   fpath$ = filer.path + filer.FileName
   text1.Text = filer.path + filer.FileName
End If
Option1.Value = False
Option2.Value = False
End Sub

Private Sub filer_DblClick()
Call DispTSD.diplay_Click
End Sub
```

```
Public Sub Form_Load()

ReDim rul$(1 To 729)
ReDim gu(1 To 729)
ReDim ba(1 To 729)
ReDim ac(1 To 729)

CommonDialog1.DialogTitle = "Where is the rulesheet (output) file located?"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowOpen
filepath2$ = CommonDialog1.FileName
filepath$ = Left(filepath2$, (Len(filepath2$) - Len("rulesheet.csv")))
Open filepath2$ For Input As #1
filepather$ = filepath$ + "trips.csv"
Open filepather$ For Input As #2
direr.path = filepath$

Input #2, title1$, title2$, title3$, emptySpace$, title4$
'Reinput data from metric files created in previous form
For r = 1 To 729
Input #1, ju$, bu$, rul(r)
Input #2, juju$, gu(r), ba(r), bubu$, ac(r)
Next r

Close #1
Close #2

SelMode.start.Enabled = False
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
Call Form_Unload(0)
End Sub

Private Sub Form_Unload(Cancel As Integer)
'Written by Nitin Sharma
If runi >= 1 Then
Set xlsheet = Nothing
'xlbook.Close
'xlApp.Workbooks.Close
'Set xlbook = xlApp.Workbooks.Close(fpath$)
xlApp.quit
Set xlApp = Nothing
runi = 0
End If
```

```
SelMode.text1.Text = "Before Running Display Mode again please quit this window and
rerun the application"
SelMode.dispmode.Enabled = False
DispTSD.Hide
Set DispTSD = Nothing
DetectExcel
End Sub
Sub DetectExcel()
'Written by Nitin Sharma
Dim MyXL As Object   ' Variable to hold reference
                     ' to Microsoft Excel.
Dim ExcelWasNotRunning As Boolean   ' Flag for final release.

On Error Resume Next
' Test to see if there is a copy of Microsoft Excel already running.
' Getobject function called without the first argument returns a
' reference to an instance of the application. If the application isn't
' running, an error occurs.
Set MyXL = GetObject(, "Excel.Application")

If MyXL <> Empty Then
   SelMode.text1.Text = "Excel is running and should quit on it's own when you close
SelMode, else close manually"
   MyXL.Visible = True
End If

'MyXL.quit
Err.Clear   ' Clear Err object in case error occurred.

End Sub

Private Sub quit2_Click()
OLE2.Close
'Set xltrap = GetObject(, "Excel.Application")
'xlApp.Application.quit
'xlApp.Visible

Call Form_Unload(0)
End Sub
```

The code for the form *opti.frm* is listed below.

```
'This program optimizes the initial rule base
'using the following 2 selection criteria:
'1. Number of good trips >=2
'2. MERIT >=1
'It is used in the selection mode.
Dim rule$()
Dim rulnum() As Single
Dim num() As Single
Dim stmt$()
Dim gudtrip() As Single 'number of goos trips
Dim baddtrip() As Single 'number of bad trips
Dim gtrip() As Single
Dim btrip() As Single
Dim acc() As Single 'MERIT of each rule
Dim acc1() As Single
Dim counter As Integer
Dim path1$
Dim filepath7$
Dim filepath51$
Dim rulerank(1 To 729)


Private Sub Command1_Click()
Opti.Hide
Set Opti = Nothing
End Sub

Private Sub Form_Load()
'Sub for input of data

ReDim rulnum(1 To 729)
ReDim num(1 To 729)
ReDim stmt$(1 To 729)
ReDim gudtrip(1 To 729)
ReDim baddtrip(1 To 729)
ReDim gtrip(1 To 729)
ReDim btrip(1 To 729)
ReDim acc(1 To 729)
ReDim acc1(1 To 729)
ReDim rule$(1 To 729)

CommonDialog1.DialogTitle = "Please Choose the rulesheet file"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
```

```
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowOpen
filepath7$ = CommonDialog1.FileName

Open filepath7$ For Input As #2

path1$ = Left(filepath7$, (Len(filepath7$) - Len("rulesheet.csv")))
filepath51$ = path1$ + "trips.csv"
Open filepath51$ For Input As #101
Input #101, title1$, title2$, title3$, blah$, title4$

For i = 1 To 729 'input of data
   Input #2, a, b, rule$(i)
   Input #101, rulnum(i), gtrip(i), btrip(i), Dummy3$, acc(i)
Next i

Close #2
Close #101

SelMode.Command2.Enabled = False

Call optifinal

End Sub

Sub optifinal()
'sub for optimization of rule-base

Text2.Text = "Optimization has" + vbCrLf + "BEGUN"

filepath7$ = path1$ + "finalopti.csv"
Open filepath7$ For Output As #7

counter = 0
For i = 1 To 729
'checking for selection criteria
   If gtrip(i) > 1 And acc(i) >= 1 Then
      counter = counter + 1
      num(counter) = rulnum(i)
      stmt$(counter) = rule$(i)
      gudtrip(counter) = gtrip(i)
      baddtrip(counter) = btrip(i)
      acc1(counter) = acc(i)
   End If
Next i
```

```
'For i = 1 To counter
   'rulerank(i) = 0
'Next i

'For i = 1 To counter
   'For j = 1 To counter
      'If j <> i And acc1(j) > acc1(i) Then
         'rulerank(i) = rulerank(i) + 1
      'End If
   'Next j
'Next i

Print #7, "Rule no., Statement, Good trips, Bad trips"

For i = 1 To counter
   Print #7, num(i); ","; stmt$(i); ","; gudtrip(i); ","; baddtrip(i); ","; counter
   List1.AddItem (num(i))
Next i
Close #7

Text2.Text = "Optimization has been" + vbCrLf + "COMPLETED!"

End Sub

Private Sub List1_Click()
a = List1.ListIndex + 1
Text1.Text = "Good trips = " + Str(gudtrip(a)) + vbCrLf + "Bad trips = " +
Str(baddtrip(a))
End Sub
```

The code for the form *PredMode.frm* is listed below

```
'Algorithm for the prediction mode
'This program uses only the antecedents of NEW DATA to predict their outcomes for
each rule
'based on historical distribution information calculated in the selection mode.

