SANDIA REPORT

SAND2007-2758
Unlimited Release
Printed May 2007

Bayesian Methods for Estimating the Reliability in Complex Hierarchical Networks (Interim Report)

Paul T. Boggs, Youssef M. Marzouk, Philippe P. Pébay, John Red-Horse, Kathleen Diegert, and Rena Zurn

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

SAND2007-2758
Unlimited Release
Printed May 2007

Bayesian Methods for Estimating the Reliability in Complex Hierarchical Networks (Interim Report)

Paul T. Boggs
Sandia National Laboratories
M.S. 9159, P.O. Box 969
Livermore, CA 94551, U.S.A.
ptboggs@sandia.gov

Youssef M. Marzouk
Sandia National Laboratories
M.S. 9051, P.O. Box 969
Livermore, CA 94551, U.S.A.
ymarzou@sandia.gov

Philippe P. Pébay
Sandia National Laboratories
M.S. 9051, P.O. Box 969
Livermore, CA 94551, U.S.A.
pppebay@sandia.gov

John Red-Horse
Sandia National Laboratories
M.S. 0828, P.O. Box 5800
Albuquerque, NM 87185, U.S.A.
jrredho@sandia.gov

Kathleen Diegert
Sandia National Laboratories
M.S. 0830, P.O. Box 5800
Albuquerque, NM 87185, U.S.A.
kvdiege@sandia.gov

Rena Zurn
Sandia National Laboratories
M.S. 9007, P.O. Box 969
Livermore, CA 94551, U.S.A.
rmzurn@sandia.gov

Abstract

Current work on the Integrated Stockpile Evaluation (ISE) project is evidence of Sandia's commitment to maintaining the integrity of the nuclear weapons stockpile. In this report, we undertake a key element in that process: development of an analytical framework for determining the reliability of the stockpile in a realistic environment of time-variance, inherent uncertainty, and sparse available information. This framework is probabilistic in nature and is founded on a novel combination of classical and computational Bayesian analysis, Bayesian networks, and polynomial chaos expansions. We note that, while the focus of the effort is stockpile-related, it is applicable to any reasonably-structured hierarchical system, including systems with feedback.

Contents

Bayesian Methods for Estimating the Reliability in Complex Hierarchical Networks (Interim Report)

1 Introduction

Sandia has tremendous responsibilities in maintaining the integrity of the nuclear weapons stockpile and, in carrying out this mission, manages a comprehensive testing program. A major goal of this program is to determine the reliability of the stockpile and, at the same time, to calculate the confidence in that reliability estimate. This is clearly a cost-intensive activity; thus, the question arises as to whether or not the same levels of overall stockpile integrity can be achieved at lower cost, or if higher levels can be achieved within the same budget constraints.

Weapons systems are tremendously complex, multi-level affairs, and are often rife with uncertainties that can originate in any number of ways. Calculating the reliability in such settings can be extremely difficult; estimating the confidence in the reliability is often even more difficult. In this paper, we set forth our program for investigating these issues.

In this document, we take "reliability" to be the probability that system failure does not occur. In the context of our development, we define this precisely and discuss our strategy for developing estimates of reliability in complex hierarchical systems comprised of interconnected components that inherently possess uncertainty at every level of the system. We also develop a rigorous means of computing the confidence in this estimate. Moreover, due to, among other things, data gathering limitations, our mathematical descriptions of the underlying uncertainties are often incomplete. This lack of information is itself a source of uncertainty in reliability estimates. Thus, properly accounting for this uncertainty is of paramount importance, particularly in systems where it is impractical, or impossible, to perform a sufficient number of classical tests to ensure a specified confidence level.

In the sections below, we describe the development of a novel mathematical strategy, employing structured probabilistic models, for describing systems-related tests. Our goal will be to develop a rigorous means to assess the overall reliability of the systems, the associated confidence we have in this reliability, and the impact additional data acquired from subsystem and component-level tests can have. This last ability will allow us to formulate optimization problems that will determine the "best" place in the system to test to give the maximum increase in our confidence in the reliability estimate.

The basis of our strategy will be a Bayesian approach, with which we seek to update the probability distribution for the output of any component given new test data on the perfor-

mance of the system. This approach is inspired by the work of Martz and Waller [10, 9], who used a Bayesian approach to update reliability estimates based only on pass-fail data and only for special structures of the system. We extend this approach to more general Bayesian networks in which certain components may exist in, or directly affect, multiple levels of the systems. The strategy will first be applied to time-independent systems with continuous-valued data, but will be naturally extensible to time-dependent systems with continuous data. "Continuous data" refers to real-valued performance variables, e.g., voltage, that must fall between given bounds for the system to be considered working acceptably. A final foundational element of our strategy is the use of polynomial chaos expansions (PCE), with which we will represent the random variables and processes that characterize the system at all of the levels. This facilitates several important tasks: uncertainty propagation through the system; analysis of so-called uncertainties present due to lack of information; and generalizations to the time-dependent case.

Exact or approximate inference algorithms will allow probabilistic information characterizing reliability to be updated through and across systems. This is crucial for estimating the uncertainty in the overall system reliability, and for infusing appropriate updates when additional information and data are acquired.

