5-2004

Fault Detection and Isolation Expert System and Kernel Smoothing Techniques to Monitor the Continuous Automated Vault Inventory System (CAVIS)

Joseph Michael Bowling
University of Tennessee, Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Joseph Michael Bowling entitled "Fault Detection and Isolation Expert System and Kernel Smoothing Techniques to Monitor the Continuous Automated Vault Inventory System (CAVIS)." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

J. Wesley Hines, Major Professor

We have read this thesis and recommend its acceptance:

Lawrence Townsend, Laurence Miller

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Joseph Michael Bowling entitled "Fault Detection and Isolation Expert System and Kernel Smoothing Techniques to Monitor the Continuous Automated Vault Inventory System (CAVIS)." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

J Wesley Hines

J. Wesley Hines, Major Professor

We have read this thesis
and recommend its acceptance:

[illegible]

Lawrence Townsend

Laurence L Miller

Laurence Miller

Accepted for the Council:

[illegible]

Vice Chancellor and Dean of Graduate Studies

Thesis
2004
.B69

Fault Detection and Isolation Expert System and Kernel Smoothing Techniques to Monitor the Continuous Automated Vault Inventory System (CAVIS)

A Thesis

Presented for the Master of Science Degree

The University of Tennessee, Knoxville

Joseph Michael Bowling

May 2004

Dedication

This thesis is dedicated to my parents, Michael and Belinda Bowling, for their guidance, my family Justin, Brian, Bethany, and Grandparents for their support, and Sara for the encouragement to never settle and accomplish the desired end.

Acknowledgements

During the course of my work at the University of Tennessee, I have received assistance from several individuals who enabled me to complete this research and the requirements for my Master of Science degree in Nuclear Engineering. Thanks to my Thesis Committee, Dr. Wesley Hines, Dr. Larry Miller, and Dr. Lawrence Townsend for without their help and direction I never could have achieved this end.

Special thanks to my major professor Dr. Hines, for without his guidance I would have never completed this work.

I would like to thank Chris Pickett and Rick Oberer of the Y-12 National Security Complex, for overseeing the research project.

I would also like to thank Dr. Robert Uhrig for the advice and expertise he provided me throughout the research.

I would like to thank my fellow researcher on the project T Jay Harrison for all of the hard work and assistance he provided me throughout the project.

I would like to thank my fellow graduate students Ravi, Jared, Aaron, Ke, Ted, and Martin whose companionship spurred creative thought while working on this project.

Finally, I would like to thank the Y-12 National Security Complex for providing me funding so that I could carry out this study. This project was funded by BWXT Y-12 under contract document 4300017131.

Abstract

The Continuous Automated Vault Inventory System (CAVIS^{TM}) is a system designed to continually monitor the status of special nuclear materials (SNM) at the Oak Ridge based Y-12 facility. CAVIS consists of an integrated package of low-cost sensors used to continuously monitor weight and radiation attributes of the stored items. The CAVIS system detects "changes-in-state" of the special nuclear material and generates an appropriate alarm. Unfortunately, the CAVIS system is susceptible to false alarms that do not coincide with the removal of special nuclear material. These false alarms may be due to the random stochastic nature of the measurements, due to failing components, or due to external sources in the vicinity of the facility. The response to a false alarm may be an inventory check, which entails the physical verification of the attributes of the SNM. Thus, it is desirable to limit this costly response.

This thesis presents the development of a monitoring system for CAVIS to eliminate the costly responses caused by false alarms. The system merges advanced statistical algorithms, such as the sequential probability ratio test (SPRT), to extract features related to changes in the CAVIS sensors with an expert system that forms a hypothesis on the root cause of any anomaly. In addition, kernel-averaging techniques have been developed as a regional anomaly-monitoring module. This thesis presents the development of the expert system and the kernel-averaging techniques featured in the fault detection and isolation system. The implementation of these techniques will enable the monitoring of the CAVIS system and develop alternative hypothesis of the root cause of spurious CAVIS alarms. These alternative hypotheses can be investigated prior to any inventory check, thus reducing cost and lessening radiation exposures.

Table of Contents

List of Tables

List of Figures

1 Introduction

This chapter contains an introduction to the CAVIS monitoring system, a Fault Detection and Isolation (FDI) system for the CAVIS system at the Y-12 National Security complex. The introduction includes a brief description of CAVIS, the faults CAVIS is subject too, and the proposed FDI system to detect and isolate CAVIS faults. The project has two major contributions: 1) the extraction of information rich features from CAVIS data and 2) the mapping of those features to faults hypothesis through an expert system. This thesis focuses on the second contribution. Information concerning the CAVIS monitoring system feature extraction can be found in "The Sequential Probability Ratio Test (SPRT) in Feature Extraction and Expert Systems in Nuclear Material Management" [Harrison 04].

1.1 Objectives of the Present Study

The objective of the present study is to develop a Fault Detection and Identification (FDI) system to monitor the Continuous Automated Vault Inventory System (CAVIS^{TM}). The Y-12 National Security Complex in Oak Ridge Tennessee currently houses the nations supply of weapons grade uranium. CAVIS is a security system in place at the Y-12 National Security Complex that monitors this special nuclear material (SNM) to ensure its safe and secure storage. CAVIS is subject to several types of failures making it necessary to have a FDI system to monitor the CAVIS system as a whole.

CAVIS is a system designed to continually monitor the status of SNM. CAVIS is an integrated package of low-cost sensors that monitor the weight and radiation attributes of SNM. The system functions by monitoring the SNM for "changes-in-state" and generates an appropriate alarm in the event of a change.

CAVIS generated alarms may result in an inventory check of the SNM. An inventory check consists of measurements of the physical characteristics of the SNM to ensure none of the material has been removed. Inventory checks can be costly, time consuming and may expose workers to radiation. Thus, it is desirable to perform as few inventory checks as possible. To limit the number of inventory checks it is necessary to ensure that CAVIS will only alarm when tampering with the SNM has occurred. However, CAVIS is prone to generate alarms that do not coincide with the removal of SNM. Thus, a system that provides some validity to CAVIS generated alarms is desirable. The proposed CAVIS monitoring system will provide root cause analysis by monitoring the CAVIS system to ensure that generated alarms are not a result of some type of CAVIS failure. Henceforth, CAVIS alarms that coincide with the removal of SNM will be referred to as true alarms. All other CAVIS generated alarms, or alarms that do not coincide with the removal of SNM, will be referred to as false alarms.

This thesis presents the development of a monitoring system for CAVIS, which eliminates the costly responses caused by false alarms. The system merges advanced statistical algorithms, such as the sequential probability ratio test (SPRT), to extract features related to changes in the CAVIS sensors with an expert system that forms a hypothesis on the root cause of any anomaly. In addition, kernel-averaging techniques have been developed as a regional anomaly-monitoring module.

1.2 Problem Statement

The CAVIS system is suspect to false alarms, which may not coincide with the removal of special nuclear material. Several factors can result in false radiation sensors alarms. First, the statistical nature of radioactive decay and counting may cause the count rate to fall outside of the commonly used 95% confidence intervals. In such an instance, the state of the SNM has not changed and the CAVIS system incorrectly alarms. Secondly, the CAVIS system is comprised of numerous components that may fail over time. Thus, the CAVIS system may generate alarms due to component failures, which are not correlated with changes in the SNM. Thirdly, the storage area is a functioning warehouse that may have radioactive material being moved. These external radiation sources may be detected by the CAVIS system causing an alarm in the region of the warehouse where the external radiation source is located. Fourthly, the radiation sensors used in the CAVIS system display a spike behavior when they are impacted. Forklifts and other heavy equipment moving in the warehouse may cause impacts that are transmitted to the CAVIS storage vaults inducing spikes in the count rates. Finally, the CAVIS system has displayed a dependence on environmental stimuli such as heat and humidity. Thus, the environmental conditions of the storage area may cause the CAVIS system to generate false alarms. These numerous conditions can all result in CAVIS false alarms that may result in unnecessary and costly inventory checks.

1.3 Overview of the Methodology

The CAVIS monitoring system is a proposed system that will monitor the CAVIS system. In effect, the CAVIS monitoring system is a system that monitors a monitoring system. The system extracts features related to changes in CAVIS using the sequential probability ratio test and several other modules. These extract features are then analyzed by an expert system that forms a hypothesis on the root cause of any anomaly. These alternative hypotheses can be investigated prior to an inventory check to avert unwarranted inventory checks. A Regional Anomaly Monitoring Module (RAMM) was also developed to detect external stimuli that may induce alarms in CAVIS. The RAMM utilizes kernel-averaging techniques to detect and isolate root causes of abnormality that affect regions of the CAVIS system.

The SPRT is a statistical test developed by A. Wald in 1945 that is capable of monitoring statistical properties of a Gaussian distribution [Wald 1945]. The SPRT determines if an input data stream was generated by the expected, normal Gaussian distribution characterized by an expected mean and variance, or if there is a greater probability that the data stream comes from some faulted distribution characterized by a shifted mean and or altered variance. If the input comes from the faulted distribution, the SPRT will generate an appropriate alarm. This technique is capable of optimally monitoring two attributes of the radiation distribution: mean and variance, in contrast to previous techniques used in CAVIS [Bell], which only monitored the mean. By monitoring two attributes of the radiation distribution, the SPRT based system will be capable of identifying additional operational faults.

An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that may require significant human expertise [Feigenbaum 1982]. An expert system is comprised of a rule base, a knowledge base, and an inference engine [Waterman 1986]. Knowledge or facts cause rules to "fire" which in turn cause additional facts to be hypothesized. The inference engine controls program execution. When presented information about the state of a system, the expert system is capable of emulating the diagnostic actions of an expert if the system has been programmed with the correct knowledge. Expert systems have been used for fault detection and isolation in several industries including nuclear power [Bhatnagar 1990, Khartabil 1991, Miller 1994].

Kernel smoothing is a non-parametric technique used to estimate the probability density function of a data set [Wand 1995]. In this application the data are the radiation detector count rates observed at different locations in the storage facility. Kernels are used to smooth the discrete measurements resulting in an approximation of the underlying radiation field. The resulting distribution can be thought of as a weighted average of the data with the weights defined by the kernel function. Kernel smoothing is implemented to detect and identify regional radiation disturbances. Kernel smoothing is used to map the behavior of the sensors in close proximity to one another to form a neighborhood score. Certain known anomalies may affect a region of the storage facility and would induce an increase in the neighborhood scores for that region. The maximum of the neighborhood scores will occur close to the regional anomaly, enabling the location of the anomaly to be determined and its cause to be investigated.

1.4 Organization of the Thesis

The first chapter of this thesis presents the objectives of the study, the problem statement, and an overview of the methodology describing the proposed FDI system. Chapter 2 contains a literature survey of an application of expert systems in fault detection and isolation in nuclear systems and an application of kernel smoothing in fault detection and isolation. Chapter 2 also contains a description of expert systems including the definition, theory, and a discussion of their deficiencies and a discussion of the method of kernel smoothing. Chapter 3 contains a description of the CAVIS system. This chapter describes the Y-12 National Security complex, the Highly Enriched Uranium Material Facility, the various components of CAVIS, and the known possible faults CAVIS can experience including a discussion of several environmental experiments performed on CAVIS. Chapter 4 discusses the methodology of the study, including the integration of the feature extraction module with the expert system and RAMM to form the FDI system. Chapter 5 presents the results of the CAVIS monitoring system including a discussion of the performance of the FDI system and the RAMM on fabricated data and data collected by the CAVIS system. Conclusions on the study, contributions of the study, and recommendations for the future work on the system are given in chapter 6.

The CAVIS system at the Y-12 National Security complex is a security system that monitors the status of SNM by detecting "changes in state" of the SNM. The system is susceptible to false alarms, the response to which can be costly and time consuming. The proposed CAVIS monitoring system is capable of distinguishing between true alarms and the false alarms. The following chapters of this thesis describe the CAVIS monitoring system.

2 Literature Survey of Expert Systems and Kernel Smoothing

This chapter contains a literature survey of expert systems and kernel smoothing that includes a discussion of the theory and method of expert systems and kernel smoothing. Applications of expert systems and kernel smoothing in fault detection and isolation systems are also discussed.

2.1 Expert Systems

Expert systems are one of the most successful and widespread divisions of artificial intelligence or the study of computer programs that exhibit intelligent behavior. An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that may require significant human expertise [Feigenbaum 1982]. In general it is necessary to optimize expert systems to solve specific problems, as attempts to build general problem solving expert systems have not had much success [Giarratano 1994, Waterman 1986]. If optimized with the correct system knowledge, an expert system is capable of emulating the actions of an expert system operator when presented with information concerning the state of that system.

An expert system is comprised of a rule base, a knowledge base, and an inference engine [Tsoukalas 1997]. The knowledge base contains facts or information about the state of the system. The rule base contains the knowledge domain of the expert, or a collection of rules concerning the system that an expert would use to diagnosis the system. A popular method of representing the knowledge domain of an expert in the rule base is in the form of IF/THEN rules. Knowledge or facts contained in the knowledge base cause rules in the rule base to "fire" which in turn causes additional facts to be hypothesized which are further stored in the knowledge base. The continual “firing” of rules will allow the expert system to build an understanding of the systems problem until it is able to correctly diagnose the problem. The inference engine controls program execution by determining which rules are satisfied by the facts in the knowledge base, prioritizes the satisfied rules and executes the rules with the highest priority. This enables the inference engine to diagnose the problem affecting the system. Several introductory texts on expert systems are available. The presentation given here is similar to Giarratano [Giarratano 1986] and Waterman [Waterman 1986].

Expert systems can also contain Explanation and Knowledge Acquisition modules [Giarratano 1986]. The Explanation module explains the reasoning the Expert system used to arrive at its decision. The Knowledge Acquisition module enables a user to add additional rules to the rule base of the expert system without having to explicitly code the rules. This is useful as additional information concerning the expert system problem may become available, or the knowledge engineering may have not covered all possible scenarios. A Graphical User Interface allows the Expert system to communicate with the user [Giarratano 1986]. Figure 2.1 illustrates the various components and the process of data flow in a typical Expert system.

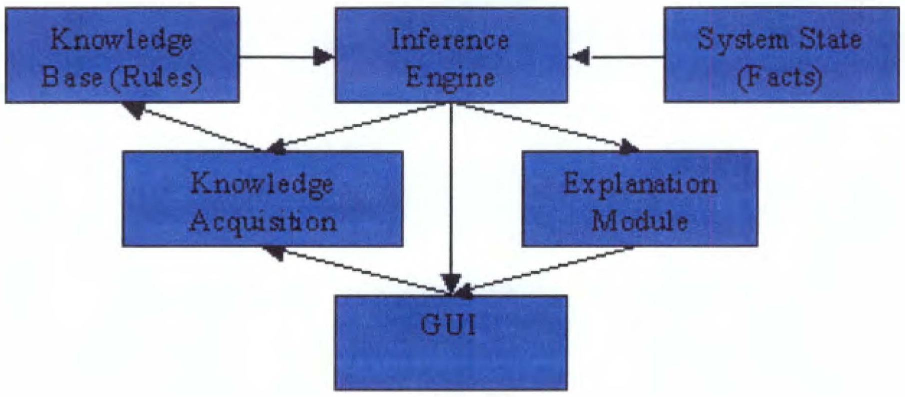


Figure 2.1. Structure of Rule-Based Expert System

2.1.1 Knowledge Engineering and Representation

In order to emulate the actions of an expert, an expert system must be programmed with significant knowledge of the problem it is to solve, and the course of actions an expert would take when presented with the problem. The gathering and programming of this knowledge is known as knowledge engineering. The knowledge engineer extracts strategies and rules of thumb for problem solving from the human expert and uses this knowledge to build the expert system [Waterman 1986]. For the expert system to be successful, it is necessary for the knowledge domain (the domain of information extract from experts) to span the entire problem domain (specific to the current problem area). An expert system containing knowledge of medicine would be ineffective at solving problem in engineering, and vice versa. Thus, the proper knowledge must be programmed into the knowledge domain of the expert system. In contrast, it is necessary to only program knowledge into the knowledge domain that is necessary to solve the specific problem. As previously mentioned, early attempts to produce systems capable of solving large classes of problems, or general problem solvers, were unsuccessful [Giarratano 1994, Waterman 1986]. As the number of classes of problems a system could solve increased, the poorer the system preformed on each individual problem [Waterman 1986]. This is because the number of rules involved in the knowledge domain became very large and it became increasing difficult for the system to distinguish between the rules and solve the problem. Thus, it is necessary for an expert system to be problem specific. The later development of Fuzzy Logic by Zadeh [Zadeh 1965] extended the applicability of expert system rule-base by allowing greater flexibility in the formation of the rules [Tsoukalas 1997]; however it is still necessary for an expert system to be problem specific.

Expert system development spawned from several disciplines of artificial intelligence. One such area is known as cognitive science and focuses on the method that human's process information or how humans think. This is important to expert systems because they attempt to emulate the actions of a human expert. Newell and Simon were able to demonstrate that human cognition can be expressed by IF/THEN rules [Newell 1972]. A simple example of human knowledge being expressed in IF/THEN rules is IF the car will not start THEN check the gas. Of course there are thousands of reasons why a car may not start but without any other operatory knowledge of the car the first reaction to a car not starting is to ensure that the vehicle has adequate fuel. Additional combinations of IF/THEN rules can be used to diagnose the more complicated failures that can occur in an automobile. Depending on the complexity of the problem the number of IF/THEN rules featured in an expert system can be hundreds or thousands. As previously mentioned, the portion of the expert system that contains the knowledge concerning the problem is known as the knowledge base. The knowledge base of an expert system contains both factual and heuristic knowledge, which can be expressed in the form of IF/THEN rules. Factual knowledge is the widely excepted and understood knowledge concerning the problem domain and contains the underlying principals of the problem domain [Engelmore 1993]. Heuristic knowledge is the knowledge of the instinct or gut feeling an expert in the field of the problem domain [Engelmore 1993]. This includes the judgments an expert in the problem domain would use to make a decision when presented with various situations in the problem domain. Heuristic knowledge contains the experiences an expert in the field has had while working in the field.

Expert systems responsible for diagnosing large complex systems may feature hundreds or thousands of rules. In these large systems it is possible for several rules in the rule base to be satisfied by the conditions in the knowledge base. Thus, it is necessary for expert system to be able to determine which rule should be executed by the current state of the system and in some cases, choose between multiple rules that could be executed. As previously mentioned, the inference engine determines which rules are satisfied by the knowledge base and determines the order of execution of the rules by prioritizing the rules in order to form an inference or make a diagnosis about the state of a system. Returning to the automobile example, suppose the car will not start and the engine will not turn over. An expert system attempting to diagnosis this problem may have a rule base featuring two rules that state 1) IF the car will not start THEN check the gas, and 2) IF the car will not start AND the engine does not turn over THEN check the battery. The antecedent for both rules is satisfied however, if the engine does not turn over the failure is probably not an empty gas tank. In this instance, the Expert systems inference engine would prioritize the satisfied rules and suggest checking the cars battery first.

The inference engine is able to determine which rules in the rule base to execute using the problem-solving paradigm to organize the problem and determine the order the problem should be solved [Giarratano 1994]. This often involves chaining the IF/THEN rules to form an order or line of reasoning. Problems can be forward or backwards chained depending on the desired direction of the problem solving [Giarratano 1994]. Forward chaining begins with a set of conditions and moves towards some conclusion using the IF/THEN rules contained in the knowledge base. Backwards chaining begins with the desired conclusion and reasons backwards to the conditions necessary for that conclusion to occur. By programming the rule base knowledge in the structure of multiple IF/THEN rules and following a line of reason to the desired goal, expert systems are able to infer the action an expert would take when presented with the necessary information concerning the state of the system.

2.1.2 Expert System Uncertainty

When experts solve problems, they have an idea of how confident they are in their solution. The uncertainty that experts have in their solution can be due to inadequate information concerning the problem and uncertainty in the accuracy of the rules and principals they used in determining their solution. Calculating an uncertainty for the conclusion of an expert system can be a difficult task. This is in part because it is difficult to assign an uncertainty to the IF/THEN rules contained in the knowledge base. One method of assigning uncertainty to an expert system is to use fuzzy logic and make the expert system a fuzzy system [Engelmore 1993]. Fuzzy logic is a problem solving method that uses fuzzy numbers that can be expressed in linguistic terms to address the imprecision of the inputs and output of the system variables [Tsoukalas 1997]. In a fuzzy system, each IF/THEN rule has a degree of fulfillment, or the degree to which the rule is satisfied. An uncertainty in the expert system conclusion can be calculated using the degree of fulfillment of each rule used by the expert system to arrive at the conclusion. Fuzzy logic can also be useful in expert systems with extremely large knowledge bases. In crisp or non-fuzzy systems, when the knowledge base becomes increasingly large, the inference engine can have an increasing difficult time determining which knowledge to use to arrive at the appropriate conclusion. Fuzzy logic can be used to remedy this issue [Tsoukalas 1997].

2.1.3 Expert System Deficiencies

Expert systems have been proven successful at diagnosing specific problems but some deficiencies in their ability still exist. These deficiencies are mainly attributed to the knowledge base of the expert system, or strictly their inability to infer from the knowledge base, and to actually understand the knowledge contained in the knowledge base. Expert systems are incapable of a complete understanding of the knowledge base and this can often cause their diagnosis to suffer [Luger 2002]. For example, MYCIN [Shortliffe 1976] one of the first expert systems was a medical application used to select the antimicrobial therapy for patient and contains no knowledge of physiology. The expert system only contains the knowledge necessary to select the antimicrobial therapy for a patient. MYCIN selects a treatment by analyzing patient records, and lab results and then asks the user a series of question to narrow in on the correct treatment. A myth concerning the system states that MYCIN actually asked if the patient was pregnant even after being told the patient was a male [Luger 2002]. Regardless of whether or not the myth is true, the example illustrates the inability of the expert system to have a profound understanding of the knowledge contained in its knowledge base. This lack of profound understanding results in the expert system having an inability to provide a profound explanation of its arrived decision. The explanation module is really just the expert system asserting which antecedents were satisfied in the knowledge base. Thus, the lack of a profound understanding of its knowledge base leads to several deficiencies in expert systems.

In addition to a lack of profound understanding of its knowledge base, expert systems are incapable of inferring outside of their knowledge base. Thus, expert systems "lack robustness and flexibility" [Luger 2002]. When presented with a problem they cannot solve, expert systems are incapable of going and searching for more knowledge on the subject or reasoning the problem out from deeper principles than define the problem. This inability of an expert system to learn or search out new knowledge is also a deficiency [Luger 2002]. In general, once the expert system is programmed and complete, the knowledge base will not extend itself. Some expert systems are programmed with knowledge acquisition modules; however, this is not the system learning on its own, but rather additional knowledge being programming into the expert system. Finally, and perhaps the most serious deficiency for this particular application is the difficulty in verifying the decision, or providing a confidence in the decision [Luger 2002]. The expert system is only as accurate as its knowledge base, and it is difficult to quantify the accuracy of the knowledge base. Where many techniques may have a 99% confidence in their result, it is difficult to give a confidence in the result of an expert system.

Despite these deficiencies, expert systems have enjoyed great success when applied to a specific problem. Expert systems have numerous applications in geology, medicine, engineering, and military science, among many others. With the adaptation of representing human knowledge as IF/THEN rules, expert systems have evolved from unsuccessful systems that attempted to be general problem solvers, to extremely successful problem specific systems. The following section describes the application of an expert system.

2.2 Expert Systems Applications

AI scientists have attempted to build computer programs capable of exhibiting intelligent behavior since the 1950s. These early attempts focused on building general problem solver that were capable of solving large classes of problems. These attempts were generally unsuccessful, however some significant result did come from this work such as

Newell and Simons demonstration that human cognition could be represented by IF/THEN type production rules. This work paved the way for the rise of knowledge-based systems that were problem specific rather than general problem solvers. Early expert system such as DENDRAL [Linsay 1980], MYCIN [Shortliffe 1976], and PROSPECTOR [Duba 1979] are all examples of problem specific expert systems that were successful in particular fields. In 1965, work began on DENDRAL, a system that identifies the molecular structure of a compound. The system uses knowledge of chemical expertise on mass spectrometry and nuclear magnetic response data to identify the unknown compound from a database of molecular structures of all compounds [Linsay 1980].

MYCIN is an expert system used in medicine to select the antimicrobial therapy for patient with viral infections of bacteremia, meningitis, and cystitis [Shortliffe 1976]. MYCIN functions by first diagnosing the infection using laboratory results, patient history, and the patients symptoms and then selects the appropriate treatment by mimicking treatments previously administered by experience physicians. MYCIN was the first expert system to separate the knowledge base and the inference engine. This enabled the core of MYCIN, known as the shell, to be reused in another expert system with a different knowledge base.

PROSPECTOR is an expert system used in geology to determine the likelihood of mineral and ore deposits in a region based on samples from the region [Duba 1979]. PROSPECTOR functioned by analyzing characteristics of the region such as the geologic setting, structural settings, and the type of rocks and minerals discovered from samples from the region. Using this information PROSPECTOR infers the likelihood of discovering deposits of certain ores by comparing and contrasting the site with previously developed models. PROSPECTOR is a large system containing over 1000 rules. These early problem specific knowledge based systems demonstrated that expert systems could be successfully used to solve real world practical problems and paved the way for the application of expert systems in other areas.

Today expert systems have application in multiple disciplines including chemistry, medicine, and geology as well as agriculture (PLANT/ds), computer systems (XCON), law (EXPERTAX), engineering (REACTOR), and military science (EPES). Expert systems have also been used for fault detection and isolation in several industries including nuclear power. The following review discusses an application of an expert system to a nuclear application.

2.2.1 Operator Advisors

Expert systems have been develop to aid nuclear power plant operators by continually monitoring plant parameters and when an abnormality is detected, diagnoses the plant using a hierarchical classification scheme that are known as operator advisors (OAs). The OAs have been developed for use on simulators of the Perry Nuclear Power Plant, and a simulator of the K reactor at the Savannah River Site as simulators enable testing of hypothetical accident scenarios that can be difficult to test and rarely seen in normal operating conditions [Miller, etc. 1994].

The OAs discussed here feature three modules; data management, monitoring, and diagnosis, with a fourth module, procedure management, still under development. The data management module performs several tasks including reading the data from a serial port, storing the data in a database that must be easily accessible for later use, and providing data to the other module in the OA when it is requested. The data management module data is organized in relational databases, which enables convenient storage and retrieval of data and trends in the data by the other modules. The monitoring module inspects the status of the plant by detecting changes in values of plant data under normal operating condition. Current plant data values are compared to baseline values that were established when the OA was initialized. If some value or values deviate from the baseline value by some threshold amount the monitoring module determines that some abnormal condition exists. If an abnormal condition exists the diagnostic module is initialized. The diagnostic module purpose is to isolate the specific component failure or plant automated action that resulted in the abnormal condition. For every component failure that can occur there exist an expected pattern of plant parameter changes that are characteristic of the component failure. Diagnosis of component failures requires matching the observed abnormal conditions with a known pattern of plant values that are characteristic of some specific component failure.

Nuclear power plants contain thousands of components that are subject to failure, thus it is necessary to reduce the search space to a manageable size in order to isolate the failed component. Organizing the plant into a hierarchy of systems and component reduces the search space. Diagnosis is performed by identifying the system, subsystem, component, and subcomponent that have failed until the specific component failure is isolated. Top systems are evaluated first. If a failure is found, the lower level systems that correspond to the top system are evaluated. If a failure is not found, the lower level systems are not evaluated. This process continues until the faulted component is isolated. Systems unrelated to the failure are not evaluated thus reducing the search space. The OA determines if a system is faulted using; 1) the status of systems which support the system, 2) the status of inputs to the system, 3) the presence of signals which initiate some automatic action by the system, and 4) the system output. This hierarchy diagnosis method is known as system based decomposition utilizing a search strategy of classification known as establish-refine. Hierarchical classification has several advantages, namely that highly compiled knowledge is easily encoded in structure programming constructs, and the search space can be easily reduced.

It is important to note that this method of organization of the knowledge base does not insure the correct diagnose of all abnormalities related to the system by the expert system. This is because it is difficult to conceive all possible interactions between systems for every possible scenario. For this reason the OA features a knowledge acquisition module. Additional knowledge is added to the OA by following a set of step where the user 1) chooses a scenario of interest to add, 2) identifies all plant systems associated with the scenario, 3) develops the proper hierarchical decomposition for the systems related to the scenario, 4) incorporates this scenario specific knowledge including the hierarchical decomposition into the expert system and 5) tests the expert systems response on the scenario.

The expert system was applied to a multiple mode Residual Heat Remover (RHR) in a Boiling Water Reactor (BWR) to diagnose and manage operations and procedures. The RHR system is an eight-mode system that is used to remove heat from the core under normal shutdown conditions, maintain desired pool temperature, and to maintain desired reactor water level under abnormal conditions by condensing reactor steam. The original design of the OA was unable to distinguish between transients caused by operator actions such as a change in power level, and faults in the actual system. To account for this deficiency, the system was modified so that transients were qualified as system abnormalities or changes caused by normal operator actions such that the RHR system is in a steady state, a transient state (changing from one mode to another), or an abnormal state. When the OAs monitoring module detects a change in the operating conditions of the RHR the OA uses a routine to check if the abnormality is caused by operator action or if the system is indeed in an abnormal state. If the routine determines the system is in an abnormal state the diagnostic module is initiated, else the OA initiates another module to assist the operator in whatever action is being executed.

Due to the multi-modal aspect of the RHR, it can be difficult to use classification-based decomposition to diagnose the system. This is because it is necessary to construct a different diagnostic hierarchy for each mode, or build one diagnostic hierarchy with a knowledge base for each mode. This can be remedied by using a functional base decomposition to decompose the system hierarchy. Functional decomposition operates by denoting top hypotheses that specify failures and faults in the system. The hierarchy is based on system malfunction and the function of the subsistent of the systems containing the abnormalities related to the malfunction. The top of the hierarchy states the function of the system being diagnosed, in this case the RHR. The lower level states the major functions that are necessary to perform the function of the higher levels until the hierarchy ends with the function of independent components. This system is diagnosed to determine if it is performing its function for the indicated mode. This is achieved by diagnosing the input and output conditions of the system. If some fault is detected, the expert system systematically moves down the malfunction hierarchy testing the status of each component it encounters until the failed component is isolated. The functional decomposition approaches is advantageous as the functionality of the components of nuclear power plants is well defined which provides consistent malfunction hypotheses. Also, the multiple modes of the RHR system can be diagnosed using this method of hierarchical decomposition.

The expert system was applied to a heavy water reactor to detect and diagnoses root causes of deviations from normality in the reactors process water loop including leaks. The applied system uses a diagnostic knowledge base that features a hybrid of the system-based decomposition and the functional based decomposition. This organization of the knowledge base was used because a set of specific malfunctions was chosen and experts in diagnosing these malfunctions were available [Miller 1994]. The heavy water reactor that the expert system was applied to was a 2700 MW thermal reactor used to produce plutonium, tritium, and medical isotopes. Heat from the core is transferred to the process water loop that moderates and cools the core. Heat exchangers are then used to transfer the heat from the process water loop to the cooling water system that rejects the heat into the environment. The process water loop's main function is to dissipate heat generated in the reactor by pumping heavy water (D_20) from the reactor to heat exchangers where it is cooled. The system consists of a pump suction valve, a pump, a heat exchanger, and an outlet valve.

The knowledge base for the expert system was acquired through written documents, and interviews with experts. This information was organized into a malfunction hypothesis hierarchy where the top nodes represent more general malfunction hypothesis and the lower nodes called tip nodes represent more specific malfunction hypothesis that correspond to a specific malfunction. The organization featured a hybrid of system and functional based decomposition where the choice between the decomposition is knowledge driven. The reason for choosing the hybrid representation was the goal of an expert system is to not only perform the task of an expert, but also mimic the manor in which the expert performs the task. The hybrid knowledge base was best able to obtain this goal.

Here the decomposition of a process water system malfunction is discussed to demonstrate the use of functional and system base decomposition. The top node of this hierarchy is naturally a process water malfunction. The process water system can be decomposed into two major functions of process water containment and process water transport through the reactor and heat exchangers. Thus, lower node malfunctions are process water leak and process water flow malfunction. The children nodes of the process water flow malfunction use a system based decomposition to isolate the loop where the flow malfunction occurred. A functional based decomposition is then used to classify the malfunction as a flow loss or valve malfunction in the faulted loop. This final classification leads to the isolation of the root cause of failure in the process water system. By using both types of decomposition the root cause of failure is identified in a manor similar to that an expert would use.

This expert system has undergone several structural tests to ensure its functionality and accuracy in diagnosis. Testing of the knowledge base indicated that system could correctly detect and diagnose the complete set of process water malfunctions.

The common factor in the article "Experience with the hierarchical method for diagnosis of faults in nuclear power plant systems" and the research present in this thesis is that both use an expert system for fault detection and isolation for a large complex system featuring a large number of components. Of course a nuclear power plant is immensely more complex than the CAVIS system, however, the systems are similar in that both feature multiple components that are subject to failure and these failures yield affects in other components in the system. Similar to the nuclear power plant the structure of the CAVIS system lends itself to decomposition into a hierarchy that an expert system can use to detect and isolate malfunctions and failures. Thus, a hierarchical method can be used by an expert system to detect and diagnosis faults and malfunctions in the CAVIS system similar to the method the expert system is used to diagnosis the nuclear power plant.

The structure of the OA described in the article is similar to the expert system designed to monitor the CAVIS system. The OA from the article features four modules:

a data management module, a monitoring module, a diagnosis module, and a procedure module. The expert system designed to monitor the CAVIS system currently features three modules that perform similar tasks. The CAVIS monitoring expert system features a feature extraction module that acts as a custodian and interpreter for the data collected by the CAVIS system. This module is similar to the data management module in that it processes data from the CAVIS system and provides information pertinent to the other modules of the expert system. A difference between the CAVIS monitoring expert systems feature extraction module and the OA data management module is that the feature extraction module does not currently read or log data from the CAVIS system. However, to implement the CAVIS monitoring expert system this type of functionality would need to be added to the CAVIS monitoring expert system. The CAVIS monitoring system also features a module that monitors the features extracted by the feature extraction module to determine if some abnormality exists in the CAVIS system. Comparing the value of the extracted feature to some threshold value performs this monitoring. If some extracted feature value exceeds the threshold value for the feature the CAVIS system is considered in an abnormal state and the knowledge base module of the expert system is initialized. This module is extremely similar to the monitoring module of the OA. They practically serve identical roles in identical fashions. Finally, the CAVIS monitoring expert system features a module that attempts to isolate the source of abnormality in the CAVIS system detected by its monitoring module. This isolation is performed using a classification base system decomposition hierarchical scheme. This module is similar to the OAs diagnostic module except the OAs diagnostic module features a functional based decomposition scheme as well as a system based decomposition scheme. The CAVIS monitoring expert system is similar to the application of OAs to diagnose nuclear power plant systems in that they both use a similar system architecture and method to detect and isolate faults in the systems they are designed to monitor.

The main difficulties faced by the OA diagnosing the nuclear power plant were transient state of the power plant where operator actions induce changes in state of the plant variables rather than a malfunction or faulted component. Operator actions such as power reduction induce changes in state of system variables that the expert system has difficulty diagnosing. Introducing a classification of the state of the nuclear power plant and organizing the knowledge base in a functional decomposition overcame this difficulty. This type of difficulty should not be faced in the implementation of the CAVIS monitoring expert system, as no transient states exist in the CAVIS system.

The method presented in this article is pertinent to this research in that it demonstrates a successful application of expert systems in fault detection and isolation nuclear system application. The architectures of both systems are very similar with both systems featuring basically the same modules. Also, both systems lend themselves well to a hierarchical decomposition that enable system base and for the OA function based decomposition of malfunctions and faulted components. The main difficulties faces by the OA are transient states that induce changes in state of the system variables. These difficulties were overcome by qualifying the mode of the system and organizing the knowledge base in a functional decomposition. The CAVIS monitoring system will not face these difficulties, as there are no transient states or operations utilized in the CAVIS system.

2.3 Kernel Smoothing

This section contains a description of kernel smoothing and describes the method that it is implemented. Kernel smoothing is a non-parametric technique used to estimate the probability density function of a data set using a kernel density estimator. It can be thought of as a weighted average of the data with the weights defined by the kernel function, which measures a similarity between stored examples and new observations. Suppose data exists from a random sample such that $X_1,\ldots,X_n$ taken from a continuous, univariate density f. The kernel-smoothing estimator $\hat{f}$ is simply the weighted average of each observation in the data set defined by equation 2.1

$$\hat{f}(x,h) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \qquad \text{Eq. 2.1}$$

where $K_h(x, x_i)$ represents the kernel function that defines the weighted average with a kernel width h, n is the size of the data set, x is the contributing observation, and x_i is the data point around which the kernel is placed. The kernel estimate value at the point x is the average of each of the n kernels ordinate at the point x. The kernels spread a probability density of size $\frac{1}{n}$ from each data point to its neighboring data points. Summing the contributions from each data point results in the kernel estimate of the probability density of the data set. Thus, the kernel-smoothing estimator $\hat{f}$ is a representation of the underlying behavior of the data set [Wand 1995].

A kernel function is a multivariate function for $x \in R^d$ and has the following properties [Cherkassky, 1998]:

1. $K_h(x,x_i)$ takes on a maximum value where $x=x_i$.
2. $|K_h(x,x_i)|$ decreases with abs$|x-x_i|$.
3. $K_h(x,x_i)$ is a general function of 2d variables
4. $K_h(x,x_i)$ is a non-negative function
5. $K_h(x,x_i)$ is a radially symmetric function

Any function that meets these criteria can be used as a kernel function, thus there exist many types of kernel functions such as the top hat kernel function, the Epanechnikov kernel function, the triangular kernel function, and the commonly used Gaussian kernel function defined in equation 2.2.

$$K_h(x,x_i,h) = \frac{1}{\sqrt{2\pi}\,h} e^{-\frac{(x-x_i)^2}{2h^2}} \qquad \text{Eq. 2.2}$$

In equation 2.2, x_i is the data point around which the kernel is placed, x is the contributing observation and h is a smoothing parameter or kernel width. The kernel width (h) dictates the spread of the Gaussian function and can be used as a regularization parameter. As the kernel width is increased, the system becomes more biased while its variance decreases, and vice versa. This means that as the kernel width is increased, more weight is given to the surrounding observations and the result is forced towards the mean of the observations, thus the larger the kernel width the smoother the result. The optimal kernel width is problem specific and in many instances must be estimated. Figure 2.2 displays several Gaussian kernel functions with varying kernel widths.

2.4 Kernel Smoothing Applications

The following section contains a description of an application that uses kernel smoothing for fault detection.

2.4.1 Kernel Smoothing for Fault Symptom Generation

Several methods of fault diagnosis for systems have been developed and generally can be classified as model-based fault detection and model-free fault detection methods. The

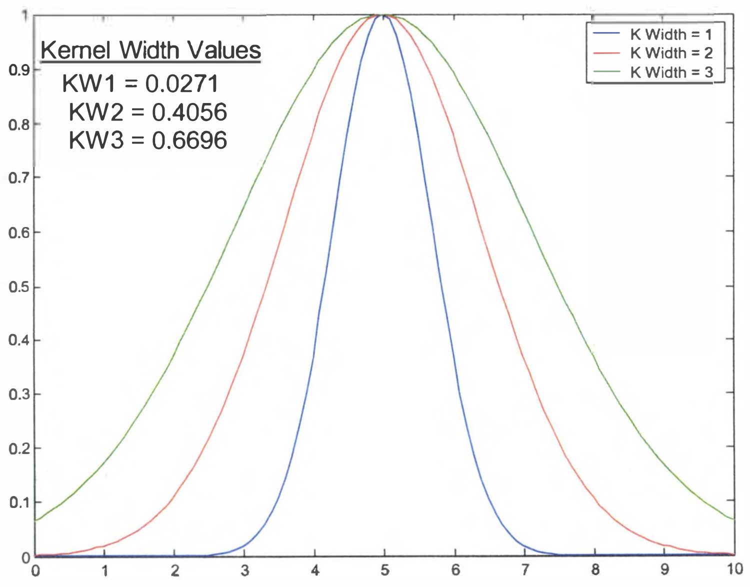


Figure 2.2. Gaussian kernel functions with various kernel widths

utilization of model-based approaches to real life process may be somewhat restricted in that an accurate process model of the system, that may not be available, is required to produce realistic and reliable results. Model based fault detection approaches are also limited because they can generally only be applied to linear processes. If an accurate and reliable model is available, model based techniques are superior, however if a model is not available a non-model based technique may be more appropriate. Suggested by their name, model-free approaches do not require a model for fault symptom generation. These methods function by comparing the measured and desired values of the controlling variables in a system to determine the operating mode for the system.

In the article "A note on nonparametric kernel smoothing for model-free fault symptom generation" [Fenu 1999], a kernel smoothing method is discussed for model free fault symptom generation for nonlinear systems. As previously discussed, kernel based smoothing is a statistical method used to smooth noisy or scattered data to determine the underlying probability density function of the data. Here a method of applying kernel smoothing for fault detection is proposed that employs the kernel bandwidth as a fault symptom. The basic method is when some fault occurs in the system, some correlated change will occur in the smoothness characteristics of the time behavior of the variable being smoothed. The change in the smoothness characteristics will be reflected in the features of the kernel smoother, namely the optimal kernel bandwidth. A change in the optimal kernel bandwidth may signify a fault in the system.

Thus, the optimal kernel bandwidth can be interpreted as a fault symptom, which can be monitored for fault detection.