Public p As Variant, q As Variant
Public n As Variant

Dim time()
Dim t1()       'temperature 1
Dim f1()       'flow rate 1
Dim f2()       'flow rate 2
Dim t3s()      'temperature 3 after short delay
```

```
Dim t3m()      'temperature 3 after medium delay
Dim t3l()      'temperature 3 after long delay
Dim cp()       'combined persistance

'mua()=(T1(L,M,H),F1(L,M,H),F2(L,M,H),P(L,M,H))
Dim mua() As Double 'membership of antecedent
Dim filepath$
Dim filepersist$
Dim filepatha$

Dim i As Integer    'counter of data point
Dim pa() As Double  'individual persistence of every variable in antecedent
Public progress As Integer 'variable to update the progress bar

Sub lingpersclassify(i, tone, fone, ftwo)
'This subroutine calculates the combined persistence of the antecedent and
'classifies it into three fuzzy categories(L,M,H)
'written by Nitin Sharma

mua(10, i) = 0
mua(11, i) = 0
mua(12, i) = 0

'the combined persistence time is the minimum persistence time among all inputs
If pa(fone, i) >= pa(tone, i) Then
   If pa(ftwo, i) >= pa(tone, i) Then
      cp(i) = pa(tone, i)
   End If
End If

If pa(tone, i) >= pa(fone, i) Then
   If pa(ftwo, i) >= pa(fone, i) Then
      cp(i) = pa(fone, i)
   End If
End If

If pa(tone, i) >= pa(ftwo, i) Then
   If pa(fone, i) >= pa(ftwo, i) Then
      cp(i) = pa(ftwo, i)
   End If
End If

'Classification of persist using 1-20-40 limits
      If cp(i) = 1 Then mua(10, i) = 1
      If cp(i) >= 40 Then mua(12, i) = 1
      If cp(i) = 20 Then mua(11, i) = 1
```

```
        If cp(i) > 1 And cp(i) < 20 Then
        mua(10, i) = (20 - cp(i)) / (19)
        mua(11, i) = (cp(i) - 1) / (19)
        boogi = cp(i)
        End If
        If cp(i) > 20 And cp(i) < 40 Then
        mua(11, i) = (40 - cp(i)) / (20)
        mua(12, i) = (cp(i) - 20) / (20)
        End If
End Sub

Sub TATC(n, mua() As Double, pa() As Double)
'Subroutine for calculating the truth of the antecedent , using the predefined values of
linguistic variables
'in the previous SUB. The number of iterations are Nx3x3x3x3x3x3 i.e. 7 loops
'in total cover N points and 729 rules in 81 categories.
'Written By Nitin Sharma, Modified by Preetica Kumar

'This subroutine also calculates the number of antecedent hits for new data
'Written by Preetica Kumar

Dim zz, zx As Integer

Dim slope As Double
Dim Y As Double
Dim X As Double
Dim ta() As Double
Dim tasum() As Double
Dim ta2() As Double
Dim kmax() As Single


Dim numpts()
Dim hits() As Double
Dim row, column As Integer
Dim ii, jj, pp, qq As Integer
Dim num As Integer
Dim k, l As Single

Dim filepath2$
Dim filepath3$

'Redimensioning of variables and initialization of certain linguistic string constants

ReDim cp(1 To n)
```

```
ReDim ta(1 To n)

ReDim hits(1 To 729, 1 To 5)

ReDim ta2(1 To n)
ReDim tasum(1 To n)
ReDim kmax(1 To 729)

'Asking the user where he wishes to save the files
answer:
answer$ = InputBox("Continue saving output files in same directory? Please type y
(Yes)// n (No) // q (Quit)", "Enter Information")
If answer$ = "y" Then
zz = Len(filepersist$)
zx = Len(filepaths$)
filepath$ = Left$(filepaths$, zx - zz)
ElseIf answer$ = "n" Then
filepath$ = InputBox("Please Enter target directory for output files", "Enter Information",
"e:\vbprogs\output\")
ElseIf answer$ = "q" Then
End
Else
Text1.Text = "Please input  y or n or q to proceed"
GoTo answer
End If

'Opening output files in specified filepath

filepath4$ = filepath$ + "Predcombop.csv"
Open filepath4$ For Output As #14
filepath101$ = filepath$ + "Prednumpoints.csv"
Open filepath101$ For Output As #51

Print #51, "Rule,Antecedent Hits in each Ta Zone for NEW DATA"
Print #14, "r,i,cp,mua10,mua11,mua12"

r = 0
cat = 0
For theta = 1 To 3
      For tthree = 1 To 3
      cat = cat + 1
         For tone = 1 To 3
               For fone = 4 To 6
                     For ftwo = 7 To 9
                           For pers = 10 To 12
```

```
'this is indexing for the consequent part
a2 = tthree + 3
a3 = tthree + 6

'now calculate rule number
r = r + 1

progress = progress + 1
ProgressBar1.Value = progress

For i = 1 To 5
hits(r, i) = 0
Next i

'Main iteration for all datasets begins here

For i = 1 To n - 1

Call lingpersclassify(i, tone, fone, ftwo)

Print #14, r; ","; i; ","; cp(i); ","; mua(10, i); ","; mua(11, i); ","; mua(12, i)

'calculation of truth of antecedent

If theta = 1 Then
kmax(r) = 2
ElseIf theta = 2 Then
kmax(r) = 10
ElseIf theta = 3 Then
kmax(r) = 50
End If

tasum(i) = 0
If i <= 60 Then
tasum(i) = 0
ta2(i) = (mua(tone, i) * mua(fone, i) * mua(ftwo, i) * mua(pers, i)) ^ (1 / 4)
ta(i) = ta2(i)
Else:
ta2(i) = (mua(tone, i) * mua(fone, i) * mua(ftwo, i) * mua(pers, i)) ^ (1 / 4)
For ff = 0 To kmax(r)
mudelay = 1 - ((kmax(r) - ff) / (kmax(r)))
tasum(i) = tasum(i) + (mudelay * ta2(i - ff))
Next ff
ta(i) = tasum(i) / kmax(r)
End If
```

