[illegible] I-22588 DR [illegible] (12)

UCID--20508

DE85 017693

COMPUTING SYSTEM FAILURE PROBABILITIES: A COMPARISON OF SIGMA PI WITH OTHER METHODS

R. M. Thatcher
G. C. Corynen

August 1985

Lawrence Livermore National Laboratory

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

EXECUTIVE SUMMARY

Performing a risk assessment of a nuclear power plant requires the study of complex systems. Because of this complexity, the issues of size, speed, timing, and accuracy become important when considering the uses of a computer in the analysis. In addressing issues concerning analysis of probabilistic events, the Lawrence Livermore National Laboratory (LLNL) has developed the SIGMA PI method of computing the probabilistic performance of complex systems.

A detailed description of SIGMA PI has been reported previously in Ref. 3. Included are some performance comparisons among leading alternative methods of evaluating system failure probability, P(S), from a set of minimum cut sets.

Purpose

The purpose of this report is to demonstrate that SIGMA PI is the first method which addresses all of the major problems that have prevented successful evaluations of P(S) in the past. To accomplish this purpose:

1) We present a description of these major problems,
2) Then we describe how leading alternative methods, including SIGMA PI, address these major problems, and
3) Then we use the performance comparisons of SIGMA PI with leading alternatives, that were reported previously, to display the uniqueness of the SIGMA PI method.

The major problems are of two types: algebraic and probabilistic.

Algebraic Problems

Algebraic problems arise because minimum cut set probabilities cannot be combined arithmetically to produce an accurate value of P(S). Therefore, the original set of cut sets must be manipulated algebraically to form an equivalent set of product sets whose probabilities can be so combined.

Probabilistic Problems

Probabilistic problems deal with the probability calculation of product sets, which are Boolean products of basic events. Product sets with dependent basic events can require more calculations for evaluation than even the fastest computers can provide, whereas, sets with independent basic events can be calculated very quickly.

Another probabilistic problem is the development of adequately defined stopping rules for terminating probability calculations when sufficient accuracy has been attained. Stopping rules can result in significant reductions in computer running times.

Available alternatives to SIGMA PI are of two types: bounding methods and exact methods.

Bounding Methods

Before results of research on algebraic problems cited above were available, bounding methods, which tried to by-pass these problems, were necessary for evaluating large fault trees. Bounding methods are still in use, even though their formulas are not expected to give correct probabilities, even when the given formula is calculated accurately.

In this report, we present results from the application of two upper bounding methods to five nuclear power plant safety system fault trees. These two methods are known as "SUM" and "MINIMUM CUT." A comparison with accurate failure probabilities for these same fault trees computed by SIGMA PI demonstrate the unreliability of bounding methods in general.

We also explain in this report why the formula for Hunter's bound can require more computer time to compute an upper bound to P(S) than that required by SIGMA PI to compute P(S) accurately.

Exact Methods

The only viable exact methods available today are disjointing methods, which compute a set of disjoint sets from the original set of minimum cut sets. Disjointing is a technical solution to the algebraic problems cited above; i.e., the disjoint set probabilities add up to P(S).

By 1981, the three leading disjointing methods today were available: 1) Nakazawa (1977), 2) Abraham (1979), and 3) STOP (1981). In this report, we present results from the application of these three methods to the same series of cut set problems. Each method computed the same value of P(S) "exactly," i.e., to machine accuracy.

These results show that the relative speed of STOP increased from about five to eight times the speed of Nakazawa, and from about eleven to fourteen times the speed of Abraham, as the number of basic events in the tree increased from 30 to 45. Comparisons on larger trees were not made because it was not feasible to run the computer long enough to let the Nakazawa and Abraham methods finish their runs.

The SIGMA PI Method

Although disjointing methods solved many problems, they presented some new problems of their own. All of these problems motivated the design and development of the SIGMA PI method.

The special features used by SIGMA PI to address the above problems are summarized as follows:

Algebraic Problems:

1) The SIGMA method uses the [illegible]OP algorithm to generate disjoint sets for quick accuracy.
2) It also uses a technique called DECOMP to exploit special structures in the cut sets to increase the speed of the disjointing process.

Probabilistic Problems:

1) The SIGMA method avoids lengthy calculations by using a technique called PROB to generate only sufficient disjoint sets required for a prespecified accuracy. PROB provides firm upper and lower limits to P(S), which it uses as the basis for a stopping rule, and which verify that the specified accuracy has been attained. Thus, PROB retains full control over accuracy.
2) The PI method further reduces computation times by converting, where possible, any set of independent basic events into multiple sets of conditionally independent basic events.

ABSTRACT

SIGMA PI is the fastest method available today of accurately calculating probabilities associated with large fault trees. To demonstrate this, we describe the major problems that have prevented successful evaluation of these probabilities by other methods in the past. Then, we describe how SIGMA PI addresses all of these problems, and we compare the performance of SIGMA PI with that of leading alternative methods under selected problem scenarios.

1. BACKGROUND - MAJOR ISSUES - COMPUTATIONAL PROBLEMS

Performing a risk assessment of a nuclear power plant requires the study of complex systems. Because of this complexity, the issues of size, speed, timing, and accuracy become important when considering the uses of a computer in the analysis.

Analysts are under increased pressure from decision makers to develop more defensible analyses of probabilistic events. To adequately plan for such events, analysts must calculate event probabilities with as much precision as data and computational resources allow. Because the causes of such events are so involved, current methods are imprecise, and bounding methods have been employed. Even when exact determination of absolute probabilities is not important, it is essential to accurately assess variations in these probabilities as factors or parameters are varied. This assessment has been difficult to do because available bounding methods can only employ bounds on variations that were estimated with bounds in the first place.

1.1. EVALUATION OF FAULT TREES

The first step in evaluating P(S), the failure probability of a system (S), is to convert its fault tree into a Boolean sum of minimum cut sets. This set of minimum cut sets is called a problem set. P(S), the probability of the top event of the fault tree, is the probability of the union of the minimum cut sets.

Fault tree analysis programs, such as FTAP (see Ref. 13), exist for generating problem sets from fault trees. But there are many problems that impact upon the efficient calculation of P(S) from a generated problem set.

In addressing problems concerning analysis of fault trees, the Lawrence Livermore National Laboratory (LLNL) developed the SIGMA PI method of computing P(S) from a fault tree problem set.

A detailed description of SIGMA PI has been reported previously in Ref. 3, which is the source of information for SIGMA PI descriptions given throughout this report. Included are some performance comparisons among leading alternative methods of evaluating system failure probability, P(S), from a set of minimum cut sets.