Consider some single input single output nonlinear discrete time closed-loop plant system such that

$$x_{t=1} = f(x_t, r_t, t),$$
$$y_t = h(x_t, t)$$

where x_t ε $\Re^n$ represents the state vector, y_t ε $\Re$ denotes the output measurement vector, and r_t ε $\Re$ is the set-point input. Also consider the sampling of some unknown continuous random variable y(t) from the system over some finite time interval [t-T,t]. The density of the variable can be determined using the kernel-smoothing estimator $\hat{f}$. As previously discussed, the kernel bandwidth h plays a role in the kernel smoothing result and can be used in fault symptom generation. Several criteria can be used to determine the optimal kernel bandwidth one such being a minimization of an average cost function such that

$$h_t^\circ = \arg\min_{h_t} \int_{t-T}^{t} E[y(\tau) - \hat{f}(\tau)]^2 d\tau$$

where $E[x]^2$ is the square of the expectation. Assuming that the kernel estimator $\hat{f}$ is twice continuously differentiable and that the error in the estimate is a white random process with constant variance σ^2, then it is possible to obtain the optimal kernel bandwidth such that

$$h_t^\circ = \left[\frac{\sigma^2 c_k}{n d_k^2}\right]^{1/5} \frac{1}{\left(\int_{t-T}^{t} [\hat{f}''(x)]^2 dx\right)^{1/5}}$$

where $c_k = \int K^2(x)dx$ is the integral over the square of kernel function K(x), $d_K^2 = \int x^2 K(x)dx$ is the second moment of the kernel function K(x), $\hat{f}''$ (x) denotes the second derivative of the scalar function $\hat{f}(\cdot)$, and n is the number of samples of y(t) over the time interval [t-T,t].

The technique developed in the article is able to use the kernel bandwidth as a fault symptom because the kernel smoothing provides a smooth estimate of the density of the unknown variable y(t). The optimal kernel smoothing bandwidth h_t° is a regularization parameter and dictates the smoothness of the kernel smooth estimator $\hat{f}$. Any change in the system will be reflected in a change in the optimal kernel bandwidth h_t°. Thus, the time behavior of the optimal kernel bandwidth can be used as a fault symptom making the kernel smoother a change detector that can be used to correlate a fault symptom with the current set point vector and any other available variables. It is important to note that the equation for the optimal kernel bandwidth h_t° is difficult to apply in practice because its calculation involves complex unknowns that must be estimated from the data. A good approximation of the optimal kernel bandwidth can be determined using the leave one out cross validation method. Leave one out cross validation partitions the data set into many training and testing sets in which one observation is left out and computes an average squared error over the various sets. Using this method it is possible to define the cross validation function that validates the ability to predict y(t) such that

$$CV(h_t) = \frac{1}{T+1} \sum_{i=t\_T}^{t} [y(t) - \hat{f}(t)]^2$$

The estimation of the optimal kernel bandwidth is then given by the minimizing the average cost function of the cross validation function such that

$$h_t^\circ = \arg\min_{h_t} CV(h_t)$$

This equation is an approximate estimation of the optimal kernel bandwidth and simulations have shown that approximation errors are negligible if the time interval T is not to small.

The developed method was applied to a well-known FDI industrial actuator benchmark Blanke et al. The benchmark at Aalborg University in Denmark is based on an electro-mechanical test facility. The benchmarks actuator is a brushless synchronous DC motor connected by an arm and an epicyclic gear train to a rod. A similar motor is mounted in parallel to the rod that allows the desire external load torque to be simulated. Four sequences of data from a real-time monitoring of the process were analyzed by the kernel smoother with the size of the moving batch set at 150 milli-seconds. The four sequences were no load disturbance, a position measurement fault, a load step disturbance, and a current fault. During the no load disturbance sequence the actuator is operating without an external load torque being applied by the motor. The position measurement fault sequence simulates a malfunction of the position measurement of the motors power drive. The current fault simulates a malfunction that results in the power drive being able to only deliver positive current. The load step disturbance simulates an external torque load applied to the actuator. The developed method was able to successfully diagnose the faulted sequences in the process as the position and current faults resulted in significant changes in the kernel bandwidth, while the kernel bandwidth remained unaffected by the load disturbance. Thus, the optimal kernel bandwidth was successfully used as a fault symptom in this nonlinear system and the kernel smooth estimator could be used as a fault generator by monitoring the smoothness characteristics of the time behavior of the measured variables.

The common factor in the article on nonparametric kernel smoothing for model-free fault symptom generation and the research present in this thesis is that both use kernel-smoothing techniques to monitor a system for a faulted state. The method presented in the article monitors the parameters of the kernel smoothing for changes that signify some fault state of the system. Here the optimal kernel bandwidth is calculated for each setpoint and monitored for changes in order to generate faults. This research implements kernel smoothing to detect and identify regional radiation disturbances caused by regional anomalies in the CAVIS system. Here the actual kernel smoother estimator $\hat{f}$ is monitored for an abrupt change, as the probability function of the CAVIS sensor being monitored by the kernel smoother should center about a mean value of zero given no abnormalities in the CAVIS system.

The method presented in this article is pertinent to this research in that it demonstrates a successful application of kernel smoothing in a fault detection application. While the two applications do not monitor a common fault symptom both are based on similar reasoning that fault detection is possible by monitoring the smoothness characteristics of the systems they are assessing. A problem encountered in the method presented in the article is the calculation of the optimal kernel bandwidth. This is because the calculation of the optimal kernel bandwidth involves complex unknowns that must be estimated from the data. This problem was remedied by estimating the optimal kernel bandwidth using a leave one out cross validation method. A similar problem exists in this research in that it is difficult to choose an optimal kernel bandwidth for the kernel smoothing. This is because the kernel bandwidth is dependent on the physical distance between the sensors in the CAVIS system. The physical distance can be difficult to estimate, as the CAVIS storage area is a functional area where the equipment is frequently being moved. Thus, the optimal kernel bandwidth will have to be determined empirically when implemented at the Y-12 site.

3 Y-12 National Security Complex and CAVIS Background

This chapter contains a description of the Y-12 National Security Complex, and the equipment and proposed facility for the storage and monitoring of the SNM. This includes a discussion of the Highly Enriched Uranium Material Facility (HEUMF) and the Continuous Automated Vault Inventory System (CAVIS^{TM}).

3.1 Y-12 National Security Complex

The Y-12 National Security complex, shown in figure 3.1, is an 811-acre facility in Oak Ridge Tennessee built in 1943. Part of the Manhattan Project, the original mission of the Y-12 facility was to process uranium for the first atomic bomb [Yesterday at the Y-12 National Security Complex]. Since the end of World War II, the mission of Y-12 has changed to including the manufacturing and remanufacturing of unique nuclear weapon components, the dismantling, storage and evaluation of returned weapons, and the storage and management of enriched uranium material known as special nuclear material (SNM) [Defense Programs at Y-12].

As nuclear weapons are retired from the national stockpile or returned for dismantlement under strategic arms reduction treaties, the size of the nations stockpile of weapons grade uranium will increase. The safe, secure, & reliable storage of this material is essential to national security. The Y-12 National Security complex currently houses the nations supply of weapons grade uranium and construction is currently

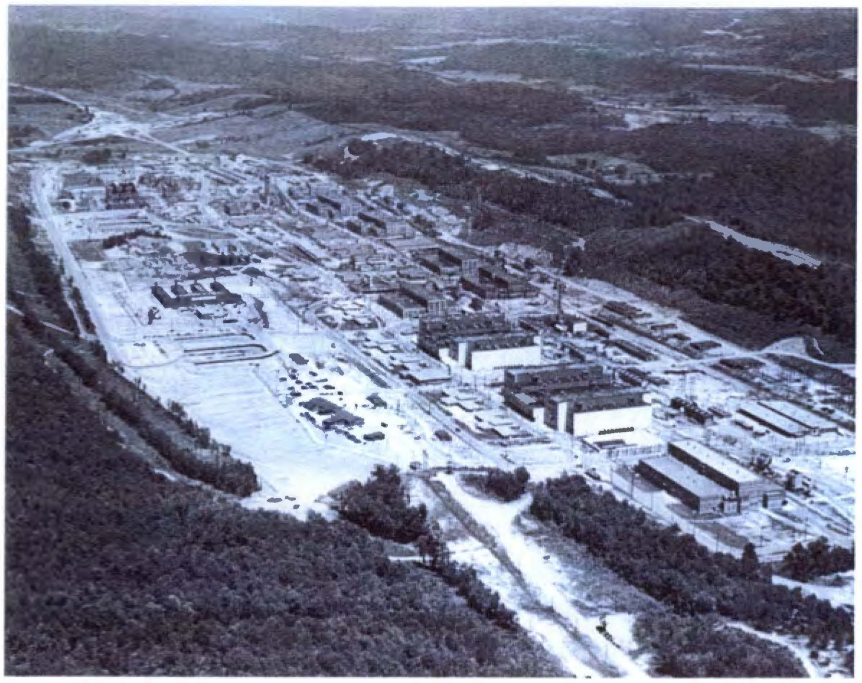


Figure 3.1. Y-12 National Security Complex [Society for the Historical Preservation of the Manhattan Project 2004]

underway on the Highly Enriched Uranium Material Facility (HEUMF) that will function as a modern storage facility for the SNM.

3.2 Highly Enriched Uranium Material Facility (HEUMF)

The Highly Enriched Uranium Material Facility (HEUMF) will act as the nations repository for highly enriched uranium (SNM) [Parson 2002]. Currently, there are five separate facilities at the Y-12 National Security complex where SNM is stored. Upon completion, the HEUMF will act as the single repository of the SNM [Parson 2002]. Construction began on the facility in 2001 and should be completed in 2005. A modern monitoring system termed CAVIS will monitor the storage of the SNM in the HEUMF. Figure 3.2 is a picture of the proposed HEUMF.

3.3 Continuous Automated Vault Inventory System (CAVIS)

One of the missions of the Y-12 National Security complex is the storage of SNM. The Department of Energy (DOE) requires that the status of the SNM inventories be confirmed periodically [Younkin 1999]. The purposed of the inventory status is to ensure that the SNM is secure. One method is to continuously verify that the weight and radiation attributes of the SNM have not changed. Currently, inventory verifications are performed manually, resulting in an expensive process that exposes workers to radiation.

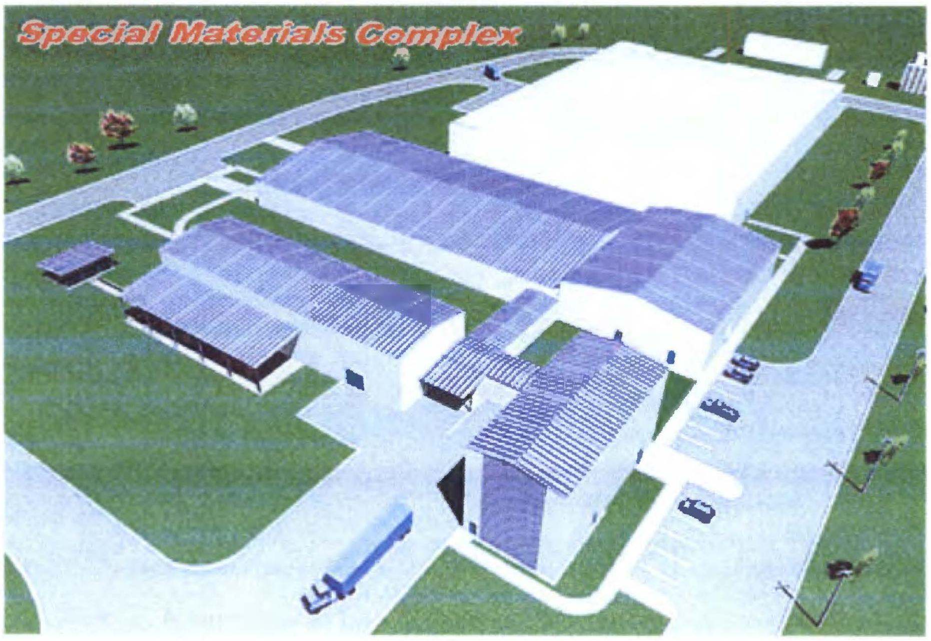


Figure 3.2. Highly Enriched Uranium Material Facility [Parsons 2001]

CAVIS was developed to provide a method to remotely perform the inventory confirmation [Pickett 1999]. CAVIS is an integrated package of low-cost sensors used to continuously monitor the weight, radiation, and temperature attributes of SNM [Pickett 2003 A]. The CAVIS system detects "changes-in-state" of these attributes for the special nuclear material and generates an appropriate alarm. CAVIS continually receives the radiation, weight, and temperature signals and forms a hierarchical network of components including Power and Communication Distribution Units (PCDU) and Sensor Concentrations. Figure 3.3 illustrates the structure of the CAVIS system.

Several sensor types are available for monitoring the SNM attributes. This research projects focuses solely on the radiation sensors. The available radiation sensors for use in CAVIS are RADSiP™, RADTELL™, or other ORSENS™ radiation sensors. The radiation sensors selected for use in the CAVIS system are RADSiP^{TM} Photodiode

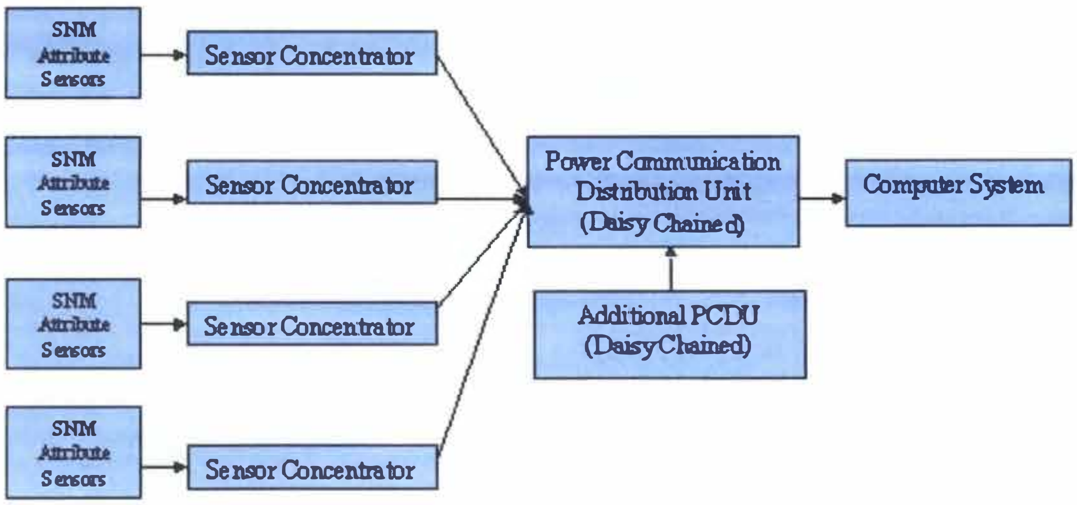


Figure 3.3. CAVIS System Diagram

Gamma Ray Sensors. These sensors are small, inexpensive, and are well suited to monitor stored nuclear materials for long periods of time. Suggested by their name, the sensors monitor the gamma ray emission of a radioactive source. The sensor continually monitors the gamma ray emission radiation level of the SNM. Due to the long half-life of the nuclear material, the radiation level remains approximately constant, so any deviation suggests an abnormal status [Harrison 04].

The radiation signal, detected by the RADSiP detectors, propagates through a series of junctions. The signal is first sent to a sensor concentrator, manufactured by ORSENS, for processing. Here the signal collected by the detector is summed to calculate a radiation count rate. The signal is then sent to a Power and Communication Distribution Unit. This component provides power to the Sensor Concentrator and RADSiP sensors and relays the radiation count rate to a central computer system. The monitoring computer system can be a desktop personal computer running either a National Instruments LabView® (Windows® 95) application or the GraFIC™ software package on a Windows® NT system [Pickett 2003 A]. The computer system requirement is an Intel Pentium133 MHz based computer or higher. The computer system logs the count and performs the calculation necessary to determine if a change in state of the special nuclear material has occurred.

Currently, a change in state of a radiation signal is defined as a certain deviation in the count rate, measured in the standard deviation of the signal, from the norm. Radiation counting is a Poisson process, thus the standard deviation of the radiation signal is the square root of the signal's mean. If an observation occurs outside of a three-sigma confidence interval placed on the mean then the CAVIS system will alarm. The CAVIS monitoring system's feature extraction module will monitor for a change in state using the SPRT, an optimal method, rather than the three-sigma confidence interval.

The CAVIS system contains numerous components in the system hierarchy whose failure will result in a disturbance of the propagation of the radiation signal. CAVIS features an error and anomaly reporting module that attempts to quantify the status of the radiation signal based on the communication between a microprocessor module contained in the sensor concentration and the central computer system. If proper communication exists between the microprocessor module and the central computer, the status of the radiation signal for all RADSiP sensors common to the microprocessor is classified as "GOOD." If the microprocessor and the central computer are not properly communicating, the status of the radiation signal for all RADSiP sensors common to the microprocessor is classified as "BAD." This sensor signal status is incorporated as one of the features used by the CAVIS monitoring systems expert system to isolate root causes in the CAVIS system.

Due to the integration of the sensor concentrator network, a single computer may monitor thousands of SNM storage vaults. Figure 3.4 is a picture of the CAVIS system used in this research project.

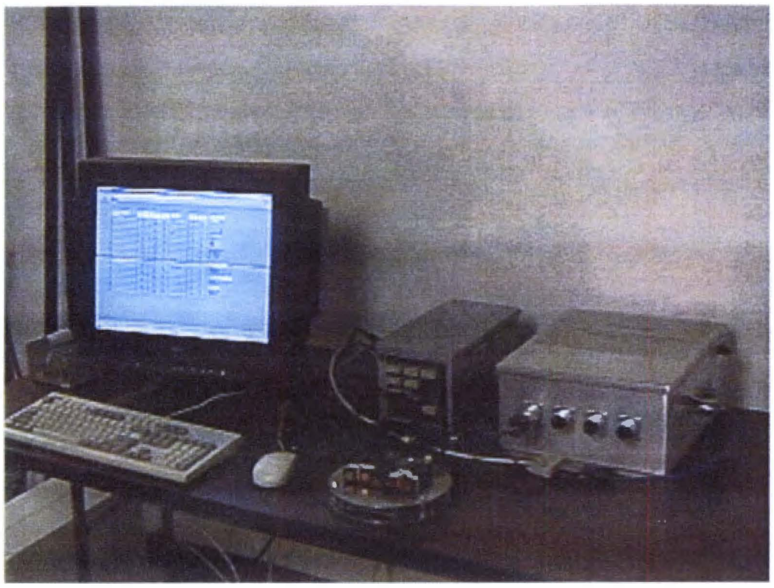


Figure 3.4. CAVIS System

3.3.1 RADSiP Radiation Sensors

The RADSiP™ photodiode gamma ray sensor is used to monitor the radiation attribute of the SNM in the CAVIS system. The sensor is small (1.5 cm wide by 8.75 cm long), inexpensive, and well suited for continual long-term monitoring of the SNM. Suggested by their name, the sensors monitor the gamma ray emission of a radioactive source. The sensor requires a single +12 V power supply for electronics power and the sensor is capable of monitoring the 20 keV to 100 keV gamma-ray energy band. The components of the sensors are a Silicon-PIN photodiode, a low-noise preamplifier, and a pulse-shaping amplifier [Pickett 2003 C]. Gamma ray interaction within the photodiode produce pulses at approximately 500 counts per second per R per hour. The produced voltage pulse is shaped by filters in the pulse-shaping amplifier to provide an impulse response with a width of 20 microseconds. The detected signal level can be selected using a pulse height discriminator. The discriminator lower level is set at the energy peak of americium-241 (60 keV), the proper lower-level adjustment for precise gamma ray monitoring of ^{235}U. The discriminator upper level is adjusted to the highest energy of the Compton interaction pulses generated in silicon. Thus, the hardware is adjusted to monitor the 60–100 keV range. ^{235}U decays to ^{231}Th by an alpha decay and the daughter emits several characteristics gamma rays while decaying to the ground state. The most prevalent characteristic gamma ray is the 185.715 keV gamma ray, which has a relative intensity of 57.2 [Schmorak 2004]. This gamma ray fluoresces neighboring uranium atoms due to the photoelectric effect, resulting in several K shell X-rays energy peaks in the 60-100 keV energy range, the range monitored by the RADSiP sensor. The complete specification sheet for the RADSiP radiation sensors can be seen in appendix A. Figure 3.5 is a picture of a RADSiP photodiode gamma ray sensor.

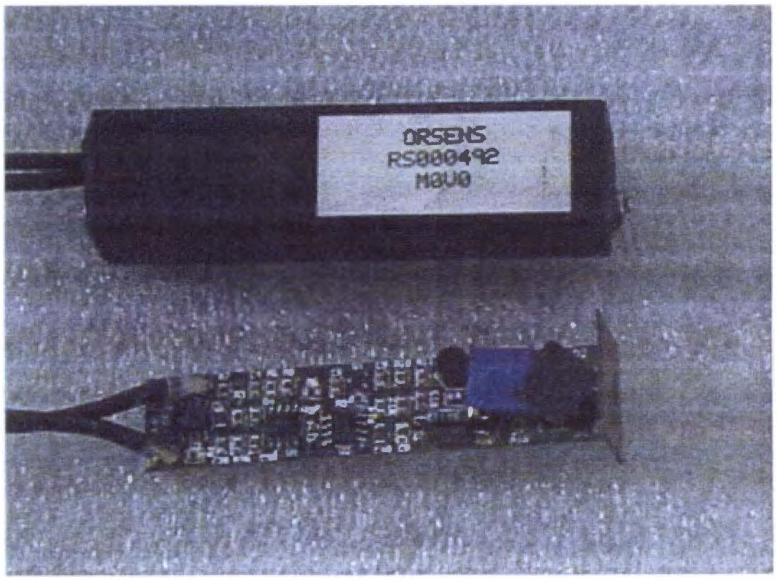


Figure 3.5. RADSiP photodiode gamma ray sensor

3.3.2 ORSENS Sensor Concentrators

The sensor concentrator acts as the first relay junction in the CAVIS system. Here the radiation signal from the RADSiP sensors is summed to calculate a radiation count rate and then further relayed on through the CAVIS system to the central computer system. The sensor concentrators used in the CAVIS system are ORSENS sensor concentrator which were developed with LonWorks® Technology. ORSENS sensor concentrators are a configurable multi channel sensor interface and signal-processing unit designed for use with the ORSENS sensors featured in the CAVIS system [Pickett 2003 B]. The unit has a voltage range for the operating power supply of 16 to 28 V DC. The sensor concentrator also has several features that ensure proper function and security such as an enclosure tamper switch, enclosure temperature sensor, and a peer-to-peer communications that allows the sensor concentrator to report error and anomalies immediately. The error and anomaly reporting is a particularly useful feature incorporated in the working memory of the proposed expert system.

The sensor concentrator has a six-slot motherboard that can accommodate four sensor interface modules and two microprocessor modules. The four sensor interface modules collect the signal from the RADSiP sensors and the two- microprocessor modules process the signal to calculate a count rate for each sensor. The ORSEN sensors all have a mating interface module that allows the connection of ten channels of each type of sensors to each sensor interface module. Thus, a maximum of forty ORSENS sensors can be connected to a single sensor concentrator. The sensor concentrator also contains a communications processor to coordinate the relay of the sensor data from the two-microprocessor modules throughout the CAVIS system. The communications processor is an Echelon 3120 Neuron® based module with its embedded LonTalk® communications protocol. Figure 3.6 is a picture of the ORSENS sensor concentrator with a sensor interface module and a microprocessor module positioned beside the sensor concentrator.

3.3.3 Power and Communication Distribution Unit

The Power and Communication Distribution Unit (PCDU) acts as the second relay junction in the CAVIS system. The PCDU relays the processed radiation signal from the sensor concentrators to the central communication computer. The PCDU unit also provides power to the RADSiP radiation sensors, and the sensor concentrators. The PCDU unit has mating interfaces available for up to four sensor concentrators. Thus, the

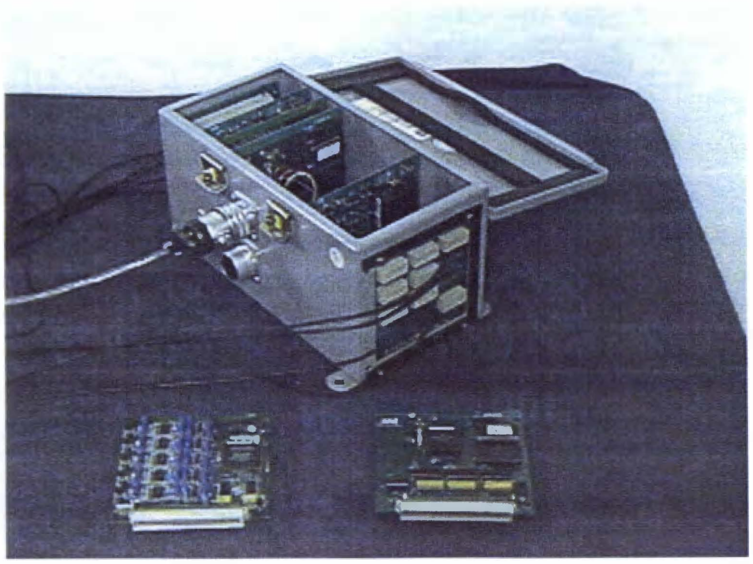


Figure 3.6. ORSENS Sensor Concentrator

PCDU unit is capable of providing power and relaying the radiation signal to the central computer for up to 160 ORSENS sensors. The PCDU units can also be daisy chained together if additional sensors are necessary. Figure 3.7 is a picture of the Power and Communication Unit used in the CAVIS system.

3.4 CAVIS Deficiencies

The CAVIS system is susceptible to alarms, which may not coincide with the removal of special nuclear material. Several factors can result in false radiation sensors alarms. First, the statistical nature of radioactive decay and counting may cause the count rate to fall outside of the commonly used 95% confidence intervals. In such an instance, the state of the SNM has not changed and the CAVIS system incorrectly alarms. Secondly,

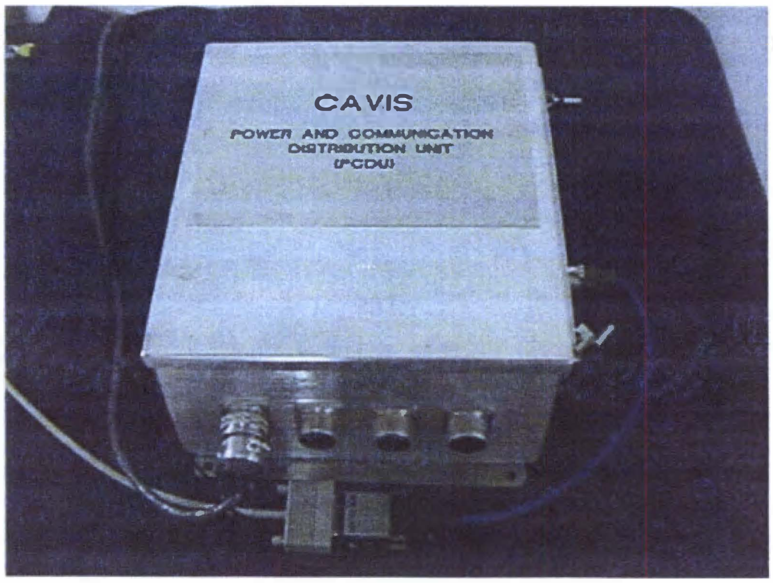


Figure 3.7. Power and Communication Distribution Unit

the CAVIS system is comprised of numerous components that may fail over time. Thus, the CAVIS system may generate alarms due to component failures, which are not correlated with changes in the SNM. Thirdly, the storage area is a functioning warehouse that may have radioactive material being moved. These external radiation sources may be detected by the CAVIS system causing an alarm in the region of the warehouse where the external radiation source is located. Fourthly, the radiation sensors used in the CAVIS system display a spike behavior when they are impacted. Forklifts and other heavy equipment moving in the warehouse may cause impacts that are transmitted to the CAVIS storage vaults inducing spikes in the count rates. In addition to inducing spikes, a collision between heavy machinery and the CAVIS storage vault may cause the RADSiP radiation sensors to move.

The radiation signals are affected by source (SNM) distance, collimation of the source, and the SNM container thickness and material. If the collision results in any movement of the RADSiP or the SNM the sensor count rate may decrease and induce CAVIS alarms. Finally, the CAVIS system has displayed a dependence on environmental stimuli such as heat and humidity. Thus, the environmental conditions of the storage area may cause the CAVIS system to generate false alarms. These numerous conditions can all result in CAVIS false alarms, which may result in unnecessary and costly inventory checks.

3.5 CAVIS Component Failure

Several experiments were performed on the CAVIS system to explore the effect various CAVIS component failures have on the CAVIS system. The CAVIS system was tested for these effects by simulating a CAVIS component failure following a normal operating condition counting experiment. Thus, it was possible to correlate any observed behavior in the reported count rate to the induced failure. Due to a limited amount of CAVIS equipment, it was not feasible to destroy or fail any of the equipment. Thus, the only method to simulate a CAVIS component failure was to remove or unplug the component to be failed. As previously discussed and illustrated by figure 3.3 the CAVIS system is a hierarchical system in which a component failure may produce a response in the components that are common to the failed component. For example, if a sensor interface modules in the sensor concentrator were to fail, the RADSiP sensors common to the interface module would exhibit a change in state of their radiation signal. This section describes the behavior of the RADSiP radiation signal when each component of the CAVIS system fails. Also, this section describes RADSiP radiation signal behavior observed in CAVIS data sets collected at the Y-12 National Security Complex. It is not possible to know how or why the component failed in the Y-12 data sets, only that the RADSiP radiation signal displayed a certain behavior corresponding to some failed component.

3.5.1 RADSiP Sensor Failure

The RADSiP radiation sensors are connected individually to the Sensor Concentrator by a two-prong wire. This connection provides power to the RADSiP sensor and relays the radiation signal from the sensor to the sensor concentrator junction. If the RADSiP fails or the wire is damaged, the radiation signal goes to and remains at zero. The CAVIS systems error and anomaly-reporting module monitors the status of the radiation signal from each RADSiP. Individual RADSiP sensor failures do not affect the communication between the microprocessor module common to the sensor and the central computer system. Thus, the CAVIS systems error and anomaly reporting module is unable to detect the failure and will continue to report a radiation signal status of "GOOD" for the failed or unplugged sensor.

3.5.2 Sensor Concentrator Failure

The sensor concentrator acts as the first relay junction in the CAVIS system where the radiation signal from the RADSiP sensors is summed to calculate a radiation count rate and then further relayed on through the CAVIS system. The Sensor Concentrator is connected by a two-prong wire to each of the RADSiP radiation sensors, which provided power to the RADSiP and relays the radiation signal from the RADSiP to the Sensor Concentrator. The Sensor concentrator is connected to the PCDU by a large gauge wire whereby the PCDU provides power to the Sensor Concentrator and receives the radiation signals. If the sensor concentrator fails or the large gauge wire is damaged, the radiation signal for each RADSiP sensor common to that concentrator will go to and remain at zero. Also, the CAVIS system self-diagnosis feature is capable of detecting this failure and will report a radiation signal status of "BAD" for each RADSiP sensor corresponding to the failure.

The sensor concentrator has a six-slot motherboard that accommodates four sensor interface modules and two microprocessor modules that are all capable of failure. The sensor interface modules collect the radiation signal from the RADSiP sensors and relay the processed signal to the PCDU junction in the CAVIS system. Five RADSiP radiation sensors are common to each sensor interface module. When a sensor interface module fails, the radiation signal for each RADSiP sensor common to the module goes to and remains at zero. Sensor interface module failures are not detected by the CAVIS system self diagnosis features and the radiation signal status will remain at "GOOD" throughout this type of failure. The two- microprocessor modules process the signal to calculate a count rate for each RADSiP sensor. Ten RADSiP radiation sensors are common to each microprocessor module. When a microprocessor module fails the radiation signal for each RADSiP sensor common to the module goes to and remains at zero. When a microprocessor module fails, the CAVIS system self-diagnosis feature will detect the failure and report a radiation signal status of "BAD" for each RADSiP sensor common to the failed sensor interface module.

3.5.3 PCDU Failure

The PCDU provides power to the RADSiP radiation sensors, and the sensor concentrators and also acts as the second relay junction in the CAVIS system. The

PCDU unit has mating interfaces for up to four sensor concentrators, thus one PCDU provides power to four-sensor concentrators power and relays the radiation signal to the central computer for 80 RADSiPs. If the PCDU experiences a power loss the radiation signal for all RADSiP sensors common to the PCDU goes to and remains at zero. Also, the radiation signal status is "BAD" for all RADSiP sensors common to the PCDU. The PCDU is connected to the central communication computer by a large gauge wire. In the event this wire become damaged or unplugged the radiation signal for the RADSiP sensors common to the PCDU goes to and remains at zero. Also, the radiation signal status is "BAD" for all RADSiP sensors common to the PCDU. The PCDU units can also be daisy chained together if additional sensors are necessary. Due to equipment limitations the effect of a PCDU failure at a point in the chain on the other PCDU in the chain is not known.

3.5.4 Y-12 Data Set Component Failure

The radiation signal from several CAVIS data sets collected over a period of several months at the Y-12 National Security Complex were analyzed for behavior representative of CAVIS component failure. The data were collected at one-hour intervals with no alterations made to the system; thus any abnormal changes to the count rate indicate a system degradation or component failure. As previously discussed, it is not possible to know how or why the component failed in the Y-12 data sets, only that the RADSiP radiation signal displayed a certain behavior that may be indicative of a failed component. Of the forty analyzed data sets, twenty-two display some behavior that appears to be related to component failure. The behavior that these component failure data sets display is RADSiP sensors that are "stuck" and common to some component. A "stuck" sensor is a radiation sensor that returns some non-zero count rate repeatedly without variation. All twenty-two of the component failure data sets contain a Sensor Concentrator wide "stick" or a portion of the data set where all twenty of the RADSiP common to a sensor concentrator are stuck. Here the assumption can be made that some failure in the Sensor Concentrator resulted in the stuck sensors. Figure 3.8 illustrates a Sensor Concentrator stick.

Further analysis of the component failure data sets reveals that prior to the sensor concentrator sticks, individual sensors will have brief instances of being stuck and then return to normal operation. These sticks may last for five or six data observations (five or

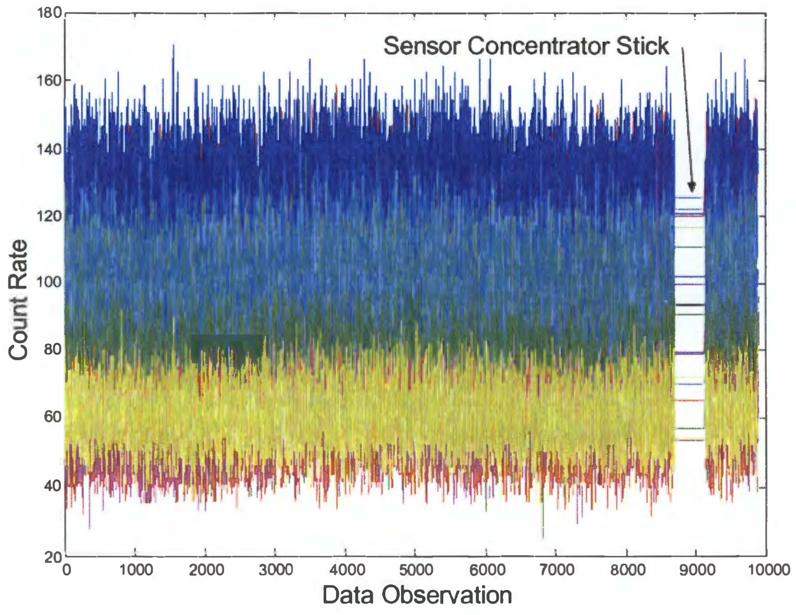


Figure 3.8. Sensor Concentrator Stick

six hours) before the sensor returns to normal operation. This behavior may be a precursor to a sensor concentrator failure that will occur later in the data set. All of the twenty-two data sets containing a sensor concentrator failure exhibited this random stuck detector behavior prior to the sensor concentrator failure. However, due to the manner in which the data sets were collected it is impossible to be certain this behavior is a precursor.

The component failure data set also contain instances in which a sensor interface module fails. In these instances, the radiation signals for five of the twenty RADSiP sensors will display some common abnormal behavior. This abnormal behavior is a zero count rate for each sensor common to the module. All five of these sensors correspond to the same sensor interface module due to their consecutive labeling (Sensor Labels: 1-5, 6-10, 11-15, 16-20). The root cause of these abnormal behaviors is unknown, only that the behavior is common to the sensor interface module. Figure 3.9 illustrates a sensor interface module failure for one module in a Sensor Concentrator.

3.6 CAVIS Environmental Effects

The CAVIS system is designed to monitor the material year round; therefore, it is necessary to ensure that changes in the environmental conditions of the storage area, which may be induced by the changing seasons, will have no effect on the radiation sensors and their radiation signal. The current CAVIS facilities have no humidity control and limited temperature control. The HEUMF will have both temperature and humidity control however; RADSiP environmental response is advantageous information to

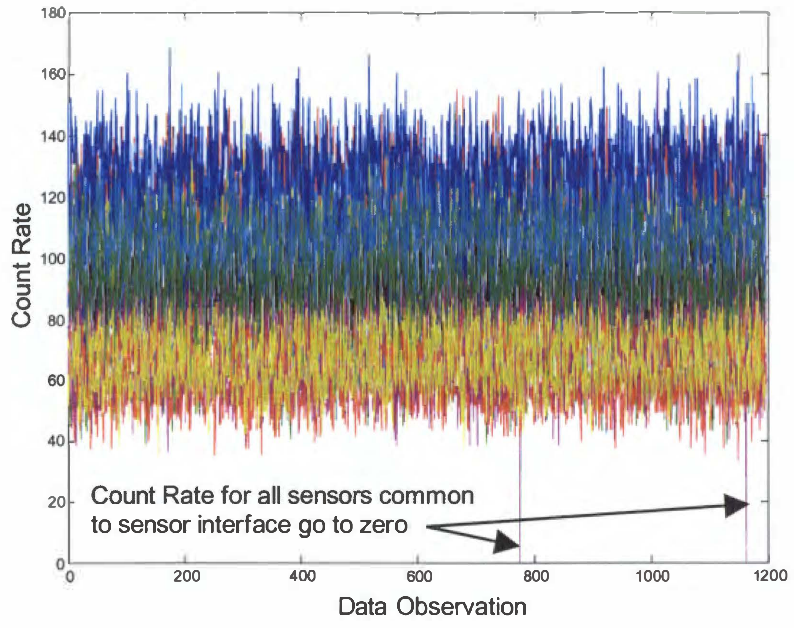


Figure 3.9. Sensor Interface Module Failure

contain in the expert systems knowledge base. Thus, several experiments were performed to determine the RADSiP sensor response to temperature and humidity variation. The Sensor Concentrator and PCDU were not tested for environmental effects due to equipment limitations. The manufacturers of the RADSiP radiation sensors have provided specifications for the sensors. The allowable operating range temperature for the sensor is –20 to 125 degrees Fahrenheit. The humidity operating range is between 0 – 100% humidity. This operating range should cover the environment in which the radiation sensors are housed. These experiments were performed to ensure that the radiation sensors do not exhibit any flawed performance in or near this operating range.

3.6.1 CAVIS Environmental Testing Methodology

The effect of environmental changes on the response of the RADSiP radiation sensors was tested using a simple environmental chamber. The environmental chamber was constructed using an aquarium with a heater or a humidifier inside it. By performing counting experiments inside of the chamber with either the heater or humidifier operating it was possible to observe the effect these environmental stimuli have on the responses of the radiation sensors. Only the sensor was stressed by the environmental changes, the effects on the power supplies, and remote signal processing electronics was not tested.

The device used to heat the chamber was a compact fan-forced heater with two-power setting of 750/1500 watts. The heater was capable of heating the environment inside of the aquarium to approximately 150 degrees Fahrenheit. The heater featured a thermostat that enables the temperature in the chamber to be held somewhat constant. A device to lower the temperature of the chamber was not incorporated in the experiment, however the ambient temperature of the room in which the experiments were performed was approximately 65 degrees Fahrenheit. Thus, the radiation sensors were subjected to a temperature range of approximately 65 – 150 degrees Fahrenheit.

To test the response of the radiation detectors to various amounts of humidity, a similar technique was incorporated. The detectors were placed inside the environmental chamber along with a humidifier. The humidifier was a 2.5-gallon model capable of humidifying an 1100 ft^2 area. The humidifier was able to completely saturate the environmental chamber. Turning the humidifier on inside of the chamber and monitoring the response of the detectors tested the detectors response to humidity. A hygrometer was not available to monitor the exact humidity of the chamber. This was not an issue as this experiment was only looking for changes in response inside the sensors operating range. The allowable operating range given in the manufactures specification sheet is 0 – 100% humidity, thus it was not possible to stress the sensor outside of its operating range.

The counting experiment performed inside of the environmental chamber featured the RADSiP radiation sensors monitoring the activity of a 10 micro Curie Barium 133 source. A Barium 133 source is utilized because a portion of its gamma emission energy (79.6139 keV 2.62% of decay, 80.9971 keV 34.06% of decay) is contained within the energy ranged monitored by the RADSiP sensor (60-100 keV). During the counting experiment the temperature or humidity inside of the environmental chamber was changed and any observed effect in the radiation signal was correlated to the environmental change. To determine if the environmental stimuli induced a change in state of the radiation signal, the signal was analyzed using the sequential probability ratio test (SPRT) [Wald 1945]. The SPRT is a test that is capable of monitoring statistical properties of a Gaussian distribution. The SPRT is capable of detecting a change in the mean or variance of the radiation signal caused by the environmental stimuli. Additional information concerning the SPRT can be found in chapter 4.3.

The result of the SPRT test used to analyze the data from the environmental experiments is a plot that illustrates when an alarm occurs. The plot is broken into subplots of the reported count rate and of the four hypothesis of the SPRT test: mean shift up (hypothesis 1), mean shift down (hypothesis 2), variance shift up (hypothesis 3), variance shift down (hypothesis 4). The SPRT plot features circles that indicate alarms that the test detects. The number of alarms generated during the SPRT test is dependent upon the number of data points in the set and the mean of the data set. The nature of a stochastic statistical distribution results in some data streams that will cause a false SPRT alarm. In addition, testing has indicated that the SPRT is slightly more likely to alarm with higher data mean values. To account for these properties, post processing has been incorporated. However when this experiment was performed, post processing was not in place and the results of SPRT must be interpreted accordingly. Testing of the SPRT with fabricated data has indicated that an alarm rate of 1 alarm per 10,000 data points can be expected. If a data set contains more than 1 alarm per 10,000 data points it is an indication that a change in the statistical properties of the data set may exist.