```
'calculation of number of hits of antecedents in each of the 5 antecedent zones

If ta(i) <= 0.6 And ta(i) > 0.5 Then
   num = 1
ElseIf ta(i) <= 0.7 And ta(i) > 0.6 Then
   num = 2
ElseIf ta(i) <= 0.8 And ta(i) > 0.7 Then
   num = 3
ElseIf ta(i) <= 0.9 And ta(i) > 0.8 Then
   num = 4
ElseIf ta(i) <= 1 And ta(i) > 0.9 Then
   num = 5
Else: GoTo continue
End If

hits(r, num) = hits(r, num) + 1

continue:
Next i

For jjj = 1 To 5
   Print #51, r; ","; hits(r, jjj)
Next jjj

                        Next pers
                    Next ftwo
                Next fone
            Next tone
        Next tthree
Next theta

Close #14
Close #51

End Sub

Public Sub classify(n As Variant, t1() As Variant, f1() As Variant, f2() As Variant, t3s()
As Variant, t3m() As Variant, t3l() As Variant)
'subprogram for linguistic variable classification of data
'Written by Nitin Sharma

ReDim mua(1 To 12, 1 To n)
ReDim muc(1 To 9, 1 To n)

CommonDialog1.DialogTitle = "Please Choose location and name for classification file"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
```

```
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowSave
filepathr$ = CommonDialog1.FileName
fileclassify$ = CommonDialog1.FileTitle
zz = Len(fileclassify$)
zx = Len(filepathr$)
filepath$ = Left$(filepathr$, zx - zz)

Open filepathr$ For Output As #2
Print #2, "time,mua1,mua2,mua3,mua4,mua5,mua6,mua7,mua8,mua9"

For i = 1 To n
'Classification of t1 using 5-50-100 limits
    If t1(i) <= 5 Then mua(1, i) = 1
    If t1(i) >= 100 Then mua(3, i) = 1
    If t1(i) = 50 Then mua(2, i) = 1
    If t1(i) > 5 And t1(i) < 50 Then
    mua(1, i) = (50 - t1(i)) / (50 - 5)
    mua(2, i) = (t1(i) - 5) / (50 - 5)
    End If
    If t1(i) > 50 And t1(i) < 100 Then
    mua(2, i) = (100 - t1(i)) / (100 - 50)
    mua(3, i) = (t1(i) - 50) / (100 - 50)
    End If

'Classification of f1 using 1-15-29 limits
    If f1(i) <= 1 Then mua(4, i) = 1
    If f1(i) >= 29 Then mua(6, i) = 1
    If f1(i) = 15 Then mua(5, i) = 1
    If f1(i) > 1 And f1(i) < 15 Then
    mua(4, i) = (15 - f1(i)) / (14)
    mua(5, i) = (f1(i) - 1) / (14)
    End If
    If f1(i) > 15 And f1(i) < 29 Then
    mua(5, i) = (29 - f1(i)) / (14)
    mua(6, i) = (f1(i) - 15) / (14)
    End If

'Classification of f2 using 1-12-23 limits
    If f2(i) <= 1 Then mua(7, i) = 1
    If f2(i) >= 23 Then mua(9, i) = 1
    If f2(i) = 12 Then mua(8, i) = 1
    If f2(i) > 1 And f2(i) < 12 Then
    mua(7, i) = (12 - f2(i)) / (11)
    mua(8, i) = (f2(i) - 1) / (11)
    End If
```

```
    If f2(i) > 12 And f2(i) < 23 Then
    mua(8, i) = (23 - f2(i)) / (11)
    mua(9, i) = (f2(i) - 12) / (11)
    End If


Print #2, time(i); ","; mua(1, i); ","; mua(2, i); ","; mua(3, i); ","; mua(4, i); ","; mua(5, i);
","; mua(6, i); ","; mua(7, i); ","; mua(8, i); ","; mua(9, i)
Next i
Reset
Erase t1, f1, f2

Close #2
End Sub

Public Sub persist(n As Variant, t1() As Variant, f1() As Variant, f2() As Variant)

'This sub introduces the persistence term into the antecedent of the rules
'persistence is calculated by counting the number of minutes the current value
'of the 3 variables has persisted based on an error function and combines it
' to get the peristence product or weighting function that should be multiplied
'into the antecedent truth later in TATC, the counter is according to data point
'and not rule because persistence changes with time.
'Written  by Nitin Sharma

Dim t1p(), f1p(), f2p()
ReDim t1p(1 To n)
ReDim f1p(1 To n)
ReDim f2p(1 To n)
ReDim cp(1 To n)

filepaths$ = filepath$ + "Predpersist.csv"

Open filepaths$ For Output As #4
t1p(1) = 0
f1p(1) = 0
f2p(1) = 0
cp(1) = 0

Print #4, "time,t1p,f1p,f2p,cp"
Print #4, time(1); ","; t1p(1); ","; f1p(1); ","; f2p(1); ","; cp(1)

For i = 2 To n
If Abs(t1(i) - t1(i - 1)) <= 0.1 Then
    t1p(i) = t1p(i - 1) + 1
    Else: t1p(i) = 0
```

```
End If
If Abs(f1(i) - f1(i - 1)) <= 0.1 Then
        f1p(i) = f1p(i - 1) + 1
        Else: f1p(i) = 0
End If
If Abs(f2(i) - f2(i - 1)) <= 0.1 Then
        f2p(i) = f2p(i - 1) + 1
        Else: f2p(i) = 0
End If
cp(i) = (t1p(i) * f1p(i) * f2p(i)) ^ (1 / 3)
Print #4, time(i); ","; t1p(i); ","; f1p(i); ","; f2p(i); ","; cp(i)
Next i
Close #4
End Sub