This report was partially supported by the Division of Risk Analysis and Operations, Office of Nuclear Regulatory Research, and is a task deliverable for their project entitled, "Efficient Probabilistic Computations and Methods for PRA" (FIN No. A0392).

1.2. PURPOSE OF THIS REPORT

The purpose of this report is to demonstrate that SIGMA PI is the first method which addresses all of the major problems that have prevented successful evaluations of P(S) in the past. To accomplish this purpose:

1) We present a description of these major problems,
2) Then we describe how leading alternative methods, including SIGMA PI, address these problems, and
3) Then we use the performance comparisons of SIGMA PI with leading alternatives, that were reported previously, to display the uniqueness of the SIGMA PI method.

We begin by describing some of the major problems that have prevented effective evaluation of P(S). In later sections of this report, we describe more problems encountered in applying disjointing techniques, so that still later we can describe SIGMA PI in a manner that demonstrates, specifically, how it addresses all of these problems.

1.3. MAJOR PROBLEMS THAT IMPACT UPON THE EFFICIENT CALCULATION OF P(S)

Major problems that impact upon the efficient calculation of P(S) are of two types: algebraic and probabilistic.

1.3.1. Algebraic Problems

P(S), the system failure probability, is the probability of the union of the minimum cut sets in the problem set:

$$P(S) = P\langle \bigcup_{j=1}^{T} C(j)\rangle = P\langle \bigcup_{j=1}^{T} \bigcap_{i=1}^{N(j)} B(i,j)\rangle \quad ,$$

where:

T is the number of minimum cut sets in the problem set,

C(j) is the jth cut set,

B(i,j) is the ith basic event in cut set C(j), and

N(j) is the number of basic events in cut set C(j).

The calculation of the union of cut sets is difficult even when accurate cut set and basic event probabilities, P<C(j)> and P<B(i,j)>, can be computed. The reason is that almost always some of the cut sets will have one or more basic events in common, so that the sets are not disjoint. Therefore, in general:

$$P\langle \bigcup_{j=1}^{T} C(j)\rangle \neq \sum_{j=1}^{T} P\langle C(j)\rangle \quad .$$

This leads to the problem of manipulating the original set of cut sets algebraically to form an equivalent set of Boolean products that is "computable," i.e., whose probabilities can be combined by computationally efficient operations to yield P(S).

In Section 2, we shall examine the two best known methods of generating equivalent sets of this kind:

1) The inclusion-exclusion method,

2) The disjointing method.

1.3.2. Probabilistic Problems

Any method of computing P(S) from a set of Boolean product sets, $\{D(j)\}$, must include a method of computing product set probabilities:

$$P(D(j)) = P\Big(\bigcap_{i=1}^{N(j)} B(i,j)\Big) \quad ,$$

where B(i,j) is the ith basic event in set D(j), and N(j) is the number of events in D(j). The time requirements for this calculation are linear in the number of statistically independent basic events, but they are usually exponential in the number of dependent basic events. Section 2.4 of Ref. 10 explains why this is so when the underlying component stress - strength probability distributions are assumed to be normal. The data given (which was derived from Ref. 9) shows that only six dependent basic events in a set can require 50,000 times as much computer time as when the same six are statistically independent. For this reason, sets with five dependencies are usually considered as a cut off limit for feasible evaluation.

Another probabilistic problem is the development of adequately defined stopping rules for terminating probability calculations when sufficient accuracy in P(S) has been attained. Stopping rules can result in considerable savings in calculations performed.

2. CURRENT METHODS OF EVALUATING P(S)

There are only two kinds of methods available for evaluating P(S): bounding methods and exact methods. In this section, we shall describe these two kinds in general, and then identify and describe three leading methods of each kind. Then, in the next section, the performance of SIGMA PI will be compared on selected problem sets with performances of the other five methods. Also, some of the differences in approaches used by these methods will be discussed.

2.1. BOUNDING METHODS

Before algebraic methods of manipulating minimum cut sets into equivalent Boolean products were developed, it was necessary to rely on bounding methods to evaluate P(S). Even today, in practice, only bounding methods appear to be available to evaluate P(S) for large system fault trees. The results are often unsatisfactory because bounding methods are not expected to give correct answers even when the given bounding formula is calculated accurately.

Also, some seemingly straightforward bounding methods, such as Hunter's bound, reveal, after implementation, that they can exceed some exact methods in their requirements for computer time. This can tempt the user to invent short cuts to the calculations for the bounding formula, ("since we are only after an approximation anyway"). This then opens the door to procedures whose outcomes are incomprehensible. An explanation of why Hunter's bound can be so difficult to compute is given in Section 3.2.2 below.

Three upper bounding methods that have been applied to nuclear power plant safety system fault trees (see Section 3.4.2 of Ref. 11) are:

1) $\sum_{j=1}^{T} P(C(j))$ "SUM"

2) $1.0 - \prod_{j=1}^{T} (1.0 - P(C(j))$ "MINIMUM CUT"

3) $\sum_{j=1}^{T} P(C(j)) - \sum_{(i,j)\in\tau} P(C(i) \cap C(j))$ "Hunter's"

where:

- $C(j)$ is the jth minimum cut set,
- T is the number of minimum cut sets in the fault tree,
- τ is a subset of T-1 pairs out of the set of all possible (i,j) pairs, selected appropriately to guarantee an upper bound, and to maximize the second sum in (3).

In comparison to (3), it is interesting to note that the first two terms of the inclusion-exclusion method:

4) $\sum_{j=1}^{T} P(C(J)) - \sum_{\text{all } (i,j)} P(C(i) \cap C(j))$,

produce a lower bound to P(S) (see Chapter IV of Ref. 4).

The first bound, "SUM," is an upper bound to P(S) unless all minimum cut sets are disjoint, in which case equality holds. Thus, the effects of any common events among minimum cut sets are not taken into account by SUM as an approximation to P(S).

The second bound, known as the "MINIMUM CUT" upper bound, is an upper bound to P(S) unless all cut sets are statistically independent, in which case equality holds. Lambert (Ref. 7) also points out that, if all basic events are statistically independent, then equality holds if and only if none of the basic events in the fault tree is replicated, i.e., if and only if each basic event is input to only one gate in the fault tree.

Both the SUM and the MINIMUM CUT upper bounds become more accurate estimates of P(S) as the basic event probabilities get smaller. Section 2.4.1 of Ref. 7 presents an example which demonstrates this fact for the MINIMUM CUT upper bound.