A Labview data acquisition system is used to record the radiation count rate and the temperature of the environmental chamber adjacent to the sensors. It should be noted that inside the radiation sensors, where the electronics exist, might be at a slightly different temperature than the temperature measured by the thermocouples. Inside the environmental chamber are the radioactive sources that are attached to the radiation detectors and the thermocouples. The heater is located on the top of the chamber and blows heated air into the chamber. When the humidifier is used, it is located inside the chamber. Figure 3.10 illustrates the basic setup for the environmental testing

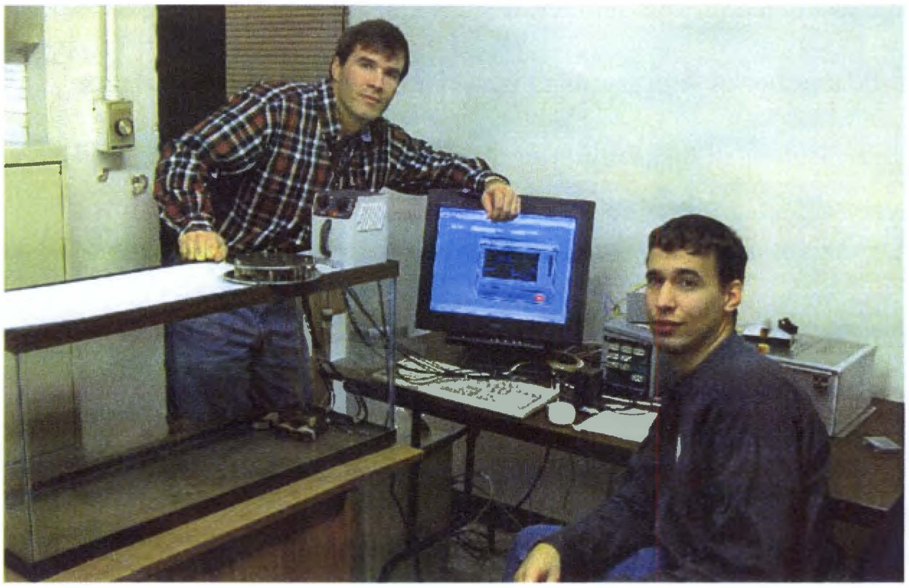


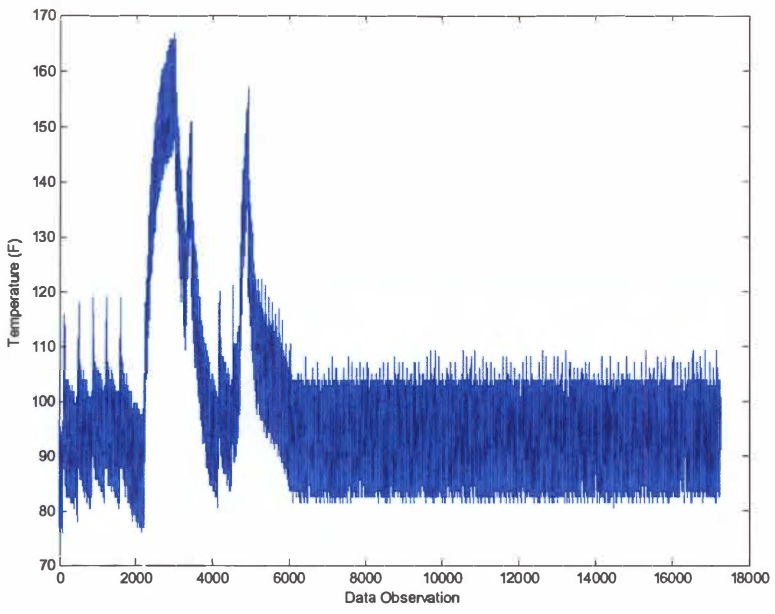
Figure 3.10. Environmental Chamber Setup

3.6.2 Temperature Experiment Results

A counting experiment was performed inside an environmental chamber. During the experiment the temperature of the chamber was either increased or decreased to test the effect that temperature variation had on the RADSiP radiation sensors. The results of the temperature variation are rather difficult to characterize. In general, the radiation sensors show no response to nominal temperature variations. However, in some instances the sensors response is sporadic and troublesome. The experiments indicate that the sporadic behavior occurs when the sensors are exposed to temperatures greater than 110 degrees Fahrenheit. This sporadic behavior is a "ramp up", or a steady increase in reported activity, as the temperature of the environment increases, followed by a "ramp down" behavior as the environment cools. This temperature effect should not be problematic as it is unlikely the sensors will be exposed to such high temperatures in their everyday use. However, the temperature effect should be noted and may be included in the rule base knowledge of the FDI system.

The following example demonstrates how a large temperature variation may result in a change in state of the RADSiP radiation signal. Data were sampled every 5 seconds during the experiment. The temperature of the environmental chamber is held fairly constant at 90 – 100 degrees Fahrenheit except for two two-hour intervals in which the temperature was increased to approximately 150 degrees. This temperature variation may seem extreme, but during most of the experiment the average temperature was approximately 95 degrees, well within the operating range of the sensors. Figure 3.11 is a plot of the temperature of the environmental chamber during the experiment, the response of the radiation sensors to the temperature variations and the SPRT analysis of the signal.

Figure 3.11 indicates that several SPRT mean shift up and variance shift up alarms occurred during the experiment. The SPRT alarms occur during the portion of the experiment when the sensor was exposed to high and greatly varying temperatures. In contrast, when the temperature is held relatively constant near 95 degrees Fahrenheit, the radiation signal does not contain any SPRT alarms except for a spurious mean shift down alarms. This and other results indicate that greatly varying temperatures above 110 degrees Fahrenheit may induce a change in the response of the sensors. The result does demonstrate some temperature dependence, however it may not be very applicable as the temperature of the CAVIS storage area should never be this high or vary this rapidly.



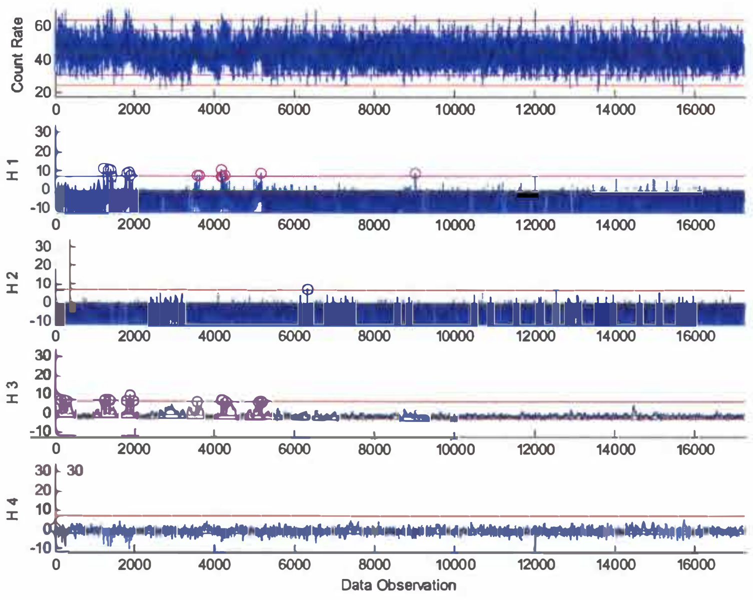


Figure 3.11 Environ. Chamber Temperature, SPRT Analysis of Sensor Response

3.6.3 Humidity Experiments Results

A counting experiment was performed inside an environmental chamber. During the experiment the humidity of the chamber was increased to test the RADSiP response to various amounts of humidity. Data were sampled every five seconds during the experiment. In general, the RADSiP sensors do no display any dependence on the humidity of the environment in which they are housed. In the following example, the RADSiP radiation sensors were exposed to various amounts of humidity to test their responses to this environmental stimulus. The chamber initially had the same humidity as the surrounding room, which will be assumed to be none. At the two-hour point of the experiment, humidity was added to the chamber using a humidifier. The humidifier was left running for 15 hours. The environmental chamber was completely saturated at the end of the experiment. Figure 3.12 displays the results of the SPRT on the sensor signal exposed to these conditions.

The SPRT indicates that the radiation signal did not experience a change in state while being exposed to the various levels of humidity although a few spurious alarms occurred. Thus, this RADSiP sensor displays no dependence on the humidity level it is exposed to. This result is characteristic of the RADSiP response to all of the performed humidity environmental experiments.

3.6.4 CAVIS Environmental Conclusions

The environmental experiments indicated that the RADSiP radiation sensors manufacturers suggested operating ranges are fairly accurate, however when the sensors

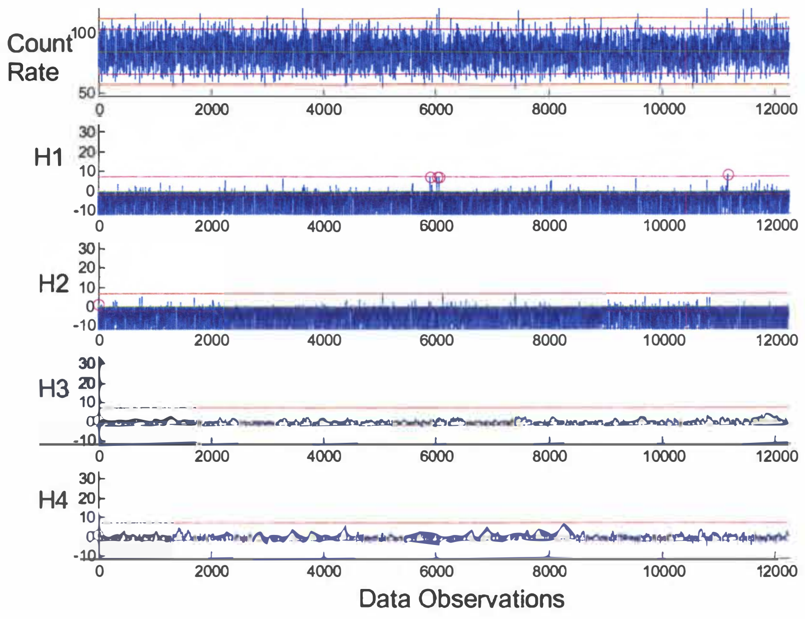


Figure 3.12. RADSiP Response to Humidity Variation, SPRT Result

are exposed to temperatures at the extremes of their operating ranges their responses can be difficult to characterize. Variations in the humidity level of the RADSiP's environment had a null effect on the sensors response. Testing of the RADSiP sensors response to temperature variations found the radiation signal remains unaffected up to 110 degrees Fahrenheit. In some instances, when the RADSiP sensors were exposed to greatly varying temperatures greater than 110 degrees Fahrenheit the radiation signals would experience mean and variance increases. This upper temperature limit is slightly less than the manufacturers specified temperature limit. However due to the rudimentary nature of these environmental experiments, it is difficult to make any suggestions for changes in the temperature operating range. Thus, the environmental experimentation indicates that the RADSiP radiation signal is unaffected by humidity variations, and the radiation signal should remain unaffected by temperature variations less than 110 degrees Fahrenheit.

4 Fault Detection and Isolation, and Regional Anomaly Monitoring Module Methodology

This chapter contains the Fault Detection and Isolation (FDI) system methodology and the Regional Anomaly Monitoring Module (RAMM) methodology that together are used to monitor the CAVIS system. The FDI system was developed to improve the CAVIS system reliability and eliminate unnecessary inventory checks. The system merges advanced statistical algorithms, such as the sequential probability ratio test (SPRT) [Wald 1945], to extract features related to changes in the CAVIS sensors with an expert system that forms a hypothesis of the root cause of any anomaly. The SPRT is a statistical method used to detect changes in the characteristics of a data stream. The SPRT is used to extract features, which are used by an expert system. An expert system is a rule-based system designed to perform functions similar to those of an expert. The RAMM features kernel-averaging techniques to detect the presence of and track the location of external radiation sources in the vicinity of the CAVIS system and other anomalies that affect regions of the CAVIS system. The contents of this chapter include a discussion of the system's software environment, an overview of the FDI system architecture, a discussion of the SPRT and feature extraction system, a discussion of the expert system, and a discussion of the RAMM.

4.1 Software and Computing Environment

The CAVIS monitoring system was implemented in MATLAB (**Mat**rix **Lab**oratory) with Microsoft EXCEL acting as the system database. MATLAB is a software package for numerical computation and graphical applications in scientific and engineering applications. MATLAB can run on many platforms including Windows, Macintosh, Linux, DEC, VAX, and Sun. MATLAB was chosen for implementation due to ease of use, availability of the software, and the ability to communicate with Microsoft EXCEL, the program implemented as the database for the FDI system. Microsoft EXCEL is a popular spreadsheet software that is a component of Microsoft Office. Excel will also run on many platforms such as Windows, Macintosh, etc.

MATLAB is a programmable language that can be implemented at a command prompt or in command files known as m-files. The language features standard programmable constructs such as IF, FOR, WHILE, etc., logical operators such as AND, OR, XOR, etc, string manipulation, file input/output, and graphical user interface abilities. MATLAB is a vectorized language that performs poorly with DO and FOR loops. However, MATLAB can be converted to C code, or compiled as stand alone applications to remedy the poor performance. Dynamic data exchange allows MATLAB to communicate with other Windows applications such as Microsoft Excel.

Microsoft Excel is a popular spreadsheet program used in many applications. Excel is capable of simple arithmetic operation, graphical applications, and data analysis. In this research Excel is used as a database. The program was chosen due to simplicity and ease of communication with MATLAB.

4.2 Overview of the FDI System Architecture

A monitoring and diagnosis system combining feature extraction, by a SPRT and other statistical measures, with an expert system was developed and optimized to monitor the CAVIS system in order to eliminate costly alarm responses and unnecessary inventory checks. Figure 4.1 is a flow chart that illustrates the overall processes of CAVIS data collection, the flow of this data into the CAVIS monitoring system for feature extraction, fault detection and fault isolation, and the final output to a graphical user interface.

CAVIS collects sensor measurements that are analyzed by the SPRT to extract several features from the radiation count rates for each sensor and writes them to a feature set. Additional features are derived from the time history of the count rates and added to the feature set. The expert system analyzes the extracted features and maps them to possible root causes of failure identified for the CAVIS system. Information concerning the root cause is sent to a graphical user interface.

Figure 4.2 is a flow chart specific to the FDI system that completely illustrates the path of data in the CAVIS monitoring system. The time and count rate of the radiation signals for each vault are collected by CAVIS and stored in the Excel Database "Counts".

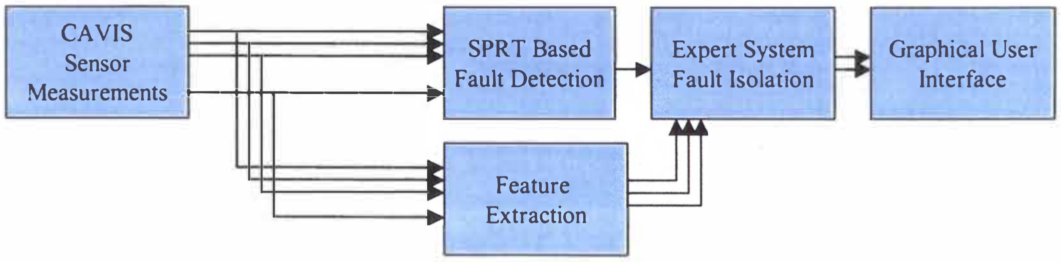

CAVIS Sensor Measurements
SPRT Based Fault Detection
Expert System Fault Isolation
Graphical User Interface
Feature Extraction

Figure 4.1. System Flow Chart

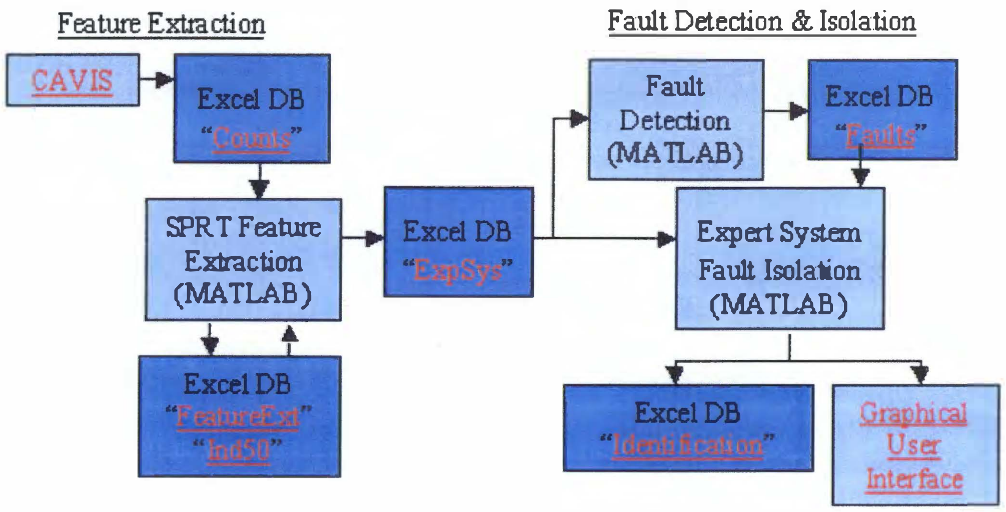

Feature Extraction
Fault Detection & Isolation
CAVIS
Excel DB "Counts"
SPRT Feature Extraction (MATLAB)
Excel DB "ExpSys"
Excel DB "FeatureExt" "Ind50"
Fault Detection (MATLAB)
Excel DB "Faults"
Expert System Fault Isolation (MATLAB)
Excel DB "Identification"
Graphical User Interface

Figure 4.2. CAVIS Monitoring System Data Flow Chart

The sequential probability test analyzes these data, produces alarms if variances or means have changed past a 3-sigma confidence interval, and records the alarms in the Excel Databases "FeatueExt" and "Ind50". The feature extraction module extracts alarms rates and other necessary features from the two databases and records them in the Excel Database "ExpSys" that acts as the working memory for the expert system. The expert system processes the working memory in an attempt to determine the source of the fault. First the expert system detection module compares the values of the extracted features in Excel Database "ExpSys" to a set of tolerances to determine if a fault in any of the radiation signals has occurred. If the extracted features do not exceed the tolerances then no faults have occurred and the system will reinitialize. If the extracted features exceed a set tolerance, then a fault has occurred and the type of fault and the identification number of the culprit sensor will be stored in Excel Database "Faults". Also, the detected fault will initialize the expert system fault isolation module that attempts to determine the source of the fault by analyzing what features were affected by the fault, and which faults had previously occurred. The isolated fault will be stored in an Excel Database "Identification", and information concerning the fault will be sent to a graphical user interface. The Y-12 procedure for the remedy of detected root causes of failure in the CAVIS system is not known, thus the CAVIS monitoring system does not feature an advisory module to assist Y-12 operators in the repair of the CAVIS system.

4.3 Sequential Probability Ratio Test (SPRT)

This section describes the SPRT that is used to analyzes the CAVIS radiation signals. The application of the SPRT in the CAVIS monitoring system is the work of T Jay Harrison and additional information concerning the application can be found in his thesis [Harrison 04]. The description of the SPRT is included in this work for completeness.

The SPRT is a statistical test developed by A. Wald in 1945 that is capable of monitoring statistical properties of a Gaussian distribution. The SPRT determines if an input data stream was generated by the expected, normal Gaussian distribution characterized by an expected mean and variance, or if there is a greater probability that the data stream comes from some alternate distribution characterized by a shifted mean and/or altered variance. If the input comes from the alternate distribution, the SPRT will generate an appropriate alarm. This technique is capable of monitoring two attributes of the radiation distribution: mean and variance, in contrast to previous techniques [Bell], which only monitored the mean. By monitoring two attributes of the radiation distribution, the SPRT-based system is capable of identifying additional operational faults.

The SPRT evaluates the radiation signal for four alternative hypotheses; increase (H_1) or decrease (H_2) in signal mean by an amount M and an increase (H_3) in variance of amount V^+ or a decrease (H_4) in signal variance by amount V^-. If the signal has a greater likelihood of having been a member of an alternative hypothesis than having been a member of the null hypothesis, the SPRT for that particular hypothesis alarms. For every data observation, the SPRT calculates a likelihood that the data stream belongs to the original distribution or one of the four alternative hypotheses. That likelihood is given by equation 4.1

$$L_n = \frac{P(Y_n \mid H_j)}{P(Y_n \mid H_0)}, j = 0:4 \qquad \text{Eq. 4.1}$$

where $P(Y_n|H_x)$ is the probability of an observed sequence Y_n given that H_x is true. The radioactive decay process is Poisson and then approximated by a Gaussian, thus the likelihood of each of the alternative hypotheses is given by equation 4.2-4.5.

$$L_1 = \exp\left[\frac{-1}{2\sigma^2}\sum_{k=1}^{n} M(M - 2y_k)\right] \qquad \text{Eq. 4.2}$$

$$L_2 = \exp\left[\frac{-1}{2\sigma^2}\sum_{k=1}^{n} M(2y_k + M)\right] \qquad \text{Eq. 4.3}$$

$$L_3 = (V^+)^{-n/2} \exp\left[\frac{-1}{2\sigma^2}\left(\frac{1-V^+}{V^+}\right)\sum_{k=1}^{n} y_k^2\right] \qquad \text{Eq. 4.4}$$

$$L_4 = (V^-)^{-n/2} \exp\left[\frac{-1}{2\sigma^2}\left(\frac{1-V^-}{V^-}\right)\sum_{k=1}^{n} y_k^2\right] \qquad \text{Eq.4.5}$$

The SPRT equations, given by equation 4.6-4.9, are defined by taking the natural log of the likelihood equations.

$$SPRT_1 = \left[\frac{-1}{2\sigma^2}\sum_{k=1}^{n} M(M - 2y_k)\right] \qquad \text{Eq. 4.6}$$

$$SPRT_2 = \left[\frac{-1}{2\sigma^2}\sum_{k=1}^{n} M(2y_k + M)\right] \qquad \text{Eq. 4.7}$$

$$SPRT_3 = \left(-\frac{n}{2}\ln V^+\right) + \left[\frac{-1}{2\sigma^2}\left(\frac{1-V^+}{V^+}\right)\sum_{k=1}^{n} y_k^2\right] \qquad \text{Eq. 4.8}$$

$$SPRT_4 = \left(-\frac{n}{2}\ln V^-\right) + \left[\frac{-1}{2\sigma^2}\left(\frac{1-V^-}{V^-}\right)\sum_{k=1}^{n} y_k^2\right] \qquad \text{Eq. 4.9}$$

The parameters for these alternative hypotheses (H_1-H_4), M, V^+, and V^-, are defined by the mean μ and variance σ^2 of the radiation signal, which are equal due to the mean-variance equality inherent in Poisson processes such as radioactive decay. The amount by which the mean shifts up or down, M, is set for three standard deviations. This mirrors a +/- 3σ band for the desired 99+% confidence interval. That is, if for a sequence of data points the SPRT alarms with a mean up or mean down indication, the alarm has a 99+% probability of *not* being a false alarm. The factors by which the variance increases V^+ or decreases V^-, are set to reflect the shifted means. Because the mean μ equals the variance σ^2, the variance for the increased mean hypothesis is $\mu + M = \mu + 3\sigma$. Thus, the ratio of the new variance to the old variance is $(\mu + M)/\mu = 1 + 3/\sigma$ for a variance increase and $1 - 3/\sigma$ for a variance decrease.

The results of the SPRT equations are compared to two parameters, ln(A) and ln(B), to determine if the radiation distribution has changed. A and B are defined by

$$A = \frac{\beta}{1-\alpha} \qquad B = \frac{1-\beta}{\alpha} \qquad \text{Eq. 4.10}$$

where the parameters α and β are the set false (Type I) and missed (Type II) alarm rates, respectively. The sensitivity of the SPRT depends on these false- and missed-alarm probabilities. This research sets α and β at 0.1% and 10%, respectively. The low value for α reflects the need to minimize the number of false alarms – roughly 1 false alarm per 1000 data observations, or 99.9% accuracy in sounding alarms. The value for β is set arbitrarily at 10% based on the assumption that if an actual alarm condition occurs but does not trigger an alarm, it will trigger an alarm at a future time step.

If the result of any SRPT equation is greater than ln(B), the SPRT alarms for that hypothesis then resets to 0 and starts a new collection sequence. If the result of any SPRT equation is less than ln(A), the SPRT resets to 0 and starts a new collection sequence. For more information of the feature extraction module see Harrison.

4.4 Feature Extraction Module

This section describes the Feature Extraction Module that is used to extracted features from the CAVIS radiation signals. This module in the CAVIS monitoring system is the work of T Jay Harrison and additional information concerning the application can be found in his thesis [Harrison 04]. The discussion of the Feature Extraction Module is presented here for completeness.

The feature extraction system (FES) acts as a pre-processor for the collected data. The FES acts as custodian and interpreter for the count rate database and extracts information useful to the expert system. It does so using the SPRT to calculate, track, and communicate trends within the collected count rates. The FES sums the number of all alternative hypothesis alarms over the last 100 and 1000 data points and tracks the interval since the last alarm for each hypothesis. The FES also extracts the variance of the last five and fifty data observations and performs a test on the sign of the residuals. This is commonly known as Run Test 2: nine consecutive points same side of average [Western Electric 1958, Nelson 1984]. Two other features used by the expert system include the current count rate and a built-in system status signal from the CAVIS hardware. The FES supplies the current count rate directly to the expert system database without processing, but the FES does not extract the system status signal, which is supplied directly from CAVIS. The expert system uses these extracted features to isolate and diagnose system faults.

4.5 Expert System

An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that may require significant human expertise [Feigenbaum 1982]. When applied to the CAVIS system, an expert system may reduce the number of unnecessary inventory checks by determining alternative explanations for the alarm, besides the removal of SNM. This will allow workers to investigate the alternate explanation first, which will save time, money, and possibly radiation exposure. For example, the radiation sensors used in the CAVIS system may display some dependence on temperature. This dependence may cause "spikes" or "ramp up" behavior in their reading of the reactivity of the source. Without an expert system the response to this behavior may be an inventory check. However, an expert system might recognize the problem as an increase in temperature. The expert system would then make an alternate suggestion to check the temperature of the storage room rather than performing an inventory check. Recall that the expert system can only make this suggestion if it has a rule base that incorporates knowledge concerning the functionality of the radiation sensors. Thus, for an expert system to work properly it requires a complete understanding of every component of the system it monitors and contain the heuristic knowledge of an expert. The following sections describe the implementation of the fault detection and isolation expert system in the CAVIS monitoring system.

4.5.1 Expert System Fault Detection

The expert system compares the extracted features values with a set of tolerances to detect any faults that have occurred. The tolerances are set to ensure a 99% or greater confidence interval in the faulted state of the feature when possible. The tolerances for the SPRT alarms are set according to the results of a parametric study, the result of which can be seen in Harrison. The tolerances for the remaining features are set using simple probabilistic calculations. Table 4.1 contains a complete listing of the tolerances for the extracted features.

If any value of the extracted features exceeds its tolerance, then a fault is generated for that sensor. For example, if the radiation signal for a particular sensor experiences five SPRT mean shift up alarms in 1000 data observations, then feature 1 is faulted for the particular sensor. A faulted feature for a sensor implies there is a 99% or greater confidence that the sensors radiation signal has experienced a "change in state" or CAVIS has experienced some failure.

The tolerances for the SPRT alarms have bases from different conceptual and experimental sources. The maximum number of SPRT alarms in the last 100 or 1000 data points is set through experiments and theory. By counting faults over a fixed

Table 4.1. Fault Detection Tolerances for Extracted Features.

Feature	Fault Tolerance: Feat. faulted if ...
F1: Mean Shift Up SPRT (1000 obs.)	5 SPRT MSU alarms in 1000 obs.
F2: Mean Shift Down SPRT (1000 obs.)	5 SPRT MSD alarms in 1000 obs.
F3: Variance Shift Up SPRT (1000 obs.)	5 SPRT VSU alarms in 1000 obs.
F4: Variance Shift Down SPRT (1000 obs.)	5 SPRT VSD alarms in 1000 obs.
F5: Successive Mean Shift Up	MSU SPRT alarms for 2 cons. data obs.
F6: Successive Mean Shift Down	MSD SPRT alarms for 2 cons. data obs.
F7: Successive Variance Shift Up	VSU SPRT alarms for 2 cons. data obs.
F8: Successive Variance Shift Down	VSD SPRT alarms for 2 cons. data obs.
F9: Run Test 2	Nine cons. data obs. on same side of mean
F10: Variance of last 50 points	Variance of last 50 data obs. equals zero
F11: Variance of last 5 points	Variance of last 5 data obs. equals zero
F12: Mean Shift Up SPRT (100 obs.)	3 SPRT MSU alarms in 100 obs.
F13: Mean Shift Down SPRT (100 obs.)	3 SPRT MSD alarms in 100 obs.
F14: Variance Shift Up SPRT (100 obs.)	3 SPRT VSU alarms in 100 obs.
F15: Variance Shift Down SPRT (100 obs.)	3 SPRT VSD alarms in 100 obs.
F16: Current Count Rate	Current count rate equals zero
F17: Communication Status of CAVIS	Communication status of CAVIS is "bad"

number of data points, this effectively measures the alarm rate. Experimentally, an unfaulted distribution will produce a total number of faults as tabulated in a parametric study [Harrison 04]. The expected total alarm rate for means ranging from 25 to 145 is around $2 * 10^{-4}$, while the alarm rates for hypotheses 1 and 2 are around $0.75 * 10^{-4}$ and hypotheses 3 and 4 are around $0.25 * 10^{-4}$. However, if the fault tolerances were set at levels that low, one random data point outside the 3σ bands would be sufficient to cause an SPRT alarm. The two-consecutive-alarms metric is intended to account for the decreased sensitivity brought about by increasing the tolerated fault rate. This therefore serves as a surrogate alarm threshold for finding drifting or wildly varying systems.

The remaining features tolerances were set according to probabilistic calculations. Since the SNM count rates range between 25-145, the radiation signal can be approximated with a Gaussian distribution. Thus, the likelihood of any particular count occurring is

$$P(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \qquad \text{Eq. 4.11}$$

where P is the probability of the count, μ is the mean of the count rate, σ is the standard deviation of the count rate ($\sigma = \mu^{1/2}$), and x is the count rate for which probability is to be determined. The tolerances for feature 10 (variance of last 50 observations), feature 11 (variance of last 5 observations), and feature 16 (current count rate) are set using equation 4.11 to ensure a 99% confidence.

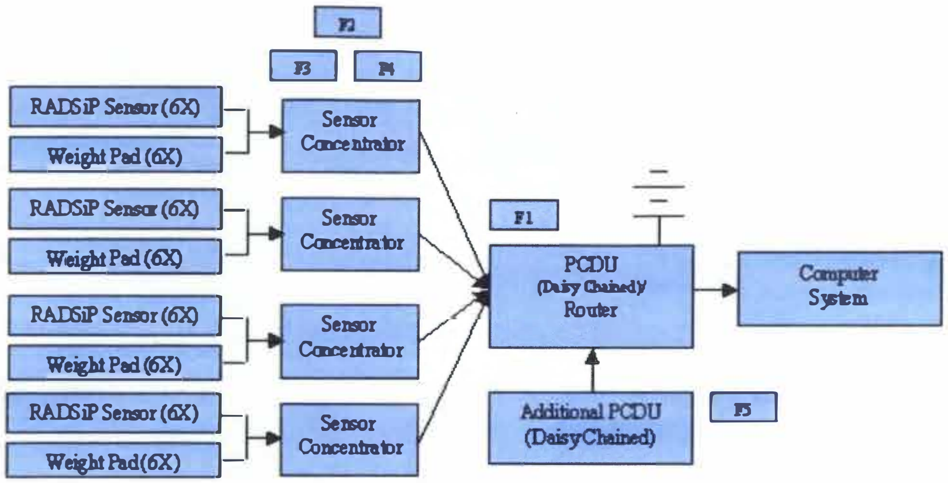
When a fault is generated, the time of the fault, the culprit sensor, and the type of fault are recorded in a database. This database is used as a log to keep a record of all fault occurrences and as a supplement to the working memory of the expert system. Also, a detected fault initializes the fault isolation portion of the expert system.

4.5.2 Expert System Fault Isolation

When a fault is detected in an extracted feature, the expert system attempts to isolate the root cause of the fault using its programmed rule base knowledge. The rule base knowledge is a collection of IF/THEN rules containing information mapping the feature space to the fault space. The rule base contains a hierarchal collection of root causes and alarmed features that may be “characteristic” of certain faults. The hierarchal rule base allows physical component failures to be isolated by comparing the number of failed sensors to the total number of sensors that correspond to a certain component. An example rule is "if all the RADSiP sensors that correspond to a certain component fail, the fault exists in the component rather than in each of the sensors." Figure 4.3 further illustrates CAVIS component failure isolation using a hierarchal knowledge base.

In addition to the hierarchal rule base, the expert system can isolate faults based on what features are alarmed for a particular sensor. In many instances, certain faults may be characteristic of certain failures that a sensor may experience. For example, a signal variance. Thus, the root cause of an increase in variance fault could be a loose electrical connection for the culprit sensor.

CAVIS testing has identified a number of abnormal behaviors that can occur in the reported radiation signals. These abnormalities are zero count rate (ZCR), stuck



F1 – Fault exists in PCDU or Router if all common sensors alarm
F2 – Fault exists in Sen. Con. if all common sensors alarm
F3 – Fault exists in Sen. Con. Processor Board if ½ common sensors alarm
F4 – Fault exists in Sen. Con. Communication Module if ¼ common sensors alarm
F5 – Fault exists in last PCDU or Router in chain if all sensors alarm in preceding chain.

Figure 4.3. Hierarchal Fault Isolation

count rate (SCR), count rate mean shift up and down (MSU/MSD), count rate variance shift up and down (VSU/VSD), and spike in count rate (SpS). Faults in certain features are characteristic of all of these abnormalities in the radiation signal. In addition, all of these abnormalities can be mapped to a root cause. Thus, it is possible to associate a set of faulted features with a root cause. Table 4.2 contains a mapping of the known root causes to a set of alarming features. The table contains the logic necessary to identify the root cause. The character ~ (not) in the table indicates that the following features must be in a non-alarmed state.

The logic contained in table 4.2 is used as the rule base knowledge of the expert system to isolate root causes in CAVIS. The knowledge base enables the detection of characteristic faults such as dead or stuck sensors, which affect individual sensors, and

Table 4.2. **Map of Feature Space to Root Cause**

Root Cause Symptom	FDI Detection Logic (Feature)
Dead Sensor	11&16
Stuck Sensor	11&~16
Temp. Inc. Sensor Drift	1&3&12& 14
External Radiation Source	1&3&12&14 or (RAMM)
Removal of SNM	~1&2&~4&~11&~12&13&~14
Loose Elec. Connection	~1&~2&3&~12&~13&14&~16
Dead Sensor PCDU	11&16 (All Sen. PCDU)
Stuck Sensor PCDU	11&~16 (All Sen. PCDU)
Loose Elec. Conn. PCDU	~1&~2&3&~12&~13&14&~16 (All Sen. PCDU)
Dead Sensor. Concentrator.	11&16 (All Sen. Conc.)
Stuck Sensor Concentrator.	11&~16 (All Sen. Conc.)
Loose Elec. Conn. Sen. Conc.	~1&~2&3&~12&~13&14&~16 (All Sen. Conc.)
Dead Sen. Conc. Processor Board	11&16 (All Sen. Conc. Board)
Stuck Sen. Conc. Processor Board	11&~16 (All Sen. Conc. Board)
Elec. Conn. Sen. Processor Conc. Board	~1&~2&3&~12&~13&14&~16 (All Sen. Conc. Board)
PCDU Power Failure	17 (All Sen. PCDU)
Sen. Conc. Unplugged	17 (All Sen. Conc.)
Sen. Conc. Board Removal (Processor)	17 (All Sen. Conc. Board)
Sen. Conc. Board Removal (Interface)	11&16&17 (All Sen. Conc. Board)
Equipment Vault Stack Collision	Regional Anomaly Monitoring Module

hierarchical faults that affect CAVIS component such as the PCDU or the sensor concentrators.

4.6 Regional Anomaly Monitoring Module (RAMM)

The Regional Anomaly Monitoring Module (RAMM) is an additional module used to detect anomalies that affect regions of the CAVIS system. Several abnormalities or operating conditions exist that can induce "change-in-state" behavior in a region of the CAVIS system, rather than changes-in-state for individual sensors or the system as a whole. These anomalies include the presence of external radiation sources in the CAVIS storage area or a sensor vault pile impacted by a piece of heavy equipment. The module is capable of detecting root causes that affect regions of the CAVIS system, rather than individual radiation sensors by mapping the behavior of neighboring sensors to form a neighborhood score. The neighborhood scores are analyzed to detect and isolate regional anomalies. The ability to detect regional faults may help to avoid unnecessary inventory checks, thus saving cost and eliminating unnecessary radiation exposure.

Kernel smoothing is a non-parametric technique used to estimate the probability density function of a data set [Wand 1995]. In this application the data are the radiation detector count rates observed at different locations in the storage facility. Kernels are used to smooth the discrete measurements resulting in an approximation of the underlying radiation field. Kernel smoothing is implemented to detect and identify regional radiation disturbances.

Each radiation sensor is expected to produce a specific count rate that is calculated as an average of a series of count rates collected under normal operating conditions. The difference between a sensor's actual count rate and its expected count rate is called a residual. The residual values of the CAVIS radiation sensors are placed in a three dimensional array according to their physical location. Under normal conditions, these residuals will have a Gaussian distribution around a mean value of zero and have a variance equal to the square root of the mean count rate.

Kernel smoothing is used to map the behavior of the sensors in close proximity to form a neighborhood score. A change in the neighborhood scores for a region of CAVIS indicates that the sensors in the region experienced the effect of the same root cause. The maximum of the neighborhood scores will occur near the root cause, enabling the location of the anomaly.

4.6.1 RAMM Methodology

The development of the RAMM module involves the implementation of a neighborhood system that monitors regions of the CAVIS system for changes in state. It is difficult for CAVIS to isolate an anomaly affecting several sensors in a region because CAVIS focuses on sensors not their interactions. A neighborhood system features a kernel based density estimation technique that reduces the noise or variability of the sensor readings, to form neighborhood scores. The neighborhood score for a sensor is calculated based on the Euclidean distance between the sensors, the behavior or count rate of the other sensors in the warehouse, and a kernel width. Kernel smoothing of the detectors residuals results in neighborhood scores near zero unless some abnormality exists in the CAVIS system. Recall that the residuals are the difference between the sensor count rate and the expected count rate (mean). Therefore, residuals have a mean of zero and a variance equal to the square root of the radiation signal mean. Thus, the implementation of a neighborhood system allows for the detection of abnormalities that affect regions of CAVIS. Figure 4.4 illustrates the process of the RAMM.

CAVIS collects a count rate for every radiation sensor and the RAMM places the residual (difference from the normal mean) of the count rate in a three-dimensional array based on the sensors physical location. The sensors are contained in concrete vaults in 4x5 sensor arrangements that are then stacked on top of each other. The neighborhood

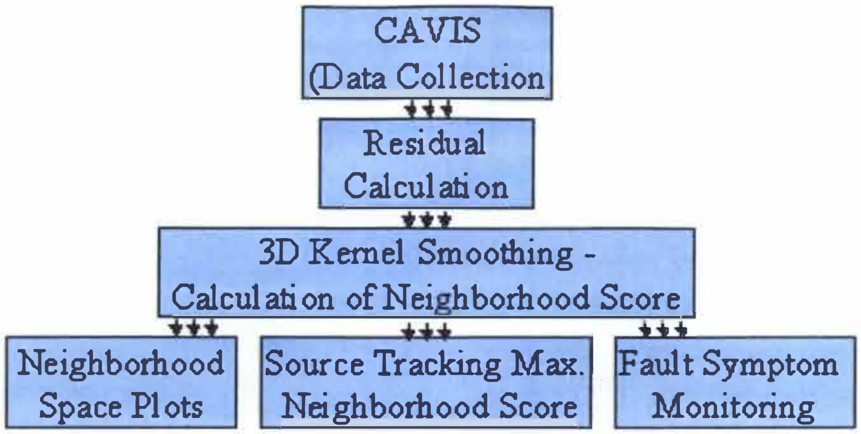


Figure 4.4. Flow Chart of Regional Anomaly Monitoring Module

score for each sensor is then determined by performing kernel smoothing of the residual values in the three dimensional array such that

$$NeighborhoodScore(x) = \sum_{i}^{x_i} \mathrm{Res}(x_i) * K_h(x, x_i) \qquad \text{Eq. 4.12}$$

where x is the sensor location in question, x_i is the sensor location around which the kernel is placed, K_h is a kernel function, and Res is the residual of the sensor x_i. A Gaussian kernel, with magnitude equal to the residual value, is placed at each sensor location and the neighborhood score for each individual sensor location is determined by summing the contributions from every sensor in the warehouse. Kernel smoothing is used to reduce the variance due to the random nature of radioactive decay. Although the mean of the residuals is zero, the value of any particular residual at any particular data observation can vary between $0 \pm 2\sigma$ with 95% confidence. Thus, it is necessary to smooth the residuals to force the neighborhood score towards the mean.

The kernel width acts as a regularization parameter and must be chosen to optimally reduce the variance of the neighborhood function. As the kernel width becomes larger, the solution will be smoother and the neighborhood score for every sensor will be pushed towards the mean of zero. If the kernel width is small, less smoothing (averaging) occurs and the model may overfit the data. As the width goes to zero the neighborhood score for each sensor will simply be its residual. Thus, it is necessary to determine the optimal kernel width to optimize the variance reduction. In this application the optimal kernel width was found to be dependent on the Euclidean distance between the sensors in the CAVIS vault pile.

In the event of some regional anomaly, the residuals of the sensors will smooth to a set of neighborhood scores with values larger than zero, which will enable the detection and isolation of the origin of the anomaly. Monitoring the maximum value of the neighborhood scores will enable regional anomaly detection. The maximum neighborhood score should vary around zero in the absence of a regional anomaly and increase to some value greater than zero in the presence of a regional anomaly. Thus, placing monitoring bands, which are determined experimentally, around the maximum neighborhood score enables regional anomaly detection.