Sub Exp()
' Sub to calculate the 'EXPECTATIONS'(Metric 2) for all the rules

Dim prno(1 To 729) As Single
Dim ppts(1 To 729, 1 To 10, 1 To 5) 'historical distribution
Dim normpts(1 To 729, 1 To 10, 1 To 5) 'normalized historical distribution
Dim pdiaghit(1 To 729, 1 To 5, 1 To 5)
Dim PredHits(1 To 729, 1 To 5) As Double 'number of new antecedent hits in each of the
5 zones
Dim HistHits(1 To 729, 1 To 5) As Double 'number of historical antecedent hits in each
of the 5 zones
Dim product(1 To 729, 1 To 10, 1 To 5)
Dim pTotsum(1 To 729) As Single
Dim pexp(1 To 729, 1 To 10) As Double 'EXPECTATIONS for each rule in each of the
10 consequent zones
Dim pNormExp(1 To 729, 1 To 10) As Double
Dim pCumulativeExp(1 To 729) As Single
Dim pcdf(1 To 729, 1 To 10) As Double 'confidence factors for each of the 10
consequent zones
Dim plowlimit(1 To 729) As Single 'lower confidence limits
Dim puplimit(1 To 729) As Single  'upper confidence limits
Dim ptcmiddle(1 To 10) As Single
Dim ptcupper(1 To 10) As Single
Dim ptclower(1 To 10) As Single
Dim pweights(1 To 729) As Single
Dim pmean(1 To 729) As Single 'weighted mean average of the expectations

filepath55$ = filepath$ + "numpoints.csv"
Open filepath55$ For Input As #105

Input #105, title1$
```

```
Input #105, title1$, title2$, title3$, title4$, title5$, title6$

For r = 1 To 729
   For i = 1 To 5
     Input #105, Dummy10$, Dummy11$, Dummy12$, Dummy13$, Dummy14$,
Dummy15$, HistHits(r, i), Dummy16$
   Next i
   For i = 1 To 10
      Input #105, prno(r), ppts(r, i, 1), ppts(r, i, 2), ppts(r, i, 3), ppts(r, i, 4), ppts(r, i, 5),
Dummy17$, Dummy18$
  Next i
Next r


filepath54$ = filepath$ + "prednumpoints.csv"
Open filepath54$ For Input As #104

Input #104, title1$, title2$
For r = 1 To 729
   For i = 1 To 5
     Input #104, Dummy17$, PredHits(r, i)
   Next i
Next r

'converting the new antecedent hits to a diagonal matrix
For r = 1 To 729
   For i = 1 To 5
      For j = 1 To 5
         If i = j Then
         pdiaghit(r, i, j) = PredHits(r, i)
         Else
         pdiaghit(r, i, j) = 0
         End If
      Next j
   Next i
Next r
'calculating the number of normalized hits in the historical database.

For r = 1 To 729
   For i = 1 To 5
      For j = 1 To 10
         If HistHits(r, i) <> 0 Then
         normpts(r, j, i) = ppts(r, j, i) / HistHits(r, i)
         Else
         normpts(r, j, i) = 0
         End If
```

```
        Next j
    Next i
Next r
'sum of total antecedent hits made by new data
'For r = 1 To 729
  ' For i = 1 To 5
     '  sumPredHits(r) = sumPredHits(r) + PredHits(r, i)
  ' Next i
'Next r

' calculating the product of the number of points
'in each 0.1 x 0.1 grid with the number of hits in the 5 antecedent zones
For r = 1 To 729
    For k = 1 To 5
    For i = 1 To 10
        For j = 1 To 5
            product(r, i, k) = product(r, i, k) + (normpts(r, i, j) * pdiaghit(r, j, k))
        Next j
  Next i
  Next k
Next r

' adding all the points in each consequent zone
'to yield absolute EXPECTATIONS!
For r = 1 To 729
    For i = 1 To 10
        For j = 1 To 5
            pexp(r, i) = pexp(r, i) + product(r, i, j)
        Next j
    Next i
Next r

For r = 1 To 729
    pTotsum(r) = 0
Next r

'normalizing pexp to yield Normalized "Expectations"
For r = 1 To 729
    'calculation of the total sum of all the expectations for each rule
    For i = 1 To 10
            pTotsum(r) = pTotsum(r) + pexp(r, i)
    Next i
    'Normalizing Expectations(0-1)
    For i = 1 To 10
            If pTotsum(r) <> 0 Then
               pNormExp(r, i) = pexp(r, i) / pTotsum(r)
```

```
        Else: pNormExp(r, i) = 0
        End If
    Next i
Next r

'calculation of center of each bin
For i = 1 To 10
    ptclower(i) = (i - 1) * 0.1
    ptcupper(i) = i * 0.1
    ptcmiddle(i) = ptclower(i) + 0.05
Next i


' file to print the cumulative sums at each stage and the corresponding conf. factors.
filepath57$ = path1$ + "PredSums.csv"
Open filepath57$ For Output As #107

For r = 1 To 729
    For i = 1 To 10
        ' cumulative distribution function for each zone(cdf)to be used during interpolation
to determine (95%) confidence limits
        pCumulativeExp(r) = pCumulativeExp(r) + pNormExp(r, i)
        pcdf(r, i) = pCumulativeExp(r)
      Print #107, i; ","; pCumulativeExp(r); ","; pcdf(r, i)
    Next i
Next r

'calculating the weighted average: pmean
For r = 1 To 729
For i = 1 To 10
    pweights(r) = pNormExp(r, i) * ptcmiddle(i) + pweights(r)
Next i
'Since the total of the normalized expectations is 1
pmean(r) = pweights(r) / 1
Next r

' calculation of confidence limits by interpolation
For r = 1 To 729
  For i = 1 To 9
      If pmean(r) <> 0 Then

        'right on target
        If pcdf(r, i) < 0.025 And pcdf(r, i + 1) > 0.975 Then
           plowlimit(r) = ptcupper(i) + (ptcupper(i + 1) - ptcupper(i)) * (0.025 - pcdf(r, i))
/ (pcdf(r, i + 1) - pcdf(r, i))
```

```
            puplimit(r) = ptcupper(i) + (ptcupper(i + 1) - ptcupper(i)) * (0.975 - pcdf(r, i)) /
(pcdf(r, i + 1) - pcdf(r, i))
            Exit For
         End If

      'lower confidence limit
         If pcdf(r, i) < 0.025 And pcdf(r, i + 1) > 0.025 Then
            plowlimit(r) = ptcupper(i) + (ptcupper(i + 1) - ptcupper(i)) * (0.025 - pcdf(r, i))
/ (pcdf(r, i + 1) - pcdf(r, i))
            GoTo up
         End If
up:
      'upper confidence limit
         If pcdf(r, i) < 0.975 And pcdf(r, i + 1) > 0.975 Then
            puplimit(r) = ptcupper(i) + (ptcupper(i + 1) - ptcupper(i)) * (0.975 - pcdf(r, i)) /
(pcdf(r, i + 1) - pcdf(r, i))
            Exit For
         End If
      End If
   Next i
Next r