This work is related to and inspired by our current work on the Integrated Stockpile Evaluation (ISE) project, but is more generally applicable to any hierarchical system. In fact, it should be applicable to any reasonably structured system, including, e.g., systems that have feedback.

This report is organized as follows: For the sake of completeness, we provide in section 2 some technical background on several topics. First, we describe the Bayesian networks that we will consider and why they are appropriate to this study. Next, we review Bayes' theorem in the form that we exploit . We then describe function analytic approaches to probability and the fundamentals of polynomial chaos expansions (PCE) that are essential to our approach. In this context, we consider the "stochastic dimension" and methods for preventing an explosion of this dimension. In section 3, we develop the basic model that we will use and consider some of its properties. We also formally define reliability in this setting. We then consider a very simple hierarchical system with linear functional relationships among the components. This allows us to work out in detail the effects of obtaining test data at various levels in the system. Some counter-intuitive results emerge from this exercise that have helped us to understand this effort better. In section 4, we develop a synthetic generator that will be used to produce hierarchical test systems with specified properties. These, in turn, will be used to test our final strategy over a variety of systems. In section 5 we conclude with a discussion of the next steps in our research.

2 Background

In this section we provide some background on some of the fundamental concepts that will be needed in our discussion.

2.1 Bayesian Networks

The systems that we are considering are assumed to possess uncertainty, either inherent or due to lack of information, and this is a fundamental characteristic that we considered in establishing an appropriate mathematical context. Here, we describe this and other important system considerations:

1. They are engineered hierarchical systems; any component may be comprised of subcomponents, which, in turn, may themselves contain subcomponents. Also, several instances of a single type of component[1] may appear in a given system. For example, a hydraulic system may contain five valves, three of which have identical model numbers and specifications. It is also possible that a particular instance of a component will play a role in more than one subsystem. A simple example of this is an automobile battery, which has functions both in the starting circuitry, and in, say, the instrument panel.

2. We identify the i-th component in the system with a variable X_i. This variable may represent voltage, impedance, yield stress, or any other quantity of interest. Because of measurement error, limited opportunities for testing, component-to-component variability, aging, and environmental influences, our knowledge of the exact value of X_i will be imprecise. Thus, we adopt a probabilistic approach and treat X_i as a random variable. Information on X_i will be expressed in this probabilistic framework, say using probability distributions, or functional representations. Furthermore, we will have the ability to update these probabilistic descriptions using Bayesian inference as more data become available.

3. Bayesian networks will be employed to aggregate information into a system-level probabilistic model, and to encode conditional independence relationships among various components as imposed by the system structure.

Regarding the last item above, consider the *joint* probability distribution of the random variables $\{X_i : i \in \mathcal{V}\}$, where $\mathcal{V}$ is the set of component indices. This distribution is high-dimensional and complex, describing the performance of every component of the system and dependencies among the performance values. We would like to estimate and update this distribution based on test data. We would also like to focus our attention on the performance of particular components and subsystems—in other words, to examine the marginal

[1] These will be referred to as "instances" of a "class;" see §3.

distribution of a particular subset $\{X_i : i \in \mathcal{U} \subset \mathcal{V}\}$. And we would like to calculate conditional probabilities—i.e., the probability of one subset of the variables given the values of another subset of the variables. These tasks will become computationally intractable unless we take advantage of the structure of the system. In particular, we propose using the engineered structure of the system to factor the joint probability distribution into a number of conditional probabilities.

The above serves as motivation for using probabilistic graphical modeling capabilities present in *Bayesian networks*. We provide a brief description of this concept here; for more details see, for example, Jensen [5] or Jordan [6]. Our notation here follows that of Jordan [6]. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a directed acylic graph (DAG) with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Let $X_{\mathcal{V}} \equiv \{X_i : i \in \mathcal{V}\}$ be a collection of random variables indexed by the nodes of the graph. Each node $v \in \mathcal{V}$ is associated with a set of "parent" nodes, i.e., all the nodes from which a directed edge points towards v. This set of parents is denoted by π_v and may be the empty set. Using any set of indices as a subscript, we let X_{π_v} denote the set of random variables associated with the parents of v. In a Bayesian network, the joint probability distribution of $X_{\mathcal{V}}$ factors as follows:

$$p(x_{\mathcal{V}}) = \prod_{v \in \mathcal{V}} p(x_v | x_{\pi_v})$$

where $p(\cdot)$ is a probability density function in the case of real-valued X and a probability mass function in the case of discrete-valued X.

In the present application, the structure of the graph $\mathcal{G}$ will reflect the structure of the engineered system. But this correspondence need not be exact; the Bayesian network based on $\mathcal{G}$ may include different types of nodes representing the class and specific instances of a particular component, and may also include nodes representing environmental conditions or other external factors that are relevant to system performance.

2.2 Bayes' Theorem

The ability to update our estimate of the reliability of a system based on newly acquired data is critical. And, since we will be using Bayesian updating to effect this, Bayes' Theorem is fundamental to our work. We now provide a brief summary of this important result. By definition, the conditional probability of event A given event B is given by

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

Similarly, the conditional probability of event B given event A is given by

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}.$$

Thus, by combining these two equations and rearranging terms, we obtain Bayes' Theorem:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}.$$

In this setting, the terms have standard names.