The third bound (3) is known as "Hunter's upper bound," and (4) is often referred to as "Hunter's lower bound" (Refs. 5, 6). Hunter's upper bound is developed as follows:

The T cut sets of a fault tree are treated as nodes of a graph with cut set intersections as arcs. Any T-1 (i,j) pairs of arcs that form a spanning tree, will guarantee that (3) will form an upper bound to P(S). In order for the bound to be minimum, the cut set arcs are selected to maximize the second sum in the formula over all spanning trees. Algorithms are in use (for example, see Ref. 12) that can find the maximal spanning tree using computer times that increase as T^2. This time does not include the time required to calculate the cut set intersection probabilities, $\{P(C(i) \cap C(j))\}$.

We can now see that the minimizing requirement for either of Hunter's bounds includes the probability calculation of all possible pair-wise combinations of the cut set intersections, $(C(i) \cap C(j))$, of which there are $T(T-1)/2$. Each of these cut set arcs are new sets formed by the basic events from two original cut sets.

2.2. EXACT METHODS

The two best known methods of generating a "computable" set of Boolean products from a problem set are:

1) the inclusion-exclusion method, and
2) the disjointing method.

Of these two, the only viable method is the disjointing method. We include a discussion of the inclusion-exclusion method here because it is so well known (see Ref. 4), and because it is of interest to identify some of the complexities involved in trying to use it to compute P(S) accurately.

2.2.1. The Inclusion-Exclusion Method

The fact that set probabilities of the well known inclusion-exclusion method converge to P(S) (Ref. 4):

$$P(S) = \sum_{j=1}^{T} P(C(j)) - \sum_{i=1}^{T} \sum_{j=1}^{T} P(C(i) \cap C(j)) + \dots \quad ,$$

where T is the number of minimum cut sets, has made it a tempting algorithm to run on a high speed computer.

Our discussion of probabilistic problems at the end of Section 1.3 makes it clear that this series cannot be computed for even small problem sets where basic events are statistically dependent, since it requires the generation and evaluation of larger and larger product sets. In fact, in Section 3.2.2

below, we show that Hunter's bounds, which involve calculations that are equivalent to only the first two sum terms of this series, is effectively eliminated when basic event dependencies exist.

A fact that is also becoming better known is that, even for medium problem sets, and with all independent basic events, converging to an accurate value of P(S) with this series of product sums is still beyond our fastest computers today. The problem is that the cumulative sum has a tendency to oscillate wildly between large positive and negative values for too many sum terms before it begins to converge.

2.2.2. The Disjointing Method - Some More Problems

The disjointing method converts the problem set, {C(j)}, into an equivalent set of disjoint sets, {D(j)}, expressed as another Boolean sum of Boolean products of basic events. The fact that the new sets are disjoint means that, if the new set probabilities can be calculated, accuracy is assured because the disjoint set probabilities add up to P(S), i.e.:

$$P(S) = P\left(\bigcup_{j=1}^{T} C(j)\right) = P\left(\bigcup_{j=1}^{V} D(j)\right) = \sum_{j=1}^{V} P(D(j)) \quad ,$$

where V is the number of disjoint sets generated.

However, there are some problems with the disjointing method:

1) The new set of disjoint sets, {D(j)}, tends to be much larger than the original problem set, {C(j)}. The first set generated will have

only a few basic events, but as new sets, D(j), are generated, they tend to increase in size in a pyramidal fashion. Also, V, the number of new sets, tends to increase considerably over T, the number in the original problem set.

2) Available disjointing algorithms are designed to generate all of the V disjoint sets, and to compute P(S) from their probabilities "to machine accuracy." In the literature, this type of method is referred to as an "exact" method.

3) From the previous discussion of probabilistic problems, we can see that basic event dependencies can prevent the probability calculation for many of the disjoint sets. If only the evaluated set probabilities are added, the result is a lower bound to P(S).

4) A stopping rule applied to a disjointing method could result in significant reductions in computer running times. If all basic events are independent, it is possible to calculate probabilities for all V disjoint sets that the computer can generate with the given disjointing algorithm. But usually two or three decimal places is all of the accuracy that is required. Since the first disjoint sets are the shortest, and the shortest tend to contain the largest probability values, only a relatively few of the first ones generated will contain enough of the required probability value to give the accuracy that is normally needed. Fortunately, the shortest sets are also the easiest ones to evaluate.

It is noteworthy that, thus far, no exact methods have been used to aid in the evaluation of risks associated with large nuclear reactor and safety systems.

2.2.3. Leading Disjointing Methods

By 1981, the following three leading methods were available:

1) Nakazawa's Method (1977, Ref. 8),
2) Abraham's Method (1979, Ref. 1), and
3) The STOP Method (1981, Ref. 2).

A complexity analysis, reported in Chapter 4 of Ref. 3, indicates that, in the worst case, STOP requires computer time that is weakly exponential in a function of N, the number of basic events in the set. A weakly exponential function of N is defined as:

$$e^{K \cdot N} \quad , \quad \text{where } K < 1 \quad .$$

In fact, K·N is equal to C(min), the size of the shortest cut set. This places STOP on the borderline with polynomial complexity because K is normally very small, i.e.:

$$K = C(\min)/N \quad ,$$

the ratio of the shortest cut set to the full set dimension. For example, if C(min) is three basic events, and N is 100 basic events, then $K = 0.03$.

An examination of the algorithms used by Abraham and Nakazawa indicates that their complexities are "strongly exponential" ($K > 1.0$).

These findings are consistent with performance test results presented in Section 3.3.

The efficiency of the STOP algorithm is due, in essence, to the parallel fashion in which it manipulates the given problem set. The first step is to decompose the problem set into disjoint and simpler problem sets along a carefully chosen coordinate. Then, each new set is similarly decomposed. This process continues until, in the limit, a collection of mutually disjoint simple sets is obtained.

This contrasts with the sequential operations in the Abraham method which does not view the cut sets as a whole. Instead, each new set is added to the union of previously processed sets, an operation that is strongly exponential in complexity.

The Nakazawa disjointing method shares a technique in common with STOP which retains "prominent" coordinates at a crucial step in the disjointing process. This technique makes Nakazawa's method a little faster than Abraham's.

1981 Technology Summary

We summarize the fault tree technology available in 1981:

1) Fault tree analysis programs (e.g., FTAP) were available to generate problem sets from fault trees.