'file to store the absolute expected distribution values
filepath56$ = filepath$ + "PredExpectation.csv"
Open filepath56$ For Output As #106
'file to store the Normalized expected distribution values
filepath65$ = filepath$ + "PredNormalizedExpectation.csv"
Open filepath65$ For Output As #115

'file to store the predicted mean and upper and lower confidence limits.
filepath58$ = filepath$ + "PredFreshMetrics.csv"
Open filepath58$ For Output As #108
'file to store the individual expectations in each antecedent zone
filepath60$ = filepath$ + "PredIndividualExpectations.csv"
Open filepath60$ For Output As #110

Print #106, "Absolute Expectations for all rules"
Print #106,
"Rule,TcZone1,TcZone2,TcZone3,TcZone4,TcZone5,TcZone6,TcZone7,TcZone8,TcZo
ne9,TcZone10"
Print #108, "Rule,Mean Expectation, L.C.L(95%), U.C.L(95%)"

Print #115, "Normalized Expectations(0-1) for all rules"
Print #115,
"Rule,TcZone1,TcZone2,TcZone3,TcZone4,TcZone5,TcZone6,TcZone7,TcZone8,TcZo
ne9,TcZone10"
```

```
For i = 1 To 729

   Print #106, prno(i); ","; pexp(i, 1); ","; pexp(i, 2); ","; pexp(i, 3); ","; pexp(i, 4); ","; pexp(i, 5) _
   ; ","; pexp(i, 6); ","; pexp(i, 7); ","; pexp(i, 8); ","; pexp(i, 9); ","; pexp(i, 10)

   Print #115, prno(i); ","; pNormExp(i, 1); ","; pNormExp(i, 2); ","; pNormExp(i, 3); ","; pNormExp(i, 4); ","; pNormExp(i, 5) _
   ; ","; pNormExp(i, 6); ","; pNormExp(i, 7); ","; pNormExp(i, 8); ","; pNormExp(i, 9); ","; pNormExp(i, 10)

   Print #108, prno(i); ","; pmean(i); ","; plowlimit(i); ","; puplimit(i)

   For j = 1 To 10
         Print #110, prno(i); ","; product(i, j, 1); ","; product(i, j, 2); ","; product(i, j, 3); ","; product(i, j, 4); ","; product(i, j, 5)
   Next j
Next i

Close #104
Close #105
Close #106
Close #108
Close #110
Close #115

End Sub

Private Sub Command3_Click()
Unload Me
Call Form_Unload(0)
End Sub


Private Sub Form_Unload(Cancel As Integer)
Unload Me
Set PredMode = Nothing
End
End Sub


Private Sub Command1_Click()
'sub to take in information about the input file
q = InputBox("Please enter number of lines in the input file including the header", "Enter Information", "2605")
```

```
p = InputBox("Please enter the value of largest delay", "Enter Information", "60")

If p = "" Or q = "" Then
   Text1.Text = "You did not enter value of variables"
   Else
   n = q - p
   Call afterstart((n))
End If

End Sub

Sub linguipersist(n, mua() As Double)
'This subroutine provides a linguistic label for the maximum persistence of each data
point
'written by Nitin Sharma

Dim X As Integer
Dim maxi() As Integer
ReDim maxi(1 To 3)
ReDim pa(1 To 9, 1 To n)

filepaths$ = filepath$ + "Predlingpers.csv"
Open filepaths$ For Output As #10
Print #10, "time,patl,patm,path,paf1l,paf1m,paf1h,paf2l,paf2m,paf2h"

For i = 2 To n
For k = 1 To 3 'k represents t1 f1 and f2 respectively
j = (3 * k - 2)

 If mua(j, i) > mua(j + 1, i) Then
    If mua(j, i) > mua(j + 2, i) Then
       maxi(k) = j
    Else: maxi(k) = (j + 2)
    End If
 ElseIf mua(j, i) < mua(j + 1, i) Then
    If mua(j + 1, i) > mua(j + 2, i) Then
       maxi(k) = (j + 1)
    Else: maxi(k) = (j + 2)
    End If
End If

   'Persistence increases by one
   X = maxi(k)
   pa(X, i) = pa(X, i - 1) + 1

   For l = j To (j + 2)
```

```
    If l <> maxi(k) Then pa(l, i) = 0
    Next l

Next k
Print #10, time(i); ","; pa(1, i); ","; pa(2, i); ","; pa(3, i); ","; pa(4, i); ","; pa(5, i); ","; pa(6, i); ","; pa(7, i); ","; pa(8, i); ","; pa(9, i)
Next i

Close #10
End Sub

Public Sub afterstart(n As Variant)

'Main subroutine that calls all other subs
'Written by Nitin Sharma
'Modified by Preetica Kumar

ReDim time(1 To n)
ReDim t1(1 To n)
ReDim f1(1 To n)
ReDim f2(1 To n)
ReDim t3s(1 To n)
ReDim t3m(1 To n)
ReDim t3l(1 To n)

Text1.Text = "Calculation" + vbCrLf + "has BEGUN..."