- $P(A)$ is called the *prior* probability. It is our state of knowledge prior to observing event B.
- $P(B \mid A)$ is the probability of B given A. This is often called the *likelihood* of A.
- $P(A \mid B)$ is called the *posterior* probability of A given B. That is, it is the updated probability of A given the new event B. If subsequent data is collected, this becomes the prior in the next round.
- The term $P(B)$ is called the *marginal probability* of B and acts as a normalizing constant.

Although we have derived Bayes' Theorem in terms of probabilities, the same result holds for probability densities. In particular, for a probability density, say $f(x),\ x \in X$, we can write, given data $y \in Y$

$$f(x \mid y) = \frac{f(y \mid x) f(x)}{f(y)}.$$

Again, the terms have standard names:

- $f(x)$ is called the *prior* distribution of X. It is our state of knowledge about the random variable X prior to observing $Y = y$.
- $f(y \mid x)$ is the likelihood function of x given $Y = y$.
- $f(x \mid y)$ is called the *posterior* distribution of X given y. That is, it is the updated probability density of X given the data $Y = y$. If subsequent independent data is collected, this distribution becomes the prior in the next round.
- The term $f(y)$ is the *marginal distribution* of y, also called the *evidence*, and acts as a normalizing constant.

The notational abuse of using f in all of these is conventional; each one is, in fact, different as is distinguished by its arguments.

Our approach is to use Bayes' Theorem to update distributions; thus we will use the latter form in our development. Finally, we note that it is easy to extend Bayes' Theorem to two or more variables. Specifically, in the context of the hierarchical system structure that we assume, we obtain the following useful result

$$f(x \mid y,z) \propto f(y \mid x) f(z \mid x) f(x),$$

where we have ignored the normalizing constant. This is true if the data y and z are conditionally independent given X.

2.3 Functional Analytic Approach to Probability

In the field of probability, there are two primary means of analysis. The first of these is the traditional probabilistic approach, in which one is concerned with properties of certain probabilistic entities, such as cumulative distribution functions, probability density functions, or a statistical moment, and their behavior under transformation or limit operations.

There is, however, an alternative approach, which we will refer to as a function analytic approach to probability. The basis of this approach is the recognition that random variables (RVs) and random fields (RFs) are functions with at least a subset of the domain of these functions being a sample space, Ω, of elementary events that is well-defined in the context of a probability space consisting of the triple, $(\Omega, \mathcal{S}, P)$. In addition to the sample space, the probability space consists of a σ-algebra $\mathcal{S}$ of subsets of Ω called events, and a probability measure P. Each of these entities has well-established and precise mathematical properties.

Using this structure, it is easy to observe that a random function is, in fact, not random at all, and that whatever randomness that exists in the framework is entirely associated with the occurrence of events. Thus, it seems reasonable to cast these random functions in a function analytic setting. Within this setting there are many analysis possibilities: algebraic, semi-group, topological, etc. Our goal is to develop approximations to exact functions in a Hilbert space setting.

We emphasize that under identical assumptions, probabilistic solutions that result from either analytical path are identical. They offer competing means to package information; the approach taken should be dictated by the particulars associated with a given application class. In our case, the primary goal is first to generalize deterministic problems to accommodate input parameters that are modeled as RVs with approximate probabilistic information. The function analytic approach is particularly well-suited to this. Based on experience, we expect no impediments to generalizing our implementation to time-dependent problems using RFs.

2.3.1 Scalar Polynomial Chaos Expansions

The polynomial chaos expansion (PCE) method was first conceived by N. Wiener as a means to integrate operators possessing differential Brownian motion, at the time viewed as chaotic, as an external forcing influence. While the random process he described is quite general, we will take a simpler route while noting that the transition to the more general case, first progressing to vectors containing RVs as components, then to more general random processes, is possible [13].

Consider two real-valued scalar RVs X and Y, defined on $(\Omega, \mathcal{S}, P)$, each with finite variance. Assume that there exists a functional transformation, T, between X and Y; that is, that $X = T(Y)$ is well-defined.

Since X has finite variance by assumption, it is a member of the class of square-integrable functions on its domain, the sample space; thus $X \in L_2(\Omega)$. It is also known that members of L_2 constitute a Hilbert space [12] of functions. Specifically, L_2 is a space of functions that is a complete, normed, inner product space. As one might expect, there are a number of mathematical properties that this membership entails. Most important to our purposes is that we know that it is possible to construct a generalized Fourier expansion [8] to approximate T, and that this can be done to arbitrary accuracy.