When a regional anomaly is detected, its origin will be in the vicinity of the maximum neighborhood score, thus enabling the isolation of the location of the anomaly. Three-dimensional plots of the neighborhood scores, with the color in the plot representing the neighborhood score, will allow the visualization of the regional anomaly's location and the visual image of the anomaly will change with time allowing a moving anomaly to be tracked. The plots generated by the RAMM feature a color scaling such that white or no color indicates no regional anomaly and a dark color in a region indicates that some regional anomaly is present.

The three plots generated by the RAMM are: 1) the current neighborhood scores, 2) the neighborhood scores when the maximum neighborhood over a time interval occurred, and 3) a time history plot of the five most recent neighborhood scores weighted by an exponential function to give more weight to the most recent scores. The plot of the current neighborhood scores describes what is currently occurring regionally in the CAVIS system.

The plot of the neighborhood scores when the maximum score occurred is useful for locating and keeping record of impacted CAVIS vault stacks. The radiation signal spikes induced by collisions are short lived and usually last only one data observation. The maximum score plot will preserve the neighborhood scores when the collision occurred allowing the impacted vault pile to be identified even after the spiked sensor behavior has pasted.

The time history plot illustrates the past behavior of the neighborhood scores and is useful in tracking external radiation sources that may be moving in the CAVIS storage area. If the external source is moving the time history plot will contain a "tail" that reveals where the source has been and the direction it is traveling. Weighting the five most recent sets of neighborhood scores by Eq. 4.13 creates the data plotted in the time history plot.

$$TimeHistory = \frac{\sum_{t=1}^{5} NS^{t} e^{\frac{-t-1}{\delta}}}{\sum_{t=1}^{5} e^{\frac{-t-1}{\delta}}} \qquad \text{Eq. 4.13}$$

In this research δ is set to three and NS^{t} are the neighborhood scores at the specified time interval t, where NS^{1} is the current neighborhood scores, NS^{2} is the previous neighborhood scores, etc. The results and plots generated by the RAMM allow Y-12 personal the ability to visualize changes-in-state of regions of the CAVIS system and detect anomalies that affect regions of CAVIS.

5 Results

The CAVIS monitoring system is capable of detecting and isolating CAVIS faults including numerous types of sensor failures, component failure, and environmental effects. It was tested on several data sets including data sets collected over several months of operation at Y-12 and data sets containing induced failures collected at the University of Tennessee. An Regional Anomaly Monitoring Module has also been developed to investigate the presence of root causes of false alarms for regions of the CAVIS system. This module is capable of detecting anomalies that affect regions of the CAVIS system such as external radiation sources in the CAVIS storage area and spiked sensor vault stacks induced by heavy equipment collisions with the CAVIS storage vaults.

5.1 Data Abnormality Monitoring - Sensor Malfunction

The following sections contain several examples of the CAVIS monitoring system analyzing CAVIS data sets containing some data abnormality for a particular sensor. The examples illustrate the ability of the CAVIS monitoring system to detect various RADSiP sensor failures that have been found to occur in the CAVIS system. The data sets feature twenty independent radiation signals that correspond to a particular sensor concentrator. Each data set contains a single sensor failure that may result in a CAVIS alarm or an unmonitored state of the SNM. An unmonitored state of the SNM means the CAVIS system will not alarm provided the failure does not result in reported count rates outside of a 99% confidence interval. The sensor failures that are shown are 1) Dead RADSiP sensor, 2) Stuck RADSiP sensor, and 3) Drifting RADSiP sensor induced by temperature extremes. The CAVIS monitoring system is able to detect and isolate each sensor malfunction and is able to produce an alternate response to any CAVIS alarm, which will alleviate any unnecessary inventory check.

5.1.1 Sensor Malfunction – Dead Sensor

The following example illustrates the CAVIS monitoring systems ability to detect and isolate a "dead" RADSiP radiation sensor. A dead sensor is a sensor that has experienced some failure that results in a repeatedly reported count rate of zero. The root cause of this abnormality can be failure of the RADSiP electronics, an unplugged or damage wire connection between the RADSiP and sensor concentrator, or a failure of one of the components the sensor is connected to in the sensor hierarchy. The data set analyzed here was collected at the University of Tennessee and RADSiP sensor 1 1 3 was unplugged from its sensor concentrator at data observation 187 and plugged back into the concentrator at data observation 935 to induced the failure. The data was collected at five-second intervals for 2 hours and 45 minutes to generate 2000 data observations. The described dead sensor data set is presented in figure 5.1.

The data set shown in figure 5.1 was analyzed by the CAVIS monitoring system for abnormal behavior. In this instance, the data anomaly is obvious from a plot of the data. Nonetheless it is necessary for the CAVIS monitoring system to detect and isolate

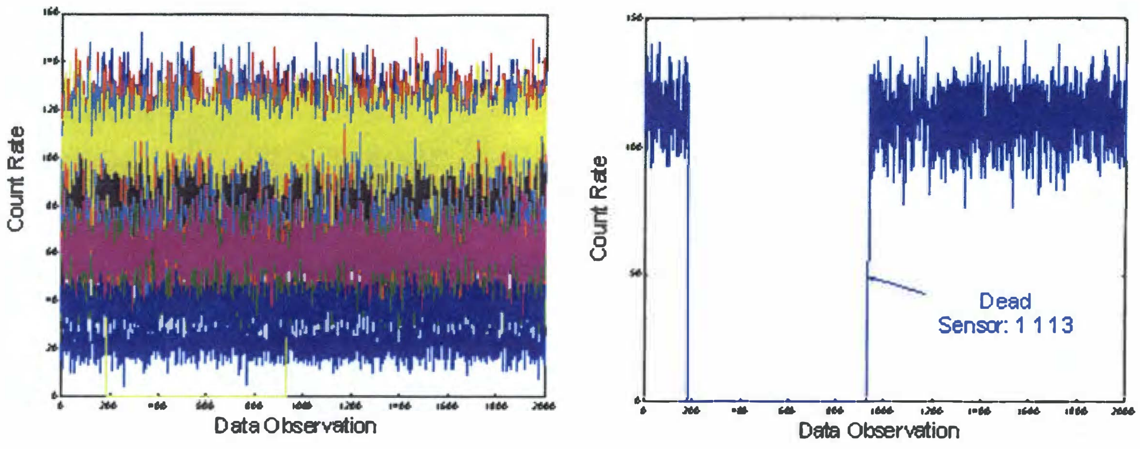


Figure 5.1. Dead Sensor Anomaly Data Set

the failure, as it is impractical to continually visually inspect data collected by CAVIS. The data was analyzed by the CAVIS monitoring system to detect any abnormalities in the count rates. The warnings and alarms generated by the CAVIS monitoring system are presented in table 5.1.

The CAVIS monitoring system was able to detect the dead CAVIS sensors in the data set. As discussed the sensor failed at data observation 187 and the CAVIS monitoring system was able to instantly generate a dead sensor warning. The warning is generated because the current count rate feature is in a faulted state. At data observation 191 the CAVIS monitoring system generated a dead sensor alarm. The alarm is generated because both the variance of the last five-observation feature and current count rate feature are in a faulted state. The dead sensor failure ends at observation 935 when the sensor was plugged into the sensor concentrator. The CAVIS monitoring system recognized that the sensor malfunction had been corrected and generated a "sensor is no longer dead" message. It is necessary for the CAVIS monitoring system to recognize this

Table 5.1. CAVIS monitoring system analysis of Dead Sensor Anomaly Data Set

Ind.	Problem	Logic
187	Warning: The RADSiP sensor may be dead: 1 1 13	Zero count rate for the sensor
191	The RADSiP sensor is dead: 1 1 13	Zero count rate for the sensor for 5 cons. obs.
935	The RADSiP sensor is no longer dead: 1 1 13	The sensors no longer has a zero count rate

failure as it may result in an unnecessary inventory check of the SNM. The CAVIS monitoring system is able to detect and isolate dead sensors and will provide an alternative response, which should alleviate the possibility of unnecessary inventory checks. It should be noted that dead sensor warnings and alarms will only be generated if a radiation signal change-in-state is not correlated with a weight signal change-in-state for the corresponding weight sensor in the SNM canister. In the event of a correlated change updates to the CAVIS monitoring system will alarm with a Removal of SNM alarms.

5.1.2 Sensor Malfunction – Stuck Sensor

The following example illustrates the CAVIS monitoring systems ability to detect and isolate a "stuck" RADSiP radiation sensor. A stuck sensor is a sensor that has experienced some failure that results in a repeatedly reported non-zero count rate. The data set analyzed here is a 2,000 data observations portion of a data set collected at Y-12 from May 3, 1999 at 14:00:56 to January 6, 2001 at 12:21:20 with a one-hour update rate. The CAVIS data sets collected at Y-12 feature a one-hour update rate to limit the size of the data set. It should be noted that the techniques used by the CAVIS monitoring would require a quicker update rate to ensure the safe storage of the SNM. With this long of an update rate the SNM could be removed for a few hours before the system would detect the induced change-in-state of the SNM. If an individual knew the time CAVIS logged the data they could completely remove the SNM shortly after the attribute data was logged and have a full hour to flee before CAVIS or the CAVIS monitoring system would alarm. An update rate of around one minute is more appropriate to securely monitor the SNM. Nonetheless, the CAVIS monitoring system is capable of detecting and isolating sensor malfunction in the data set.

It is not possible to know what root cause induced the abnormal behavior in the radiation signal because the data set were collected at Y-12; however testing at UT has found that stuck sensor behavior can be caused by failure of one of the components the sensor is connected to in the sensor hierarchy or electronic failure of the RADSiP. This particular data set contains only one stuck sensor thus the root cause cannot be a component failure. In the data set, sensor 1 1 7 becomes stuck at about data observation 400 and is repaired or the failure ceases near data observation 1350. The described stuck sensor data set is presented in figure 5.2.

The data set shown in figure 5.2 was analyzed by the CAVIS monitoring system for abnormal behavior. The warnings and alarms generated by the CAVIS monitoring system are presented in table 5.2.

The CAVIS monitoring system was able to detect the stuck CAVIS sensor in the data set that became stuck at data observation 371. The sensor stuck alarm was generated because the variance of the last five data observations feature was in a faulted state while

Table 5.2. CAVIS monitoring system analysis of Stuck Sensor Anomaly Data Set

Ind.	Problem	Logic
371	The RADSiP sensor is stuck: 1 1 7	The sensor had the same count rate for 5 cons. obs.
1365	The RADSiP sensor is no longer stuck: 1 1 7	The sensor no longer has the same count rate

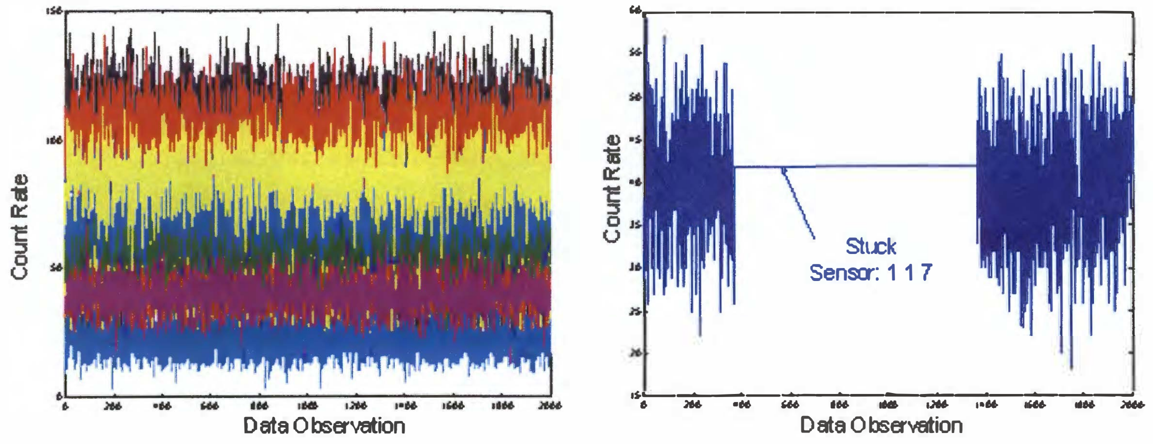


Figure 5.2. Stuck Sensor Anomaly Data Set

the current count rate feature was not faulted. The CAVIS monitoring system recognized the failure had been corrected at data observation 1365 when the sensor was repaired or ceased to fail, and generated a "no longer stuck" message for the sensor. Due to a lack of variance testing in the original CAVIS system, a stuck sensor failure would result in an unmonitored state for the SNM. However, the CAVIS monitoring system is able to detect and isolate the stuck sensors, suggest a response of component repair of replacement, and should alleviate the possibility of an unmonitored state of the SNM.

5.1.3 Sensor Malfunction – Drift in Signal Induced by Temperature Extremes

The following example illustrates the CAVIS monitoring system's ability to detect and isolate a "drift" in the RADSiP radiation signal. Drifts in the RADSiP radiation signals can be induced by extreme environmental conditions, or RADSiP electronic failure. The drift presented in this example is an increase in the mean of the signal induced by extreme environmental conditions. The data set analyzed here is a portion of a data set collected with the UT environmental chamber. The RADSiP sensor 1 1 4 was exposed to a varying temperature of 90-140 degrees Fahrenheit that induced a mean shift up behavior in the sensors radiation signal. The analyzed data set represents four hours and 10 minutes of testing with an update rate of 5 seconds to generate a total of 3000 data observations. Several RADSiP sensors were exposed to the same conditions as sensor 1 1 4, however due to the peculiar response of the sensors to temperature variation it is the only sensor in the set that demonstrated a temperature dependence. The described drifting sensor data set is presented in figure 5.3.

A visual inspection of the data set reveals that sensor 1 1 4 begins to experience a shift up in mean near data observation 900 that remains throughout the set. The count increases from a mean of about 70 at the beginning of the set to a mean of near 90 at the end of the set. The data set shown in figure 5.3 was analyzed by the CAVIS monitoring system for abnormal behavior. The warnings and alarms generated by the CAVIS monitoring system are presented in table 5.3.

The CAVIS monitoring system was able to detect the drifting CAVIS sensor in the data set. However, it was not possible for the system to diagnose the root cause of temperature-induced failure because only one RADSiP sensor exhibited the abnormal behavior and temperature data was not presented to the system. Updates to the CAVIS monitoring system may incorporate environmental data in the working memory

Table 5.3. CAVIS monitoring system analysis of Drifting Sensor Anomaly Data Set

Ind.	Problem	Logic
1354	Warning: The RADSiP sensor may be drifting: 1 1 4	The sensor is experiencing SPRT alarms
1369	The RADSiP sensor is drifting: 1 1 4	The sensor is experiencing SPRT alarms

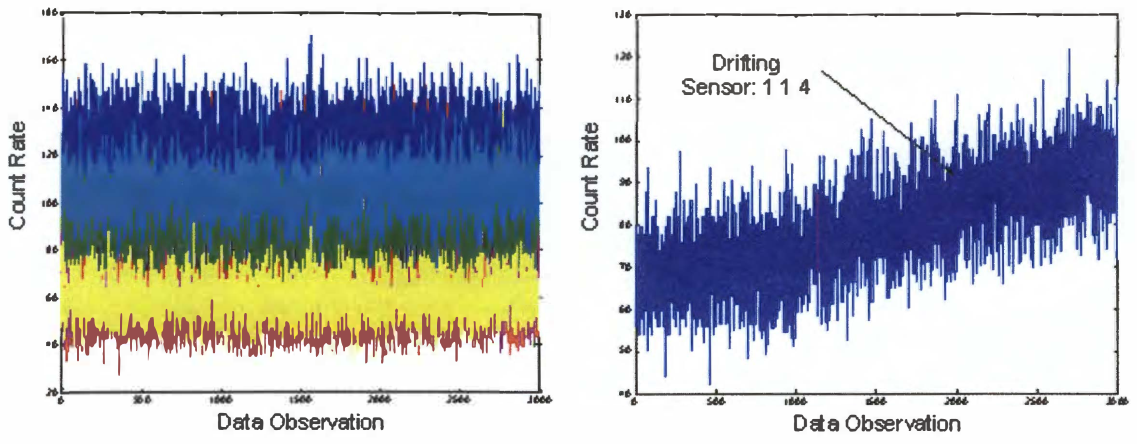


Figure 5.3. Drifting Sensor Anomaly Data Set

of the system. In the example, a drifting sensor warning was generated at data observation 1354 and the warning was upgraded to an alarm at data observation 1369. A drifting sensor warning is generated when the SPRT mean shift features are faulted for either the last 100 or 1000 data observations. The warning is raised to alarm status when the SPRT mean shift features are faulted for both the last 100 and 1000 data observations.

The visual inspection of the radiation signal revealed that the drift began near data observation 900, however it was not detected by the CAVIS monitoring system for approximately 400 data observations or about 30 minutes. The time of detection can be attributed to the magnitude of the drift in the data, the mean and standard deviation of the non-drifting radiation signal, and the set value of the SPRT shifted mean hypothesis. Before the drift began, sensor 1 1 4 radiation signal had a mean of approximately 70, and because the signal is a Poisson a standard deviation of $\sqrt{70}$. The SPRT mean shift hypothesis is set to the mean plus or minus three standard deviations; thus when the mean of a drifting data sequence is equal to the original mean plus or minus 1.5 standard deviations the SPRT will alarm [Harrison 04]. In this example, the SPRT should alarm at a count rate $70 + 1.5 * \sqrt{70} = 82.55$. Visual inspection of the data reveals that the magnitude of the drift resulted in a drifting count rate mean of approximately 83 at about data observation 1400. As expected, this is near the data observation the CAVIS monitoring system detects the drifting sensor. Thus, the time of drift detection depends magnitude of the drift, the statistical properties of the data stream before the drift occurs, and the set value of the mean shift hypothesis.

The count rate for drifting sensors may fall outside of a 99% confidence interval resulting in CAVIS alarms and unnecessary inventory checks of the SNM. The ability of the CAVIS monitoring system to detect drifting CAVIS sensors should alleviate the possibility of unnecessary inventory checks by providing Y-12 personal with an alternate response to CAVIS alarms.

5.2 Data Abnormality Monitoring - Component Failure

The following sections contain several examples of the CAVIS monitoring system analyzing CAVIS data sets containing some component failure. The examples illustrate the ability of the CAVIS monitoring system to detect component failures that are known to occur in the CAVIS system. The data sets feature twenty independent radiation signals that correspond to a particular sensor concentrator. Each data set contains a single component failure that results in a data abnormality for each RADSiP sensor that is common to the component. The component failures that are shown are 1) stuck sensor concentrator, and 2) a failed sensor concentrator communication board. The CAVIS monitoring system is able to detect and isolate each component failure and is able to produce an alternate response to any CAVIS alarm generated due to the failure, which will alleviate any unnecessary inventory check.

5.2.1 Component Failure – Stuck Sensor Concentrator

The following example illustrates the CAVIS monitoring system's ability to detect and isolate a stuck sensor concentrator component failure. A stuck sensor concentrator failure consists of the RADSiP sensors common to a sensor concentrator repeatedly reported some non-zero count rate. The data set analyzed here is a 10,000 data observations portion of a data set collected at Y-12 from May 3, 1999 at 14:00:56 to January 6, 2001 at 12:21:20 with a one-hour update rate. As previously discussed, a one-hour update rate is too long and an update rate of around one minute is more appropriate to securely monitor the SNM. Nonetheless, the CAVIS monitoring system is still capable of detecting and isolating the component failures that occur in the data set.

It is not possible to know what root cause induced the abnormal behavior in the radiation sensors common to this particular sensor concentrator because the data set was collected at Y-12. However, because each sensor common to the sensor concentrator and only these sensors failed it can be assumed that the malfunction existed in the sensor concentrator. The stuck sensor concentrator data set is presented in figure 5.4.
A visual inspection of the data reveals an obvious fault where all the sensors count rates stick near data observations 8700. The data set shown in figure 5.4 was analyzed by the

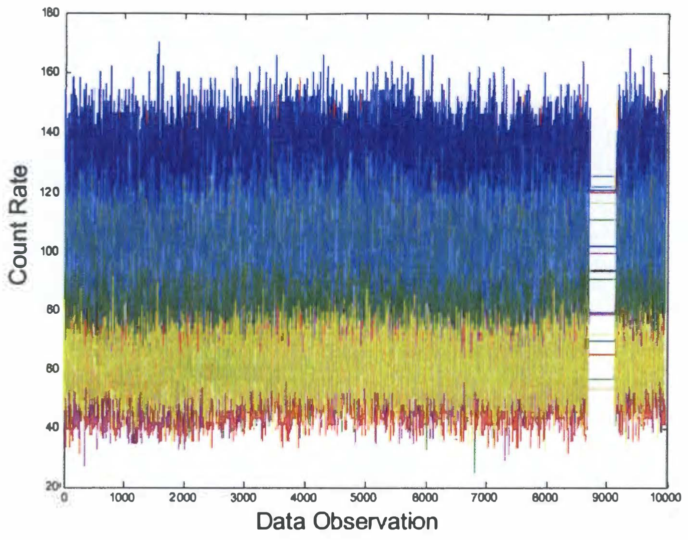


Figure 5.4. Component Failure Stuck Sensor Concentrator Data Set

CAVIS monitoring system for abnormal behavior. The warnings and alarms generated by the CAVIS monitoring system are presented in table 5.4.

The CAVIS monitoring system experienced several stuck detector alarms for various sensors prior to the common sensor concentrator communication failure at data observation 8725. However, in each of these instances the detector returned to normal behavior after a short period of time. These stuck detectors are not visible in figure 5.4 due to the amount of data and also due to the short length of time the stick occurs. As previously discussed, this faulty behavior for several of the sensors has been determined to be a precursor to the sensor concentrator failure that occurred later. At data observation 8725 the sensor concentrator experienced a stick that is detected by the monitoring system. The CAVIS monitoring system was able to detect the stick because the variance of the last five-observation feature was in a faulted state and the current

Table 5.4. CAVIS monitoring system analysis of Sensor Concentrator Failure Data

Index	Problem	Logic
5649	The RADSiP sensor is stuck: 1 1 6	The sensor had the same count rate for 5 cons. obs.
5650	The RADSiP sensor is no longer stuck: 1 1 6	The sensor no longer has the same count rate
5853	The RADSiP sensor is stuck: 1 1 12	The sensor had the same count rate for 5 cons. obs.
5854	The RADSiP sensor is no longer stuck: 1 1 12	The sensor no longer has the same count rate
6613	The RADSiP sensor is stuck: 1 1 12	The sensor had the same count rate for 5 cons. obs.
6614	The RADSiP sensor is no longer stuck: 1 1 12	The sensor no longer has the same count rate
7744	The RADSiP sensor is stuck: 1 1 17	The sensor had the same count rate for 5 cons. obs.
7745	The RADSiP sensor is no longer stuck: 1 1 17	The sensor no longer has the same count rate
8262	The RADSiP sensor is stuck: 1 1 2	The sensor had the same count rate for 5 cons. obs.
8264	The RADSiP sensor is no longer stuck: 1 1 2	The sensor no longer has the same count rate
8714	The RADSiP sensor is stuck: 1 1 3	The sensor had the same count rate for 5 cons. obs.
8716	The Sensor Conc. Process. Board 1 is stuck: 1 1	The Sen. Conc. board sensors have the same counts for 5 cons. obs.
8720	The Sensor Conc. Process. Board 4 is stuck: 1 1	The Sen. Conc. board sensors have the same counts for 5 cons. obs.
8724	The Sensor Conc. Process. Board 3 is stuck: 1 1	The Sen. Conc. board sensors have the same counts for 5 cons. obs..
8725	The Sensor Conc. is stuck: 1 1	The Sen. Conc. sensors have had the same counts for 5 cons. obs.
9143	The Sensor Conc. is no longer stuck: 1 1	The Sen. Conc., sensors no longer have the same count rate
9147	The Sensor Conc. Process. Board 4 is no longer stuck: 1 1	The Sen. Conc. board sensors no longer have the same counts
9147	Warning: The RADSiP sensor may be drifting: 1 1 18	The sensor is experiencing SPRT alarms
9151	The Sensor Conc. Process. Board 3 is no longer stuck: 1 1	The Sen. Conc. Board sensors no longer have the same counts
9151	Warning: The RADSiP sensor may be drifting: 1 1 12	The sensor is experiencing SPRT alarms
9152	The Sensor Conc. Process. Board 2 is no longer stuck: 1 1	The Sen. Conc. Board sensors no longer have the same counts

count rate feature was in an unfaulted state for every sensor common to the sensor concentrator. As previously discussed, this type of abnormality would go undetected by previous monitoring technique, due to a lack of variance testing. Thus in all likelihood, this CAVIS component failure would go unnoticed, and the SNM unmonitored, while the failure existed. The stuck sensor concentrator returns to normal operation at data observation 9152.

5.2.2 Component Failure – Failed Sensor Concentrator Communication Board

The following example illustrates the CAVIS monitoring systems ability to detect and isolate a failed sensor concentrator communication board. A failed communication board will result in a zero count rate for every sensor common to the board. The failure presented in this example was simulated at the University of Tennessee by removing communication board 1 from the sensor concentrator at data observation 215 and replacing the board at data observation 1067. Removing board 1 resulted in the count rates for sensors 1-10, which are common to the board, going to and remaining at zero. The data were collected at five-second intervals for 2 hours and 45 minutes to generate 2000 data observations. The described failed communication board data set is presented in figure 5.5.

A visual inspection of the data reveals ten of the sensors count rates go to zero near data observations 200 and return to normal operation near data observation 1000. The data set shown in figure 5.5 was analyzed by the CAVIS monitoring system for

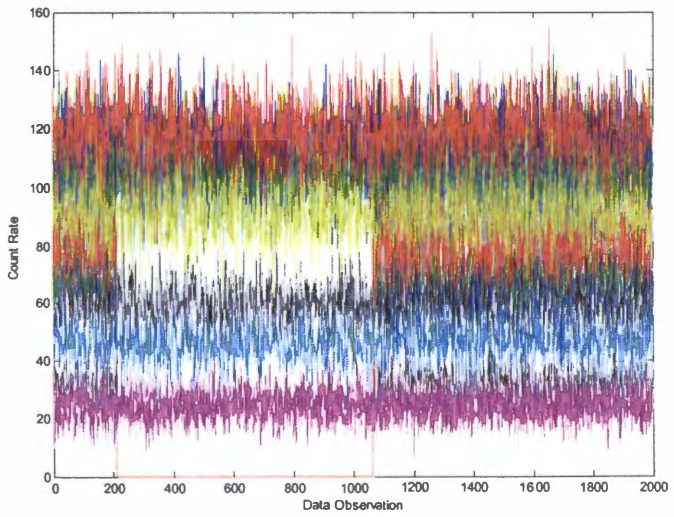


Figure 5.5. Component Failure Dead Communication Module Data Set

abnormal behavior. The warnings and alarms generated by the CAVIS monitoring system are presented in table 5.5.

The CAVIS monitoring system was able to detect the failed communication board and identified it as dead because the count rates for every sensor common to the board were zero during the failure. The communication board failed at data observation 215 and the CAVIS monitoring system was able to instantly generate a dead communication board sensor warning. The warning was generated because the current count rate feature was faulted for every sensor common to the board. At data observation 219 the CAVIS monitoring system generated a dead communication board alarm. The alarm is generated because both the variance of the last five-observation feature and current count rate feature were in a faulted state for every sensor common to the board. The dead communication board failure ends at observation 1068 when the communication board is plugged back into the sensor concentrator. The CAVIS monitoring system recognized that the board malfunction has been corrected and generated a "communication board is no longer dead message." It is necessary for the CAVIS monitoring system to recognize this type of failure as board malfunctions can occur and were frequently observed in the Y-12 data sets. These board failures would generate numerous CAVIS alarms and may result in unnecessary inventory checks of the SNM. The CAVIS monitoring system is

Table 5.5. CAVIS monitoring system analysis of Failed Communication Module

Ind.	Problem	Logic
215	Warning: The Sensor Conc. Comm. Board 1 may be dead: 1 1	Zero count rate for all sensors common to the board
219	The Sensor Conc. Comm. Board 1 is dead: 1 1	Sensors common to board zero count rate for 5 cons. obs.
1068	The Sensor Conc. Comm. Board 1 is no longer dead: 1 1	Sensors common to the board no longer have a zero count

able to detect and isolate board failures and will provide an alternative response, which should alleviate the possibility of unnecessary inventory checks.

5.3 Regional Anomaly Monitoring Module

The Regional Anomaly Monitoring Module (RAMM) is a module that detects region anomalies in the CAVIS system using a kernel smoothing technique. The following sections contain results from the RAMM including 1) a demonstration of the maximum neighborhood score as a fault symptom, 2) examples of kernel smoothing in two and three dimensions with and without an anomaly present, 3) the RAMM detection and isolation of a impacted CAVIS sensor vault pile, and 4) the RAMM detecting and tracking an external radiation source in the CAVIS storage area.

The kernel function featured in the RAMM is a Gaussian kernel whose shape is defined by the kernel width parameter. Optimization of the kernel width is performed though experimentation, and depends on the physical distance between the sensors and the various vault stacks. In this research, the kernel that produced the desired result had a kernel width of 2.5. However, the desired result is subjective, thus the optimal kernel width must be optimized for the application or at implementation.

5.3.1 Kernel Smoothing Neighborhood Score as Fault Symptom

The result of the RAMM is a collection of neighborhood scores at various locations in a three-dimensional array representing the CAVIS system. The values of these neighborhood scores should center near zero given no abnormality in the system. In the event of some abnormality, the RAMM will smooth the residual values to neighborhood scores greater than zero. Thus, the values of the neighborhood scores can be used as a fault symptom. Figure 5.6 illustrates the value of the maximum neighborhood score at each time interval during a simulation of the RAMM. The simulation featured an external source entering the CAVIS storage area for the time interval 10 to 25. The array representing the CAVIS storage area had dimensions of [12 x 15 x 5] which would correspond to 900 storage containers.

The value of the maximum neighborhood score increases to some larger value when the external source is in the CAVIS storage area. The magnitude of this increase depends on the strength of the external source. This result demonstrates that the neighborhood scores

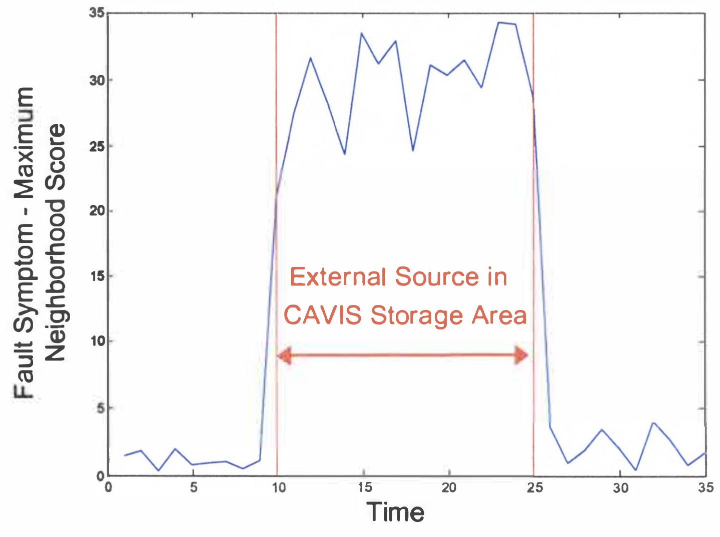


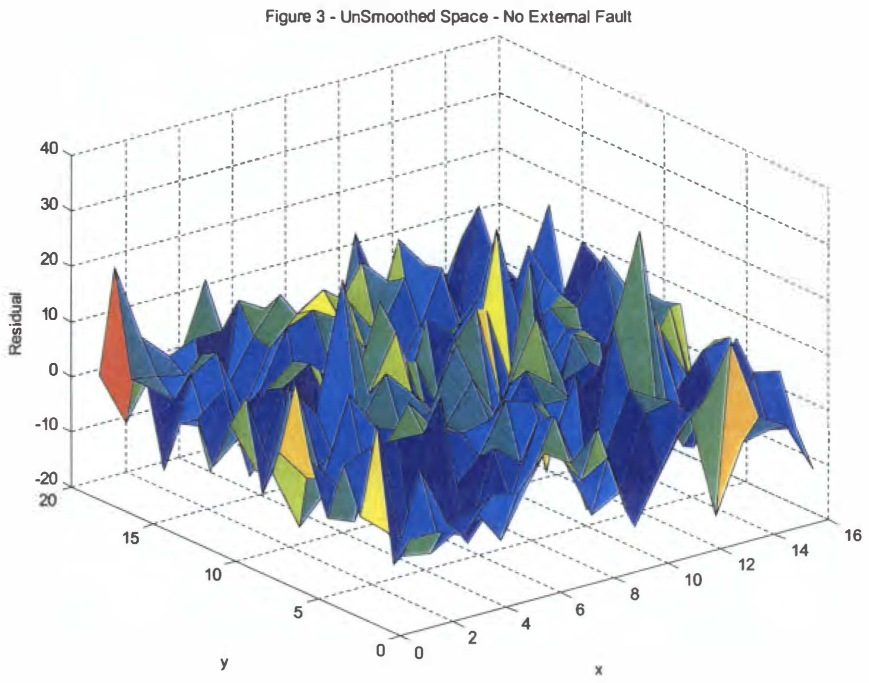
Figure 5.6. Maximum Neighborhood Score as Fault Symptom

can be used as a fault symptom to detect the presence of an abnormality affecting a region of the CAVIS system. In addition, plots of the neighborhood scores, which are described in the following sections, enable the visualization of the local anomalies.

5.3.2 Kernel Smoothing in Two Dimensions

The RAMM result is introduced by the following plots that illustrate kernel smoothing in a two-dimensional plane such as what would be seen with an unassembled sensor vault stack. The residual of the radiation sensor will be in the range of $0 \pm 3\sqrt{\sigma}$ with 99% confidence. Kernel smoothing the residual should result in neighborhood scores close to zero given no anomaly because the sensor residuals mean is zero. Figure 5.7 illustrates the smoothing of actual sensor residuals in two-dimensions without an abnormality present. The array representing the CAVIS storage area has dimensions of [20 x 16 x 1] which would correspond to 320 storage containers. To assist in visualization, the figure is shown in three-dimensions with the third dimension representing the neighborhood score.

The RAMM is able to transform the rigid plot of the residual data shown at the top of figure 5.7, into the smooth surface plot of the neighborhood scores located on the bottom of the figure. It is not possible to determine if an anomaly is present or not from the residual plot because the residual values range from –20 to 20. This variation is



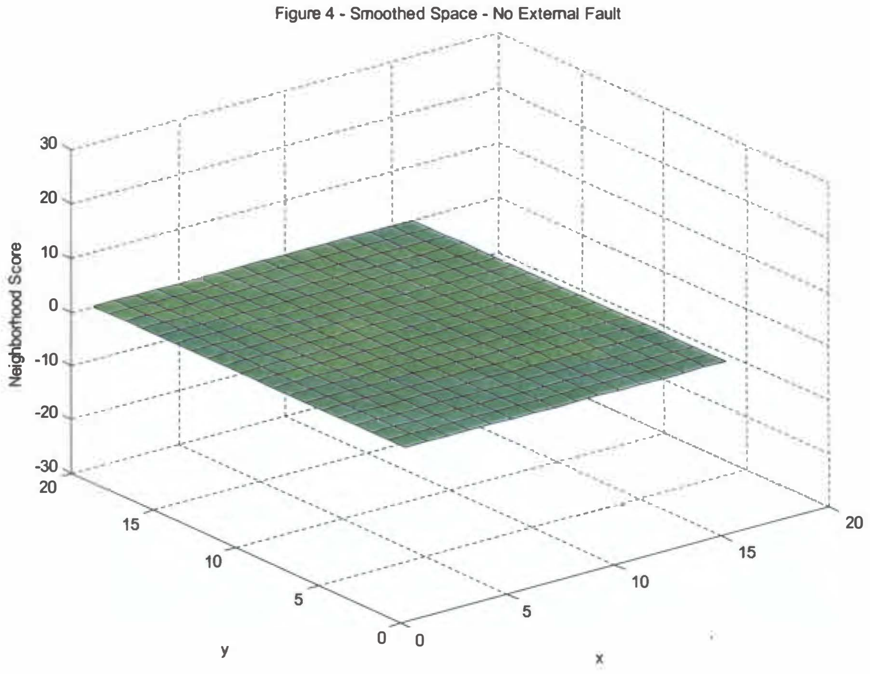
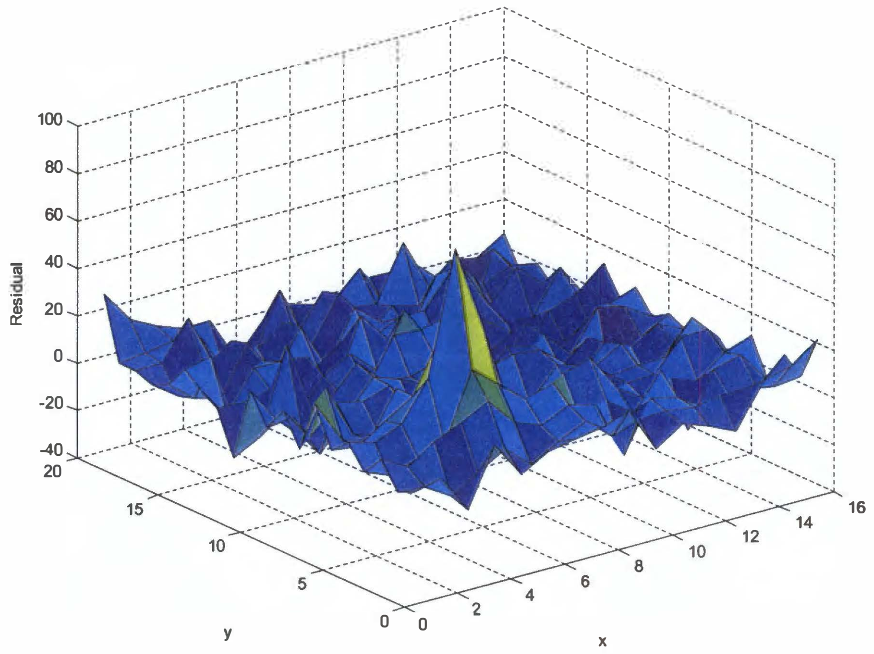


Figure 5.7. Kernel Smoothing in Two-Dimensions, No Abnormality

natural as the residual values can vary by $3\sqrt{\sigma}$ with 99% confidence. However, it can easily be seen that no anomaly is present from the plot of the neighborhood scores that is smooth. As discussed, the surface is smooth because the mean residual of the CAVIS regions center about zero because no data anomaly is present.

CAVIS collected data sets that contained regional anomalies were not available because the sensor locations were not known in the Y-12 collected data sets, and insufficient equipment was available at UT to produce these sets. Therefore, data sets featuring abnormalities had to be fabricated in order to test the RAMM system. The data set fabricated for this example had dimensions of [20 x 16 x 1] which would correspond to 320 storage containers. To simulate the abnormality in the set, a fabricated external radiation source was placed near the sensor at coordinate [5, 5, 1]. The fabricated source strength was such that it increased the residual value of this sensor by a factor of 20. The external source affected the other sensors in the data set by increasing their values by $20*\left(\frac{1}{R^2}\right)$ where R is the Euclidean distance from the sensor to the external source. The described fabricated data set is shown at the top of figure 5.8 and the result of the RAMM kernel smoothing of the data is shown at the bottom of the figure.

RAMM transforms the rough residual plot on the top of figure 5.8 into the smooth plot of the neighborhood scores located on the bottom; resulting in a clearly visible external source. Again, it is difficult to detect the data anomaly from the plot of the residuals, but the external radiation source is easily seen in the plot of the neighborhood scores. In addition, the RAMM is able to precisely determine the location of the external radiation source by determining the location of the maximum neighborhood score.



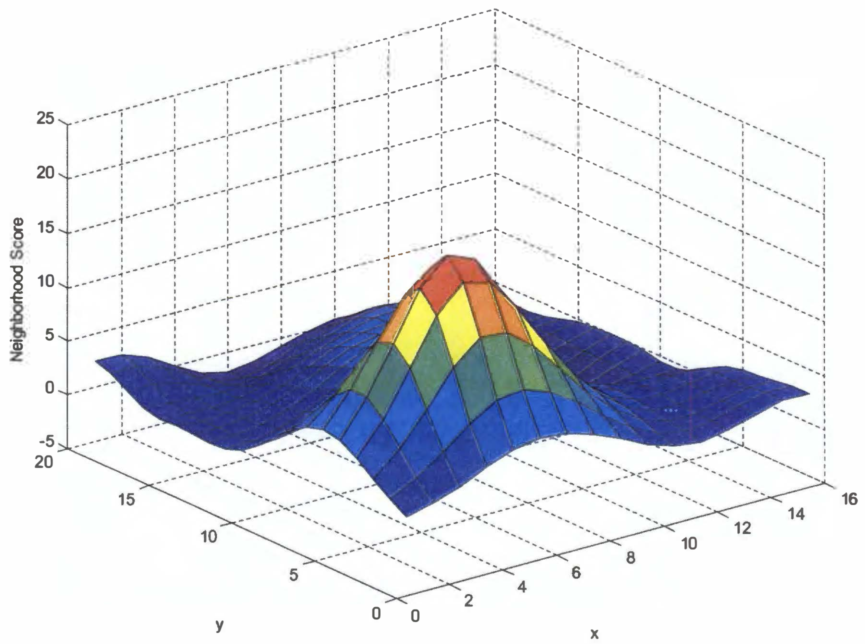


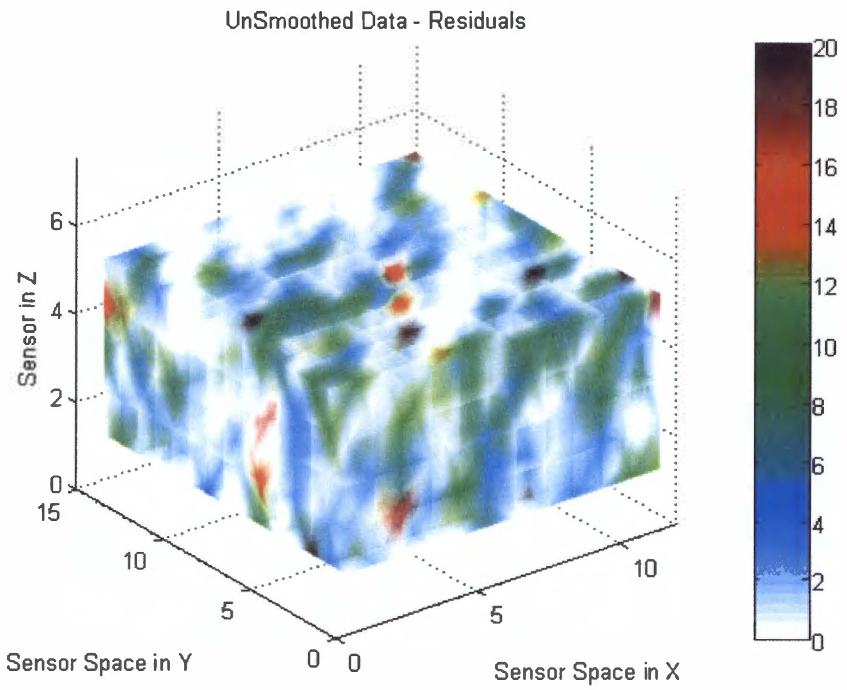
Figure 5.8. Kernel Smoothing in Two-Dimensions, Abnormality Present

5.3.3 Kernel Smoothing in Three Dimensions

The following plots illustrate kernel smoothing in three-dimensions, which represents the sensor vault stacks assembled on top of each other. As previously discussed, the residual values will range between $0 \pm 3\sqrt{\sigma}$ with 99% confidence, and kernel smoothing the residual data with no anomaly should result in neighborhood scores close to zero. Figure 5.9 illustrates the smoothing of actual sensor residuals in three-dimensions without an abnormality present. The array representing the CAVIS storage area has dimensions of [15 x 15 x 5] which would correspond to 900 storage containers. The X, Y, and Z dimensions represent the location of the neighborhood score and the value of the neighborhood score is represent by the color in the figure.