2) Disjointing algorithms were available to convert problem sets into sets of disjoint sets, whose probabilities add up to P(S).

Disjointing algorithms are a technical solution to the algebraic and probabilistic problems described in Section 1.3. However, problems with the disjointing method, described in Section 2.2.2 and in this section, still needed to be resolved.

The single most significant technical development since 1981 that justified the creation of the SIGMA PI method is the DECOMP algorithm. As described below, DECOMP works with STOP to perform disjointing of most problem sets of interest in computer time that is still borderline exponential in N for smaller problem sets, but approaches linear complexity as N gets large.

The dramatic improvement in disjointing speed due to the use of DECOMP is demonstrated by results of performance tests reported in Section 3.3.

2.2.4. The SIGMA PI Method

SIGMA PI consists of two methods, SIGMA and PI. SIGMA computes P(S) by summing ($\sum$) disjoint set probabilities. The PI method prepares independent basic events for multiplying (Π) their probabilities. By a careful merging of several analytic techniques in SIGMA with the PI method, SIGMA PI addresses all of the major problems that impact upon a successful evaluation of P(S). Thus, SIGMA PI is the first method to be dedicated to the rapid and accurate calculation of P(S) for large systems.

The SIGMA Method:

SIGMA is an efficient method of developing a sufficient number of disjoint sets, {D(j)}, from a given problem set, {C(j)}, to permit the calculation of P(S) to a desired prespecified accuracy. P(S) is calculated as the sum of the probabilities of the disjoint products:

$$P(S) = P\Big(\bigcup_j C(j)\Big) = \sum_j P(D(j)) \quad .$$

The efficiency of SIGMA is due to several techniques that it employs:

1) STOP

The STOP algorithm was developed in 1981 at the LLNL (Ref. 2). STOP is a very fast method of generating a complete set of disjoint sets from a given problem set, and then computing P(S) from them. When operating alone, STOP calculates P(S) to "within machine accuracy," i.e., STOP is an "exact" method.

2) PROB

Under the SIGMA PI mode, each time STOP develops a new level of disjoint sets, a routine called PROB computes firm upper and lower bounds to P(S), which it uses as the basis for a stopping rule. These bounds converge rapidly at first and then more slowly as the disjointing proceeds. When these bounds agree to within the accuracy specified by the user, PROB terminates the calculation. Experience shows that, for medium size problems with about 30 basic events, usually less than five percent of STOP's complete disjointing effort (which gives P(S) to machine accuracy) is sufficient to obtain three place accuracy in P(S). This percentage decreases as the number of basic events is increased.

3) DECOMP

The DECOMP algorithm examines a given problem set and finds any blocks among them. A block is a special subset, C', of the original set of cut sets, C, defined as follows: Let B be the set of basic events in the problem set, C. Then C' is a block if two conditions hold: 1) for the subset, C', a subset, B', of B are all don't cares; 2) for the rest of the cut sets, C" = C - C', the set of basic events, B" = B - B', are all don't cares. A more complete description of blocks is given in Chapter 3 of Ref. 3.

Under normal circumstances, the original problem set will have come from a single fault tree, and most often will consist of only a single block. Bu. STOP will pivot on the optimal coordinate for disjointing the problem into two disjoint problem sets. Some of the new problem sets generated by repeated pivoting will be blocks. Whenever PROB determines that more disjointing by STOP is required, control is released to DECOMP which starts a new cycle by finding any blocks which STOP may have generated during the previous cycle. DECOMP cont ols the feeding of blocks back to STOP for separate and, therefore, efficient disjointing.

The complexity analysis of the SIGMA method, reported in Ref. 3, assumes the normal circumstances described above. The "expected" complexity obtained for large problem sets approaches linearity due to the ability of DECOMP to exploit the many block structures that are normally expected in large problem sets. By contrast, the Abraham and Nakazawa complexities are still strongly exponential under normal circumstances because they are insensitive to block structure.

The PI Method:

The probabilistic problems described in Section 1.3 above provide a strong motivation for avoiding, where possible, dependencies among basic events. Such an opportunity presents itself whenever dependencies among some basic events are due to a mutual correlation with a "common-cause event." It is for this commonly occurring opportunity that the PI method was developed formally. Chapter 2 of Ref. 3 presents a detailed description of the PI method.

Briefly, PI is a method of examining input failure distributions for common-cause dependencies, making suitable choices of common-cause random variables, and conditioning on them appropriately. These steps are performed by the user as he prepares the problem set for input to the SIGMA computer program. Performance of the PI method results in the conversion of a set of dependent basic events into multiple sets of conditionally independent basic events. SIGMA now must compute multiple probabilities for the same set, D(j). Section 1.3 above explains why the time required to compute P(D(j)) only once with dependent basic events can be orders of magnitude greater.

Figure 1 illustrates the merger of the three major techniques in SIGMA with the PI method into the SIGMA PI method.

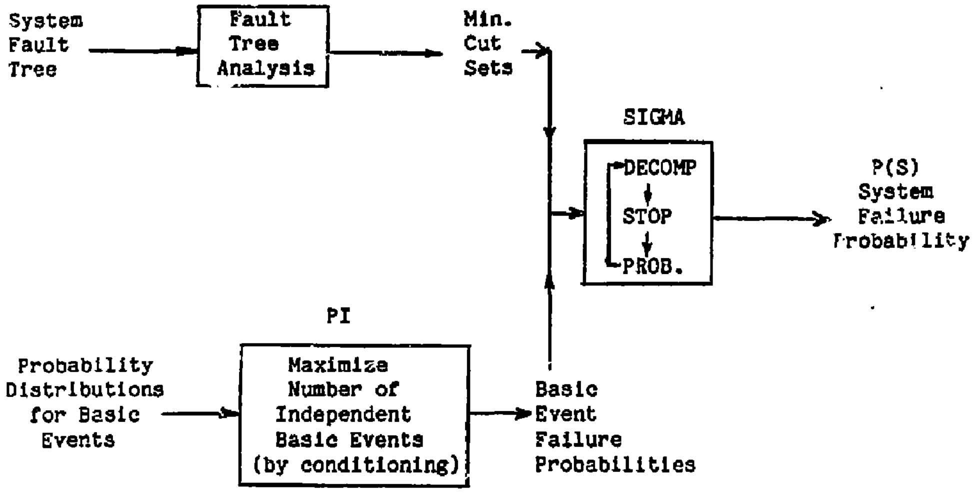
3. COMPARING SIGMA PI WITH OTHER METHODS

3.1. INTRODUCTION

SIGMA PI is the first method which addresses all of the major problems that have prevented effective evaluation of P(S) in the past. At this level of problem dedication, SIGMA PI is unique, i.e., no alternative methods have been designed to compete with SIGMA PI. The leading available alternative methods of evaluating P(S) are either exact methods or bounding methods.