For the case of PCEs, and for reasons discussed later in this report, we choose to construct this generalized Fourier expansion in terms of a set of orthogonal polynomials in a standard normal random variable, $Y = \xi$, that we will denote as $\{\Gamma_i(\xi)\}$. The mathematical construction we have described above ensures that we can construct approximations, $T^{(n)}$, such that for some norm $\|\cdot\|$, the difference $\|T - T^{(n)}\|$ can be made arbitrarily small when taking a large enough n. In other words,

$$\lim_{n\to\infty} \|T - T^{(n)}\| = 0.$$

The $\{\Gamma_i\}$ are referred to as Hermite polynomials, and their properties can be found in any of a number of references documenting orthogonal polynomials such as [1]. In one dimension, these polynomials are specified by the following formula,

$$\Gamma_i(\xi) = \sum_{j=0}^{[i/2]} (-1)^j \frac{i!}{(i-2j)!\,j!\,2^j} \xi^{i-2j} \tag{2.1}$$

where the expression $[r]$ evaluates to the largest integer less than or equal to r and $0! = 1$. The first four of these are

$$\begin{aligned} \Gamma_0(\xi) &= 1 \\ \Gamma_1(\xi) &= \xi \\ \Gamma_2(\xi) &= \xi^2 - 1 \\ \Gamma_3(\xi) &= \xi^3 - 3\xi. \end{aligned} \tag{2.2}$$

Many orthogonal polynomials derive from particular ordinary differential equations defined on Hilbert spaces [14] known as Sturm-Liouville systems. These systems induce an associated inner product weighting function that defines their orthogonality properties. For the Hermite polynomials this weighting function is, to within a constant, readily recognized to be the probability density function of a standard normal RV, which explains our choice above. With these specifics in hand, we can now exploit the orthogonality and properties of the inner product to build our PCE-based approximations:

$$X^{(n)} = T^{(n)}(\xi) = \sum_{i=0}^{n} g_i \Gamma_i(\xi), \tag{2.3}$$

where the generalized Fourier coefficients, g_i, are given by

$$g_i = \frac{\mathbb{E}\left[X\Gamma_i(\xi)\right]}{\mathbb{E}\left[\Gamma_i^2(\xi)\right]}, \tag{2.4}$$

where $\mathbb{E}[\,\cdot\,]$ is the operator of mathematical expectation.

Note that for the discrete case it is possible to generalize Eqs 2.1 and 2.3 to higher dimensions in ξ, say $\xi = (\xi_1, \ldots, \xi_m)$, where m is referred to as the stochastic dimension. We caution, though, that the process is rather complicated; we refer the reader to [3] for the details. The passage to the infinite-dimensional case, under suitable constraints, is a RF and is the subject of [13].

We also note that there are a number of other known probability density functions that also can be identified, simply by inspection, as weighting functions for inner products from different Sturm-Liouville systems. For example, an exponentially-distributed RV defined on the interval $[0, \infty)$, can be seen to be affiliated with the Laguerre polynomials. Note that none of the underlying theory, nor the operations necessary for constructing a generalized Fourier expansion rely on a particular RV/orthogonal polynomial pairing. Thus, theoretically, it is possible that PCEs can be generalized to any appropriate pairing. These generalizations, often termed Askey Expansions in the literature, are the subject of active research.

2.3.2 Additional Features of the Approach

Taking this function analytic approach, and using PCEs in particular in doing so, has some other beneficial features that we will be exploiting:

- With each n, the expansion $T^{(n)}$ has a probability distribution that is implicitly defined. Samples from this distribution, or realizations, are almost trivial to generate since they are known functional transformations of a standard normal RV.
- For the case of random vectors, the PCE representation carries statistical dependence, including, naturally, correlation, via the now vector valued, g_i.
- Since it is possible to generate large number of transformations, $T^{(n)}$, and to test against experimentally-derived statistical constraints, it is possible to use them as a tool for addressing epistemic uncertainty.
- The time-dependent problem can be accommodated through a ready extension to approximate random processes.
- Optimization Under Uncertainty (OUU) may be rigorously addressed in a function analytic framework; this will entail augmenting the associated norms by a sample space domain.

2.4 Variables

We will represent the performance values as random variables using (truncated) polynomial expansions. More specifically, there will be at least one independent "stochastic dimension" attached to each child node, in the sense that each child node will be represented by the means of a truncated WHe expansion of at least one Gaussian random variable ξ_i, and that the ξ_i's will be mutually independent. Order and coefficients of these WHe expansions will be determined using a set of relevant methods (such as projection of an experimental distribution, or moment-based expert knowledge). Therefore, an important question that immediately arises regards the "stochastic dimensionality," i.e., the number of independent Gaussians to be used in the truncated WHe expansions. In general, practical applications of WHe expansions to uncertainty propagation assign one stochastic dimension, i.e., one independent Gaussian per uncertain parameter in the system. Although this may not be sufficient in general, we will not even be able to maintain a separate stochastic dimension for each node in $\mathcal{V}$, and will need to periodically project the truncated WHe expansions onto bases of smaller stochastic dimension as the inference process advances through the network from parents to children. One approach to perform such projections is based upon the following theorem, due to Paul Lévy:

Theorem 2.1. *Let $F : \mathbb{R} \to [0,1]$ be right-continuous, increasing (not necessarily strictly), such that $\lim_{x\to-\infty} = 0$ and $\lim_{x\to+\infty} = 1$. Let F^{-1} be its* generalized inverse*:*

$$(\forall \omega \in]0,1[) \quad F^{-1}(\omega) = \inf_{x\in\mathbb{R}} \{x : F(x) \geq \omega\}.$$

If U be a uniform random variable over $[0,1]$, then $F^{-1}(U)$ is a random variable whose CDF is F.