The RAMM transforms the residual sensor data, shown at the top of figure 5.9, to produce the smoothed data plotted in the bottom of the figure. The congested plot on the residual values ranges from –27 to 30. It is impossible to determine if an anomaly is present from the residual plot because of the wide range of residual values. However, these large residual values do not occur repeatedly in a region and are thus smoothed to neighborhood scores of near zero represented by the lack of color shown in the smoothed plot. This lack of color signifies the neighborhood scores center about zero, which indicates that no external source is present in the system.

Again, because CAVIS collected data sets containing anomalies were not available so it was necessary to fabricate data sets in order to test the RAMM system. The data set fabricated for this example had dimensions of [12 x 15 x 5] which would correspond to 900 storage containers. To simulate the abnormality in the set, a fabricated



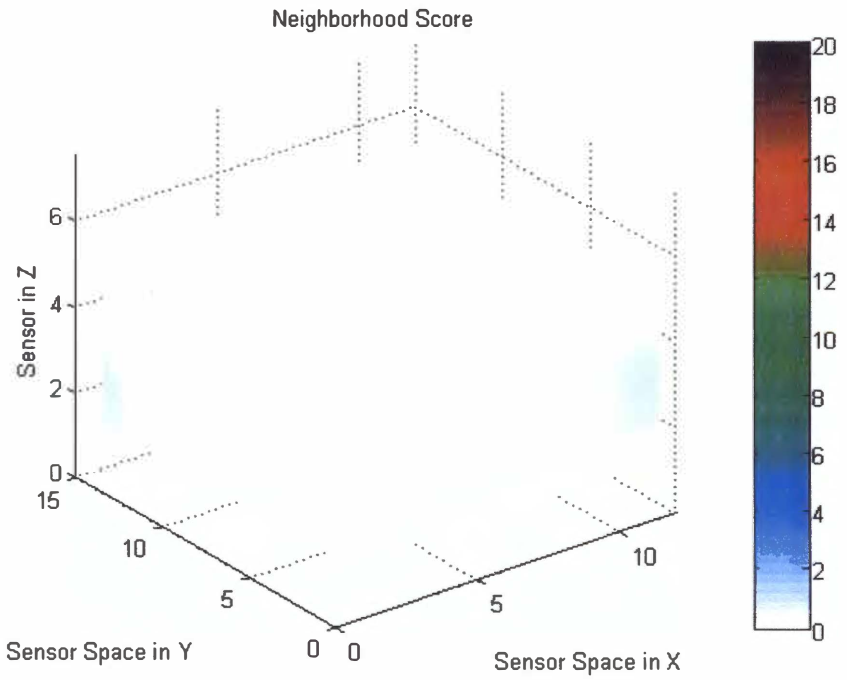


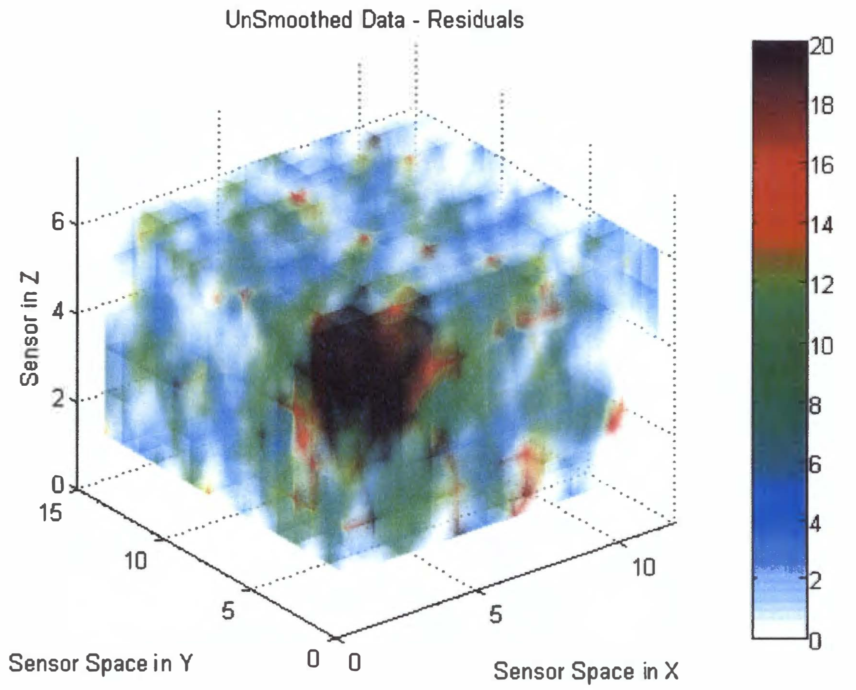
Figure 5.9. Kernel Smoothing in Three-Dimensions, No Abnormality

external radiation source was placed near the sensor at coordinate [2, 2, 5]. As before, the fabricated source increased the residual value of the sensor by twenty and increased the residual values of the other sensors in the set according to a $\left(\frac{1}{R^2}\right)$ relation. The described fabricated data set is shown at the top of figure 5.10 and the result of the RAMM kernel smoothing of the data is shown at the bottom of the figure.

The RAMM transforms the rough residual plot on the top of the figure 5.10 to the smooth plot of the neighborhood scores located on the bottom, resulting in a clearly visible external source. Again, it is difficult to detect the data anomaly from the plot of the residuals, but the external radiation source is easily seen in the plot of the neighborhood scores. Also, the RAMM is able to precisely determine the location of the external source by determining the location of the maximum neighborhood score.

5.3.4 RAMM Monitoring – Collision Induced Spike Vault Pile

The following example illustrates the ability of the RAMM to detect and isolate the location of vault stacks that have experienced spikes in their sensors count rates due to a collision with a heavy piece of machinery. The CAVIS storage area is a functioning warehouse that may have moving heavy equipment. It is possible that forklifts or other types of machinery moving in the CAVIS storage area may collide with one of the CAVIS sensor vault stacks resulting in a physical shock to each sensor in the stack. As previously discussed, the RADSiP sensors featured in the CAVIS system are sensitive to physical shock. When impacted the count rate of the sensor spikes for one and only one



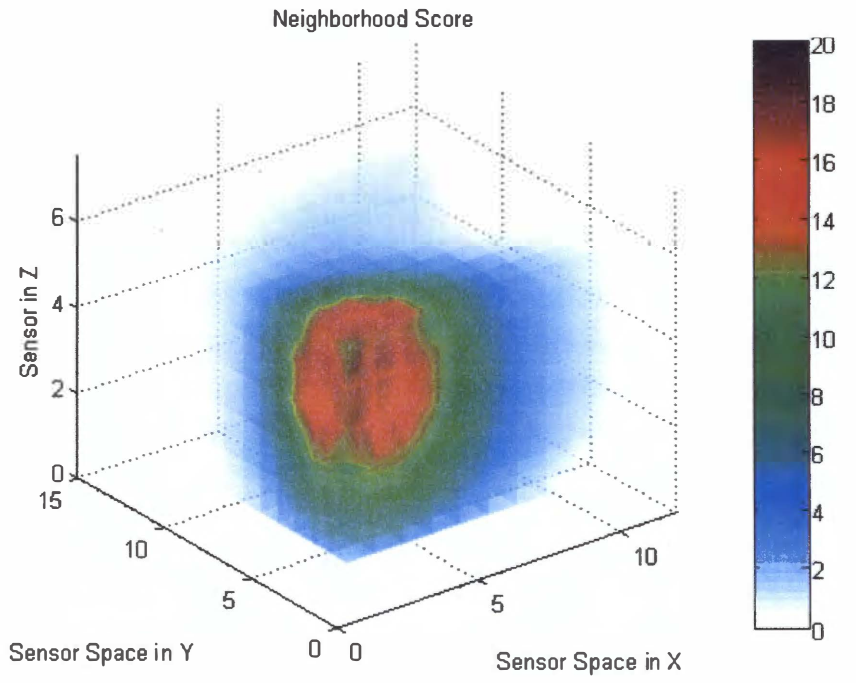
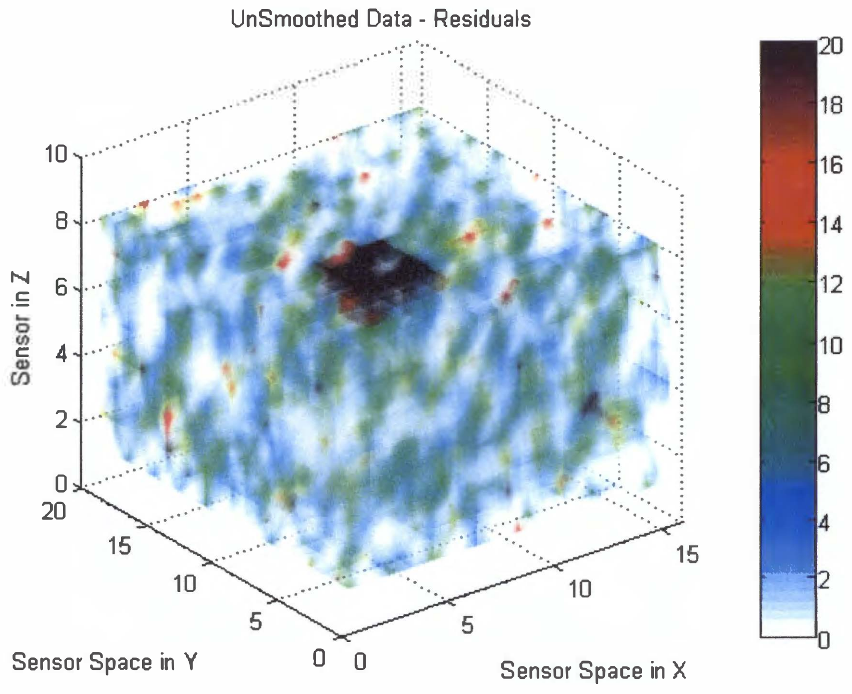


Figure 5.10. Kernel Smoothing in Three-Dimensions, Abnormality Present

data observation. The magnitude of these spikes varies and can be anywhere between 30% to 100% increase in the reported count rate. These collision induced spikes represent a change-in-state of the radiation signals in the vault stack, and may generate CAVIS alarms for the sensors that may result in inventory checks.

The RAMM can alleviate the possibility of unnecessary inventory checks due to collision induced CAVIS alarms, by detecting the collision. Because the spikes will be generated in the sensors common to the vault stack, the neighborhood scores for the region near the vault stack will dramatically increase for the one data observation in which the spike is recorded. These neighborhood scores will be recorded in the maximum score plot generated by the RAMM creating a record of the collision and an alternative hypothesis that can be inspected before an inventory check is performed.

CAVIS collected data sets of collision induced spike behavior were not available so the sets were fabricated. The data set fabricated for this example had dimensions of [16 x 20 x 8] which would correspond to sixteen-vault stacks in a 4 x 4 arrangement and 2560 storage containers as each vault stack contains 20 sensors in a 4 x 5 arrangement. To simulate the collision in the set, the count rate of each sensor common to the impacted vault stack was increase by 100%, which increased the value of the residual by the count rate. In this example, the collision was simulated for vault stack [2 x 2], which is one of the vault stacks in the middle of the arrangement. The described fabricated data set is shown at the top of figure 5.11 and the result of the RAMM kernel smoothing of the data is presented in the maximum neighborhood score plot shown at the bottom of the figure.



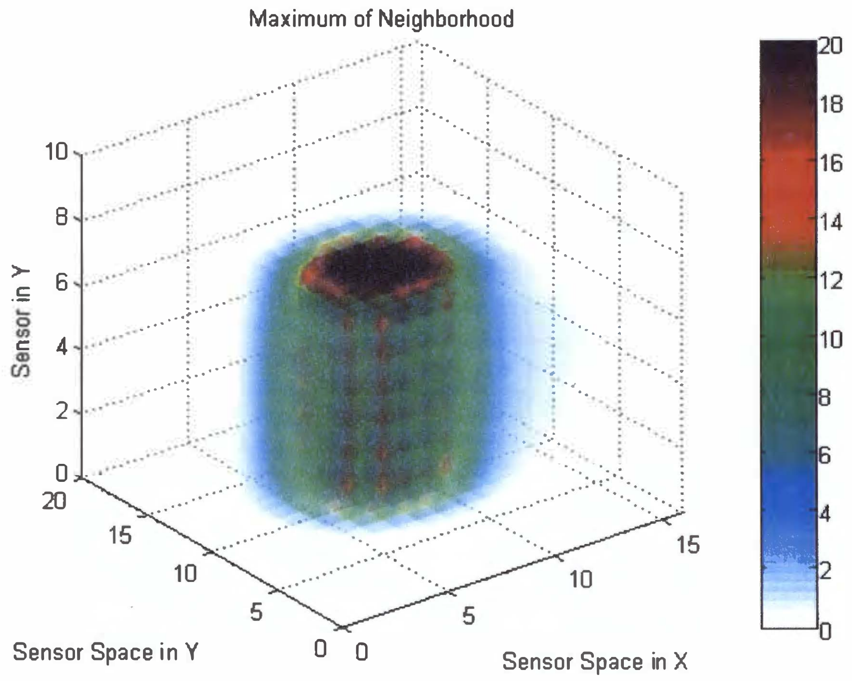


Figure 5.11. RAMM Analysis of Vault Stack Collision Induce Spiked Sensors

The RAMM transforms the residual plot on the top of figure 5.11 into the smooth plot of the neighborhood scores located on the bottom; resulting in a clearly visible data anomaly that is the impacted vault stack. The large neighborhood scores for the sensors common to the middle vault stack suggest that the anomaly originated from the stack. The RAMM records the impact in the maximum neighborhood score plot and is able to precisely determine the location of the impacted vault stack with zero error. Figure 5.11 is a portion of an impacted sensor vault simulation in which the collision was simulated at time interval three. The entire simulation including the maximum neighborhood score plot, the current residual plot, and the time history plot can be seen in appendix A.

5.3.5 RAMM Monitoring – Tracking of External Radiation Source

The following example illustrates the ability of the RAMM to track an external radiation source as it travels in the CAVIS storage area. The CAVIS storage area is a functioning warehouse in which radioactive material is frequently being moved. These external radiation sources may be detected by the CAVIS system causing an alarm in the region of the warehouse where the external radiation source is located. The RAMM can detect and track the location of external radiation sources and may alleviate the number of unnecessary inventory checks performed.

CAVIS collected data sets containing moving external radiation sources were not available, thus it was necessary to fabricate data sets in order to test the RAMM system. The data array fabricated for this example had dimensions of [16 x 20 x 8] which would correspond to 2560 storage containers. To simulate the abnormality moving in the set, a fabricated external radiation source was placed in the array at time index 3 at coordinate [1, 1, 5] and moved randomly in the z direction along a set path towards the opposite x-y planar corner of the array until the simulation ended at time index 10. The fabricated source increased the residual value of the sensor by seventy-five and increased the residual values of the other sensors in the set occurring to a $\left(\frac{1}{R^2}\right)$ relation. The described fabricated data at time index 9 is shown at the top of figure 5.12, the time history plot result of the RAMM kernel smoothing of the data is shown in the middle of the figure, and a rotation that shows a top view of the time history plot is shown at the bottom of the figure.

The RAMM transforms the residual plot on the top of figure 5.12 to the smooth time history plot of the neighborhood scores located in the middle plot; resulting in a clearly visible external source. Recall that the time history plot is a combination of the five most recent neighborhood scores weighted by an exponential function to give more weight to the most recent scores. The time history plot illustrates the past behavior of the neighborhood scores and contains a “tail” that reveals where the source has been and the direction it is traveling. The bottom plot in figure 5.12 is a top view of the time history plot that further demonstrates the tail in the time history plot. Figure Appendix A contains the residual plots and the time history plot produced by the RAMM for the entire simulation.

The RAMM determines the location of the external radiation source by locating the maximum neighborhood score. Figure 5.13 is a plot of the magnitude of the error in the RAMM predicted location of the external radiation source for the simulation.

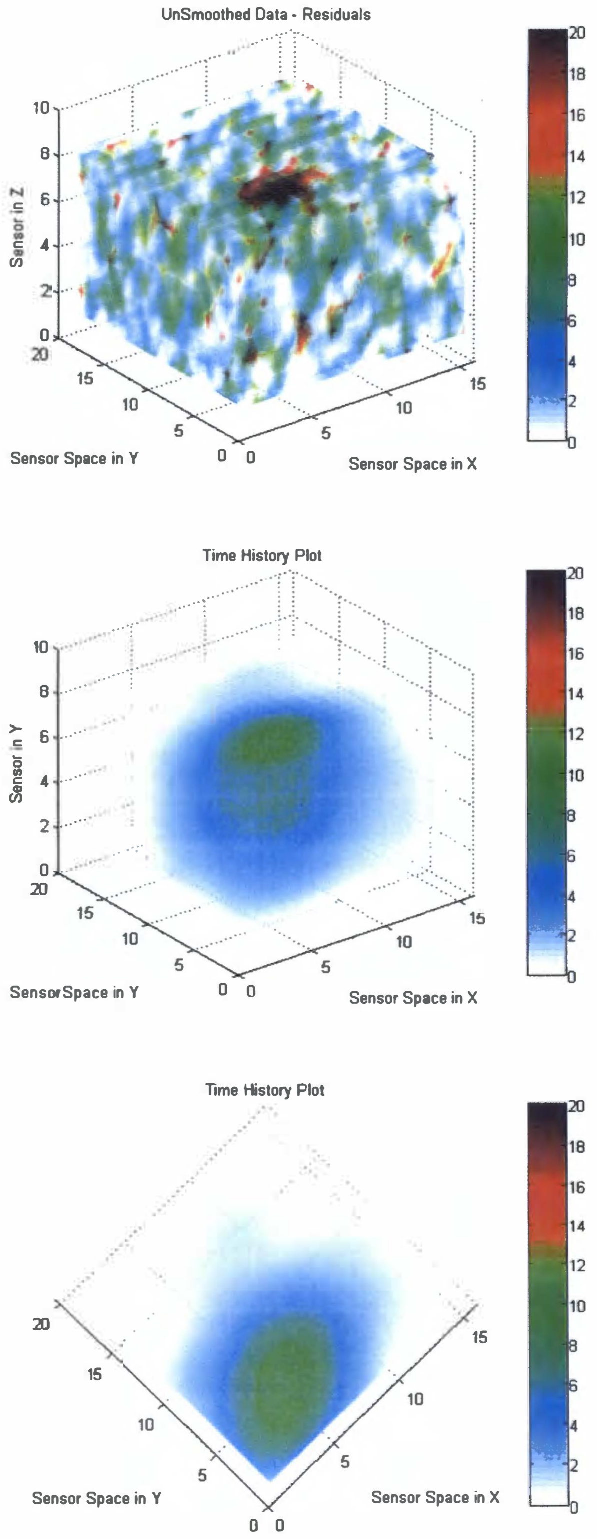


Figure 5.12. RAMM Analysis of Moving External Source, Time Index = 9

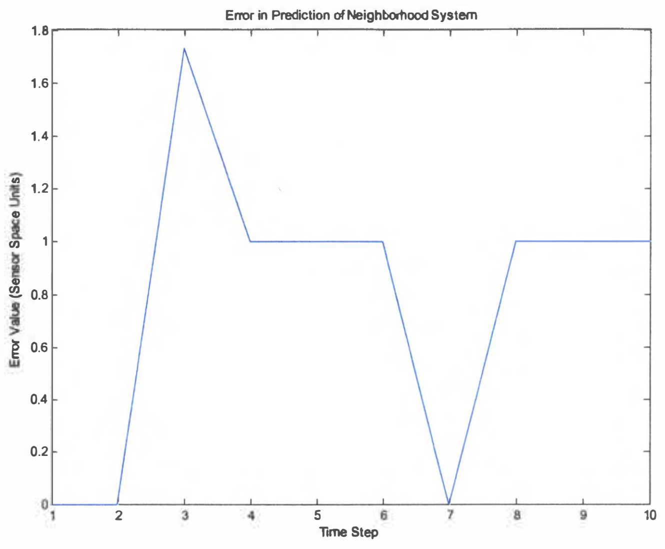


Figure 5.13. RAMM Error in Location of External Radiation Source

The RAMM was very successful at locating the external radiation source moving in the CAVIS storage area during the simulation. The average error of the prediction was 0.97 in sensor units and the maximum error of 1.75 occurred at the beginning of the simulation when the location of the external source was at the edge of the array. The higher error at the edges is expected, as kernel-smoothing techniques are susceptible to edge effects due to reduced amounts of data at the boundaries.

6 Conclusions, Contributions, and Recommendations for Future Work

6.1 Conclusions

The CAVIS monitoring system was developed as a fault detection and isolation system to monitor the CAVIS system at the Y-12 National Security Complex. CAVIS is an integrated package of sensors that continually monitors the physical attributes of special nuclear material at Y-12. The CAVIS system is subject to several types of failures such as environmental effects and component failure. The CAVIS monitoring system was able to improve the reliability of CAVIS by monitoring the system for failures. The monitoring was performed using the sequential probability ratio test, other key feature extraction algorithms, and the fault detection and isolation expert system.

The SPRT and feature extraction system mined the necessary information from the radiation signal to determine the current state of the CAVIS system. That data acted as the working memory for the expert system. The feature extraction module was the work of T Jay Harrison presented in his thesis "The Sequential Probability Ratio Test (SPRT) in Feature Extraction and Expert Systems in Nuclear Material Management [Harrison 04]."

The expert system detected and isolated all pre-enumerated faults that may occur in the CAVIS system by analyzing the data from the feature extraction module. Failures were detected by comparing the numerical values of extracted features to a set of tolerances. The expert system rule base knowledge was applied to detected failures and the system isolated the root cause of the failure. Thus, the expert system was able to reduce the alarm response cost by categorizing the alarms generated by the CAVIS system based on its knowledge base and suggesting alternate explanations for the alarm. If implemented, the FDI system can reduce operational costs by reducing the number of unnecessary inventory checks and minimizing other responses.

The Regional Anomaly Monitoring Module was developed as an additional fault detection module to monitor regions of the storage area for changes in state. The module enabled the detection of external radiation sources and collisions between equipment and CAVIS vault stacks. The module used a kernel smoothing technique to smooth the sensor residuals based on their proximity to one another. The smoothing resulted in a parameter referred to as the neighborhood score that can be monitored as a fault symptom. An abnormal increase in the neighborhood scores for a region of CAVIS indicates a common root cause such as the presence of an external radioactive source. Also, the modules results were plotted in a three-dimensional mapping that represented the regional behavior of the storage area.

The diagnostic system was capable of monitoring the condition of the CAVIS system and could perform system prognosis that resulted in early warning of component failures. If employed, its operation could allow the implementation of economical condition-based maintenance practices rather than more expensive reactive maintenance. The combination of CAVIS and its monitoring system allows for the safe, reliable, and economical monitoring of SNM.

6.2 Contributions

The primary contribution of this work to the field of nuclear material management is the demonstration of an artificial intelligence expert system to monitor a SNM security system. The research exhibits how an expert system, paired with a SPRT base feature extraction module, is capable of diagnosing root causes of abnormality in a large nuclear material management system.

The kernel smoothing method presented in this work has wide contributions as it can be applied to any detection system featuring multiple sensors in a well-defined lattice. In these systems, kernel smoothing can be used to determine the underlying behavior of the sampled variable in the system. One such application may be the proposed Homeland Security systems that feature multiple radiation sensors monitoring activities in large cities. In these systems, sensors are placed throughout metropolitan areas to monitor for radioactive materials that may be contained in bombs. Due to the random nature of radioactive decay, the radiation signal of the sensors will center about some mean value with a certain deviation, which makes it difficult to locate the origin of the radioactive material. The kernel smoothing methods presented here could be applied to smooth the sensor data to determine the underlying radiation behavior in the sensor lattice and thus locate the origin of the radiation signal.

6.3 Future Work

The future work of this project involves the integration of the CAVIS weight sensors into the developed system. This will require an analysis of failure effects, the identification of features and integration into the CAVIS monitoring system rule base knowledge.

Once the CAVIS weight sensors are integrated and optimized, the developed system may be implemented at Y-12. Implementation of the system will require adapting the rule base knowledge to any variation between the CAVIS systems at Y-12 and UT. Also, the kernel width in the RAMM will require optimization according to the physical distance between the sensors and the various vault stacks.

Future work may also include expansion of the expert system rule base knowledge to incorporate currently unknown fault scenarios. If additional knowledge of the CAVIS system is gained, or if additional components are incorporated into the system, the rule base knowledge should be updated to account for these changes. Additionally, modification of the detection threshold tolerances will make the FDI system more or less sensitive to changes-in-state as needed. The optimal value for the thresholds may be different depending on need as the values presented in this thesis are experimentally and empirically derived and set.

List of References

1. Bell, Z, T.W. Hickerson, J.E. Gaby, J.A. Williams, "Analysis of a radiation attribute from uranium in storage", *Computer Physics Communications.*
2. Bhatnagar, R., Miller, D. W., Hajek, B. K. and Stasenko, J. E., "An integrated operator advisor system for plant, monitoring, procedure management, and diagnosis," *Nucl. Technol.* (Mar.) (1990) 281-317.
3. Cherkassky, V.S., and F. Mulier, *Learning From Data*, John Wiley & Sons, New York, 1998.
4. Defense Programs at Y-12., Y-12 National Security Complex, [online] 2 September 2003, http://www.y12.doe.gov/bwxt/y12/y12-missions.html (Accessed: 29 Dec. 2003).
5. Duba, R. O., P. E. Hart, K. Konolige, and R. Reboh, "A Computer-based Consultant for Mineral Exploration," *Technical Report, SRI International.* (Sept.) (1979).
6. Feigenbaum, E. and Engelmore, R.S., "1993 Knowledge Based Systems in Japan," (Japanese Technology Evaluation Center), [online] May 1993, http://www.wtec.org/loyola/kb/ (18 November 2003).
7. Feigenbaum, E., "Knowledge Engineering in the 1980's", *Dept. of Computer Science, Stanford University*, Stanford CA, 1982.
8. Fenu, G. and T. Parisini, "A note on nonparametric kernel smoothing for model-free fault symptom generation," *Automatica*, 35 (1999) 1175-1179.

9. Giarratano, J., and G. Riley, *Expert Systems – Principles and Programming*, PWS Publishing Company, 1994.

10. Harrison, T Jay, "The Sequential Probability Ratio Test (SPRT) in Feature Extraction and Expert Systems in Nuclear Material Management," Masters Thesis in Nuclear Engineering, May 2004.

11. Hastie, T. and R. Tibshirani, *Generalized Additive Models*, CRC Press, 1990.

12. Hamamatsu, "Si PIN Phtotdiode S3590 series" (Hamamatsu) [online], http://usa.hamamatsu.com/assets/pdf/parts_S/S3590-01.pdf (Accessed 23 March 2004).

13. Khartabil, L., Hajek, B. K. and Miller, D. W., "An expert system for monitoring functionally diagnosing, and managing operations of a multiple mode nuclear plant system," In Proc. *A191 Frontiers in Innovative Computing for the Nuclear Industry*, Jackson, WY, 15-18 Sept. 1991.

14. Knoll, G.F., *Radiation Detection and Measurement*, John Wiley & Sons, New York, 1999.

15. Linsay, R. K., B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, *Applications of Artificial Intelligence for Organic Chemistry: The Dendral Project*, McGraw-Hill, New York, 1980.

16. Luger, George F., *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 4th Edition, Pearson Addison Wesley, 2002.

17. Miller, D. J. W. Hines, B. K. Hajek, L. Khartabill, Charles R. Hardy, Martin A. Haas and Lane Robbins, "Experience with the hierarchical method for diagnosis of faults in nuclear power plant systems," *Reliability Engineering & System Safety*, **44**(3) (1994) 297-311.

18. Nelson, Lloyd S. "The Shewhart Control Chart - Tests for Special Causes." Journal of Quality Technology, 16.4 (October, 1984), pp. 237-239.

19. Newell Allen and Herbert A. Simon, Human Problem Solving, Prentice-Hall, 1972.

20. Parson, Paul, "Storage facility 'anchor tenant' for Y-12 effort," (The Oak Ridger) [online] 28 March 2002, http://www.oakridger.com/stories/032802/bus_0328020012.html (Accessed 27 December 2003).

21. Parson, Paul, " Updated facilities are in DOE's future," (The Oak Ridger) [online] 3 July 2001, http://www.oakridger.com/stories/070401/new_0703010071.html (Accessed 16 March 2004).

22. Pickett C.A., K.M. Baldwin, Z.W. Bell, et al., "Automated Systems for Safeguarding and Accountancy of Stored Nuclear Materials," presented at the European Safeguards Research and Development Association (ESARDA) 21st Annual Meeting on Safeguards and Nuclear Material Management in Sevilla, Spain, 1999.

23. Pickett, Chris, "Continuous Automated Vault Inventory System (CAVIS™) for Accountability for Special Nuclear Materials," (Oak Ridge Systems for Enhanced Nuclear Safeguards), [online] 23 September 2003 A, http://www.y12.doe.gov/orsens/cavis.htm (26 December 2003).

24. Pickett, Chris, "ORSENS™ Sensor Concentrator," (Oak Ridge Systems for Enhanced Nuclear Safeguards), [online] 23 September 2003 B, http://www.y12.doe.gov/orsens/cavis.htm (26 December 2003).

25. Pickett, Chris, "RADSiP™ II Photodiode Gamma Ray Sensor Unit," (Oak Ridge Systems for Enhanced Nuclear Safeguards), [online] 23 September 2003 C, http://www.y12.doe.gov/orsens/cavis.htm (26 December 2003).

26. Schmorak, M., "235U A Decay," (Table of Nuclides), [online] 1993, http://atom.kaeri.re.kr/cgi-bin/decay?U-235+A (23 March 2004).

27. Shortliffe, E. H., *Computer-based Medical Consultations: MYCIN*, Elsevier, New York, 1976.

28. Society for the Historical Preservation of the Manhattan Project, "Photo: P-047 (Y-12 Plant)," (Children of the Manhattan Project), [online] 10 March 2004, http://www.childrenofthemanhattanproject.org/OR/Photo-Pages/ORP-047.htm (16 March 2004).

29. Tsoukalas, Lefteri H. and Robert E. Uhrig, *Fuzzy and Neural Approaches in Engineering*, John Wiley and Sons, Inc., New York, 1997.

30. Wald, A., "Sequential Tests of Statistical Hypothesis", *Ann. Math. Statist.*, **16**, 1945, 117-186.

31. Wald, Abraham *Selected Papers in Statistics and Probability*, Stanford University Press, California 1957.

32. Wald, Abraham *Sequential Analysis.* John Wiley and Sons, Inc., New York, 1947.

33. Wald, Abraham *Statistical Decision Functions*, Chelsea Publishing Company, 1971.

34. Wand, M.P., and M.C. Jones, *Kernel Smoothing*, Chapman & Hall/CRC, Boca Raton, Fl, 1995.

35. Waterman, D. A., *A guide to expert systems*, Addison-Wesley, Reading, Mass, 1986.

36. Western Electric Company, Inc. Statistical Quality Control Handbook. 2nd ed. New York: Western Electric Company, Inc., 1958.

37. Yesterday at the Y-12 National Security Complex., Y-12 National Security Complex, [online] 2 September 2003, http://www.y12.doe.gov/bwxt/y12/y12-yesterday.html (Accessed: 27 Dec. 2003).

38. Younkin, J. R., D. W. Carver, R. L. Lawson, et al., "A Secure Sensor System for Protected Asset Remote Verification," presented at the INMM 40th Annual Meeting, Pointe Hilton Resort at Squaw Peak, Phoenix, Arizona, July 25-29, 1999.

39. Zadeh, L. A., "Fuzzy Sets", *Information and Control*, Vol 8 (1965) 338-353.

Appendices

Appendix A: Selected Figures

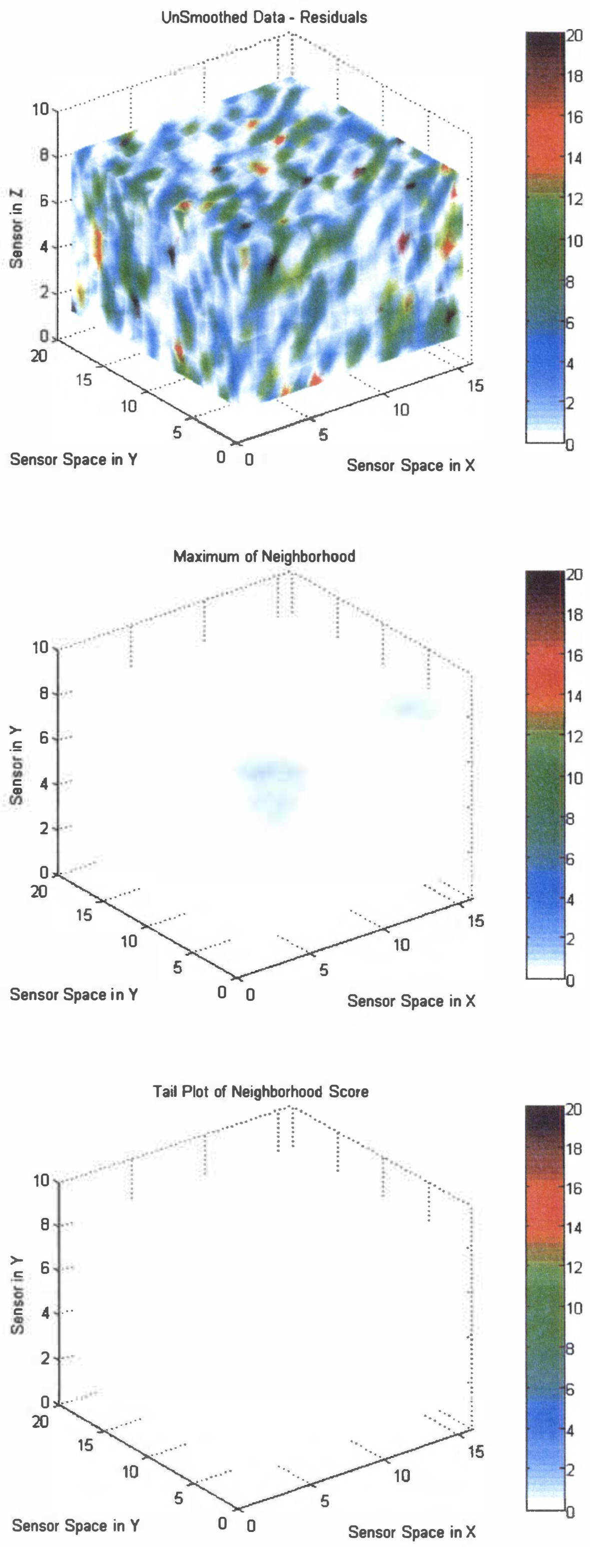
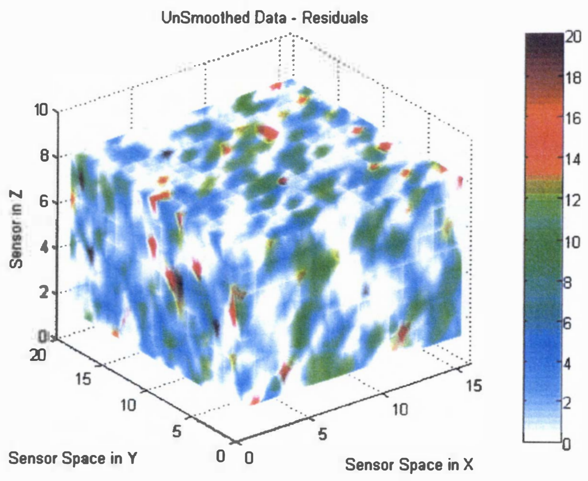
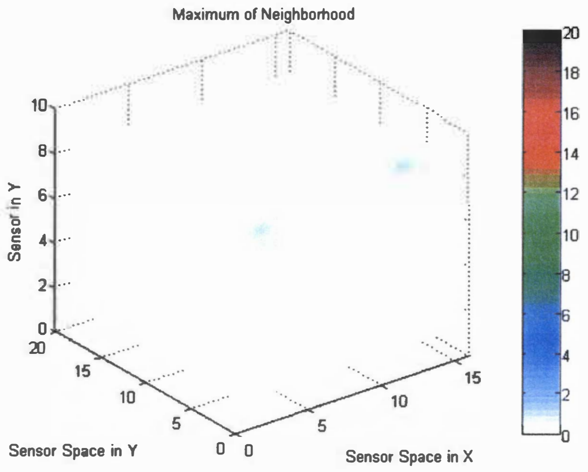