Strictly speaking, SIGMA PI should not be classified as an exact method because it can control accuracy, and exact methods are designed to calculate P(S) to machine accuracy every time. Since it can also compute P(S) exactly, we shall call SIGMA PI a "controlled exact method."

Neither can SIGMA PI be classified as a bounding method in the usual *sense because the upper and lower bounds which it computes are firm, and will* converge to the "exact" value eventually, if processing is allowed to continue. By contrast, bounding method formulas that are still in use to evaluate large systems are not expected to be correct even when the given formula is calculated "exactly."


System Fault Tree
Fault Tree Analysis
Min. Cut Sets
SIGMA
DECOMP
STOP
PROB.
P(S) System Failure Probability
PI
Probability Distributions for Basic Events
Maximize Number of Independent Basic Events (by conditioning)
Basic Event Failure Probabilities

THE SIGMA PI METHOD

Figure 1

Due to this uniqueness in the SIGMA PI method, it is desirable to begin our tests by comparing individual features of SIGMA PI with alternative methods that perform the same function. We do this in our performance comparisons below by first stripping away some of the techniques that SIGMA PI uses for full performance.

Our most basic assumption is that each comparison alternative method applies the PI method to the original basic events equally. We need this assumption for comparing with SIGMA because it provides a common basis for fair comparisons, i.e., it assumes that all methods deal with the same basic event dependencies in the tests. This assumption was invoked in all of the performance tests reported below by running them with statistically independent basic events.

This independence assumption ignores the difficulties incurred in computing set probabilities when basic events are dependent; a difficulty that is common to all comparison methods. The approaches taken by the alternative methods to deal with these difficulties are compared qualitatively in Section 3.4.

3.2. BOUNDING METHODS

3.2.1. <u>SUM and MINIMUM CUT</u>

Failure probabilities, P(S), for five nuclear reactor subsystems subjected to a heavy earthquake, were computed by upper bounding methods (1) and (2), as defined in Section 2.1 above. These values were compared with accurate values of P(S) computed by the SIGMA PI method. The results are displayed in Table 1.

Table 1. P(S) Probabilities for Five Nuclear Power Plant Systems

System Number	No. of Basic Events	No. of Cut Sets	Upper Bound Formula: SUM $\sum P(C(j))$	Upper Bound Formula: MIN CUT $1-\Pi(1-P(C(j)))$	SIGMA PI: P(S) Probability	SIGMA PI: Computer *Time - Cray
1	48	234	0.0095	0.00939	0.00886	0.625
2	137	129	3.53	0.942	0.942	2.48
3	199	295	2.04	0.989	0.989	6.87
4	72	64	3.59	0.991	0.991	0.59
5	122	309	4.64	0.989	0.939	1.46

* Seconds

As predicted, SUM is a close upper bound to SIGMA PI's accurate value of P(S) for the low probability system (P(S) = 0.00886); and obtains impractical bounds (exceeding unity) for the other four systems, in which the true failure probabilities exceed 0.93.

The fact that the MINIMUM CUT upper bound (2) attained close agreement with SIGMA's accurate values for all five systems would not have been predicted, because it was known that many of the cut sets were statistically dependent. However, two conditions could have existed that would still explain this unexpected result. The MINIMUM CUT formula would be accurate if:

1) The total value of MINIMUM CUT was determined by only a subset of the cut sets, and
2) The cut sets in this dominant subset were nearly mutually independent, staristically.

We present some arguments below in support of the possibility that these two conditions existed, based on the fact that these systems had unusual algebraic and probabilistic structures.

When these five systems were evaluated for Table 1, the PI method was applied to the common-cause earthquake in a manner outlined in a prior section. This provided independent basic events for this problem. However, a fairly large number of basic events were observed to be repeated often in many of the cut sets, so on the surface, many of the cut sets would have a high degree of statistical dependence among them. Under this condition, the close agreement with SIGMA PI's accurate values which we obtained would not have been predicted.

Two other unusual conditions existed among these five problem sets, however:

1) Most of the cut sets had only one or two basic events, so that the average cut set size over the five problem sets was 2.1 basic events. From Table 1, we can see that there were over half as many basic events as there were cut sets. It follows that if enough basic events were repeated quite often, then probably there was a fairly large subset of cut sets that had no basic events in common, i.e., they would be mutually independent among themselves.

2) Many basic event probabilities were "tiny," i.e., in the 10^{-12} range, while most of the balance of the basic event probabilities were "significant," i.e., between 0.0001 and 0.5.

This dichotomy extends to the cut sets; i.e., those that have no tiny basic events are the only ones that could have any significant influence on the true value of P(S). Therefore, if most of the repeated basic events were the tiny ones, then the subset of mutually independent cut sets would tend to be the ones that have significant probability values. These are the conditions under which MINIMUM CUT becomes an accurate estimator of P(S).

In general, we can say that whenever a dichotomy of basic event probabilities exists, as was observed for these five problem sets, the accuracy of the MINIMUM CUT formula tends to increase, as the number of repeated basic events that are significant decreases.

Calculation Times:

By using the PI method to obtain conditionally independent basic events, the time requirements to calculate the cut set probabilities, P(C(j)), for the two upper bounding formulas (1) and (2) became negligible.

The times shown for the SIGMA method were measured under difficult test conditions, and include considerable input/output time for processing data files. It is believed that the true times consumed by SIGMA were less than half of the values shown.

MINIMUM CUT Conclusions:

We have just demonstrated that the MINIMUM CUT formula can give an accurate estimate of P(S) when conditions are "right" among the problem set and the basic event probabilities. We also know that MINIMUM CUT is an inaccurate estimator of P(S) under most problem set conditions. Perhaps the selective use of MINIMUM CUT to save computer time could be justified if an efficient method were available for detecting when "right" conditions exist that would guarantee a given level of accuracy in the computed value of P(S).

The modest added cost of using SIGMA PI to calculate P(S) is justified by the accuracy guarantee that it gives on all problem sets which it evaluates.