More specifically, Theorem 2.1 allows us to make products of the form $< X\Psi_i >$ meaningful (when X is an absolutely continuous random variable), which they are not by default since the random variables X and Ψ_i have no reason to be defined over the same probability space. Inverse mapping them to the same random variable (uniform, in the context of Theorem 2.1) ensures that the following product is well defined:

$$< X\Psi_i >= \int_\Omega F_X^{-1}(U) F_{\Psi_i}^{-1}(U)\, \mathrm{d}P,$$

where U is a uniform random variable over a probability space $(\Omega, \mathfrak{S}, P)$, and F_X and F_{Ψ_i} are the respective CDFs of X and Ψ_i. For instance, consider the case of $\Psi_0 = 1$, for which a straightforward calculation yields $F_{\Psi_0}^{-1}(U) = 1$. Therefore, thanks to Theorem 2.1, one has

$$< X\Psi_0 >= \int_\Omega F_X^{-1}(U)\, \mathrm{d}P = \mathbb{E}\left[F_X^{-1}(U)\right] = \mathbb{E}\left[X\right],$$

since $F_X^{-1}(U)$ is a random variable with the same CDF as X. This result confirms that the projection strategy retrieves, at 0-th order, the expectation. At first order, if $X = \xi$, then one has

$$< X\Psi_1 >= \int_\Omega F_\xi^{-1}(U)^2\, \mathrm{d}P = \mathbb{E}\left[F_\xi^{-1}(U)^2\right] = \mathbb{E}\left[\xi^2\right] = 1,$$

which shows that in that case, at first order, the approximation is exact and is indeed obtained using the projection method. However, in general, the $< X\Psi_i >$ integrals cannot be analytically calculated even when F_X is known, and the matter is further complicated when only a sample of realizations of X is provided; therefore, one has to resort to numerical quadrature to evaluate these integrals.

3 The Model

3.1 Class *versus* instance

In the context of our model, we will distinguish between *class* networks (that are applicable to classes of devices) and *instance* networks (that are applicable to particular instances in a class of devices). This is motivated by the fact that, whereas the knowledge that can be gleaned as to a collection of devices is directly applicable to each individual device that pertains to that class, the converse is not true. In other words, there is direct *inheritance* from the class to the instance, while in the opposite direction, there can merely be an update of the class properties based on an individual device's properties.

Note that how such an update must be performed in practice is not obvious: for example, if the class properties are determined from the statistics of samples of a particular device, then the information obtained by inspection of a new instance of this class of devices must be appropriately weighted in order to reflect its size (1, in this case) with respect to the size of the initial sample. However, this method cannot be used if the class properties have been inferred using a non-sampling technique, or if sampling has been used but the sampling parameters are no longer available. A simple example to make this discussion less abstract is the following: assume that it is known that the predicted height (in cm) in a given age group is $\bar{h} = 120$, with a standard deviation of $\sigma_h = 5$, based on a sample of $n = 100$ children. Consequently, if an other child (instance) in this age group (class) is examined and found to have a height of $h^* = 130$, then the class properties can be straightforwardly updated as follows:

$$\begin{aligned}\bar{h}' &= \frac{n}{n+1}\bar{h} + \frac{1}{n+1}h^* \\ &= \frac{100}{101}120 + \frac{1}{101}130 \\ &= 120.1,\end{aligned}$$

and

$$\begin{aligned}\sigma'_h &= \sqrt{\frac{n}{n+1}(\sigma_h^2 + \bar{h}^2) + \frac{1}{n+1}h^{*2} - \bar{h}'^2} \\ &= \sqrt{\frac{100}{101}(5^2 + 120^2) + \frac{1}{101}130^2 - 120.1^2} \\ &\approx 5.05.\end{aligned}$$

On the other hand, say that we are provided instead with the class property that in this age group, height in cm is distributed according to a normal distribution with mean 120 and standard deviation 5. Although this is a much more informative characterization (with, in particular, the same information at first and second orders), it becomes now unclear how the observation of an additional instance with $h^* = 130$ can be fed back to the class property.

To overcome this issue, several strategies can be considered; one simply consists of sampling a non-frequentist property. For instance, the normal distribution can be arbitrarily sampled, then the new instance added to the sample, and finally, an updated distribution fitted to the resulting sample. This approach is easy to implement, but it suffers shortcomings: in particular, the size of the sample to be used is entirely arbitrary, and will influence the final result. Also, the fitting process requires that a model is used for the distribution. One can, for instance, use a truncated PCE, but again, this choice will influence the final result.

A more principled—and indeed Bayesian—approach consists of formulating the *parameters* of the class property distribution in probabilistic terms. In the previous example, these parameters are the mean and variance of the normally-distributed height; a Bayesian approach would describe the mean and variance with probability distributions of their own. The widths of these distributions affect how strongly the class property distribution $p(h|\mu_h, \sigma_h^2)$ is influenced by new data. Beginning with rather uninformative priors for μ_h and σ_h, Bayes' rule describes how to condition on successive property instances; the distributions of μ and σ will typically narrow as more data become available.