Figure A-1. RAMM Analysis of Impacted Vault Stack, Time Index = 1





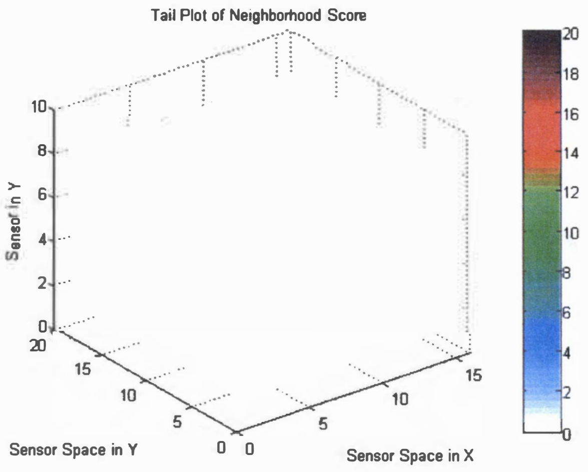


Figure A-2. RAMM Analysis of Impacted Vault Stack, Time Index = 2

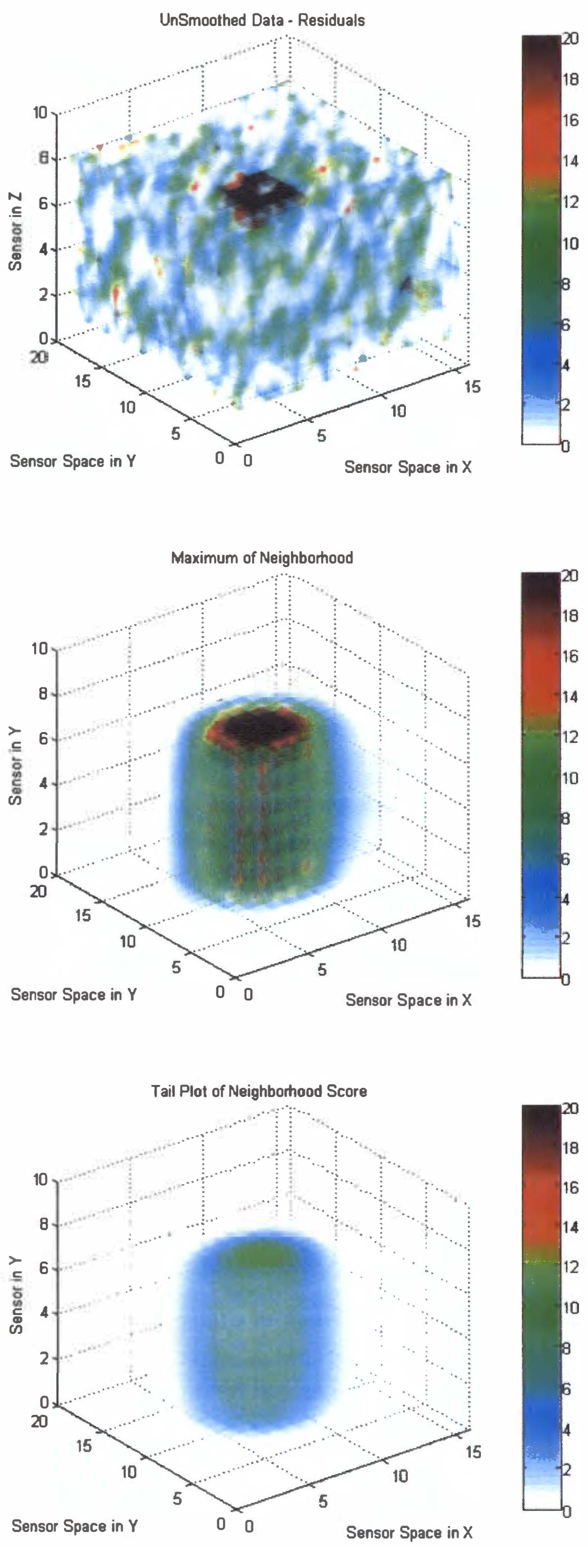


Figure A-3. RAMM Analysis of Impacted Vault Stack, Time Index = 3

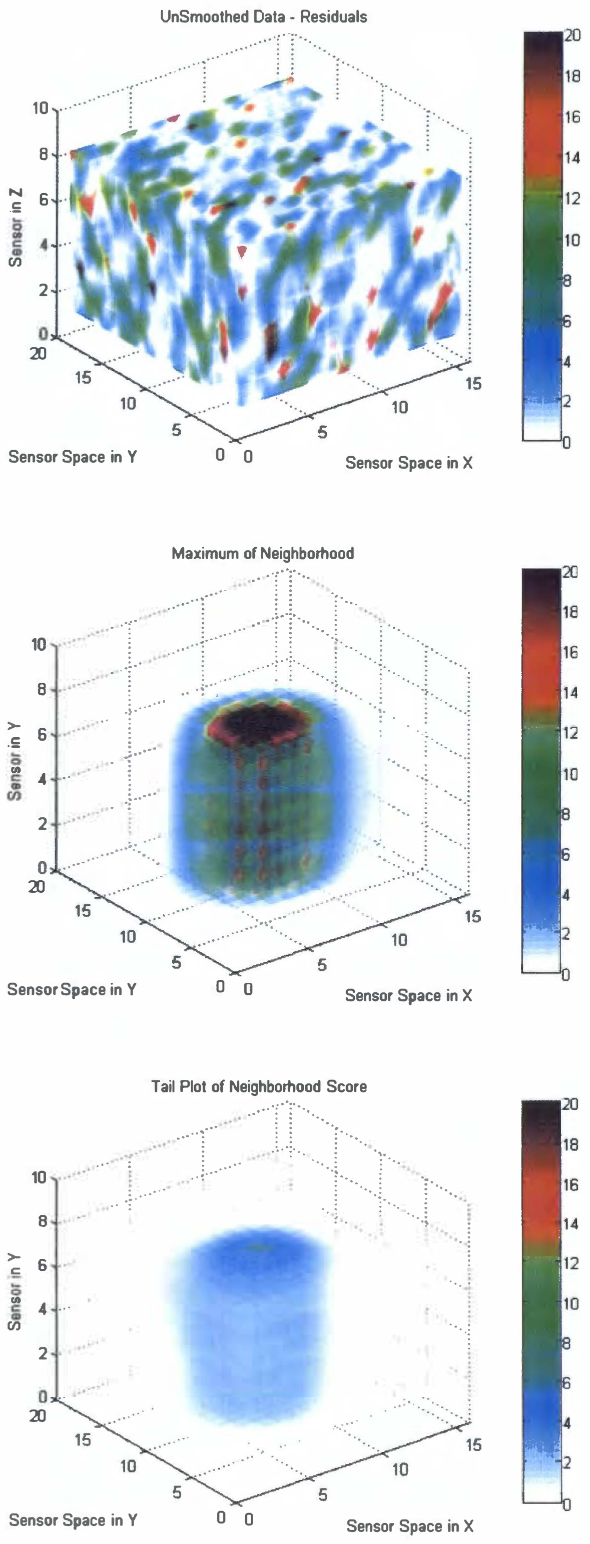
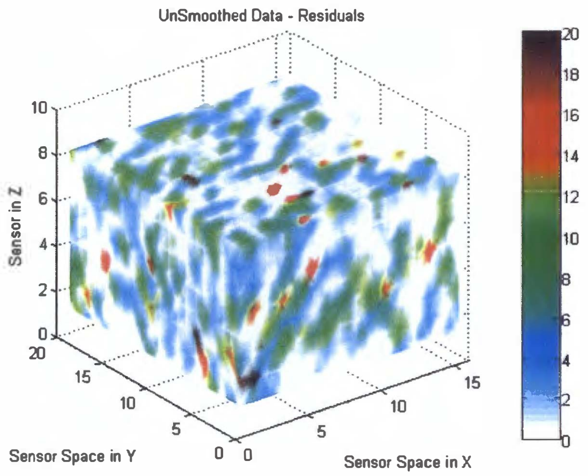
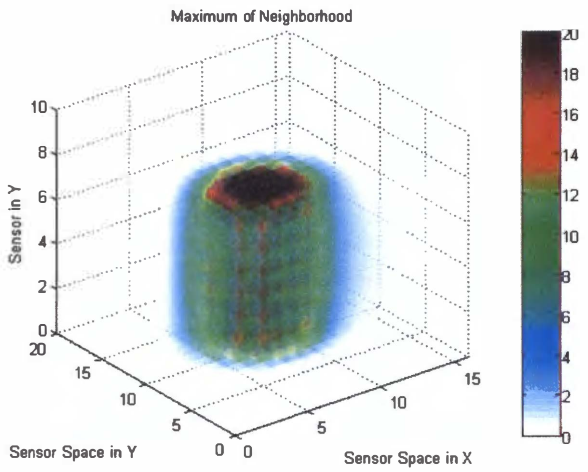


Figure A-4. RAMM Analysis of Impacted Vault Stack, Time Index = 4





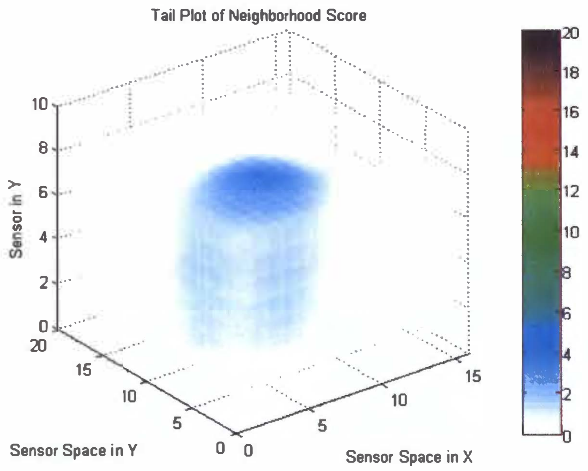


Figure A-5. RAMM Analysis of Impacted Vault Stack, Time Index = 5

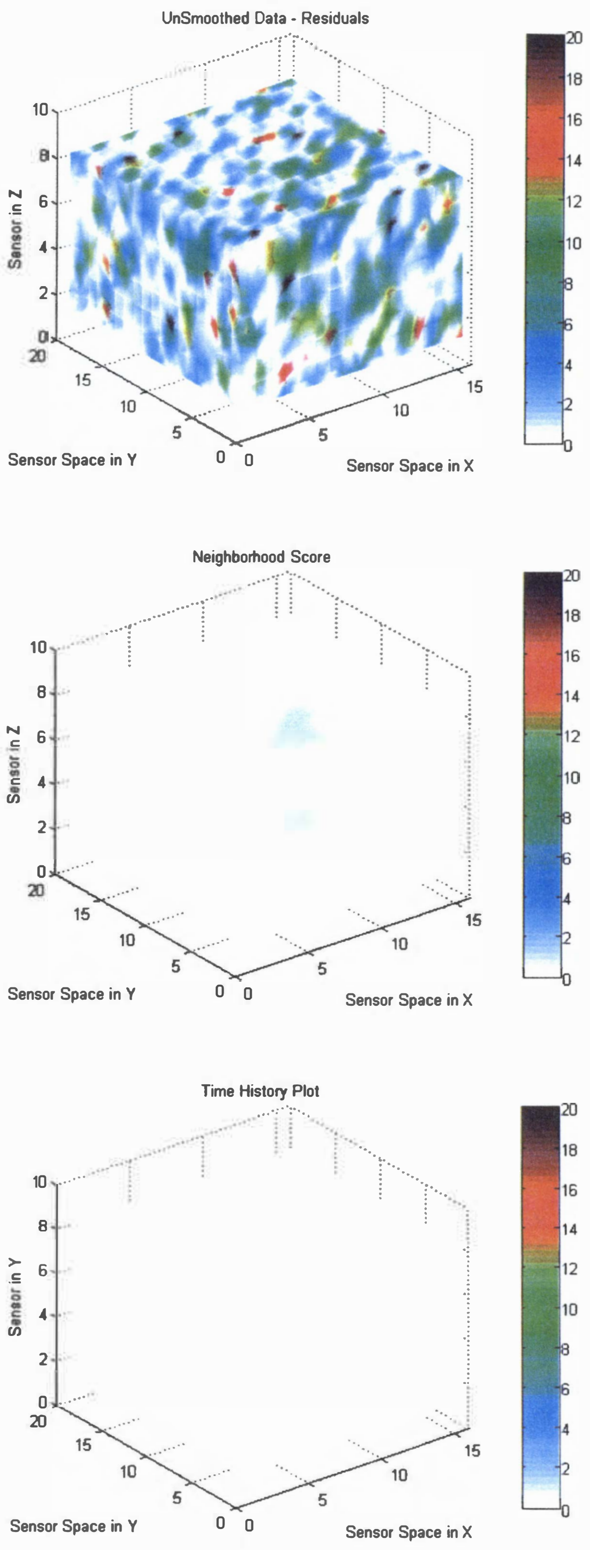
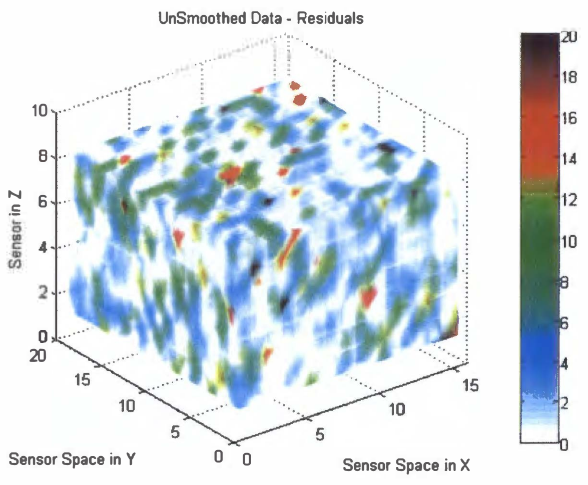
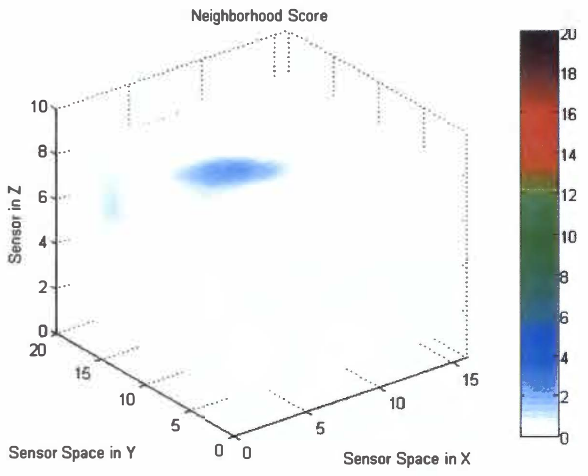


Figure A-6. RAMM Analysis of Moving External Source, Time Index = 1





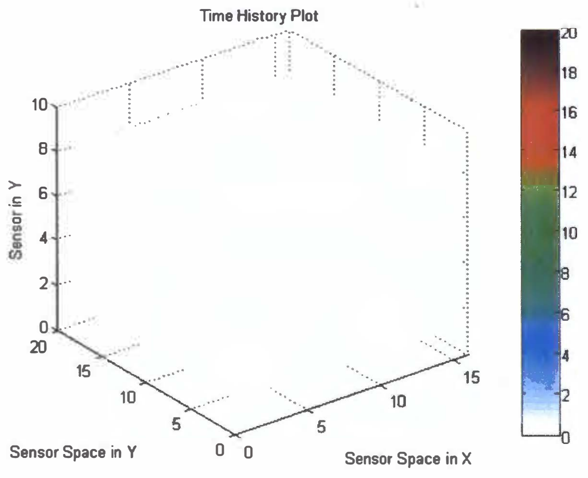
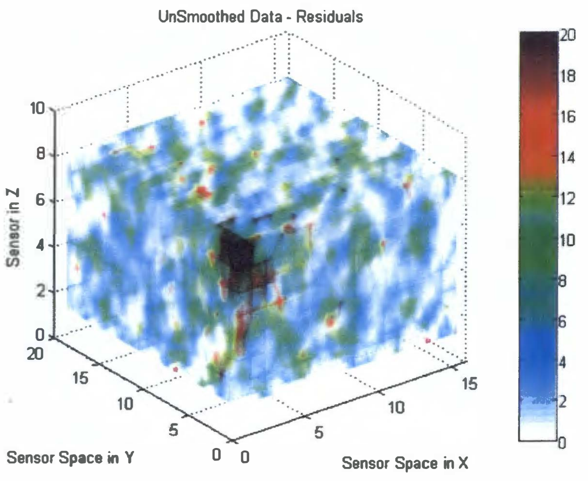
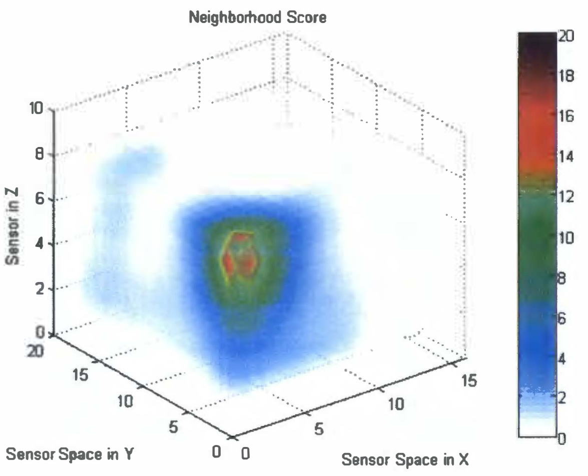


Figure A-7. RAMM Analysis of Moving External Source, Time Index = 2





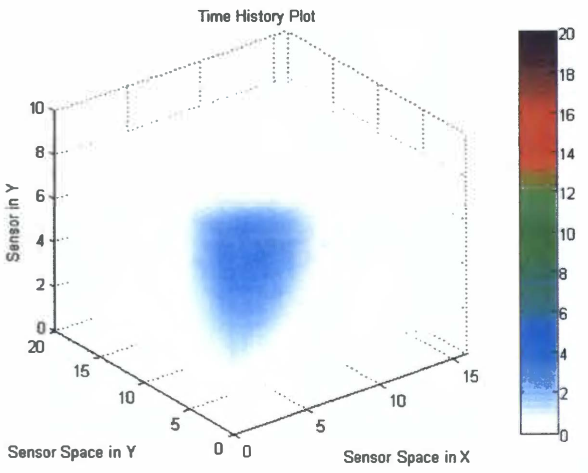
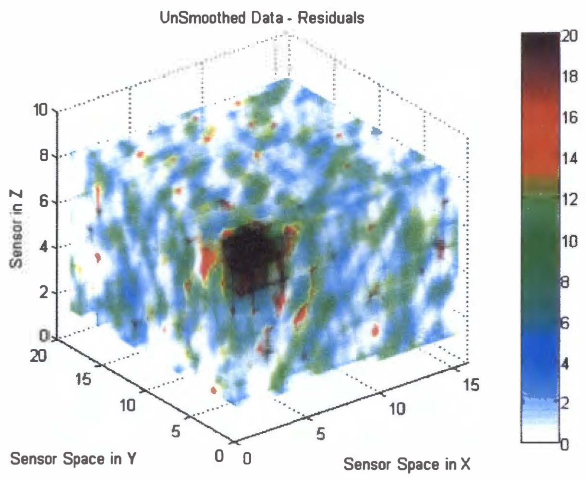
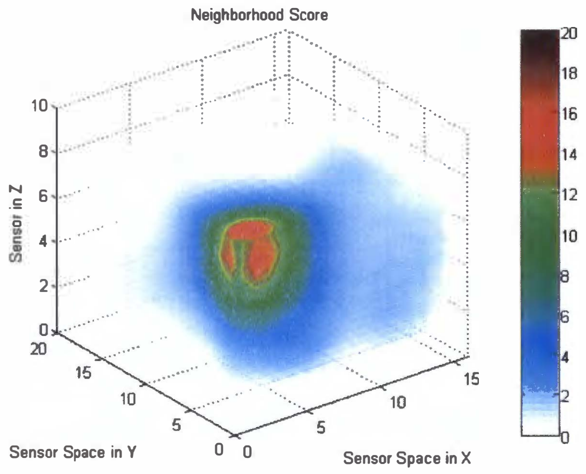


Figure A-8. RAMM Analysis of Moving External Source, Time Index = 3





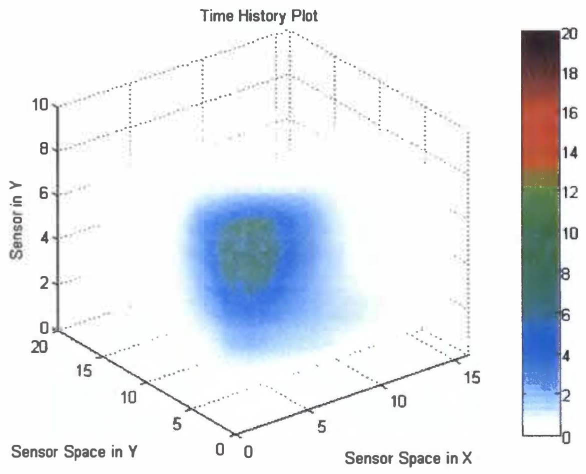


Figure A-9. RAMM Analysis of Moving External Source, Time Index = 4

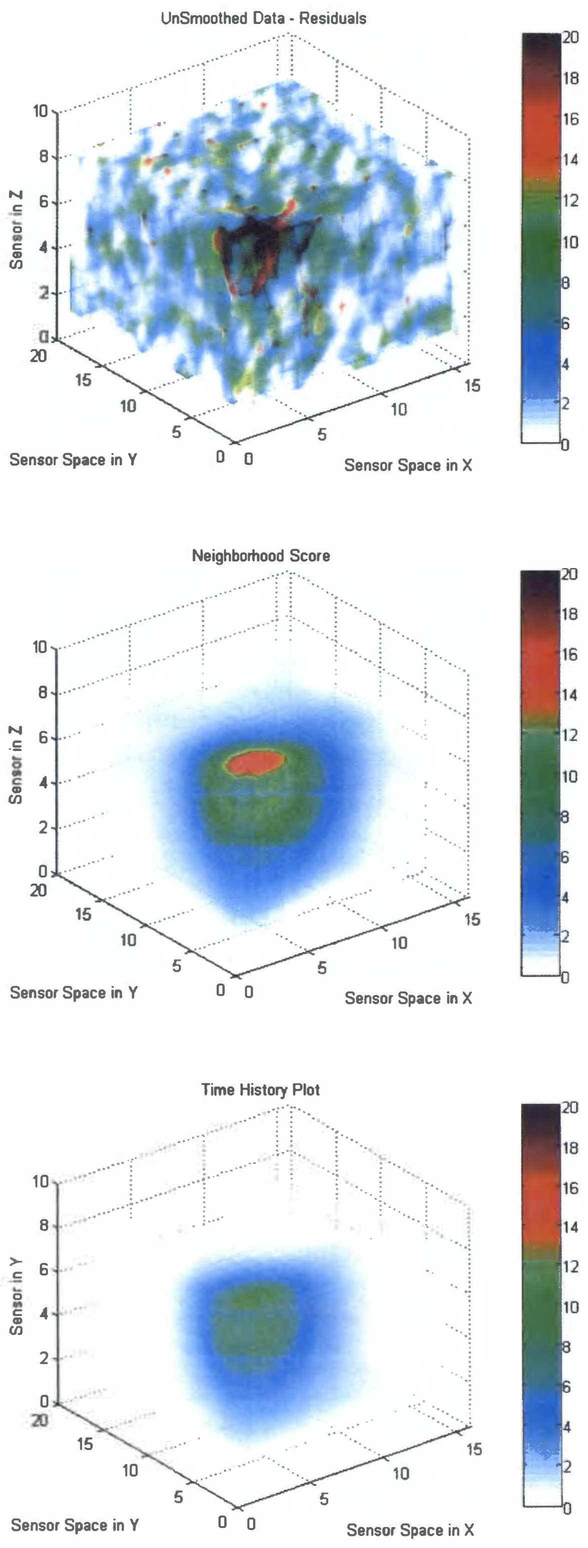


Figure A-10. RAMM Analysis of Moving External Source, Time Index = 5

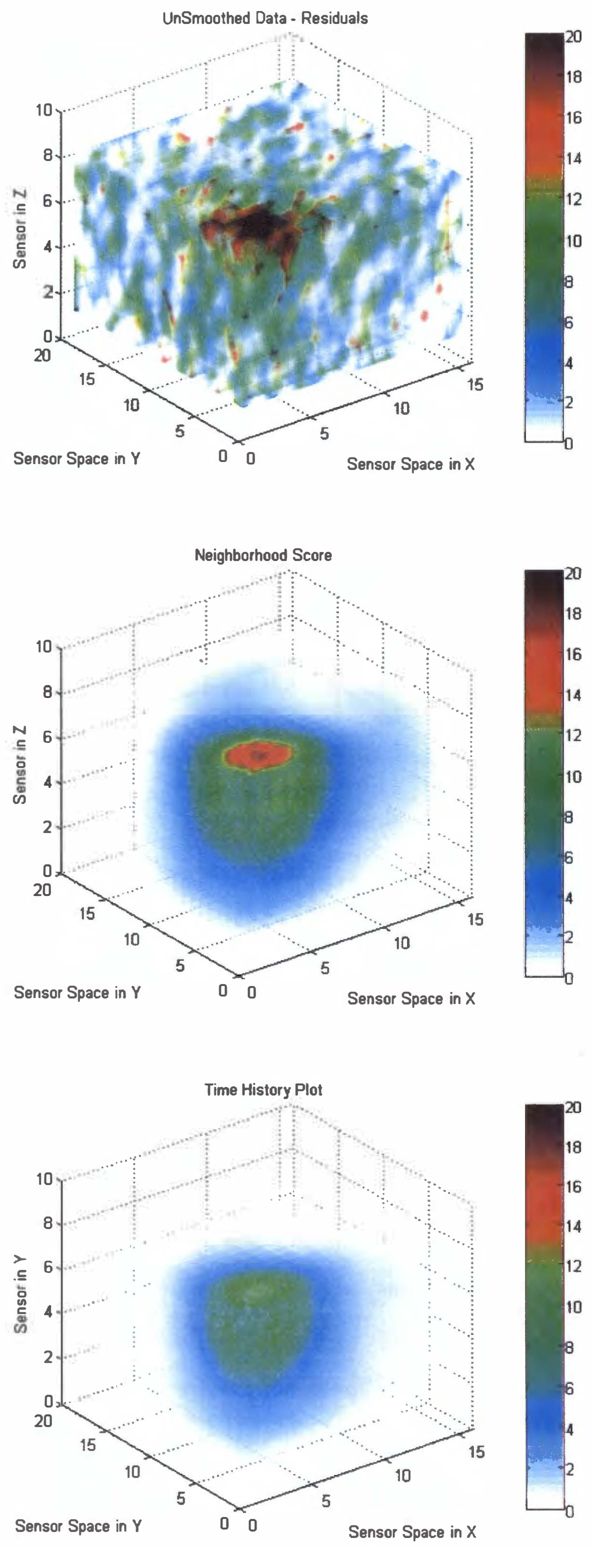


Figure A-11. RAMM Analysis of Moving External Source, Time Index = 6

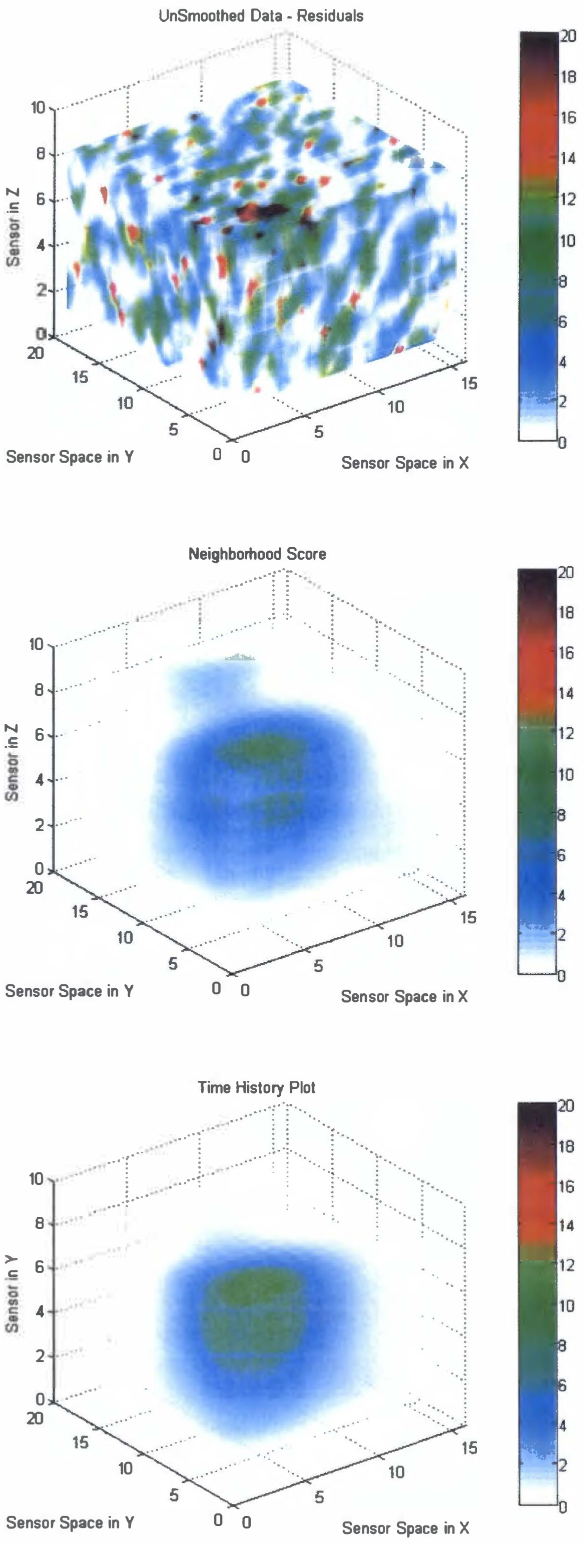


Figure A-12. RAMM Analysis of Moving External Source, Time Index = 7

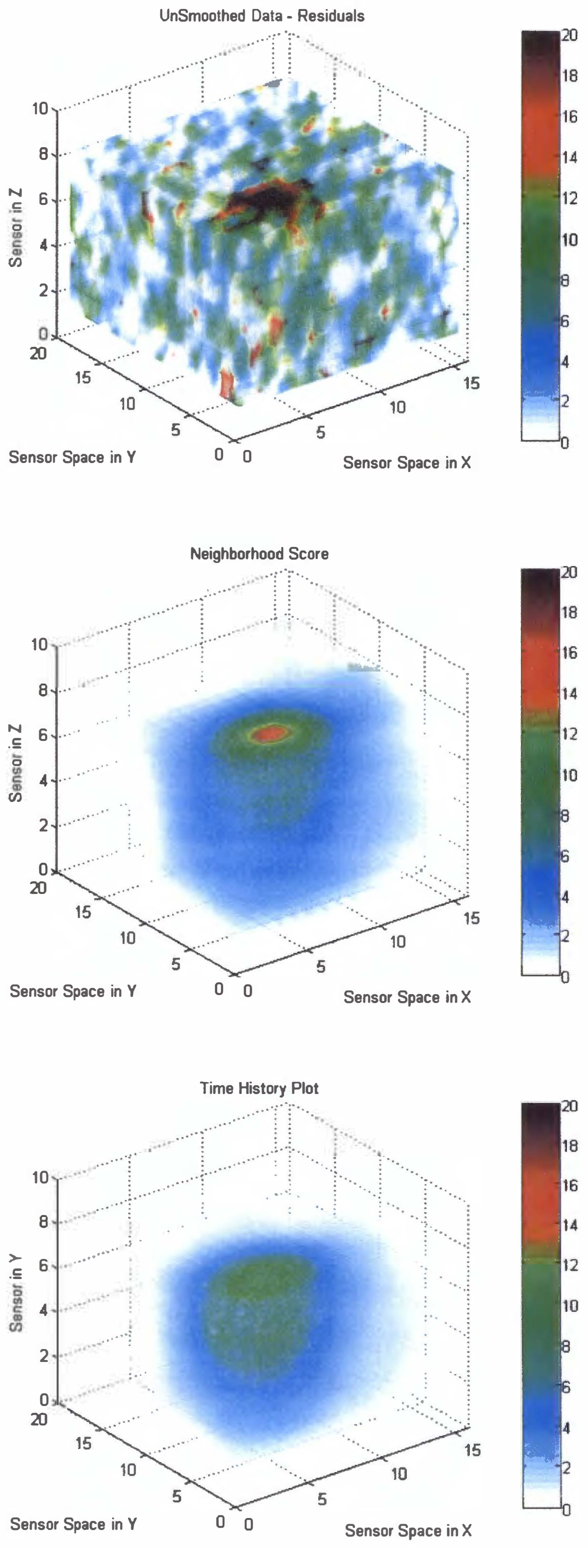


Figure A-13. RAMM Analysis of Moving External Source, Time Index = 8

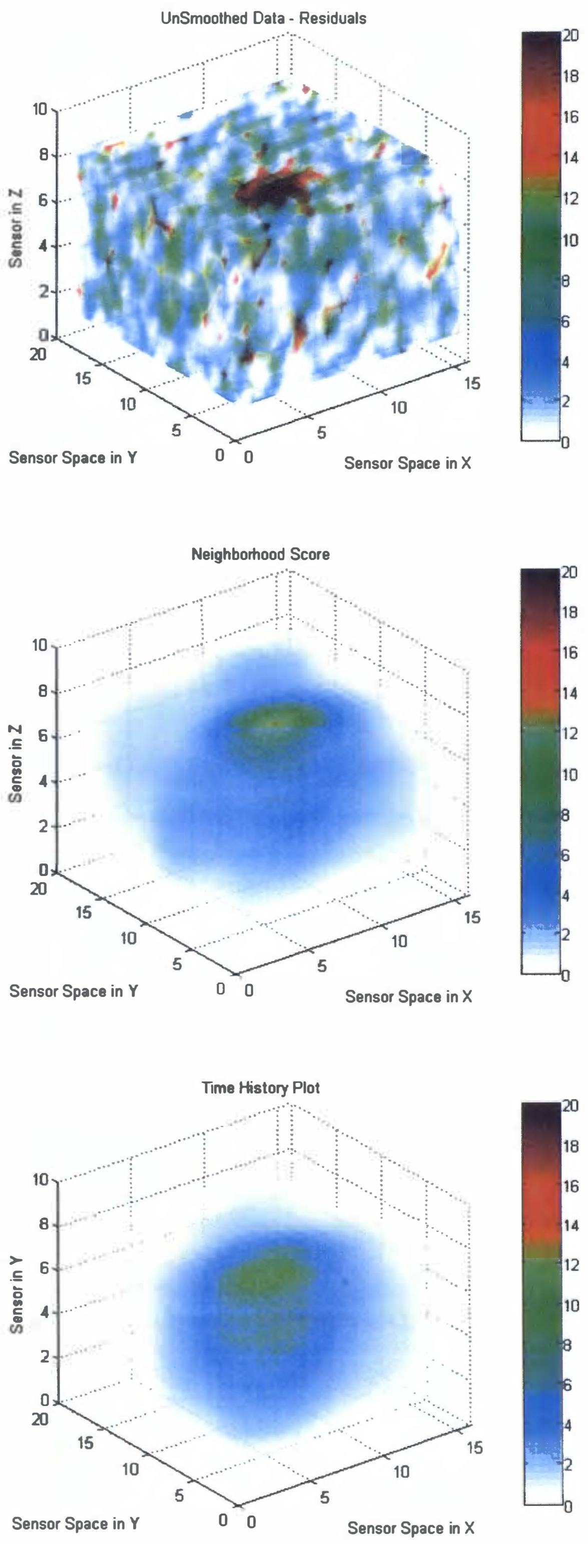


Figure A-14. RAMM Analysis of Moving External Source, Time Index = 9

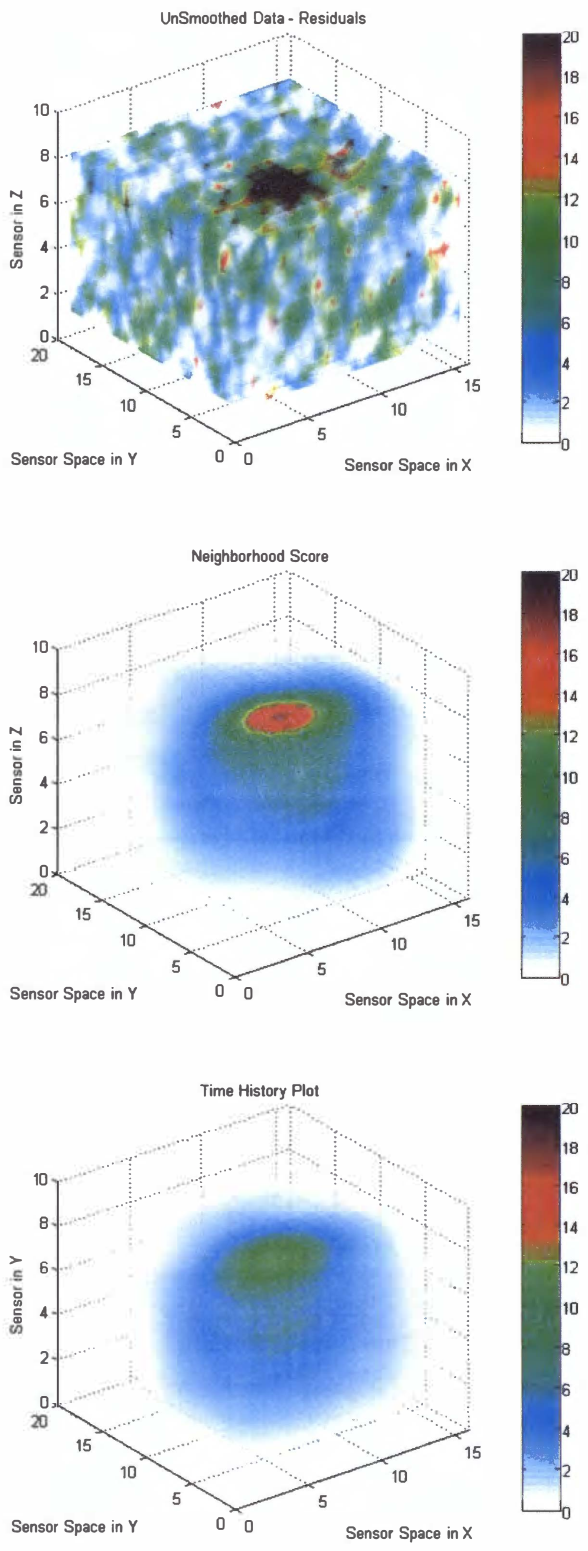
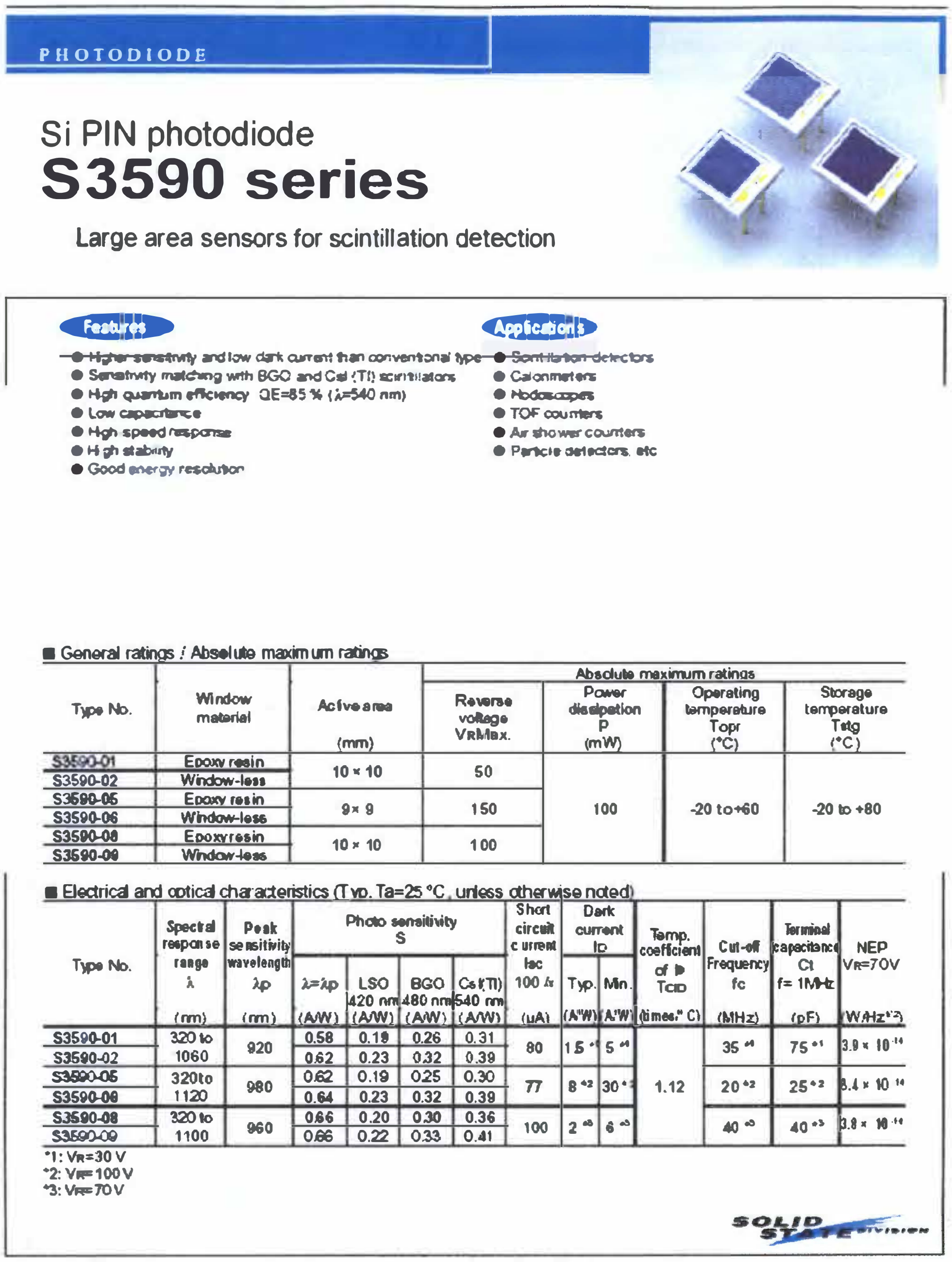


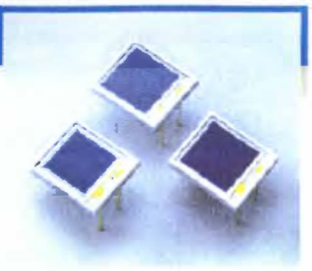
Figure A-15. RAMM Analysis of Moving External Source, Time Index = 10


PHOTODIODE

Si PIN photodiode

S3590 series

Large area sensors for scintillation detection



Features
- Higher sensitivity and low dark current than conventional type
- Sensitivity matching with BGO and CsI (Tl) scintillators
- High quantum efficiency QE=85 % (λ=540 nm)
- Low capacitance
- High speed response
- High stability
- Good energy resolution

Applications
- Scintillation detectors
- Calorimeters
- Hodoscopes
- TOF counters
- Air shower counters
- Particle detectors, etc

■ General ratings / Absolute maximum ratings

Type No.	Window material	Active area (mm)	Absolute maximum ratings Reverse voltage VR Max.	Power dissipation P (mW)	Operating temperature Topr (°C)	Storage temperature Tstg (°C)
S3590-01	Epoxy resin	10 × 10	50	100	-20 to +60	-20 to +80
S3590-02	Window-less					
S3590-05	Epoxy resin	9 × 9	150			
S3590-06	Window-less					
S3590-08	Epoxy resin	10 × 10	100			
S3590-09	Window-less					

■ Electrical and optical characteristics (Typ. Ta=25 °C, unless otherwise noted)

Type No.	Spectral response range λ (nm)	Peak sensitivity wavelength λp (nm)	Photo sensitivity S λ=λp (A/W)	LSO 420 nm (A/W)	BGO 480 nm (A/W)	CsI(Tl) 540 nm (A/W)	Short circuit current Isc 100 lx (uA)	Dark current ID Typ. (A"W)	Min. (A"W)	Temp. coefficient of ID TCID (times/°C)	Cut-off Frequency fc (MHz)	Terminal capacitance Ct f= 1MHz (pF)	NEP VR=70V (W/Hz^1/2)
S3590-01	320 to 1060	920	0.58	0.19	0.26	0.31	80	1.5 *1	5 *1	1.12	35 *1	75 *1	3.9 × 10⁻¹⁴
S3590-02			0.62	0.23	0.32	0.39							
S3590-05	320 to 1120	980	0.62	0.19	0.25	0.30	77	8 *2	30 *2		20 *2	25 *2	8.4 × 10⁻¹⁴
S3590-06			0.64	0.23	0.32	0.39							
S3590-08	320 to 1100	960	0.66	0.20	0.30	0.36	100	2 *3	6 *3		40 *3	40 *3	3.8 × 10⁻¹⁴
S3590-09			0.66	0.22	0.33	0.41							

*1: V_R=30 V
*2: V_R=100 V
*3: V_R=70 V

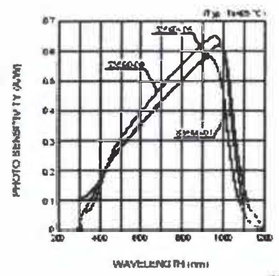
SOLID STATE DIVISION

HAMAMATSU

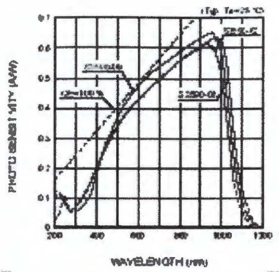
Figure A-16. RADSiP Specification Sheet Page 1 [Hamamatsu]

Si PIN photodiode S3590 series

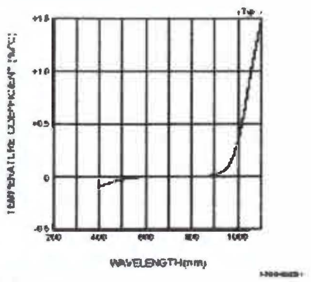
■ Spectral response



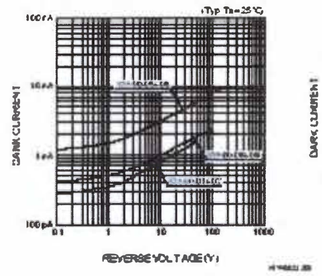
■ Spectral response (without window)



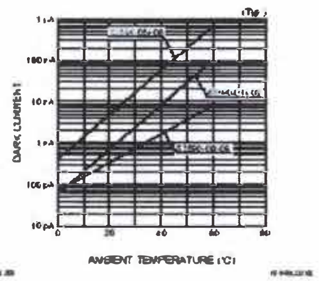
■ Photo sensitivity temperature characteristic



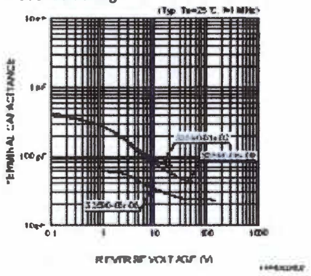
■ Dark current vs. reverse voltage



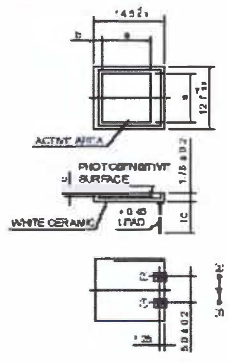
■ Dark current vs. ambient temperature



■ Terminal capacitance vs. reverse voltage



■ Dimensional outline (unit: mm)



The coating resin may extend a maximum of 0.1 mm beyond the upper surface of the package

	-01	-05	-08
a	10.0	9.0	10.0
b	[illegible]	[illegible]	[illegible]
c	0.8	0.5	0.7

HAMAMATSU

HAMAMATSU PHOTONICS K.K., Solid State Division
1126-1 Ichino-cho, Hamamatsu City, 435-8558 Japan, Telephone: (81)053-434-3311, Fax: (81)053-434-5184, http://www.hamamatsu.com
[illegible]

Cat. No. KPIN1052E02
Jan. 2002 DN

Figure A-17. RADSiP Specification Sheet Page 2 [Hamamatsu]

Appendix B: MATLAB Computer Code

Appendix

This appendix contains the coding for the FDI system. This includes the scripts and functions necessary to operate the expert system, and several scripts and function that run a demonstration of the RAMM.