3.2.2. Hunter's Bound

In Section 2.1, we saw that either of Hunter's bounds (upper or lower) requires the probability calculation of all possible pair-wise combinations of the cut set intersections, $(C(i) \cap C(j))$, of which there are T(T-1)/2, and that each combination is a new set formed by the basic events from two original cut sets.

If the basic events are mutually independent, the set probabilities can be computed as products of their basic event probabilities. Even so, this calculation requires a significant computational effort for merely an upper bound (polynomial in T).

If many of the basic events are statistically dependent, no computer today has the capability of calculating Hunter's bound, unless each cut set is limited to a maximum of two dependent basic events. This effectively eliminates the use of Hunter's bound for even small systems with dependent basic events.

The following example displays the enormous complexities caused by dependent basic events even when all cut sets are limited to two.

Example: Comparison of Dependent vs. Independent Basic Events.

The largest system in Table 1 above has 309 cut sets. Assume each cut set has two basic events. We shall compare the number of ESVNIs (Equivalent Single Variate Normal Integrals) required to compute Hunter's bound:

1) When all basic events are independent, vs.

2) When all basic events are dependent.

For the first term in Hunter's bound, we must calculate 309 cut set probabilities, each of which has two basic events. For the second term, we must compute (309 * 308)/2 = 47,586 set probabilities, most of which has four basic events.

1) Independent Basic Event Calculations

			ESVNIs
309 C.S. * 2 B.E./C.S.	=		618
47,586 Sets * 4 B.E./Set	=		190,344
		TOTAL	190,962

2) Dependent Basic Event Calculations

From the Appendix of Ref. 10, we get:

* Each two-variate normal integral requires 14 ESVNIs,
* Each four-variate normal integral requires 2,642 ESVNIs.

			ESVNIs
309 C.S. * 14 ESVNIs/C.S.	=		4,326
47,596 Sets * 2,642 ESVNIs/C.S.	=		125,748,632
		TOTAL	125,752,958

<u>Conclusion</u>:

Even when each of the 309 cut sets is restricted to two basic events per cut set, Hunter's bound requires:

191 thousand ESVNIs for independent basic events, and
126 million ESVNIs for dependent basic events.

This problem set requires nearly 660 times as much computer effort for dependent basic events as for independent basic events.

This ratio escalates when larger cut sets are used in the problem set.

3.3. EXACT METHODS

For the first performance tests PROB and DECOMP are stripped away from SIGMA, leaving STOP, the exact method, to be compared with two leading alternative exact methods. The results verify that STOP is a good disjointing algorithm to use in the SIGMA method.

Further performance tests demonstrate that increasingly larger systems can be evaluated as SIGMA features are added, successively, back to STOP.

Performance comparison data presented in this section was taken from experiments that were reported previously in Ref. 3. The Abraham method and the Nakazawa method are leading exact methods of computing P(S) from minimum cut sets, which we compare below with various levels of SIGMA. As another exact method, STOP, when operating alone, computes the same value for P(S) as these other two methods. In the performance tests below, we are interested in comparing the computer times required to compute this value.

For these tests, we adhere to a strict interpretation of the capabilities of these two alternative disjointing methods, as presented in their references given above.

Two comparative experiments were conducted.

3.3.1. Experiment #1 - Comparing SIGMA Without PROB

For experiment 1, each algorithm was run on the same set of random problems, with N, the number of components ranging between 30 and 50. T, the number of cut sets was fixed at 20, and the proportion of don't care values fixed at 0.8. We compared the computer time required to compute P(S) to machine accuracy, i.e., 16 decimal places for the machine used. The results are presented in Table 2, and in graphic form in Fig. 2.

The first three lines of Table 2 show the results of solving random coherent problems with single block structure by Abraham, Nakazawa, and by STOP alone, (no DECOMP). These are the worst case complexity functions for STOP because random problem sets have less structure than those from regular fault tree problems. It is interesting to note that for the three tree sizes tested, the relative speed of STOP alone increased from approximately five to eight times as fast as Nakazawa, and from about eleven to fourteen times as fast as Abraham, as N increased from 30 to 45. For trees with over 45 components, it was not feasible to run the computer on Abraham's and Nakazawa's methods long enough to calculate P(S).

Although STOP is significantly faster than Abraham and Nakazawa in the worst case, the advantage is not as significant as when DECOMP is used together with STOP on some of the same problem sets. The results of this comparison are shown on line A of Table 2. We see that, for a fault tree of size 45 components, STOP alone reduced computation time from Abraham's 400s to 30s, and DECOMP and STOP together further reduced this time to ten seconds.

SIGMA (STOP plus DECOMP) required far less time when problems with block structure were tested. As shown on lines A, B, and C for 50 component fault

Table 2. Performance Comparisons of Exact Methods

Times in Seconds to Compute P(S) - Cray Computer

Exact Method (16 decimal places)	Test Fault Tree (Number of Components, T) 30	40	45	50
ABRAHAM	11.0	107.0	400.0	--
NAKAZAWA	5.0	63.0	228.0	--
STOP ALONE	1.0	10.0	29.5	64.2
STOP + DECOMP:				
A Single Block	--	5.0	10.0	29.5
B Two Blocks	--	--	--	0.6
C Four Blocks	--	--	--	0.12
D Five percent of line A $\sum$ (3 place accuracy)	--	0.25	0.50	1.48