3.2 Reliability

We propose to define *reliability* as the probability that a random variable has a value within a prescribed range. Formally, this means that the reliability of X with given acceptable bounds $x_{\min}$ and $x_{\max}$ is:

$$\mathcal{R}_X(x_{\min}, x_{\max}) = P(x_{\min} < X < x_{\max})$$

For example, if the range of acceptable values is $[-2, 2]$, the reliability of a standard normal random variable is *ca.* 95%.

3.3 First Problem

In this section we consider a simple system and a formal statement of the question(s) that we want to be able to answer.

Our simple system is illustrated in figure 3.3 and consists of a system, C, with two subsystems, A and B. The assumptions we make are as follows:

1. Assume that both A and B have one output value that is transmitted to C. Let a be the output of A and b be the output of B.
2. Assume that C provides a known functional form for combining these parent values. As a first case, assume that this form is linear with additive error, i.e., C has output c given by

$$c = \alpha a + \beta b + \eta$$

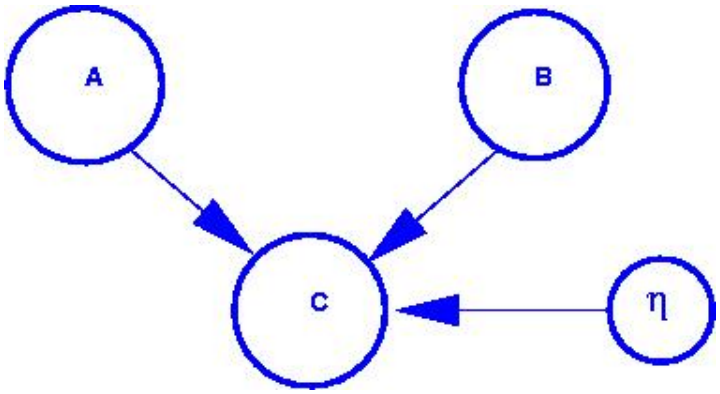

Figure 1. Simple system consisting of one system, C, that has two subsystems, A and B (A and B are considered the "parents" of C).

where α and β are given, i.e., we know what the designer had in mind, and η is an additive error. Assume that η has a known distribution, say normal.

3. Assume that the distributions on a and b correspond to PC expansions of the form

$$a = \sum_{i=0}^{N_a} a_i H_i(\xi)$$
$$b = \sum_{i=0}^{N_b} b_i H_i(\xi).$$

4. Assume that we have prior distributions for the coefficients a_i and b_i; these distributions will be updated as we gather data. This is justified by the Bayesian notion that the coefficients are determined by data, and that our state of knowledge about the coefficients' values can be represented by (conditional) probability distributions. Distributions on a_i and b_i will give rise to families of distributions for a and b. The priors we choose for a_i and b_i may reflect the intended operating conditions or nominal outputs of components A and B.

5. Assume that we are given limits of acceptability for each node in the system, i.e., upper and lower bounds of acceptability of the output.

Now we assume that we have collected a new piece of data. This data may come in various forms, depending on where the tests were conducted. Let $\hat{a}$, $\hat{b}$, and $\hat{c}$ denote data collected at A, B, and C, respectively. We must consider all possibilities of a new piece of data. For example, we could perform a complete test of the system and obtain the triple

$$\hat{d} = (\hat{a}, \hat{b}, \hat{c})$$

or we could do only a test at one of the nodes to obtain, e.g.,

$$\hat{d} = (\hat{c}),$$

or tests at two of the nodes to obtain, e.g.,

$$\hat{d} = (\hat{a}, \hat{c}),$$

or any other combination.

Given our definition of reliability as stated in section 3.2, we can pose the following questions:

1. How do we update the probability distribution at each node? Even if we do not perform a test directly on one of the nodes, we may still be able to update the distribution on that node.

2. What is our estimate of the reliability of the system? As stated in section 3.2, this is just the integral between the limits provided.

3. What is our confidence in this estimate? That is, since we have distributional information on the coefficients, this should give us enough information to be able to get a distribution of the reliability estimates. For example, we could simply sample the distributions at the parent levels and use these to create a distribution for the reliability estimates.

4. Since we want to be able to control the growth of the stochastic dimension as we proceed through the system, how can we reduce the implied dimension at the child level? That is, even if there is only one stochastic dimension at each of the, say P, parent systems (two in our simple example), there would be at least $P+1$ stochastic dimensions at the child node (C in our example). We can think of a couple of possibilities: First, one can simply truncate the PCE. If we know, for example, that one of the parent nodes is extremely stable (its variance is very small), then it may make sense to drop the stochastic term corresponding to this node. Second, if the desired dimension at the child level is one, the theorem 2.1 can be used. Third, we may be able to impose constraints on the projection to create a projection onto more than one stochastic dimension or to create different objective functions for doing the projection.

3.4 Basic Methodology

Here we assume that we have the system as described in figure 3.3 and recall that the node η is never directly testable, but, as we show below, is just a device to allow us to deal with the error in C in a consistent manner.