Expert System Code, CAVIS monitoring system script

This script initiates and runs the CAVIS monitoring system.

```
fprintf('\n\nWelcome to Eskimo: The CAVIS monitoring system\n\n');
warning off MATLAB:divideByZero
warning off MATLAB:m_warning_or_and_precedence
Data_Choice = input('Do you wish to load Fabricated Data or Y-12 Data (1:Fab, 2:Y-12) \n');
Location = input('Is the program running on the UTK Network(1), Josephs House(2), or elsewhere(3): \n');
if Location == 1
   if Data_Choice == 1
      fprintf('Available Data Files to be loaded\n');
      dir('\\nepc126\Y-12\Fab_Data\')
      data_set = input('Input the data set you wish to run: ','s');
      eval(['load \\nepc126\Y-12\Fab_Data\',data_set]);
   end
   if Data_Choice == 2
      fprintf('Available Data Files to be loaded\n');
      dir('\\nepc126\Y-12\Y-12_Data\')
      data_set = input('Input the data set you wish to run: ','s');
      eval(['load \\nepc126\Y-12\Y-12_Data\',data_set]);
   end
elseif Location == 2
   if Data_Choice == 1
      fprintf('Available Data Files to be loaded\n');
      dir('C:\Y-12\Fab_Data\')
      data_set = input('Input the data set you wish to run: ','s');
      eval(['load C:\Y-12\Fab_Data\',data_set]);
   end
   if Data_Choice == 2
      fprintf('Available Data Files to be loaded\n');
      dir('C:\Y-12\Y-12_Data\')
      data_set = input('Input the data set you wish to run: ','s');
      eval(['load C:\Y-12\Y-12_Data\',data_set]);
   end
else
   fprintf('The only available data files are in the current directory\n');
   dir *.mat
   data_set = input('Input the data set you wish to run: ','s');
   eval(['load ',data_set]);
end
figure(1)
plot(dets)
title('Data set to be analyzed by Eskimo');
xlabel('Data Observation');ylabel('Activity');
ch1 = ddeinit('excel','featextexcel/Counts');
```

```
ch2 = ddeinit('excel','featextexcel/FeatureExt');
ch3 = ddeinit('excel','featextexcel/ExpSys');
ch4 = ddeinit('excel','featextexcel/Faults');
ch5 = ddeinit('excel','featextexcel/Ind50');
ch6 = ddeinit('excel','featextexcel/Identification');
ch7 = ddeinit('excel','featextexcel/IdentificationNew');
%Define variables for the state of the state of the system
PCDUQuan = 1;   %input('Enter the number of PCDU in the system : ');
SenConQuan = 1;  %input('Enter the number of Sensor Concentrators in the system : ');
index=0;
clear RADSiPNum state previousstate;
for i = 1:PCDUQuan
   for j = 1:SenConQuan
      for k = 1:20
         index=index+1;
         RADSiPNum(index,:) = [i j k];
         state(index,:) = [i j k];
         previousstate(index,:) = [i j k];
      end
   end
end
RADSiPNum=RADSiPNum';
state = state';
state(4:20,:) = 0;
previousstate = previousstate';
previousstate(4:20,:) = 0;
ConSignFault = zeros(1,20);
faultindex = 1;
ProbExcInd = 1;
Problem = {'0' '0'};
PreProblem = {'0' '0'};
Logic = {'0'};
ProbMat = zeros(30,9);ProbMat(3,1) = 1;ProbMat(4:10,1:2) = 1;
I = [ones(20,1) ones(20,1) (1:20)'];ProbMat(11:30,1:3) = I;
PreProbMat = ProbMat;
ProbMatOrg = ProbMat;
PExInd = 1;
offset = 0;
for indexa = 1:length(dets);
   indexb = indexa + offset;
   H = datestr(now);
   data = round(dets(indexa,:));
   k = indexa + 2 + offset;
   j = num2str(k);
   posit1 = ['r' j 'c1'];
   posit2 = ['r' j 'c2'];
   posit3 = ['r' j 'c3:r' j 'c22'];
   %  Write fabricated data
   rc = ddepoke(ch1,posit1,H(1:12));
   rc = ddepoke(ch1,posit2,H(13:20));
   rc = ddepoke(ch1,posit3,data);
   %  Perform reading/writing
   for sens = 1:20
      mu = mus(sens);
      featexpdat2(k,mu,sens,ch1,ch2,ch3,ch5);
   end
   %Call features from Excel Database "ExpSys"
   Data = ddereq(ch3,'r2c2:r21c17')';
   Data(17,:) = 1;
   %Fault Detection
   [Faults,prefault,NumSensor,state,ConSignFault] = Tolerance(Data,RADSiPNum,indexb,faultindex,state,ConSignFault,ch5);
   %Fault Identification New
   if any(any(state))
      [ProbMat]=RulesNew(Faults,ProbMat,RADSiPNum,NumSensor,state,ConSignFault);
   end
   %ProbMatStuff(:,:,indexb) = ProbMat;
   %Save Faults, Problem, Logic in Excel Database New;
   ProbStatus = [ProbMat(:,1:3) ProbMat(:,4:9)-PreProbMat(:,4:9)];
   if any(any(ProbStatus(:,4:9)))
      [ProblemNew,LogicNew,ColorIdent] = Inference(ProbMat,PreProbMat,ProbStatus);
      [PExInd] = ExcelESNew(ProblemNew,LogicNew,ColorIdent,H,PExInd,ch7,indexb);
   end
   %Define Variables for following loop
   previousstate = state;
   LastFaults = Faults;
   PreProblem = Problem;
   Problem = {'0' '0'};
   Logic = {'0'};
   PreProbMat = ProbMat;
   ProbMat = ProbMatOrg;
end
```

Tolerance Function

This function acts as the diagnostic module for detection faults in the CAVIS system.

```
function [Faults,prefault,NumSensor,state,ConSignFault] = Tolerance(Data,RADSiPNum,index,faultindex,state,ConSignFault,ch5)
%  Inference - Inference engine for an expert system to monitor CAVIS system
%  Written by Joseph Bowling
%  3/10/03
%
%  Data = State of System
%  RADSiPNum = Sensor identification [PCDU#; SensorConcentrator#; Sensor#]
%  index = current index of data point
%  faultindex = current fault index (Used in database entry)
%  state = current state of system (Faulted or Unfaulted) used in print
%      statement
%
%  Faults = Faults of System
%  NumSensor = Total number of sensors in system
%  state = current state of system (Faulted or Unfaulted) used in print
%      statement
%
%
%  data1  = Mean UP alarm in last 1000 data points
%  data2 = Mean DOWN alarm in last 1000 data points
%  data3 = Variance UP alarm in last 1000 data points
%  data4 = Variance DOWN alarm in last 1000 data points
%  data5 = Time between successive Mean UP alarm
%  data6 = Time between successive Mean DOWN alarm
%  data7 = Time between successive Variance UP alarm
%  data8 = Time between successive Variance DOWN alarm
%  data9 = The number of same signs (+) of the residual
%  data10 = Variance of last 50 or 100 data points
%  data11 = Variance of last 5 data points
%  data12 = Mean UP alarm in last 100 data points
%  data13 = Mean DOWN alarm in last 100 data points
%  data14 = Variance UP alarm in last 100 data points
%  data15 = Variance DOWN alarm in last 100 data points
%  data16 = Current count rate
%  data17 = Communication status of CAVIS (1=Good, 2 = Bad)
%Test for fault to locate problematic sensors
[NumData NumSensor]=size(Data);
FaultSen = zeros(3,80);
%Get previous faults
%ch5 = ddeinit('excel','featextexcel/Ind50');
Counter = 1;
Counter2 = 0;
prefault = zeros(5,80);
for i = 1:20
```

```
    a = Counter;
    b = Counter + 3;
    A = num2str(a);
    B = num2str(b);
    a = a-Counter2;
    b = b-Counter2;
    positcheck = ['r4c' A ];
    check = ddereq(ch5,positcheck);
    if check == 0
        Filler = [0];
        positfiller = ['r4c' A ];
        rc = ddepoke(ch5,positfiller,Filler);
    end
    positprefault = ['r4c' A ':r8c' B];
    prefaultnum = ddereq(ch5,positprefault);
    [prex prey] = size(prefaultnum);
    prefault(1:prex,a:b-(4-prey)) = prefaultnum;
    Counter = Counter + 5;
    Counter2 = Counter2 + 1;
end
Counter5 = 1;
Counter6 = 2;
Counter7 = 3;
Counter8 = 4;
for j = 1:NumSensor
        if Data(1,j) >= 5;  %6e-5;
           F(1,j) = 1;
           state(1,j) = 1;
        else F(1,j) = 0;
            state(1,j) = 0;
        end
        if Data(2,j) >= 5;  %8e-5;
           F(2,j) = 1;
           state(2,j) = 1;
        else F(2,j) = 0;
            state(2,j) = 0;
        end
        if Data(3,j) >= 5;  %4e-6;
           F(3,j) = 1;
           state(3,j) = 1;
        else F(3,j) = 0;
            state(3,j) = 0;
        end
        if Data(4,j) >= 5;  %2e-5;
           F(4,j) = 1;
           state(4,j) = 1;
        else F(4,j) = 0;
            state(4,j) = 0;
        end
        if (prefault(1,Counter5) == prefault(2,Counter5) + 1) & (index > 2);
            F(5,j) = 1;
            state(5,j) = 1;
        else F(5,j) = 0;
            state(5,j) = 0;
        end
        Counter5 = Counter5 + 4;
        if (prefault(1,Counter6) == prefault(2,Counter6) + 1) & (index > 2);
            F(6,j) = 1;
            state(6,j) = 1;
        else F(6,j) = 0;
            state(6,j) = 0;
        end
        Counter6 = Counter6 + 4;
        if (prefault(1,Counter7) == prefault(2,Counter7) + 1) & (index > 2);
            F(7,j) = 1;
            state(7,j) = 1;
        else F(7,j) = 0;
    state(7,j) = 0;
  end
  Counter7 = Counter7 + 4;
  if (prefault(1,Counter8) == prefault(2,Counter8) + 1) & (index > 2);
    F(8,j) = 1;
    state(8,j) = 1;
  else F(8,j) = 0;
    state(8,j) = 0;
  end
  Counter8 = Counter8 + 4;
  if abs(Data(9,j)) >= 15;
    F(9,j) = 1;
    state(9,j) = 1;
    ConSignFault(1,j) = ConSignFault(1,j) + 1;
  else F(9,j) = 0;
    state(9,j) = 0;
  end
  if (Data(10,j) == 0) & (index > 50);
    F(10,j) = 1;
    state(10,j) = 1;
  else F(10,j) = 0;
    state(10,j) = 0;
  end
  if (Data(11,j) == 0) & (index > 3);
    F(11,j) = 1;
    state(11,j) = 1;
  else F(11,j) = 0;
    state(11,j) = 0;
  end
  if Data(12,j) >= 3;  %6e-5;
    F(12,j) = 1;
    state(12,j) = 1;
  else F(12,j) = 0;
    state(12,j) = 0;
  end
  if Data(13,j) >= 3;  %8e-5;
    F(13,j) = 1;
    state(13,j) = 1;
  else F(13,j) = 0;
    state(13,j) = 0;
  end
  if Data(14,j) >= 3;  %4e-6;
    F(14,j) = 1;
    state(14,j) = 1;
  else F(14,j) = 0;
    state(14,j) = 0;
  end
  if Data(15,j) >= 3;  %2e-5;
    F(15,j) = 1;
    state(15,j) = 1;
  else F(15,j) = 0;
    state(15,j) = 0;
  end
  if Data(16,j) == 0;
    F(16,j) = 1;
    state(16,j) = 1;
  else F(16,j) = 0;
    state(16,j) = 0;
  end
  %Communication Status
  if Data(17,j) ~= 1;
    F(17,j) = 1;
    state(17,j) = 1;
  else F(17,j) = 0;
    state(17,j) = 0;
  end
  for i=1:NumData
```

```
      if F(i,j) == 1
         FaultSen(:,j) = RADSiPNum(:,j);
      end
      end
      end
C=FaultSen(1,:);
C=find(C);
FaultSen=FaultSen(:,C);
F=F(:,C);
Faults = [FaultSen;F];
%Define state of system for print statement - Only print for change in
%state of system
[NumData NumSen]=size(Faults);
state(4:15,:) = 0;
for j =1:NumData
   for i=1:NumSen
      Z = (80*Faults(1,i))-80 + (20*Faults(2,i))-20 + Faults(3,i);
      state(j,Z) = Faults(j,i);
   end
end
```

RulesNew Function

This function acts as the fault isolation module. It contains the characteristic fault portion of the knowledge base. The function also calls other function to establish facts additional facts.

```
function [ProbMat]=RulesNew(Faults,ProbMat,RADSiPNum,NumSensor,state,ConSignFault);
%  Rule base for Expert System to Monitor CAVIS System
%  Written by Joseph Bowling
%  3/11/03
%  Last Modified June 5,2003
%
%  Rules based on Hierarchical System
%
%  Faults = Faults detected by Inference
%  RADSiPNum = Radiation Sensor Identification
%  NumSensor = Total number of sensor in CAVIS system
%  state = current state of system (Faulted or Unfaulted) used in print
%      statement
%  previousstate = Previous state of system (Faulted or Unfaulted) used in
%      print state to only print during change of state
%
%  ProbMat = Matrix containing root causes of CAVIS fault
%  previousstate = Previous state of system (Faulted or Unfaulted) used in
%      print state to only print during change of state
%Define necessary variables for rules
[NumData NumSen] = size(Faults);
%Hierarchal Rule Base
if NumSen > 3   %Set the tolerance for the minimum number of sensor to perform hierarchal rule base
   [PCDUSen,SCSen,PCDUSenFail,SCSenFail] = SenCount(Faults,RADSiPNum);
   [ProbMat] = HeirarchalNew(ProbMat,Faults,state,PCDUSen,SCSen,PCDUSenFail,SCSenFail);
end
%Sensor Failures
%Dead Detector Alarm & Warning
for i=1:20
   if (state(14,i)==1)&(state(19,i)==1);
      ProbMat(i+10,4) = 2; %Dead Detector Alarm
   elseif (state(14,i)~=1)&(state(19,i)==1);
```

```
      ProbMat(i+10,4) = 1;   %Dead Detector Warning
   end
end
%Stuck Detector Alarm
for i=1:20
   if (state(14,i) == 1)&(state(19,i)~=1);
      ProbMat(i+10,5) = 2; %Stuck Detector Alarm
   end
end
%Loose Electrical Connection Alarm & Warning (Variance Shift Up)
for i=1:20
   if
(state(4,i)==1)&(state(5,i)==1)&(state(6,i)==1)&(state(13,i)~=1)&(state(14,i)~=1)&(state(15,i)==1)&(state(16,i)==1)&(state(17,i)==
1)&(state(18,i)~=1);
      ProbMat(i+10,6) = 2; %Loose Detector Alarm
   elseif (state(13,i)~=1)&(state(14,i)~=1)&(state(15,i)==1)&(state(16,i)==1)&(state(17,i)==1)&(state(18,i)~=1);
      ProbMat(i+10,6) = 1; %Loose Detector Warning
   elseif (state(4,i)==1)&(state(5,i)==1)&(state(6,i)==1)&(state(13,i)~=1)&(state(14,i)~=1)&(state(18,i)~=1);
      ProbMat(i+10,6) = 1; %Loose Detector Warning
   end
end
%Removal of Stored Nuclear Material Alarm (Mean Shift Down)
for i=1:20
   if
(state(4,i)~=1)&(state(5,i)==1)&(state(6,i)==1)&(state(13,i)~=1)&(state(14,i)~=1)&(state(15,i)~=1)&(state(16,i)==1)&(state(17,i)==
1)&(state(19,i)~=1);
      ProbMat(i+10,7) = 2; %Removal of material alarm
   elseif (state(4,i)~=1)&(state(5,i)==1)&(state(6,i)==1)&(state(13,i)~=1)&(state(14,i)~=1)&(state(19,i)~=1);
      ProbMat(i+10,7) = 1; %Removal of material warning
   elseif (state(13,i)~=1)&(state(14,i)~=1)&(state(15,i)~=1)&(state(16,i)==1)&(state(17,i)==1)&(state(19,i)~=1);
      ProbMat(i+10,7) = 1; %Removal of material warning
   end
end
%Run Test Warning
%for i=1:20
%   if ConSignFault(1,i) > 3;
%      ProbMat(i+10,8) = 1; %Drifting Sensor Warning
%   end
%end
%Drifting Sensor Warning (Mean Shift Down or Mean Shift Up)
for i=1:20
   if ((state(4,i)==1 | state(5,i)==1)&(state(15,i)==1 | state(16,i)==1))&(state(13,i)~=1)&(state(14,i)~=1)&(state(19,i)~=1);
      ProbMat(i+10,9) = 2; %Drifting Sensor Alarm
   elseif ((state(4,i)==1)|(state(5,i)==1)|(state(15,i)==1)|(state(16,i)==1))&(state(13,i)~=1)&(state(14,i)~=1)&(state(19,i)~=1);
      ProbMat(i+10,9) = 1; %Drifting Sensor Warning
   end
end
%Power or CAVIS System Failure
if (NumSensor == NumSen) & (NumSensor > 20);
   ProbMat(1,4) = 1; %Total CAVIS System Failure
end
%External Source in Warehouse Alarm (Mean Shift Up)
if
(NumSen>10)&(((sum(Faults(4,:))>=0.5*NumSen)&(sum(Faults(5,:))==0)|((sum(Faults(15,:))>=0.5*NumSen)&(sum(Faults(16,:))==
0))));
   ProbMat(1,5) = 1;   %Mean shift up in several sensors
end
%Temperature Increase in warehouse inducing fault in sensor (Extreme Mean Shift Up)
if
(NumSen>10)&(((sum(Faults(4,:))>=0.5*NumSen)&(sum(Faults(5,:))==0)|((sum(Faults(15,:))>=0.5*NumSen)&(sum(Faults(16,:))==
0))));
   ProbMat(1,6) = 1;   %Mean shift up in several sensors
end
%Note External Source and Temperature alarms are the same
```

SenCount Function

This function is a portion of the isolation module that determines the number of RADSiP sensor present in the CAVIS system.

```
function [PCDUSen,SCSen,PCDUSenFail,SCSenFail] = SenCount(Faults,RADSiPNum)
% SenCount - Determines the number of RADSiP sensors for every
%     PCDU & Sensor Concentrator
%
% Written by T Jay Harrison & Joseph Bowling
%     April 14, 2003
%     Last Modified June 5, 2003
%
% Faults - Faults in system detected by Inference
% RADSiPNum - Sensor Identification
%
% PCDUSen - Number of sensors for each PCDU
% SCSen - Number of sensors for each Sensor Concentrator
% PCDUSenFail - Number of faulted sensors for each PCDU
% SCSenFail - Number of faulted sensors for each Sensor Concentrator
[NumData NumSensor]=size(RADSiPNum);
PCDUSen=0;
SCSen=0;
PCDUSenFail=0;
SCSenFail=0;
%Identify the number of sensor for each PCDU
counter1 = 1;
counter2 = 0;
counter3 = 1;
for i = 1:NumSensor;
   if RADSiPNum(1,i) == counter1;
      counter2 = counter2 + 1;
   elseif RADSiPNum(1,i) ~= counter1;
      PCDUSen(1,counter3) = counter1;
      PCDUSen(2,counter3) = counter2;
      counter1 = counter1 + 1;
      counter2 = 1;
      counter3 = counter3 + 1;
   end
   PCDUSen(1,counter3) = counter1;
   PCDUSen(2,counter3) = counter2;
end
%Identify the number of sensor for each Sensor Concentrator
counter1 = 1;
counter2 = 1;
counter3 = 0;
counter4 = 1;
for i = 1:NumSensor;
   if (RADSiPNum(1,i) == counter1)&(RADSiPNum(2,i) == counter2);
      counter3 = counter3 + 1;
   elseif (RADSiPNum(1,i) == counter1)&(RADSiPNum(2,i) ~= counter2);
      SCSen(1,counter4) = counter1;
      SCSen(2,counter4) = counter2;
      SCSen(3,counter4) = counter3;
      counter2 = counter2 + 1;
      counter3 = 1;
      counter4 = counter4 + 1;
   elseif (RADSiPNum(1,i) ~= counter1)&(RADSiPNum(2,i) ~= counter2);
      SCSen(1,counter4) = counter1;
      SCSen(2,counter4) = counter2;
      SCSen(3,counter4) = counter3;
      counter1 = counter1 + 1;
      counter2 = 1;
```

```
        counter3 = 1;
        counter4 = counter4 + 1;
    end
    %Needed if only one PCDU & one Sensor Concentrator
    if (counter1 == 1)&(counter2 == 1);
        SCSen(1,counter4) = counter1;
        SCSen(2,counter4) = counter2;
        SCSen(3,counter4) = counter3;
    end
    %Need to enter last data entry
    SCSen(1,counter4) = counter1;
    SCSen(2,counter4) = counter2;
    SCSen(3,counter4) = counter3;
end
[x1 y1] = size(Faults);
counter = zeros(1,4);
counter1 = zeros(4);
for index2 = 1:y1
    steve = Faults(1,index2);
    sammy = Faults(2,index2);
    if steve == 1
        counter(1) = counter(1) + 1;
        if sammy == 1
            counter1(1,1) = counter1(1,1) + 1;
        elseif sammy == 2
            counter1(1,2) = counter1(1,2) + 1;
        elseif sammy == 3
            counter1(1,3) = counter1(1,3) + 1;
        elseif sammy == 4
            counter1(1,4) = counter1(1,4) + 1;
        end
    elseif steve == 2
        counter(2) = counter(2) + 1;
        if sammy == 1
            counter1(2,1) = counter1(2,1) + 1;
        elseif sammy == 2
            counter1(2,2) = counter1(2,2) + 1;
        elseif sammy == 3
            counter1(2,3) = counter1(2,3) + 1;
        elseif sammy == 4
            counter1(2,4) = counter1(2,4) + 1;
        end
    elseif steve == 3
        counter(3) = counter(3) + 1;
        if sammy == 1
            counter1(3,1) = counter1(3,1) + 1;
        elseif sammy == 2
            counter1(3,2) = counter1 (3,2) + 1;
        elseif sammy == 3
            counter1(3,3) = counter1(3,3) + 1;
        elseif sammy == 4
            counter1(3,4) = counter1(3,4) + 1;
        end
    elseif steve == 4
        counter(4) = counter(4) + 1;
        if sammy == 1
            counter1(4,1) = counter1(4,1) + 1;
        elseif sammy == 2
            counter1(4,2) = counter1 (4,2) + 1;
        elseif sammy == 3
            counter1(4,3) = counter1(4,3) + 1;
        elseif sammy == 4
            counter1(4,4) = counter1(4,4) + 1;
        end
    end
end
%PCDUSenFail(1,:) = 1:1;
```

```
for k = 1:4
   PCDUSenFail(1,k) = k;
   PCDUSenFail(2,k) = counter(k);
end
SCSenFail(1,1:4) = 1;
SCSenFail(1,5:8) = 2;
SCSenFail(1,9:12) = 3;
SCSenFail(1,13:16) = 4;
SCSenFail(2,1:4) = 1:4;
SCSenFail(2,5:8) = 1:4;
SCSenFail(2,9:12) = 1:4;
SCSenFail(2,13:16) = 1:4;
r = 1;
for j = 1:4
   for m = 1:4
      SCSenFail(3,r) = counter1(j,m);
      r = r + 1;
   end
end
```

HierarchalNew Function

This function is a portion of the isolation module. It contains the hierarchal portion of the knowledge base.

```
function [ProbMat] = HeirarchalNew(ProbMat,Faults,state,PCDUSen,SCSen,PCDUSenFail,SCSenFail);
%Hierarchal - Fault identification function based on hierarchal rule base
%  Identifies faulty component in CAVIS system if all sensors that
%  correspond to component fail
%  Written by Joseph Bowling, May 5, 2003
%  Last modified June 4, 2003
%
%  Faults - Matrix containing faulty sensor identification and nature of
%         faults
%  RADSiPNumFail - Number of failed sensors
%  PCDUSen - Number of sensors that correspond to each PCDU
%  SCSen - Number of sensors that correspond to each Sensor Concentrator
%  PCDUSenFail - Number of failed sensors that correspond to each PCDU
%  SCSenFail - Number of failed sensors that correspond to each Sensor Concentrator
%
%  Problem - Isolated fault in CAVIS system
%Define necessary variables are function
[NumData NumSen] = size(Faults);
[PCDUx NumPCDU] = size(PCDUSen);
[SCx NumSC] = size(SCSen);
%PCDU Problem Identification
%IF every sensor from a particular PCDU experiences a fault
%Then the problem is in the PCDU
for i = 1:NumPCDU
   if (PCDUSen(2,i) == PCDUSenFail(2,i)) & (PCDUSen(2,i)>20);
      if (sum(Faults(14,:)) >= 80) & (sum(Faults(19,:)) >= 80) & (sum(Faults(20,:)) < 2);
         ProbMat(3,4) = 2;  %PCDU Dead Alarm
      elseif (sum(Faults(14,:)) < 2) & (sum(Faults(19,:)) >= 80) & (sum(Faults(20,:)) < 2);
         ProbMat(3,4) = 1;  %PCDU Dead Warning
      end
      if (sum(Faults(14,:)) >= 80) & (sum(Faults(19,:)) < 2) & (sum(Faults(20,:)) < 2);
         ProbMat(3,5) = 2;  %PCDU Stuck Alarm
      end
      if
(sum(Faults(4,:))>=60)&(sum(Faults(5,:))>=60)&(sum(Faults(6,:))>=60)&(sum(Faults(13,:))<2)&(sum(Faults(14,:))<2)&(sum(Faults
(15,:))>=60)&(sum(Faults(16,:))>=60)&(sum(Faults(17,:))>=60)&(sum(Faults(19,:))<2);
```

```
         ProbMat(3,6) = 2;  %PCDU loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 80)
         ProbMat(3,7) = 2;  %PCDU Unplugged, lost communication Alarm
      end
      ProbMat(3,8) = 1;  %PCDU has failed, all corresponding sensors have faults
   end
end
%Sensor Concentrator Fault
%IF every sensor from a particular sensor concentration experiences a fault
%Then the problem is in the Sensor Concentrator
for i = 1:NumSC
   if (SCSen(3,i) == SCSenFail(3,i)) & (SCSen(3,i) > 10);
      if (sum(Faults(14,:)) == 20) & (sum(Faults(19,:)) == 20) & (sum(Faults(20,:)) < 2)
         ProbMat(4,4) = 2;  %Sen Conc Dead Alarm
      elseif (sum(Faults(14,:)) < 2) & (sum(Faults(19,:)) == 20) & (sum(Faults(20,:)) < 2)
         ProbMat(4,4) = 1;  %Sen Conc Dead Warning
      end
      if (sum(Faults(14,:)) == 20) & (sum(Faults(19,:)) < 2) & (sum(Faults(20,:)) < 2);
         ProbMat(4,5) = 2;  %Sen Conc Stuck Alarm
      end
      if
(sum(Faults(4,:))>=15)&(sum(Faults(5,:))>=15)&(sum(Faults(6,:))>=15)&(sum(Faults(13,:))<2)&(sum(Faults(14,:))<2)&(sum(Faults
(15,:))>=15)&(sum(Faults(16,:))>=15)&(sum(Faults(17,:))>=15)&(sum(Faults(19,:))<2);
         ProbMat(4,6) = 2;  %Sen Conc loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 20)
         ProbMat(4,7) = 2;  %Sen Conc Unplugged, lost communication Alarm
      end
      ProbMat(4,8) = 1;  %Sen Conc has failed, all corresponding sensors have faults
   end
end
%Sensor Concentrator Board Fault
%IF 1/2 of the sensors from a particular sensor concentration corresponding to a SC board experiences a fault
%Then the problem is in the Sensor Concentrator Board May need floor or
%ceil in front of (1/2)*SCSen(3,i)
%Communication Board 1
for i = 1:NumSC
   if ((1/2)*SCSen(3,i) == (SCSenFail(3,i))) & (SCSen(3,i) > 5) & (sum(Faults(3,:)) == 55);
      if (sum(Faults(14,:)) >= 10) & (sum(Faults(19,:)) >= 10) & (sum(Faults(20,:)) < 1);
         ProbMat(5,4) = 2;  %Sen Conc Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 10) & (sum(Faults(20,:)) < 1);
         ProbMat(5,4) = 1;  %Sen Conc Dead Warning
      end
      if (sum(Faults(14,:)) >= 10) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
         ProbMat(5,5) = 2;  %Sen Conc Stuck Alarm
      end
      if
(sum(Faults(4,:))>=7)&(sum(Faults(5,:))>=7)&(sum(Faults(6,:))>=7)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=7)&(sum(Faults(16,:))>=7)&(sum(Faults(17,:))>=7)&(sum(Faults(19,:))<1);
         ProbMat(5,6) = 2;  %Sen Conc loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 20)
         ProbMat(5,7) = 2;  %Sen Conc Unplugged, lost communication Alarm
      end
      ProbMat(5,8) = 1;  %Sen Conc has failed, all corresponding sensors have faults
   end
end
%Communication Board 2
for i = 1:NumSC
   if ((1/2)*SCSen(3,i) == (SCSenFail(3,i))) & (SCSen(3,i) > 5) & (sum(Faults(3,:)) == 155);
      if (sum(Faults(14,:)) >= 10) & (sum(Faults(19,:)) >= 10) & (sum(Faults(20,:)) < 1);
         ProbMat(6,4) = 2;  %Sen Conc Comm Board Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 10) & (sum(Faults(20,:)) < 1);
         ProbMat(6,4) = 1;  %Sen Conc Comm Board Dead Warning
      end
      if (sum(Faults(14,:)) >= 10) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
```

```
        ProbMat(6,5) = 2;   %Sen Conc Comm Board Stuck Alarm
      end
      if
(sum(Faults(4,:))>=7)&(sum(Faults(5,:))>=7)&(sum(Faults(6,:))>=7)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=7)&(sum(Faults(16,:))>=7)&(sum(Faults(17,:))>=7)&(sum(Faults(19,:))<1);
        ProbMat(6,6) = 2;   %Sen Conc Comm Board loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 10)
        ProbMat(6,7) = 2;   %Sen Conc Comm Board Unplugged, lost communication Alarm
      end
      ProbMat(6,8) = 1;   %Sen Conc Comm Board has failed, all corresponding sensors have faults
    end
end
%Sensor Concentrator Board Fault
%IF 1/4 of the sensors from a particular sensor concentration corresponding to a SC board experiences a fault
%Then the problem is in the Sensor Concentrator Board, May need floor or
%ceil in front of (1/4)*SCSen(3,i)
%Process Board 1
for i = 1:NumSC
    if ((1/4)*SCSen(3,i) == (SCSenFail(3,i))) & (sum(Faults(3,:)) == 15);
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(7,4) = 2;   %Sen Conc Process Board Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(7,4) = 1;   %Sen Conc Process Board Dead Warning
      end
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
        ProbMat(7,5) = 2;   %Sen Conc Process Board Stuck Alarm
      end
      if
(sum(Faults(4,:))>=3)&(sum(Faults(5,:))>=3)&(sum(Faults(6,:))>=3)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=3)&(sum(Faults(16,:))>=3)&(sum(Faults(17,:))>=3)&(sum(Faults(19,:))<1);
        ProbMat(7,6) = 2;   %Sen Conc Process Board loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 5)
        ProbMat(7,7) = 2;   %Sen Conc Process Board Unplugged, lost communication Alarm
      end
      ProbMat(7,8) = 1;   %Sen Conc Process Board has failed, all corresponding sensors have faults
    end
end
%Process Board 2
for i = 1:NumSC
    if ((1/4)*SCSen(3,i) == (SCSenFail(3,i))) & (sum(Faults(3,:)) == 40);
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(8,4) = 2;   %Sen Conc Process Board Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(8,4) = 1;   %Sen Conc Process Board Dead Warning
      end
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
        ProbMat(8,5) = 2;   %Sen Conc Process Board Stuck Alarm
      end
      if
(sum(Faults(4,:))>=3)&(sum(Faults(5,:))>=3)&(sum(Faults(6,:))>=3)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=3)&(sum(Faults(16,:))>=3)&(sum(Faults(17,:))>=3)&(sum(Faults(19,:))<1);
        ProbMat(8,6) = 2;   %Sen Conc Process Board loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 5)
        ProbMat(8,7) = 2;   %Sen Conc Process Board Unplugged, lost communication Alarm
      end
      ProbMat(8,8) = 1;   %Sen Conc Process Board has failed, all corresponding sensors have faults
    end
end
%Process Board 3
for i = 1:NumSC
    if ((1/4)*SCSen(3,i) == (SCSenFail(3,i))) & (sum(Faults(3,:)) == 65);
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(9,4) = 2;   %Sen Conc Process Board Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
```

```
        ProbMat(9,4) = 1;  %Sen Conc Process Board Dead Warning
      end
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
        ProbMat(9,5) = 2;  %Sen Conc Process Board Stuck Alarm
      end
      if
(sum(Faults(4,:))>=3)&(sum(Faults(5,:))>=3)&(sum(Faults(6,:))>=3)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=3)&(sum(Faults(16,:))>=3)&(sum(Faults(17,:))>=3)&(sum(Faults(19,:))<1);
        ProbMat(9,6) = 2;  %Sen Conc Process Board loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 5)
        ProbMat(9,7) = 2;  %Sen Conc Process Board Unplugged, lost communication Alarm
      end
      ProbMat(9,8) = 1;  %Sen Conc Process Board has failed, all corresponding sensors have faults
   end
end
%Process Board 4
for i = 1:NumSC
   if ((1/4)*SCSen(3,i) == (SCSenFail(3,i))) & (sum(Faults(3,:)) == 90);
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(10,4) = 2;  %Sen Conc Process Board Dead Alarm
      elseif (sum(Faults(14,:)) < 1) & (sum(Faults(19,:)) >= 5) & (sum(Faults(20,:)) < 1);
        ProbMat(10,4) = 1;  %Sen Conc Process Board Dead Warning
      end
      if (sum(Faults(14,:)) >= 5) & (sum(Faults(19,:)) < 1) & (sum(Faults(20,:)) < 1);
        ProbMat(10,5) = 2;  %Sen Conc Process Board Stuck Alarm
      end
      if
(sum(Faults(4,:))>=3)&(sum(Faults(5,:))>=3)&(sum(Faults(6,:))>=3)&(sum(Faults(13,:))<1)&(sum(Faults(14,:))<1)&(sum(Faults(15,
:))>=3)&(sum(Faults(16,:))>=3)&(sum(Faults(17,:))>=3)&(sum(Faults(19,:))<1);
        ProbMat(10,6) = 2;  %Sen Conc Process Board loose wire or connection Alarm
      end
      if (sum(Faults(20,:)) >= 5)
        ProbMat(10,7) = 2;  %Sen Conc Process Board Unplugged, lost communication Alarm
      end
      ProbMat(10,8) = 1;  %Sen Conc Process Board has failed, all corresponding sensors have faults
   end
end
```

Inference Function

This function acts as the inference engine for the expert system. It determines which rules should fire by calling an additional sub inference engine specific to the CAVIS system state.

```
function [Problem,Logic,ColorIdent] = Inference(ProbMat,PreProbMat,ProbStatus);
%Inference - Determines which rules fire for the expert system
%   Written by Joseph Bowling
%   11-13-03
PInd = 1;
Problem = {'0' '0'};
Logic = {'0'};
ColorIdent = 0;
%Dead Sensor Inference Engine
if any(ProbStatus(:,4));
   [Problem,Logic,ColorIdent,PInd] = DeadInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%Stuck Sensor Inference Engine
if any(ProbStatus(:,5));
```

```
   [Problem,Logic,ColorIdent,PInd] = StuckInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%Loose Wire Inference Engine
if any(ProbStatus(:,6));
   [Problem,Logic,ColorIdent,PInd] = LooseWireInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%Unplugged Component Inference Engine
if any(ProbStatus(3:10,7));
   [Problem,Logic,ColorIdent,PInd] = UnpluggedInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%Removal of SNM
if any(ProbStatus(11:30,7));
   [Problem,Logic,ColorIdent,PInd] = SNMRemoveInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%Component Failure Inference Engine
%if any(ProbStatus(3:10,8)) & (max(max(ProbStatus(3:10,3:7)))>0);
%   [Problem,Logic,ColorIdent,PInd] = CompFailInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%end
%Sensor Drift Run Test
%if any(ProbStatus(11:30,8));
%   [Problem,Logic,ColorIdent,PInd] = DriftRunInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%end
%Sensor Drift SPRT
if any(ProbStatus(11:30,9));
   [Problem,Logic,ColorIdent,PInd] = DriftSPRTInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
end
%CAVIS warehouse or system faults
if any(ProbStatus(1,:))
   if (ProbMat(1,4)==1) & (ProbStatus(1,4)~=0);
      Problem(PInd,:) = {'CAVIS has failed'};
      Logic(Pind,:) = {'CAVIS has failed because all sensors have failed'};
      ColorIdent(Pind,:) = 2;
      PInd = PInd + 1;
   end
   if (ProbMat(1,4)==0) & (ProbStatus(1,4)==-1);
      Problem(PInd,:) = {'CAVIS is no longer failing'};
      Logic(Pind,:) = {'All or some of the CAVIS sensors are functional'};
      ColorIdent(Pind,:) = 3;
      PInd = PInd + 1;
   end
   if (ProbMat(1,5)==1) & (ProbStatus(1,5)~=0);
      Problem(PInd,:) = {'Warning: An external radiation source may be in warehouse'};
      Logic(Pind,:) = {'1/2 of the CAVIS sensors have experienced a mean shift up'};
      ColorIdent(Pind,:) = 1;
      PInd = PInd + 1;
   end
   if (ProbMat(1,5)==0) & (ProbStatus(1,5)==1);
      Problem(PInd,:) = {'An external radiation source is no longer detected'};
      Logic(Pind,:) = {'A mean shift up is no longer detected in 1/2 of the CAVIS sensors'};
      ColorIdent(Pind,:) = 3;
      PInd = PInd + 1;
   end
   %if (ProbMat(1,6)==1) & (ProbStatus(1,6)~=0);
   %   Problem(PInd,:) = {'Warning: The environ. cond. may have induced CAVIS drifts'};
   %   Logic(Pind,:) = {'1/2 of the CAVIS sensors have experienced a mean shift up'};
   %   ColorIdent(Pind,:) = 1;
   %   PInd = PInd + 1;
   %end
   %if (ProbMat(1,6)==0) & (ProbStatus(1,6)==-1);
   %   Problem(PInd,:) = {'The environ. cond. drifts are no longer detected'};
   %   Logic(Pind,:) = {'A mean shift up is no longer detected in 1/2 of the CAVIS sensors'};
   %   ColorIdent(Pind,:) = 3;
   %   PInd = PInd + 1;
   %end
end
```