CommonDialog1.DialogTitle = "Please Choose input file"
CommonDialog1.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
CommonDialog1.FilterIndex = 2
CommonDialog1.ShowOpen
filepatha$ = CommonDialog1.FileName

Open filepatha$ For Input As #1
Input #1, a$, b$, c$, d$, e$, f$, g$ '***dummy variables for first line to remove the header containing name of variables
'Print a$, b$, c$, d$, e$, f$, g$

For i = 1 To n 'input of data
Input #1, time(i), t1(i), f1(i), f2(i), t3s(i), t3m(i), t3l(i)
Next i

Close #1
Call classify(n, t1(), f1(), f2(), t3s(), t3m(), t3l())
progress = progress + 1
ProgressBar1.Value = progress
```

```
Call linguipersist(n, mua())
progress = progress + 1
ProgressBar1.Value = progress

Call TATC(n, mua(), pa())

Call Exp
progress = progress + 1
ProgressBar1.Value = progress

Call historical_info
progress = progress + 1
ProgressBar1.Value = progress

Command1.Enabled = False

Text1.Text = "DONE!"
ProgressBar1.Value = 740

SelMode.start.Enabled = False

End Sub

Private Sub Command2_Click()

   Load DispExpect
   DispExpect.Show
   Command2.Enabled = False

End Sub
Sub historical_info()

On Error GoTo errorhandler

ReDim rno(1 To 729)
ReDim pts(1 To 729, 1 To 10, 1 To 5)
ReDim hitno(1 To 729, 1 To 5)
ReDim sum(1 To 729, 1 To 10)
ReDim sum1(1 To 729)
ReDim totsum(1 To 729)
ReDim cdf(1 To 729, 1 To 10)
ReDim tcmiddle(1 To 10)
ReDim tclower(1 To 10)
ReDim tcupper(1 To 10)
ReDim lowlimit(1 To 729)
```

```
ReDim uplimit(1 To 729)
ReDim mean(1 To 729)
ReDim weights(1 To 729)

filepath53$ = filepath$ + "numpoints.csv"
Open filepath53$ For Input As #103

Input #103, title1$
Input #103, title1$, title2$, title3$, title4$, title5$, title6$
For r = 1 To 729
   For i = 1 To 5
     Input #103, Dummy10$, Dummy11$, Dummy12$, Dummy13$, Dummy14$,
Dummy15$, hitno(r, i), Dummy16$
   Next i
   For i = 1 To 10
      Input #103, rno(r), pts(r, i, 1), pts(r, i, 2), pts(r, i, 3), pts(r, i, 4), pts(r, i, 5),
Dummy17$, Dummy18$
  Next i
Next r

Close #103

'Adding all the points in each consequent zone.(row-wise)
For r = 1 To 729
For i = 1 To 10
   For j = 1 To 5
      sum(r, i) = sum(r, i) + pts(r, i, j)
   Next j
Next i
Next r
'Calculation of center of each bin
For i = 1 To 10
tclower(i) = (i - 1) * 0.1
tcupper(i) = i * 0.1
tcmiddle(i) = tclower(i) + 0.05
Next i

For r = 1 To 729
totsum(r) = 0
Next r

For r = 1 To 729
For i = 1 To 10
   totsum(r) = sum(r, i) + totsum(r)
Next i
Next r
```

```
filepath106$ = filepath$ + "sums.csv"
Open filepath106$ For Output As #56

For r = 1 To 729
   For i = 1 To 10
      ' Cumulative Distributive Function for each zone(CDF)to be used during
intrapolation of (95%) confidence limits
      If totsum(r) <> 0 Then
         sum1(r) = sum1(r) + sum(r, i)
         cdf(r, i) = sum1(r) / totsum(r)
      End If
      Print #56, i; ","; sum1(r); ","; cdf(r, i)
   Next i
Next r

'Calculating weighted average
For r = 1 To 729
For i = 1 To 10
   weights(r) = sum(r, i) * tcmiddle(i) + weights(r)
Next i
If totsum(r) <> 0 Then
mean(r) = weights(r) / totsum(r)
Else
mean(r) = 0
End If
Next r

'Calculation of confidence limits by intrapolation
For r = 1 To 729
  For i = 1 To 9
      If mean(r) <> 0 Then

         'right on target
         If cdf(r, i) < 0.025 And cdf(r, i + 1) > 0.975 Then
            lowlimit(r) = tcupper(i) + (tcupper(i + 1) - tcupper(i)) * (0.025 - cdf(r, i)) /
(cdf(r, i + 1) - cdf(r, i))
            uplimit(r) = tcupper(i) + (tcupper(i + 1) - tcupper(i)) * (0.975 - cdf(r, i)) /
(cdf(r, i + 1) - cdf(r, i))
            Exit For
         End If

      'lower confidence limit
         If cdf(r, i) < 0.025 And cdf(r, i + 1) > 0.025 Then
            lowlimit(r) = tcupper(i) + (tcupper(i + 1) - tcupper(i)) * (0.025 - cdf(r, i)) /
(cdf(r, i + 1) - cdf(r, i))
```

```
            GoTo up
         End If
up:
      'upper confidence limit
         If cdf(r, i) < 0.975 And cdf(r, i + 1) > 0.975 Then
            uplimit(r) = tcupper(i) + (tcupper(i + 1) - tcupper(i)) * (0.975 - cdf(r, i)) /
(cdf(r, i + 1) - cdf(r, i))
            Exit For
         End If
      End If
   Next i
Next r

filepath104$ = filepath$ + "expectation.csv"
filepath105$ = filepath$ + "Freshmetrics.csv"

Open filepath104$ For Output As #54
Open filepath105$ For Output As #55

For i = 1 To 729
   Print #54, rno(i); ","; sum(i, 1); ","; sum(i, 2); ","; sum(i, 3); ","; sum(i, 4); ","; sum(i,
5) _
   ; ","; sum(i, 6); ","; sum(i, 7); ","; sum(i, 8); ","; sum(i, 9); ","; sum(i, 10)

   Print #55, rno(i); ","; mean(i); ","; lowlimit(i); ","; uplimit(i)
Next i

Close #54
Close #55
Close #56

'End of weighted mean calculations
Exit Sub

errorhandler:
If Err.number = 75 Or Err.number = 70 Then
   Resume Next
End If
Err.Raise Err

End Sub
```

The VBA code for the form *DispExpect.frm* is listed below.

```
'Program to display the 'expectations' for all rules in form of histograms.
```

```
'It is used in the prediction mode.