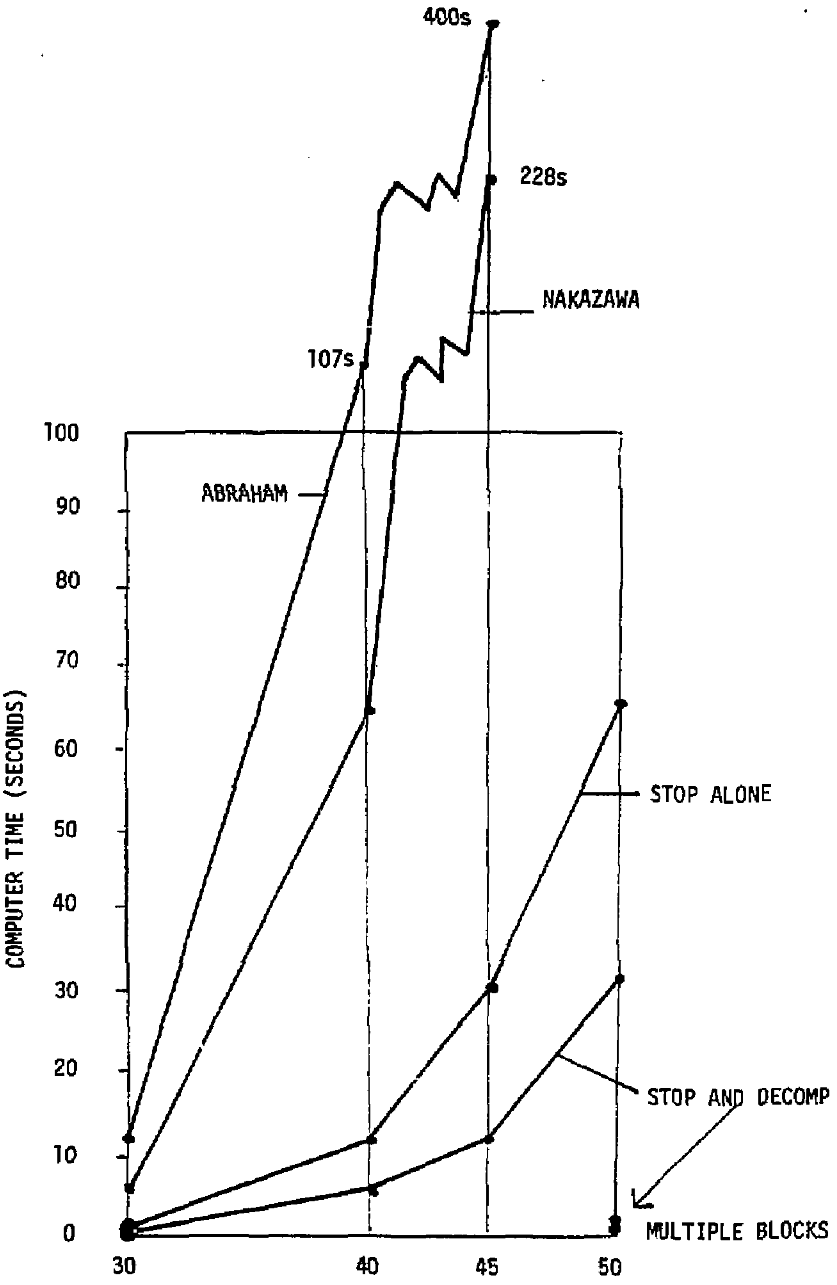


Figure 2
PERFORMANCE COMPARISON OF EXACT METHODS

trees with block structure, the 29.5s for a single block is reduced to 0.6s with two blocks, and, with four blocks, to 0.12s. This can be compared to Abraham and Nakazawa, whose times would have far exceeded 400s, had they been allowed to complete their calculations.

For completeness, we note that these performance tests were all run on a CRAY computer, which has some optimizing features for performing certain Boolean operations. These features were used to give added speed to SIGMA algorithms where they could use them. Because of differences in techniques used by Abraham and Nakazawa, similar advantages could not be taken fully for these methods on the CRAY.

The overall results presented here are unchanged by the use of these features, because at most, they could have caused a minor shift in the SIGMA results a bit to the right on the graph of Fig. 2. The shape of each graph would remain the same if these optimizing features had not been employed.

3.3.2. Experiment #2 - Comparing the Full SIGMA Method

The second experiment demonstrates the computer effort that can be saved when PROB is used to control accuracy. We accomplished this by running SIGMA on a single problem with 30 basic events and 100 cut sets. Although of medium size, this is a challenging problem set for SIGMA in that it consists of a single block, and the basic event probabilities range uniformly between 0.1 and 0.9. By contrast, convergence tends to be considerably faster when event failure probabilities are between 0.0 and 0.1.

Figure 3 displays the computation accuracy attained by SIGMA as a function of computation time for the first 15 seconds. The left hand ordinate

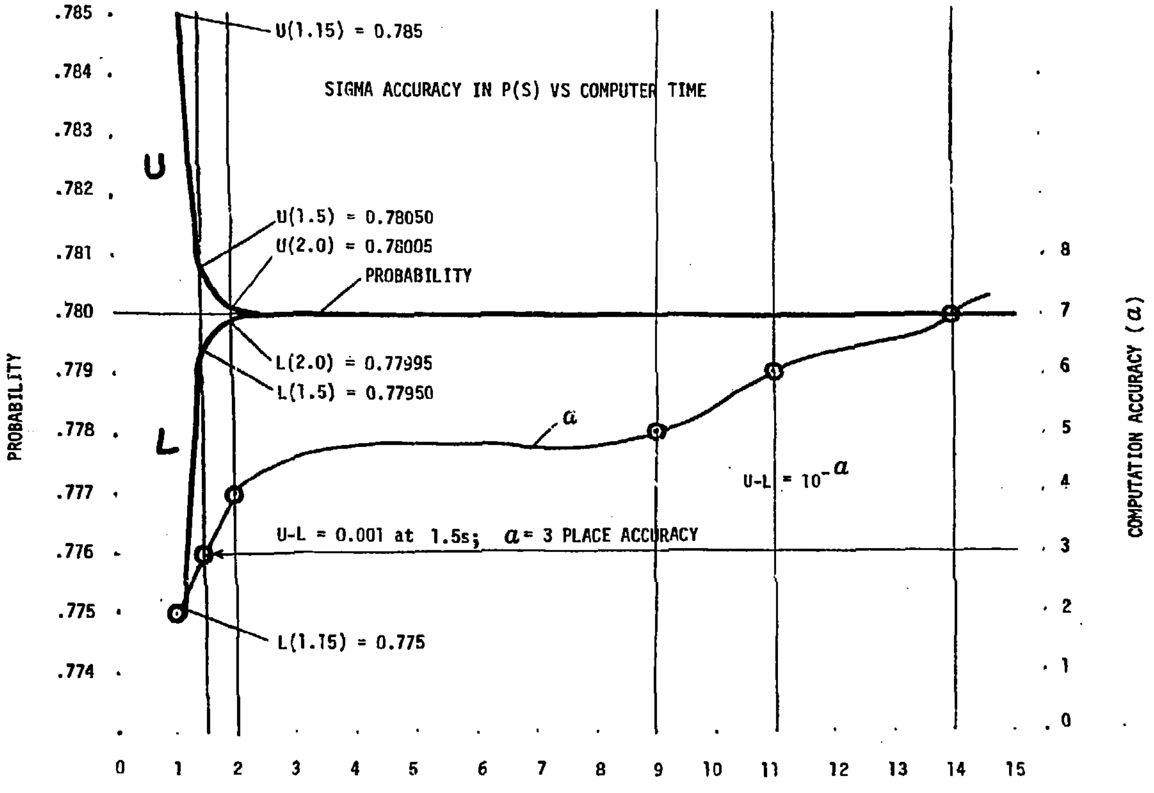

SIGMA ACCURACY IN P(S) VS COMPUTER TIME
U(1.15) = 0.785
U(1.5) = 0.78050
U(2.0) = 0.78005
PROBABILITY
L(2.0) = 0.77995
L(1.5) = 0.77950
L(1.15) = 0.775
U
L
a
$U-L = 10^{-a}$
U-L = 0.001 at 1.5s; a = 3 PLACE ACCURACY
PROBABILITY
COMPUTATION ACCURACY (a)
.785
.784
.783
.782
.781
.780
.779
.778
.777
.776
.775
.774
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
SECONDS OF COMPUTER TIME