DeadInference Function

This function acts as a sub inference engine for the detection of dead or failed CAVIS components.

```
function [Problem,Logic,ColorIdent,PInd] = DeadInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%DeadInference - Inference Engine for the dead sensors
%  Written by Joseph Bowling
%  11/13/03
if (ProbMat(3,4) == 2) & (ProbStatus(3,4) ~= 0);
   Problem(PInd,:) = {'The PCUD is dead: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the PCDU for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(3,4) == 1) & (ProbStatus(3,4) ~= 0);
   Problem(PInd,:) = {'Warning: The PCUD may be dead: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the Sen. Conc.'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(3,4) == 0) & ((ProbStatus(3,4) == -1) | (ProbStatus(3,4) == -2));
   Problem(PInd,:) = {'The PCUD is no longer dead: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'The sensors corresponding to the PCDU no longer have a zero count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(4,4) == 2) & (ProbStatus(4,4) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. is dead: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the Sen. Conc. for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(4,4) == 1) & (ProbStatus(4,4) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. may be dead: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the Sen. Conc.'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(4,4) == 0) & ((ProbStatus(4,4) == -1) | (ProbStatus(4,4) == -2));
   Problem(PInd,:) = {'The Sensor Conc. is no longer dead: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the Sen. Conc. no longer have a zero count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(5,4) == 2) & (ProbStatus(5,4) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 is dead: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(5,4) == 1) & (ProbStatus(5,4) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. Comm. Board 1 may be dead: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(5,4) == 0) & ((ProbStatus(5,4) == -1) | (ProbStatus(5,4) == -2));
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 is no longer dead: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(6,4) == 2) & (ProbStatus(6,4) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 is dead: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(6,4) == 1) & (ProbStatus(6,4) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. Comm. Board 2 may be dead: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 1;
```

```
    PInd = PInd + 1;
elseif (ProbMat(6,4) == 0) & ((ProbStatus(6,4) == -1) | (ProbStatus(6,4) == -2));
    Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 is no longer dead: ' num2str(ProbMat(6,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(7,4) == 2) & (ProbStatus(7,4) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 is dead: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(7,4) == 1) & (ProbStatus(7,4) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 1 may be dead: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(7,4) == 0) & ((ProbStatus(7,4) == -1) | (ProbStatus(7,4) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 1 is no longer dead: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(8,4) == 2) & (ProbStatus(8,4) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 is dead: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(8,4) == 1) & (ProbStatus(8,4) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 2 may be dead: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(8,4) == 0) & ((ProbStatus(8,4) == -1) | (ProbStatus(8,4) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 2 is no longer dead: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(9,4) == 2) & (ProbStatus(9,4) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 is dead: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(9,4) == 1) & (ProbStatus(9,4) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 3 may be dead: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(9,4) == 0) & ((ProbStatus(9,4) == -1) | (ProbStatus(9,4) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 3 is no longer dead: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(10,4) == 2) & (ProbStatus(10,4) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 is dead: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board for 5 cons. obs.'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(10,4) == 1) & (ProbStatus(10,4) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 4 may be dead: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'Zero count rate for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(10,4) == 0) & ((ProbStatus(10,4) == -1) | (ProbStatus(10,4) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 4 is no longer dead: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a zero count rate'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif any(ProbMat(11:30,4)) | any(ProbStatus(11:30,4))
```

```
  for k = 1:20
    if (ProbMat(k+10,4) == 2) & (ProbStatus(k+10,4) ~= 0)
      Problem(PInd,:) = {'The RADSiP sensor is dead: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'Zero count rate for the sensor for 5 cons. obs.'};
      ColorIdent(PInd,:) = 2;
      PInd = PInd + 1;
    end
    if (ProbMat(k+10,4) == 1) & (ProbStatus(k+10,4) ~= 0)
      Problem(PInd,:) = {'Warning: The RADSiP sensor may be dead: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'Zero count rate for the sensor'};
      ColorIdent(PInd,:) = 1;
      PInd = PInd + 1;
    end
    if (ProbMat(k+10,4) == 0) & ((ProbStatus(k+10,4) == -1) | (ProbStatus(k+10,4) == -2));
      Problem(PInd,:) = {'The RADSiP sensor is no longer dead: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'The sensors no longer has a zero count rate'};
      ColorIdent(PInd,:) = 3;
      PInd = PInd + 1;
    end
  end
end
```

StuckInference Function

This function acts as a sub inference engine for the detection of stuck CAVIS components.

```
function [Problem,Logic,ColorIdent,PInd] = StuckInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%StuckInference - Inference Engine for the stuck sensors
%  Written by Joseph Bowling
%  11/13/03
if (ProbMat(3,5) == 2) & (ProbStatus(3,5) ~= 0);
  Problem(PInd,:) = {'The PCUD is stuck: ' num2str(ProbMat(3,1))};
  Logic(PInd,:) = {'All sensor corresponding to the PCDU have same count rate for 5 cons. obs.'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(3,5) == 0) & (ProbStatus(3,5) == -2);
  Problem(PInd,:) = {'The PCUD is no longer stuck: ' num2str(ProbMat(3,1))};
  Logic(PInd,:) = {'The sensors corresponding to the PCDU no longer have the same count rate'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
elseif (ProbMat(4,5) == 2) & (ProbStatus(4,5) ~= 0);
  Problem(PInd,:) = {'The Sensor Conc. is stuck: ' num2str(ProbMat(4,1:2))};
  Logic(PInd,:) = {'All sensor corresponding to the Sen. Conc. have same count rate for 5 cons. obs.'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(4,5) == 0) & (ProbStatus(4,5) == -2);
  Problem(PInd,:) = {'The Sensor Conc. is no longer stuck: ' num2str(ProbMat(4,1:2))};
  Logic(PInd,:) = {'The sensors corresponding to the Sen. Conc. no longer have the same count rate'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
elseif (ProbMat(5,5) == 2) & (ProbStatus(5,5) ~= 0);
  Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 is stuck: ' num2str(ProbMat(5,1:2))};
  Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(5,5) == 0) & (ProbStatus(5,5) == -2);
  Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 is no longer stuck: ' num2str(ProbMat(5,1:2))};
  Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
```

```
elseif (ProbMat(6,5) == 2) & (ProbStatus(6,5) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 is stuck: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(6,5) == 0) & (ProbStatus(6,5) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 is no longer stuck: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(7,5) == 2) & (ProbStatus(7,5) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 is stuck: ' num2str(ProbMat(7,1:2))};
   Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(7,5) == 0) & (ProbStatus(7,5) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 is no longer stuck: ' num2str(ProbMat(7,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(8,5) == 2) & (ProbStatus(8,5) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 is stuck: ' num2str(ProbMat(8,1:2))};
   Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(8,5) == 0) & (ProbStatus(8,5) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 is no longer stuck: ' num2str(ProbMat(8,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(9,5) == 2) & (ProbStatus(9,5) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 is stuck: ' num2str(ProbMat(9,1:2))};
   Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(9,5) == 0) & (ProbStatus(9,5) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 is no longer stuck: ' num2str(ProbMat(9,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(10,5) == 2) & (ProbStatus(10,5) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 is stuck: ' num2str(ProbMat(10,1:2))};
   Logic(PInd,:) = {'All sensor corresponding to the board have same count rate for 5 cons. obs.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(10,5) == 0) & (ProbStatus(10,5) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 is no longer stuck: ' num2str(ProbMat(10,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have the same count rate'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif any(ProbMat(11:30,5)) | any(ProbStatus(11:30,5))
   for k = 1:20
      if (ProbMat(k+10,5) == 2) & (ProbStatus(k+10,5) ~= 0)
         Problem(PInd,:) = {'The RADSiP sensor is stuck: ' num2str(ProbMat(10+k,1:3))};
         Logic(PInd,:) = {'The sensor had the same count rate for 5 cons. obs.'};
         ColorIdent(PInd,:) = 2;
         PInd = PInd + 1;
      end
      if (ProbMat(k+10,5) == 0) & (ProbStatus(k+10,5) == -2)
         Problem(PInd,:) = {'The RADSiP sensor is no longer stuck: ' num2str(ProbMat(10+k,1:3))};
         Logic(PInd,:) = {'The sensor no longer had the same count rate'};
         ColorIdent(PInd,:) = 3;
         PInd = PInd + 1;
      end
   end
end
```

LooseWireInference Function

This function acts as a sub inference engine for the detection of loose and damage wire connections of CAVIS components.

```
function [Problem,Logic,ColorIdent,PInd] = LooseWireInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%LooseWireInference - Inference Engine for the loose wire sensors
%  Written by Joseph Bowling
%  11/13/03
if (ProbMat(3,6) == 2) & (ProbStatus(3,6) ~= 0);
   Problem(PInd,:) = {'The PCUD has a loose wire conn.: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the PCDU'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(3,6) == 1) & (ProbStatus(3,6) ~= 0);
   Problem(PInd,:) = {'Warning: The PCUD may have a loose wire conn.: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the PCDU'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(3,6) == 0) & ((ProbStatus(3,6) == -1)|(ProbStatus(3,6) == -2));
   Problem(PInd,:) = {'The PCUD no longer has a loose wire conn.: ' num2str(ProbMat(3,1))};
   Logic(PInd,:) = {'The sensors corresponding to the PCDU no longer have a count rate variance inc.'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(4,6) == 2) & (ProbStatus(4,6) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. has a loose wire conn.: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the Sen. Conc.'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(4,6) == 1) & (ProbStatus(4,6) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. may have a loose wire conn.: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the Sen. Conc.'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(4,6) == 0) & ((ProbStatus(4,6) == -1)|(ProbStatus(4,6) == -2));
   Problem(PInd,:) = {'The Sensor Conc. no longer has a loose wire conn.: ' num2str(ProbMat(4,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the Sen. Conc. no longer have a count rate variance inc.'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(5,6) == 2) & (ProbStatus(5,6) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 has a loose wire conn.: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(5,6) == 1) & (ProbStatus(5,6) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. Comm. Board 1 may have a loose wire conn.: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 1;
   PInd = PInd + 1;
elseif (ProbMat(5,6) == 0) & ((ProbStatus(5,6) == -1)|(ProbStatus(5,6) == -2));
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 no longer has a loose wire conn.: ' num2str(ProbMat(5,1:2))};
   Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(6,6) == 2) & (ProbStatus(6,6) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 has a loose wire conn.: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(6,6) == 1) & (ProbStatus(6,6) ~= 0);
   Problem(PInd,:) = {'Warning: The Sensor Conc. Comm. Board 2 may have a loose wire conn.: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 1;
```

```
    PInd = PInd + 1;
elseif (ProbMat(6,6) == 0) & ((ProbStatus(6,6) == -1)|(ProbStatus(6,6) == -2));
    Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 no longer has a loose wire conn.: ' num2str(ProbMat(6,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(7,6) == 2) & (ProbStatus(7,6) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 has a loose wire conn.: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(7,6) == 1) & (ProbStatus(7,6) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 1 may have a loose wire conn.: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(7,6) == 0) & ((ProbStatus(7,6) == -1)|(ProbStatus(7,6) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 1 no longer has a loose wire conn.: ' num2str(ProbMat(7,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(8,6) == 2) & (ProbStatus(8,6) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 has a loose wire conn.: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(8,6) == 1) & (ProbStatus(8,6) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 2 may have a loose wire conn.: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(8,6) == 0) & ((ProbStatus(8,6) == -1)|(ProbStatus(8,6) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 2 no longer has a loose wire conn.: ' num2str(ProbMat(8,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(9,6) == 2) & (ProbStatus(9,6) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 has a loose wire conn.: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(9,6) == 1) & (ProbStatus(9,6) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 3 may have a loose wire conn.: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(9,6) == 0) & ((ProbStatus(9,6) == -1)|(ProbStatus(9,6) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 3 no longer has a loose wire conn.: ' num2str(ProbMat(9,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif (ProbMat(10,6) == 2) & (ProbStatus(10,6) ~= 0);
    Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 has a loose wire conn.: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 2;
    PInd = PInd + 1;
elseif (ProbMat(10,6) == 1) & (ProbStatus(10,6) ~= 0);
    Problem(PInd,:) = {'Warning: The Sensor Process Board 4 may have a loose wire conn.: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'Count rate variance inc. for all sensors corresponding to the board'};
    ColorIdent(PInd,:) = 1;
    PInd = PInd + 1;
elseif (ProbMat(10,6) == 0) & ((ProbStatus(10,6) == -1)|(ProbStatus(10,6) == -2));
    Problem(PInd,:) = {'The Sensor Process Board 4 no longer has a loose wire conn.: ' num2str(ProbMat(10,1:2))};
    Logic(PInd,:) = {'The sensors corresponding to the board no longer have a count rate variance inc.'};
    ColorIdent(PInd,:) = 3;
    PInd = PInd + 1;
elseif any(ProbMat(11:30,6)) | any(ProbStatus(11:30,6))
```

```
  for k = 1:20
    if (ProbMat(k+10,6) == 2) & (ProbStatus(k+10,6) ~= 0)
      Problem(PInd,:) = {'The RADSiP sensor has a loose wire conn.: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'Count rate variance inc. for the sensor'};
      ColorIdent(PInd,:) = 2;
      PInd = PInd + 1;
    end
    if (ProbMat(k+10,6) == 1) & (ProbStatus(k+10,6) ~= 0)
      Problem(PInd,:) = {'Warning: The RADSiP sensor may have a loose wire conn.: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'Count rate variance inc. for the sensors'};
      ColorIdent(PInd,:) = 1;
      PInd = PInd + 1;
    end
    if (ProbMat(k+10,6) == 0) & ((ProbStatus(k+10,6) == -1)|(ProbStatus(k+10,6) == -2));
      Problem(PInd,:) = {'The RADSiP sensor no longer has a loose wire conn.: ' num2str(ProbMat(10+k,1:3))};
      Logic(PInd,:) = {'The sensor no longer has a count rate variance inc.'};
      ColorIdent(PInd,:) = 3;
      PInd = PInd + 1;
    end
  end
end
```

UnpluggedInference Function

This function acts as a sub inference engine for the detection of unplugged CAVIS components.

```
function [Problem,Logic,ColorIdent,PInd] = UnpluggedInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%UnpluggedInference - Inference Engine for unplugged component failure
%  Written by Joseph Bowling
%  11/13/03
if (ProbMat(3,7) == 2) & (ProbStatus(3,7) ~= 0);
  Problem(PInd,:) = {'The PCUD has been unplugged: ' num2str(ProbMat(3,1))};
  Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the PCDU'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(3,7) == 0) & (ProbStatus(3,7) == -2);
  Problem(PInd,:) = {'The PCUD is no longer unplugged: ' num2str(ProbMat(3,1))};
  Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the PCDU'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
elseif (ProbMat(4,7) == 2) & (ProbStatus(4,7) ~= 0);
  Problem(PInd,:) = {'The Sensor Conc. has been unplugged: ' num2str(ProbMat(4,1:2))};
  Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the Sen. Conc.'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(4,7) == 0) & (ProbStatus(4,7) == -2);
  Problem(PInd,:) = {'The Sensor Conc. is no longer unplugged: ' num2str(ProbMat(4,1:2))};
  Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the Sen. Conc.'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
elseif (ProbMat(5,7) == 2) & (ProbStatus(5,7) ~= 0);
  Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 has been unplugged: ' num2str(ProbMat(5,1:2))};
  Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
  ColorIdent(PInd,:) = 2;
  PInd = PInd + 1;
elseif (ProbMat(5,7) == 0) & (ProbStatus(5,7) == -2);
  Problem(PInd,:) = {'The Sensor Conc. Comm. Board 1 is no longer unplugged: ' num2str(ProbMat(5,1:2))};
  Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
  ColorIdent(PInd,:) = 3;
  PInd = PInd + 1;
```

```
elseif (ProbMat(6,7) == 2) & (ProbStatus(6,7) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 has been unplugged: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(6,7) == 0) & (ProbStatus(6,7) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Comm. Board 2 is no longer unplugged: ' num2str(ProbMat(6,1:2))};
   Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(7,7) == 2 ) & (ProbStatus(7,7) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 has been unplugged: ' num2str(ProbMat(7,1:2))};
   Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(7,7) == 0) & (ProbStatus(7,7) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 1 is no longer unplugged: ' num2str(ProbMat(7,1:2))};
   Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(8,7) == 2 ) & (ProbStatus(8,7) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 has been unplugged: ' num2str(ProbMat(8,1:2))};
   Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(8,7) == 0) & (ProbStatus(8,7) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 2 is no longer unplugged: ' num2str(ProbMat(8,1:2))};
   Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(9,7) == 2 ) & (ProbStatus(9,7) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 has been unplugged: ' num2str(ProbMat(9,1:2))};
   Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(9,7) == 0) & (ProbStatus(9,7) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 3 is no longer unplugged: ' num2str(ProbMat(9,1:2))};
   Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
elseif (ProbMat(10,7) == 2) & (ProbStatus(10,7) ~= 0);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 has been unplugged: ' num2str(ProbMat(10,1:2))};
   Logic(PInd,:) = {'The communication is lost for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 2;
   PInd = PInd + 1;
elseif (ProbMat(10,7) == 0) & (ProbStatus(10,7) == -2);
   Problem(PInd,:) = {'The Sensor Conc. Process. Board 4 is no longer unplugged: ' num2str(ProbMat(10,1:2))};
   Logic(PInd,:) = {'The communication has been restored for all sensors corresponding to the board'};
   ColorIdent(PInd,:) = 3;
   PInd = PInd + 1;
end
```

SNMRemoveInference Function

This function acts as a sub inference engine for the detection of the removal of SNM from the CAVIS system.

```
function [Problem,Logic,ColorIdent,PInd] = SNMRemoveInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%SNMRemoveInference - Inference Engine for the removal of SNM
```

```
% Written by Joseph Bowling
% 11/13/03
for k = 1:20
  if (ProbMat(k+10,7) == 2) & (ProbStatus(k+10,7) ~= 0)
     Problem(PInd,:) = {'Possible removal of SNM at : ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensor has experienced a mean shift down in count rate'};
     ColorIdent(PInd,:) = 2;
     PInd = PInd + 1;
  end
  if (ProbMat(k+10,7) == 1) & (ProbStatus(k+10,7) ~= 0)
     Problem(PInd,:) = {'Warning: The SNM may have been removed: ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensor has experienced a mean shift down in count rate'};
     ColorIdent(PInd,:) = 1;
     PInd = PInd + 1;
  end
  if (ProbMat(k+10,7) == 0) & ((ProbStatus(k+10,7) == -1) | (ProbStatus(k+10,7) == -2))
     Problem(PInd,:) = {'The SNM removal state no longer exists: ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensor is no longer experiencing a mean shift down in count rate'};
     ColorIdent(PInd,:) = 3;
     PInd = PInd + 1;
  end
end
```

DriftSPRTInference Function

This function acts as a sub inference engine for the detection of drifting RADSiP radiation sensors.

```
function [Problem,Logic,ColorIdent,PInd] = DriftSPRTInference(ProbMat,ProbStatus,Problem,Logic,ColorIdent,PInd);
%DriftSPRTInference - Inference Engine for drifting sensors by SPRT
% Written by Joseph Bowling
% 11/13/03
for k = 1:20
  if (ProbMat(k+10,9) == 2) & (ProbStatus(k+10,9) ~= 0) & (max(ProbMat(k+10,4:7))==0);
     Problem(PInd,:) = {'The RADSiP sensor is drifting: ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensors is experiencing SPRT alarms'};
     ColorIdent(PInd,:) = 2;
     PInd = PInd + 1;
  end
  if (ProbMat(k+10,9) == 1) & (ProbStatus(k+10,9) ~= 0) & (max(ProbMat(k+10,4:7))==0);
     Problem(PInd,:) = {'Warning: The RADSiP sensor may be drifting: ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensors is experiencing SPRT alarms'};
     ColorIdent(PInd,:) = 1;
     PInd = PInd + 1;
  end
  if (ProbMat(k+10,9) == 0) & ((ProbStatus(k+10,9)==-1)|(ProbStatus(k+10,9) == -2)) & (max(ProbMat(k+10,4:7))==0);
     Problem(PInd,:) = {'The RADSiP sensor is no longer drifting: ' num2str(ProbMat(10+k,1:3))};
     Logic(PInd,:) = {'The sensors is no longer experiencing SPRT alarms'};
     ColorIdent(PInd,:) = 3;
     PInd = PInd + 1;
  end
end
```

ExcelNew Function

This function acts as a communication module between the CAVIS monitoring system and the EXCEL database.

```
function [PExInd] = ExcelESNew(Problem,Logic,ColorIdent,H,PExInd,ch7,index);
%ExcelESNew - script that writes the problem to the excel database.
%   Written by Joseph Bowling
%   11-13-03
for m = 1:length(Problem(:,1));
   PExInd = PExInd + 1;
   R = num2str(PExInd);
   TimePos = ['r' R 'c1'];
   IndPos = ['r' R 'c2'];
   ProPos = ['r' R 'c3'];
   LogPos = ['r' R 'c4'];
   %ResPos = ['r' R 'c5'];
   ColPos = ['r' R 'c6'];
   ProbChar = char(Problem(m,:));
   LogChar = char(Logic(m,:));
   %ResChar = char(Response(m,:));
   rc = ddepoke(ch7,TimePos,H);
   rc = ddepoke(ch7,IndPos,index);
   rc = ddepoke(ch7,ProPos,ProbChar');
   rc = ddepoke(ch7,LogPos,LogChar');
   %rc = ddepoke(ch7,ResPos,LogChar');
   rc = ddepoke(ch7,ColPos,ColorIdent(m,:));
end
```

RAMM Code

This script initiates and runs a demonstration of the RAMM.

```
%NeighborRes - Script file that runs the neighborhood system using the
%residual of the sensor.
%   Written by Joseph Bowling
%   7/21/03
clear all;
%Define the Warehouse
VaultRow = 4;
VaultCol = 5;
VaultHeight = input('Enter the height the concentrators are stacked to: ');
WareX = input('Enter the number of vaults (x coordinate) in the warehouse: ');
WareY = input('Enter the number of vaults (y coordinate) in the warehouse: ');
%Create the Sensor Space
[Space] = FabricateSen(VaultRow,VaultCol,VaultHeight,WareX,WareY);
%Create the radiation signal
time = input('Enter the time length of the demo: ');
[t,mu1] = FabricateSignal(VaultRow,VaultCol,VaultHeight,WareX,WareY,time);
%Define the fault to induce in the warehouse
choice = input('Simulate a source of mean shift up in the warehouse: 1:Source, 2:Crash Spike: ');
if choice == 1
   faultindex = input('Enter the index the external source appears in the warehouse: ');
   source = input('Enter the x and y coordinate of the source in sensor units [x y z]: ');
   [tex] = extsource(time);
   %Define the parameters for the moving source
   XS = [1;VaultRow*WareX];XY = [1;VaultCol*WareY];
   [centerspace] = centroid([XS XY]);
   Arrive = 0;
end
```

```
if choice == 2
   faultindex = input('Enter the index the crash occurs in the warehouse: ');
   Crash = input('Enter the SC vault stack the crash occurred in [WareX WareY]: ');
   if (Crash(1,1)>WareX) | (Crash(1,2)>WareY)
      error('Spike is outside dimension of Warehouse');
   end
   A = (Crash(:,1) - 1) * VaultRow +1;
   B = Crash(:,1) * VaultRow;
   C = (Crash(:,2) - 1) * VaultCol +1;
   D = Crash(:,2) * VaultCol;
   Ind = 1;
end
%Kernel Smoothing Parameters
width = input('Enter the Kernel Smoothing Width: ');
Vresolution = input('Enter the vertical resolution of the Kernel Smoothing ');
Hresolution = 1; %input('Enter the resolution of the Kernel Smoothing');
[gauss,Interest] = LoadGauss(VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,Hresolution,width);
%Define plotting parameters and loop
[ts ty] = size(t);
x = [1:1:VaultRow*WareX];
y = [1:1:VaultCol*WareY];
preMike = 0;
PreNeigh1 = zeros(VaultRow*WareX*VaultCol*WareY*(VaultHeight*Vresolution-(Vresolution-1)),1);
PreNeigh2 = zeros(VaultRow*WareX*VaultCol*WareY*(VaultHeight*Vresolution-(Vresolution-1)),1);
PreNeigh3 = zeros(VaultRow*WareX*VaultCol*WareY*(VaultHeight*Vresolution-(Vresolution-1)),1);
PreNeigh4 = zeros(VaultRow*WareX*VaultCol*WareY*(VaultHeight*Vresolution-(Vresolution-1)),1);
PreNeigh5 = zeros(VaultRow*WareX*VaultCol*WareY*(VaultHeight*Vresolution-(Vresolution-1)),1);
Sara = 0;
for i = 1:ty
   Sara = Sara + 1;
  %Insert count rate in Space 4
   Space(:,4) = t(:,i);
   %Insert Faults into Space
   if (Sara >= faultindex) & (choice == 1);
      %Add external noise source to the signal
      source(:,4) = tex(:,i);
      [Space] = addext(Space,source);
   end
   if (Sara == faultindex) & (choice == 2);
      %Add spike to faulted vaults
      for k = 1:VaultHeight;
         CC = C-1;
         for j = C:D
            CC = CC + 1;
            AA = A;
            for j = A:B
               F = ((k-1)*VaultRow*WareX*VaultCol*WareY) + ((CC-1)*VaultRow*WareX) + AA;
               G(Ind,:) = F;
               Ind = Ind + 1;
               AA = AA + 1;
            end
         end
      end
      [Gx Gy]=size(G);
      for k = 1:Gx
         Gcount = G(k,:);
         Space(Gcount,4) = 3*Space(Gcount,4);
      end
   end
   %Calculate the residual
   Space(:,4) = Space(:,4) - mu1;
   %Kernel Smoothing
   [SmoothSpace] = KernalSmooth(Space,gauss,VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,Interest);
   %Plot the Neighborhood
   figure(1)
   AlphaPlot(SmoothSpace,VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,[0 20]);
   zlabel('Sensor in Z');
```

```
    xlabel('Sensor Space in X');ylabel('Sensor Space in Y');
    title('Neighborhood Score');
    drawnow;
    %Plot the UnSmoothed Data
    figure(4)
    AlphaPlot(Space,VaultRow,VaultCol,WareX,WareY,VaultHeight,1,[0 20]);
    zlabel('Sensor in Z');
    xlabel('Sensor Space in X');ylabel('Sensor Space in Y');
    title('UnSmoothed Data - Residuals');
    drawnow;
    %Record largest residuals.
    Mike = max(Space(:,4));
    if Mike > preMike
       figure(2);
       AlphaPlot(SmoothSpace,VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,[0 20]);
       %axis([-1 VaultRow*WareX+1 -1 VaultCol*WareY+1 -100 400]);
       title('Maximum of Neighborhood');
       zlabel('Sensor in Y');
       xlabel('Sensor Space in X');ylabel('Sensor Space in Y');
       preMike = Mike;
       drawnow;
    end
    %Plot Tails
    Tails = SmoothSpace;
    Tails(:,4) = (SmoothSpace(:,4)+(exp(-1/3))*PreNeigh1+(exp(-2/3))*PreNeigh2+(exp(-3/3))*PreNeigh3+(exp(-
4/3))*PreNeigh4+(exp(-5/3))*PreNeigh5)./2.0503;
    figure(3);
    AlphaPlot(Tails,VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,[0 20]);
    title('Tail Plot of Neighborhood Score');
    zlabel('Sensor in Y');
    xlabel('Sensor Space in X');ylabel('Sensor Space in Y');
    drawnow;
    PreNeigh5 = PreNeigh4;
    PreNeigh4 = PreNeigh3;
    PreNeigh3 = PreNeigh2;
    PreNeigh2 = PreNeigh1;
    PreNeigh1 =  SmoothSpace(:,4);
    %Find the maximum of the neighborhood space
    if ((Sara<faultindex)&(choice==1)) | ((Sara~=faultindex)&(choice==2))
       SourceLocPre(Sara,:) = [0 0 0];
    elseif (((Sara>=faultindex)&(choice==1)) | ((Sara==faultindex)&(choice==2)));
       ind = find(max(SmoothSpace(:,4)) == SmoothSpace(:,4));
       A = SmoothSpace(ind,1:3);
       %Chart position of source prediction location
       SourceLocPre(Sara,:) = [A];
       if choice == 1;
          Error(Sara,:) = source(:,1:3)-A;
       end
       if choice == 2;
          CrashPre = [ceil(center(1,1)/VaultRow) ceil(center(1,2)/VaultCol)];
          Error = Crash - CrashPre;
       end
    end
    %Chart position of source location
    if choice == 1;
       SourceLoc(Sara,:) = [source(1,1:3)];
    end
    %Alter the coordinates of the source
    if (choice == 1) & (Sara >= faultindex);
       [source,Arrive] = Alter(source,centerspace,Arrive,WareX,WareY,VaultRow,VaultCol,VaultHeight);
    end
    if Sara == 1
       pause
    end
end
%Plot position of source and prediction of source location
if choice == 1
```

```
   for i = faultindex:Sara
      figure(5);
      Brian = plot3(SourceLoc(i,1),SourceLoc(i,2),SourceLoc(i,3),'r +');hold
on;plot3(SourceLocPre(i,1),SourceLocPre(i,2),SourceLocPre(i,3),'b O')
      title('Location of Source');
      zlabel('Neighborhood Score');
      xlabel('Sensor Space in X');ylabel('Sensor Space in Y');
      legend('Source Location','Source Location Prediction',0);
      grid on;
      axis([-1 VaultRow*WareX+1 -1 VaultCol*WareY+1 0 VaultHeight+1]);
      set(Brian);
      drawnow;
      pause(.2)
      cla
   end
   %plot the error in the prediction
   figure(6)
   ErrorTotal = sqrt(Error(:,1).^2+Error(:,2).^2+Error(:,3).^2);
   plot(ErrorTotal(faultindex:Sara,:));xlabel('Time Step');ylabel('Error Value (Sensor Space Units)');
   title('Error in Prediction of Neighborhood System');
end
%Error in Crash fault
if choice == 2
   ErrorTotal = sqrt(Error(:,1).^2+Error(:,2).^2);
   fprintf('\nA crash spike occurred in Sensor Concentrator Vault %g %g \n',CrashPre(1,1),CrashPre(1,2));
   fprintf('The actual spike occurred in Sensor Concentrator Vault %g %g \n',Crash(1,1),Crash(1,2));
   fprintf('The error in the prediction is %g in Vault Units',ErrorTotal);
end
```

FabricateSen Function

This function creates the three dimensional array and the RADSiP sensor information for the RAMM.

```
function [Space] = FabricateSen(VaultRow,VaultCol,VaultHeight,WareX,WareY)
%Fabricate - Fabricates the data for the neighborhood function.
%  Written by Joseph Bowling
%  VaultRow - The number of rows in the vault
%  VaultCol - The number of columns in the vault
%  VaultHeight - The height the vaults are stacked to
%  VaultX - The number of vaults x coordinate
%  VaultY - The number of vaults y coordinate
%
%  [Space] = Fabricate(VaultRow,VaultCol,VaultHeight,VaultX,VaultY);
%Create the Sensor Space
index = 1;
for k = 1:VaultHeight   %Change the 1 to a zero to make ground 0
   for j =1:VaultCol*WareY
      for i =1:VaultRow*WareX
         Count = [i j k 0];
         Space(index,:) = Count;
         index = index + 1;
      end
   end
end
```

FabricateSignal Function

This function creates the RADSiP radiation signal for use in the RAMM.

```
function [t,mu1] = FabricateSignal(VaultRow,VaultCol,VaultHeight,WareX,WareY,time)
%Fabricate - Fabricates the data for the neighborhood function.
%   Written by Joseph Bowling
%   VaultRow - The number of rows in the vault
%   VaultCol - The number of columns in the vault
%   VaultHeight - The height the vaults are stacked to
%   VaultX - The number of vaults x coordinate
%   VaultY - The number of vaults y coordinate
%
%   [Space] = Fabricate(VaultRow,VaultCol,VaultHeight,VaultX,VaultY);
%Fabricate the data
mu1(1:VaultRow*VaultCol*VaultHeight*WareX*WareY) = round(100*rand(1,VaultRow*VaultCol*VaultHeight*WareX*WareY))
+ 20;
sig(1:VaultRow*VaultCol*VaultHeight*WareX*WareY) = sqrt(mu1);
for index = 1:time
   randy = randn(1,VaultRow*VaultCol*VaultHeight*WareX*WareY);
   data = round(mu1(1:VaultRow*VaultCol*VaultHeight*WareX*WareY) + sig.*randy);
   t(index,:) = data;
end
t = t';
mu1 = mu1';
```

Extsource Function

This function creates the external source abnormality that the RAMM demonstration will track.

```
function [tex] = extsource(time);
%extsource - Function that creates the external source to add to the fabricated signal
%in the neighborhood system.
%Fabricate the external source
exmu1(1:1) = round(100*rand(1,1)) + 40;
exsig(1:1) = sqrt(exmu1);
for index = 1:time
   randy = randn(1,1);
   data = round(exmu1(1:1) + exsig.*randy);
   tex(index,:) = data;
end
tex = tex';
```

LoadGauss Function

This function creates and loads the kernel function information necessary for the kernel smoothing of the RADSiP radiation signal.

```
function [gauss,Interest] = LoadGauss(VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,Hresolution,width);
%LoadGauss - Loads of creates the gaussian data for the Neighborhood System
%   distance - the distance from each point to each other.
% gauss - gaussian value of distance with width
% VaultRow - The number of rows in the vault
% VaultCol - The number of columns in the vault
% VaultHeight - The height the vaults are stacked to
% WareX - The number of vaults x coordinate
% WareY - The number of vaults y coordinate
% h - smoothing parameter
% Vresolution - Vertical resolution of the smoothing (Number of Nodes)
% Hresolution - Horizontal resolution of the smoothing (Number of Nodes)
%
% Written by Joseph Bowling, August 6,2003
if (VaultRow==4)&(VaultCol==5)&(WareX==4)&(WareY==4)&(VaultHeight==3)&(Vresolution==1)&(width==2.5)
   load gauss-4-4-3-2_5
elseif (VaultRow==4)&(VaultCol==5)&(WareX==4)&(WareY==4)&(VaultHeight==3)&(Vresolution==1)&(width==3)
   load gauss-4-4-3-3
elseif (VaultRow==4)&(VaultCol==5)&(WareX==5)&(WareY==5)&(VaultHeight==2)&(Vresolution==1)&(width==3)
   load gauss-5-5-2-3
elseif (VaultRow==4)&(VaultCol==5)&(WareX==3)&(WareY==3)&(VaultHeight==2)&(Vresolution==1)&(width==5)
   load gauss-3-3-2-5
elseif (VaultRow==4)&(VaultCol==5)&(WareX==4)&(WareY==4)&(VaultHeight==5)&(Vresolution==1)&(width==3.5)
   load gauss-4-4-5-3_5
elseif (VaultRow==4)&(VaultCol==5)&(WareX==3)&(WareY==3)&(VaultHeight==15)&(Vresolution==1)&(width==4)
   load gauss-3-3-15-4
elseif (VaultRow==4)&(VaultCol==5)&(WareX==4)&(WareY==4)&(VaultHeight==1)&(Vresolution==1)&(width==4)
   load gauss-4-4-1-4
elseif (VaultRow==4)&(VaultCol==5)&(WareX==3)&(WareY==3)&(VaultHeight==2)&(Vresolution==1)&(width==4)
   load gauss-3-3-2-4
elseif (VaultRow==4)&(VaultCol==5)&(WareX==2)&(WareY==2)&(VaultHeight==6)&(Vresolution==1)&(width==4)
   load gauss-2-2-6-4
elseif (VaultRow==4)&(VaultCol==5)&(WareX==2)&(WareY==2)&(VaultHeight==6)&(Vresolution==1)&(width==2.5)
   load gauss-2-2-6-2_5
elseif (VaultRow==4)&(VaultCol==5)&(WareX==2)&(WareY==2)&(VaultHeight==3)&(Vresolution==2)&(width==2.5)
   load gauss-2-2-3-2_5-2
elseif (VaultRow==4)&(VaultCol==5)&(WareX==3)&(WareY==3)&(VaultHeight==5)&(Vresolution==2)&(width==2.5)
   load gauss-3-3-5-2_5-2
elseif (VaultRow==4)&(VaultCol==5)&(WareX==3)&(WareY==3)&(VaultHeight==6)&(Vresolution==1)&(width==2.5)
   load gauss-3-3-6-2_5
else
   fprintf('\n\nThe gauss matrix does not exist for the specified architecture.\n');
   fprintf('Please wait will it is being created and saved.\n');
   filename = input('Enter the gauss data matrix filename (gauss-WareX-WareY-VaultHeight-Width-Vresolution): ','s');
   fprintf('Please make the appropriate changes to the function LoadGauss');
   [distance,gauss,Interest] = Distance(VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,Hresolution,width);
   eval(['save ' filename ' gauss Interest']);
end
%Experiment with this to avoid having to enter the gauss matrix name
%filename = char('gauss',WareXs,'-',WareYs,'-',VaultHeights);
```

Addext Function

This function adds the external radiation source to the radiation signals in the RAMM demonstration.

```
function [Space] = addext(Space,source);
%addext - Adds external source to the radiation signal based on the
%distance from the source to the sensor in question.
% Written by Joseph Bowling
% 7/23/03
%Determine the distance between the source and the sensor in question
```

```
[spacex spacey] = size(Space);
index = 1;
for i = 1:spacex
   distance(index,:) = sqrt(sum((source(1,1:3)-Space(i,1:3)).^2));
   index = index + 1;
end
%Determine the effect the source would have on the sensor at the various
%distances.
A = 1./((distance(:,1).^2+1))*source(:,4);
%Add the effect of the external source to the existing count rate for the
%sensor
Space(:,4) = Space(:,4) + A;
```

AlphaPlot Function

This function creates a transparent three-dimensional plot of the RAMM result.

```
function AlphaPlot(SmoothSpace,VaultRow,VaultCol,WareX,WareY,VaultHeight,Vresolution,Caxis);
%AlphaPlot - Slice plotting for the Neighborhood System
%
%  Written By Joseph Bowling
%  Aug 3, 2003
%
%  SmoothSpace - Smooth Data
%  VaultRow - The number of rows in the vault
%  VaultCol - The number of columns in the vault
%  VaultHeight - The height the vaults are stacked to
%  VaultX - The number of vaults x coordinate
%  VaultY - The number of vaults y coordinate
%
%  AlphaPlot(SmoothSpace,VaultRow,VaultCol,WareX,WareY,VaultHeight);
x = SmoothSpace(:,1);%Space(:,1); %SmoothSpace(:,1);
y = SmoothSpace(:,2);%Space(:,2);   %SmoothSpace(:,2);
z = SmoothSpace(:,3);%Space(:,3); %SmoothSpace(:,3);
res = SmoothSpace(:,4);%Space(:,4);  %SmoothSpace(:,4);
dimen = VaultRow*WareX*VaultCol*WareY;
for i = 1:VaultHeight * Vresolution - (Vresolution - 1);
   Count2 = dimen*i-dimen;
   for j = 1:VaultRow*WareX
      Count1 = Count2;
      Count2 = Count2 + 1;
      Count1 = Count1 + 1;
      for k = 1:VaultCol*WareY
         Res(k,j,i) = res(Count1,1);
         Count1 = Count1 + VaultRow*WareX;
      end
   end
end
[x,y,z] = meshgrid(1:1:VaultRow*WareX,1:1:VaultCol*WareY,1:1/Vresolution:VaultHeight);
h = slice(x,y,z,Res,1:1:VaultRow*WareX,1:1:VaultCol*WareY,1:1/Vresolution:VaultHeight);
%axis([0 16 0 20 0 4])
set(h,'EdgeColor','none','FaceColor','interp',...
   'FaceAlpha','interp');
axis([0 VaultRow*WareX 0 VaultCol*WareY 0 max(VaultHeight,VaultCol*WareY/2)])
alpha('color');
%alpha('color');
colormap('NeighColorMap');
caxis(Caxis);
colorbar;
%VIEW(AZ,EL)
%alphamap('rampdown')
%alphamap('increase',.1)
%colormap(hsv)
```

Alter Function

This function alters the position of the external radiation source in the CAVIS array during the RAMM demonstration.

```
function [source,Arrive] = Alter(source,centerspace,Arrive,WareX,WareY,VaultRow,VaultCol,VaultHeight);
%Alter - Alters the coordinates of the source for the NeighMove demo
%   Written by Joseph Bowling
%   source - location of external source in sensor space
%
%   [source] = Alter(source,centerspace);
CS = round(centerspace);
source(:,1:2) = round(source(:,1:2));
FFF = rand;
if (FFF  > 0.5) & (source(:,3) ~= VaultHeight)
   source(:,3) = source(:,3) + 1;
end
if (FFF  < 0.5) & (source(:,3) ~= 1)
   source(:,3) = source(:,3) - 1;
end
if Arrive == 0
   Diff = CS - source(:,1:2);
   if Diff(:,1) > 0
      source(:,1) = source(:,1) + 1;
   elseif Diff(:,1) < 0
      source(:,1) = source(:,1) - 1;
   elseif Diff(:,1) == 0
      source(:,1) = source(:,1);
   end
   if Diff(:,2) > 0
      source(:,2) = source(:,2) + 1;
   elseif Diff(:,2) < 0
      source(:,2) = source(:,2) - 1;
   elseif Diff(:,2) == 0
      source(:,2) = source(:,2);
   end
end
if Arrive == 1
  source(:,1) = source(:,1) + 1;
  source(:,2) = source(:,2);
end
if Arrive == 2
   source(:,1) = source(:,1) - 1;
   source(:,2) = source(:,2) - 1;
end
if Arrive == 3
   source(:,1) = source(:,1) - 1;
   source(:,2) = source(:,2) + 1;
end
if Arrive == 4
   source(:,1) = source(:,1) + 1;
   source(:,2) = source(:,2) + 1;
end
if Arrive == 5
   source(:,1) = source(:,1) + 1;
   source(:,2) = source(:,2) - 1;
end
if CS == source(:,1:2)
   Arrive = 1;
end
if  WareX*VaultRow == source(:,1)
   Arrive = 2;
end
```

```
if  0 == source(:,2)
   Arrive = 3;
end
if  0 == source(:,1)
   Arrive = 4;
end
if WareY*VaultCol == source(:,2)
   Arrive = 5;
end
```

Vita

Joseph Bowling was born on December the 8th 1979 in Lexington, KY. He attended grade school at Southern Elementary in Lexington and later at Kit Carson Elementary in Richmond, KY. Joseph attended Middle school in Richmond at Clark Moores and graduated from Madison Central High School in 1998. After High school he attended Eastern Kentucky University and received a B.S. in Physics and Engineering Physics and a minor in Mathematics in 2002.

Joseph is currently pursuing a Masters of Science degree in Nuclear Engineering at the University of Tennessee Knoxville.