Here are the steps to deal with this system:

1. Initialize: At parent node A, supply a prior pdf for the a_i. For example, if

$$a = a_0 + a_1\Psi_1(\xi_a) + a_2\Psi_2(\xi_a) + \ldots$$

where we can take $\bar{a_0}$ to be the designed (or target) value of a, $\bar{a_1}$ to be the variance of a based on engineering judgment or experience, and $\bar{a_i}$ to be zero for all $i > 1$. Of course, we have to specify actual prior distributions for the a_i, satisfying these simple constraints. These could be broad uniform distributions, or normal distributions, all independent of each other.

2. Initialize B in the same way.

3. Initialize η by assuming it is normal and provide initial distributions for the mean and variance. Call the values η_i. This is just a first order PCE; it could also be a higher order expansion, if desired. (Then we would have to initialize it as we did for A and B.)

4. We want to update $p(a_i, b_i, \eta_i)$ when we are given some data that we collect at the nodes. At the beginning, this probability density factors as follows, since the specifications above are independent:

$$p_0(a_i, b_i, \eta_i) = p_0(a_i)p_0(b_i)p_0(\eta_i).$$

We are also using shorthand for the successive indices; i.e., a_i really means $a_1 \ldots a_n$ if there are n coefficients in the PC expansion of a.

5. Assume we collect data, say $\hat{a}$. Then, applying Bayes rule,

$$p_1(a_i, b_i, \eta_i \mid \hat{a}) \propto p(\hat{a} \mid a_i)p_0(a_i)p_0(b_i)p_0(\eta_i),$$

since $\hat{a}$ is independent of b_i and η_i.

6. Assume we next collect data, say $\hat{b}$. Then

$$p_2(a_i, b_i, \eta_i \mid \hat{a}, \hat{b}) \propto p(\hat{b} \mid a_i, b_i, \eta_i; \hat{a})p_1(a_i, b_i, \eta_i),$$

where $p(\hat{b} \mid a_i, b_i, \eta_i; \hat{a})$ is the likelihood function of the coefficients a_i, b_i, η_i with the new data, $\hat{b}$. Since $\hat{b}$ is independent of $\hat{a}$, and is *also* independent of a_i and η_i, it follows that

$$p_2(a_i, b_i, \eta_i \mid \hat{a}, \hat{b}) \propto p(\hat{b} \mid b_i)p_1(a_i, b_i, \eta_i).$$

7. Assume we next collect data, say $\hat{c}$ at the child node. Then

$$p_3(a_i, b_i, \eta_i \mid \hat{a}, \hat{b}, \hat{c}) \propto p(\hat{c} \mid a_i, b_i, \eta_i; \hat{a}, \hat{b})p_2(a_i, b_i, \eta_i).$$

8. Expanding this last expression, we obtain

$$\begin{aligned} p_3(a_i, b_i, \eta_i \mid \hat{a}, \hat{b}, \hat{c}) &\propto p(\hat{c} \mid a_i, b_i, \eta_i)p_2(a_i, b_i, \eta_i) \\ &\propto p(\hat{c} \mid a_i, b_i, \eta_i)p(\hat{b} \mid b_i)p(\hat{a} \mid a_i)p_0(a_i)p_0(b_i)p_0(\eta_i) \\ &\propto p(\hat{c} \mid a_i, b_i, \eta_i)p_0(\eta_i) \\ &\quad \times p(\hat{b} \mid b_i)p_0(b_i) \\ &\quad \times p(\hat{a} \mid a_i)p_0(a_i). \end{aligned}$$

9. We can easily see that the addition of one new piece of data simply adds an appropriate factor to one of the lines above. For example, suppose we get $\hat{a_2}$. Then we have

$$\begin{aligned} p_4(a_i, b_i, \eta_i \mid \hat{a}, \hat{b}, \hat{c}, \hat{a_2}) &\propto p(\hat{c} \mid a_i, b_i, \eta_i) p_0(\eta_i) \\ &\quad \times p(\hat{b} \mid b_i) p_0(b_i) \\ &\quad \times p(\hat{a_2} \mid a_i) p(\hat{a} \mid a_i) p_0(a_i). \end{aligned}$$

It follows from this formula that the order of collecting the data does not matter, i.e., each new observation puts a multiplicative factor in the above formula.

10. Next consider the collection of two pieces of data at the same time. For example, suppose we collect $\{\hat{a_+}, \hat{c_+}\}$. (Note: we use the bracket notation to indicate that we have collected the new data at the same time.) This presents a new situation. Recall that

$$c = \alpha a + \beta b + \eta.$$

Thus, we can think of this as having new data for a and for the combination $\beta b + \eta$. Now,

$$\begin{aligned} p(a_i, b_i, \eta_i \mid \{\hat{a_+}, \hat{c_+}\}) &\propto p(\{\hat{a_+}, \hat{c_+}\} \mid a_i, b_i, \eta_i) p(a_i, b_i, \eta_i) \\ &\propto p(\{\hat{a_+}, \widehat{\beta b + \eta}\}) p(a_i, b_i, \eta_i) \\ &\propto p(\hat{a_+} \mid a_i) p(\widehat{\beta b + \eta} \mid b_i, \eta_i) p(a_i, b_i, \eta_i). \end{aligned}$$

Here we see that, although the joint probability function does not factor, the data does. Clearly the same will hold for new data of the form $\{\hat{b_+}, \hat{c_+}\}$ and will add a factor of the form $p(\widehat{\alpha a + \eta} \mid a_i, \eta_i)$ to the above. Similarly, new data of the form $\{\hat{a_+}, \hat{b_+}, \hat{c_+}\}$ will add a factor of the form $p(\hat{\eta} \mid a_i, b_i, \eta_i)$.