Public clickDisp1 As Integer
Dim xlApp As Excel.Application
Dim xlbook As Excel.Workbook
Dim xlsheet As Excel.Worksheet
Dim rul$()
Dim psum11() As Double
Dim runo() As Single
Dim plowcon(1 To 729) As Single
Dim pupcon(1 To 729) As Single
Dim pexp(1 To 729) As Single
Dim mean(1 To 729) As Single
Dim number() As Single
Dim good(1 To 729) As Single
Dim bad(1 To 729) As Single
Dim selected As Integer

Public filepath212$
Public path1$
Public filepath210$
Public rownum1 As Integer

Public Sub initializer()
Set xlApp = New Excel.Application

End Sub

Private Sub Display1_Click()
On Error GoTo errorhandler
clickDisp1 = clickDisp1 + 1

If clickDisp1 = 1 Then
   Call initializer
   xlApp.Application.DisplayAlerts = False

'ElseIf clickDisp1 > 1 Then
'   xlApp.Workbooks.Close
End If
Set xlbook = xlApp.Workbooks.Open(filepath210$)
Set xlsheet = xlbook.ActiveSheet

With xlApp

xlsheet.Range(xlsheet.Cells(rownum1 + 2, 2), xlsheet.Cells(rownum1 + 2, 11)).Select
```

```
    Charts.Add
        ActiveChart.ChartType = xlBarClustered
        ActiveChart.SetSourceData Source:=xlsheet.Range(xlsheet.Cells(rownum1 + 2, 2),
xlsheet.Cells(rownum1 + 2, 11)), PlotBy:= _
        xlRows
        ActiveChart.PlotArea.Select
        Selection.ClearFormats
        ActiveChart.PlotArea.Select
    With Selection.Border
        .Weight = xlThin
        .LineStyle = xlAutomatic
    End With
    Selection.Interior.ColorIndex = xlNone
    'ActiveChart.Location Where:=xlLocationAsObject, Name:="sheet1"
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = "Expectation Histogram"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Tc"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Expectation"
    End With
    With ActiveChart.Axes(xlCategory)
        .HasMajorGridlines = False
        .HasMinorGridlines = False
    End With
    With ActiveChart.Axes(xlValue)
        .HasMajorGridlines = False
        .HasMinorGridlines = False
    End With

    ActiveChart.HasLegend = False
    ActiveChart.SeriesCollection(1).Select
    With ActiveChart.ChartGroups(1)
        .Overlap = 0
        .GapWidth = 0
        .HasSeriesLines = False
        .VaryByCategories = True
    End With
End With
Set Chart1 = xlbook.ActiveChart
OLE2.CreateLink filepath210$
OLE2.SizeMode = 1

Text2.Text = "NEW DATA" + vbCrLf + "**********" + vbCrLf + "Weighted mean
average" + vbCrLf + "TcMean =" & Str(pexp(rownum1)) _
```

```
+ vbCrLf + "Lower confidence(95%) limit" + vbCrLf + "TcLow =" &
Str(plowcon(rownum1)) _
+ vbCrLf + "Upper confidence(95%) limit" + vbCrLf + "TcHi =" &
Str(pupcon(rownum1)) _
+ vbCrLf + "HISTORICAL DATA" + vbCrLf + "**********" + vbCrLf + "Weighted
mean average" + vbCrLf + "TcMean =" & Str(mean(rownum1))

For i = 1 To selected
   If val(List1.Text) = number(i) Then
   Text3.Text = "GOOD RULE!"
   Exit For
   End If
Next i
If good(rownum1) = 0 And bad(rownum1) = 0 Then
Text3.Text = "Not Expressed Sufficiently In Data"
ElseIf (good(rownum1) = 1 And bad(rownum1) = 0) Then
Text3.Text = "Insufficient Corroboration"
ElseIf good(rownum1) <= bad(rownum1) Or (good(rownum1) = 0 And bad(rownum1) =
1) Then
Text3.Text = "BAD RULE"
End If

Exit Sub
errorhandler:
   Err.Raise Err
   Resume
End Sub

Private Sub Form_Load()
On Error GoTo errorhandler

ReDim psum11(1 To 729, 1 To 10)
ReDim runo(1 To 729)
ReDim rul$(1 To 729)

clickDisp1 = 0

CommonDialog2.DialogTitle = "Where is the rulesheet (output) file located?"
CommonDialog2.Filter = "All Files (*.*)|*.*|Comma Delimited Input (*.csv)|*.csv"
CommonDialog2.FilterIndex = 2
CommonDialog2.ShowOpen
filepath212$ = CommonDialog2.FileName
path1$ = Left(filepath212$, (Len(filepath212$) - Len("rulesheet.csv")))
Open filepath212$ For Input As #212
```

```
filepath210$ = path1$ + "PredNormalizedExpectation.csv"
Open filepath210$ For Input As #210

filepath204$ = path$ + "trips.csv"
Open filepath204$ For Input As #204
Input #204, title1$, title2$, title3$, blah$, title4$

filepath205$ = path1$ + "PredFreshMetrics.csv"
Open filepath205$ For Input As #205

filepath305$ = path1$ + "FreshMetrics.csv"
Open filepath305$ For Input As #305

filepath206$ = path$ + "finalopti.csv"
Open filepath206$ For Input As #206

Input #210, title1$
Input #210, title1$, title2$, title3$, title4$, title5$, title6$, title7$, title8$, title9$, title10$
Input #205, title1$, title2$, title3$, title4$
For i = 1 To 729
   Input #212, Dummy1$, runo(i), rul$(i)
   Input #210, Dummy2$, psum11(i, 1), psum11(i, 2), psum11(i, 3), psum11(i, 4),
psum11(i, 5), psum11(i, 6), psum11(i, 7), psum11(i, 8), psum11(i, 9), psum11(i, 10)
    Input #205, Dummy8$, pexp(i), plowcon(i), pupcon(i)
    Input #305, Dummy10$, mean(i), Dummy11$, Dummy12$
    Input #204, Dummy4$, good(i), bad(i), Dummy5$, Dummy6$
Next i

Input #206, dummy30$, dummy31$, dummy32$, dummy33$
Input #206, dummy35$, dummy36$, dummy37$, dummy38$, selected
Close #206

ReDim number(1 To selected)

filepath206$ = path$ + "finalopti.csv"
Open filepath206$ For Input As #206
Input #206, dummy30$, dummy31$, dummy32$, dummy33$
For i = 1 To selected
Input #206, number(i), dummy36$, dummy37$, dummy38$, dummy39$
Next i
Close #206

For i = 1 To 729
   List1.AddItem (i)
Next i
```

```
Close #204
Close #212
Close #210
Close #205
Close #305

Text1.Text = ""
Text2.Text = ""

Exit Sub

errorhandler:
If Err.number = 75 Or Err.number = 70 Then
   Resume Next
Else: Err.Raise Err

End If

End Sub
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
   Call Form_Unload(0)
End Sub
Private Sub Form_Unload(Cancel As Integer)

If clickDisp1 >= 1 Then
Set xlsheet = Nothing
Set xlbook = Nothing
xlApp.quit
Set xlApp = Nothing
OLE2.Close
clickDisp1 = 0
End If

DispExpect.Hide
Set DispExpect = Nothing

End Sub

Private Sub List1_Click()
rownum1 = List1.ListIndex + 1
Text1.Text = rul$(rownum1)

End Sub

Private Sub quit_Click()
OLE2.Close
```

```
Call Form_Unload(0)
End Sub
```

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
```

APPENDIX B1 FINAL RULE DATA-BASE (WITHOUT-NOISE)

The following rules were selected for the case of no noise, and with a threshold of 5 data points.