Figure 3

measures upper (U) and lower (L) probability limits computed by SIGMA, which converged rapidly to attain three decimal place accuracy in about 1.5 seconds. The right hand ordinate indicates the computation accuracy in terms of alpha (α), the number of decimal places, i.e., $U - L = 10^{-\alpha}$. We can see that it took 14 seconds to attain about seven place accuracy. Not shown is the final accuracy attained: 12 place accuracy in 20 seconds.

This experiment demonstrates the general case in which convergence of the upper and lower bounds is very rapid at first, then tapers off for a while as intermediate accuracy levels are reached, and then speeds up again before the "exact" solution is computed.

The times shown in Fig. 3 are consistent with general experience with various problem sets of this approximate size, which has shown that SIGMA usually reaches three place accuracy within five percent of the time required for machine accuracy (16 decimal places). For larger problem set sizes, time requirements for three place accuracy tend to diminish below five percent.

We can now estimate the overall performance of the SIGMA method over that of STOP alone for general problem set sizes used in experiments 1 and 2. Line A of Table 2 represents times required by STOP + DECOMP for three tests. To estimate the time required by the full SIGMA method to attain three place accuracy (STOP + DECOMP + PROB), we multiplied the three values on this line by 0.05. The three results are displayed on line D of Table 2. Based on these results, we estimate that STOP alone took from 40 to 70 times as long as SIGMA would have taken to attain three place accuracy.

3.4. COMPARISONS WHEN BASIC EVENTS ARE DEPENDENT

The performance tests in the previous sections compare the relative speeds and accuracies between SIGMA PI and the five alternative methods when the calculation of set probabilities is no real problem. As explained in Section 3.1, this condition was purposely invoked by running all performance tests with statistically independent basic events.

But, as explained in Section 1.3, serious computational difficulties can arise when too many dependencies among basic events remain after all attempts to remove them, such as an application of the PI method, have been made. We can show that even though all evaluation methods will have similar difficulties in computing the set probabilities, P(D(j)), the SIGMA method alone can still produce useful results when the results from alternatives are meaningless.

Suppose many cut sets contain more dependent basic events than any of the methods can evaluate, say, over five dependencies per set. (See Probabilistic Problems in Section 1.3 above.)

In the case of exact evaluation alternatives, all possible disjoint sets can be generated, but only those with five or less dependent basic events can be evaluated. If only these probabilities are added, only a lower bound to P(S) is produced. Similarly, in the case of an upper bounding method, only a lower bound to the upper bound is computable, so the result isn't even known to be an upper bound. In either case, an estimated value of P(S) can be given, but the user will end up with little indication of its accuracy.

In the case of SIGMA PI, either:

1) The accuracy required for P(S) will be attained and reported as such; or,

2) SIGMA will run out of disjoint sets that it can evaluate before the required accuracy is reached. In this case, SIGMA will report an estimated value for P(S) and give firm upper and lower bounds to its value.

ACKNOWLEDGMENTS

The authors greatfully acknowledge the support of the Risk Analysis and Operations Division of the Office of Nuclear Regulatory Research. The continued encouragement and technical discussions provided by Dale Rasmuson are appreciated.

REFERENCES

1. J. A. Abraham, "An Improved Algorithm for Network Reliability," IEEE Transactions on Reliability, R-28, No. 1, 58-62, April 1979.

2. G. C. Corynen, "'STOP': A Fast Procedure for the Exact Computation of the Performance of Complex Probabilistic Systems," UCRL-53230, January 1982.

3. G. C. Corynen, "Evaluating the Response of Complex Systems to Environmental Threats: The SIGMA PI Method," UCRL-53399, May 1983.

4. W. Feller, "An Introduction to Probability Theory and Its Applications," Vol. 1, John Wiley and Sons, Inc., New York, Second Edition, 1957.

5. D. Hunter, "Bounds for the Probability of a Union," TR73, Series 2, Department of Statistics, Princeton University, February 1974.

6. D. Hunter, "An Upper Bound On the Probability of a Union," Journal of Applied Probability, 13, 597-603, 1976.

7. H. E. Lambert, "Fault Trees for Decision Making in Systems Analysis," UCRL-51829, October 1975.

8. H. Nakazawa, "A Decomposition System for Computing System Reliability by a Boolean Expression," IEEE Transactions on Reliability, R-26, No. 4, October 1977.

9. R. C. Milton, "Computer Evaluation of the Multivariate Normal Integral," Technometrics, Vol. 14, No. 4, November 1972.

10. R. M. Thatcher, "Evaluating the Effects of Seismic Events on Systems In A Nuclear Facility Using the SIGMA PI Method," UCID-19810, May 1983.

11. J. E. Wells, et al., "SSMRP Phase I Final Report - Systems Analysis (Project VII)," NUREG/CR-2015, Vol. 8, November 1981.

12. V. K. M. Whitney, "Algorithm 422 - Minimal Spanning Tree [H]," Department of Electrical Engineering, University of Michigan, August 1971.

13. R. R. Willie, "Computer-Aided Fault Tree Analysis," ORC 78-14, Operations Research Center, University of California, Berkeley, CA, August 1978.

PI is not programmed for computer operation. Instead, the user applies the PI method to the given problem set as he prepares it for input to the SIGMA program.

In this report, we present results from the application of the SIGMA PI method on a single problem of medium size. In this experiment, accuracy in P(S) as a function of computer time expended is plotted. From this experiment, we estimate that the full SIGMA algorithm would compute P(S) to three place accuracy 40 to 70 times as fast as the speed attained by STOP alone. The relative speed of SIGMA over STOP in attaining three place accuracy tends to increase as fault tree size increases.