Rule no.	**Statement**
97	IF Temp1 is LOW & F1 is MED & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED
115	IF Temp1 is MED & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED
121	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
122	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED
124	IF Temp1 is MED & F1 is MED & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED
127	IF Temp1 is MED & F1 is HIGH & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be MED
130	IF Temp1 is MED & F1 is HIGH & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
138	IF Temp1 is HIGH & F1 is LOW & F2 is LOW & Persistence is HIGH THEN after SHORT delay Temp3 will be MED
148	IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED

APPENDIX B2 FINAL RULE DATA-BASE (WITH NOISE)

The following rules were selected for the case of with noise, and using a threshold of 7 data points.

Rule no.	Statement
7	IF Temp1 is LOW & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be LOW
10	IF Temp1 is LOW & F1 is MED & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be LOW
19	IF Temp1 is LOW & F1 is HIGH & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be LOW
94	IF Temp1 is LOW & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
95	IF Temp1 is LOW & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED
110	IF Temp1 is MED & F1 is LOW & F2 is LOW & Persistence is MED THEN after SHORT delay Temp3 will be MED
112	IF Temp1 is MED & F1 is LOW & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
113	IF Temp1 is MED & F1 is LOW & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED
115	IF Temp1 is MED & F1 is LOW & F2 is HIGH & Persistence is LOW THEN after SHORT delay Temp3 will be MED
118	IF Temp1 is MED & F1 is MED & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be MED
119	IF Temp1 is MED & F1 is MED & F2 is LOW & Persistence is MED THEN after SHORT delay Temp3 will be MED
121	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
122	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED
123	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is HIGH THEN after SHORT delay Temp3 will be MED
127	IF Temp1 is MED & F1 is HIGH & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be MED
128	IF Temp1 is MED & F1 is HIGH & F2 is LOW & Persistence is MED THEN after SHORT delay Temp3 will be MED
129	IF Temp1 is MED & F1 is HIGH & F2 is LOW & Persistence is HIGH THEN after SHORT delay Temp3 will be MED

	Statement
139	IF Temp1 is HIGH & F1 is LOW & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
148	IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is LOW THEN after SHORT delay Temp3 will be MED
149	IF Temp1 is HIGH & F1 is MED & F2 is MED & Persistence is MED THEN after SHORT delay Temp3 will be MED
226	IF Temp1 is HIGH & F1 is MED & F2 is LOW & Persistence is LOW THEN after SHORT delay Temp3 will be HIGH
607	IF Temp1 is MED & F1 is MED & F2 is MED & Persistence is LOW THEN after LONG delay Temp3 will be MED

VITA

Preetica Kumar

Candidate for the Degree of

Master of Science

Thesis: IMPROVED QUALITY METRICS FOR LINGUISTIC RULE SELECTION

Major Field: Chemical Engineering

Biographical:

Personal Data: Born in Bhopal, M.P, India, on July 31st, 1982, the daughter of Kumar and Geetha Ranganathan.

Education: Received Bachelor of Science degree in Chemical Engineering from Osmania University College of Technology, Hyderabad, A.P, India in May 2003. Completed the Requirements for the Master of Science degree with a major in Chemical Engineering at Oklahoma State University in July, 2005.

Experience: Conducted research at the Indian Institute of Chemical Technology, Hyderabad, A.P, India; employed by Oklahoma State University, Department of Chemical Engineering as a graduate research assistant; Oklahoma State University, Department of Chemical Engineering, 2004 to present; employed by Oklahoma State University, Department of Chemical Engineering as a teaching assistant; Oklahoma State University, Department of Chemical Engineering, 2003 to present.

Professional Memberships: Phi Kappa Phi Honor Society; Omega Chi Epsilon Honor Society; Instrumentation, Systems and Automation Society.

Name: Preetica Kumar Date of Degree: May, 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: IMPROVED QUALITY METRICS FOR LINGUISTIC RULE SELECTION

Pages in Study: 203 Candidate for the Degree of Master of Science

Major Field: Chemical Engineering

Scope and Method of Study: The objective of this study was to explore improved robust metrics for the selection and evaluation of linguistic cause and effect rules and to predict future outcomes to some extent. A hot and cold water mixer simulator was used to generate the data which was then processed to incorporate temporal information about the process. After the data was fuzzified, an exhaustive search produced the initial rule database. A Truth Space Diagram (TSD) was then generated for each rule. The first metric "Merit" was calculated based on the number of "trips" made by each rule into Quadrants II, IV of the TSD. This metric was used to select the best rules, along with a minimum corroboration criterion. The second metric "Expectation" was determined from historical data to predict the expected truth of consequent given the truth of antecedent, for new data. Histograms were used to denote the absolute and normalized expectations.

Findings and Conclusions: The concept of using "trips" into the quadrants helped isolate independent events in the data and corroboration. Thus rules could be termed good or bad with more certainty. The metric "Merit" was very simple to calculate, did not require normalization, and was independent of the total number of points in the data-set. The two selection criteria proposed were found to select good and only good rules. The metric "Expectation" was used to determine the certainty of a selected rule's prediction, with a confidence limit of 95%. Adding noise to the data did not affect the efficiency of the selection procedure.
Thus, the final rule database consisted only of the best rules. Truth of the consequent was successfully predicted based on historical data.

ADVISOR'S APPROVAL: Dr. R. Russell Rhinehart