Some comments are in order:

1. The joint probability $p(a_i, b_i, \eta_i)$ will continue to factor into $p(a_i)p(b_i)p(\eta_i)$ until data is collected at C. At that point, the data at C induces a dependence between the a_i and the b_i.

2. The joint posterior distribution of a_i, b_i, η_i at each step can be explored by Markov chain Monte Carlo (MCMC). MCMC encompasses a broad class of methods that simulate drawing samples from complex probability distributions [4]. In particular, Metropolis-Hastings algorithms provide for the construction of a Markov chain with stationary distribution equal to our posterior, using only evaluations of the density $p_i(a_i, b_i, \eta_i|\cdot)$ up to a multiplicative constant. From this chain of samples, it is easy to compute means, variances, and marginal distributions for any of the PC coefficients.

3. Evaluation of the joint posterior density $p_i(a_i, b_i, \eta_i|\cdot)$ at any stage of the inference process detailed above will entail evaluating likelihood functions such as $p(\hat{a} \mid a_i)$ or $p(\widehat{\beta b + \eta} \mid b_i, \eta_i)$. Using a Metropolis-Hastings algorithm requires that we be able to evaluate these functions at arbitrary a_i, b_i, η_i, with arbitrary data. The most robust method for evaluating the likelihood may be via sampling [2]. To illustrate this procedure, consider the likelihood function $L(a_i) \equiv p(\hat{a} \mid a_i)$. Given values for the PC coefficients a_i (e.g., drawn from the proposal distribution at each MCMC iteration), we can sample the random variable ξ_a underlying the PC expansion of a. Evaluate $a^{(s)} = \sum_i a_i \Psi_i(\xi_a^{(s)})$ at each sample $\xi_a^{(s)}$. A histogram of the resulting samples $\{a^{(s)}\}$, properly normalized, will provide an estimate of the density $p(a \mid a_i)$. Then we evaluate this density at each $\hat{a}$ to obtain the likelihood.

   Strictly speaking, evaluating the density of a at an arbitrary argument, using a set of samples $\{a^{(s)}\}$, involves some form of kernel density estimation:

$$p(a) = \frac{1}{n} \sum_{s=1}^{n} K\left(a|a^{(s)}\right)$$

   where $K\left(a|a^{(s)}\right)$ is a density concentrated around $a^{(s)}$ [4]. Here, we would typically use a one-dimensional Gaussian kernel, $K = N(a^{(s)}, \sigma_s^2)$ with bandwidth σ_s selected by the user or via various optimality considerations.

   An alternative to sampling and kernel density estimation may be to evaluate analytical expressions for the probability density of a (functional) transformation of a random variable, given the density of the latter [11]. For example, if the likelihood depends on a one-dimensional PC expansion, we have the transformation $a = f(\xi_a; a_i)$, where f represents the PC expansion of a. Then the probability density of a is given by

$$p(a) = \sum_{\xi^* \in \Xi_a} \frac{p_\xi(\xi^*)}{|f'(\xi^*)|}$$

   where p_ξ is the known probability density of ξ_a and Ξ_a is the set of roots of $a - f(\xi)$.

Next we have to calculate the reliability estimate and the confidence in that estimate. Assume that we want to do this at A and that we have upper and lower bounds for the acceptable performance, say $\underline{a}$ and $\overline{a}$, respectively.

Then we proceed as follows:

1. Compute N samples of the reliability by the following procedure:
   (a) Set $k = 1$.
   (b) Obtain one joint sample of the a_i; this is one possible set of PC coefficients for a. Using this set of PC coefficients, we must obtain the probability density of a. This can be done in two ways: (1) sample the underlying ξ_a and, with the

help of kernel density estimation, build an approximation to $p(a|a_i)$; (2) use an analytical formula that requires finding the (real) roots of the polynomial $a - \sum a_i \Psi_i(\xi_a)$ and the derivatives at those roots [11]. Either of these procedures will be used to evaluate each likelihood function during MCMC (see the discussion at comment #2 above).

(c) Compute

$$r_k = \int_{\underline{a}}^{\overline{a}} p(a \mid a_i) da$$

(d) Set $k = k + 1$; repeat as necessary.

2. Compute whatever statistics are desired about this set $\{r_k\}$, e.g., the mean and variance, quantiles, etc.

Clearly, the computations at B and C will be the same. And note that samples of the PC coefficients a_i, b_i, etc, used in step 1 above will be generated by the MCMC algorithm. Thus many of the calculations can be reused.

4 Specification of a Synthetic System Generator

The purpose of this section is to develop a means of describing a complex hierarchical system in a compact form and then using this form to generate automatically a collection (or sample) of a specified number of instances of this system. This sample can then be used to test our methods for estimating the reliability. Since we have the "true" system and we can control the reliability of it, we will have a way of comparing our results to the "truth."

The idea is that a "system" in our context consists of a collection of subsystems, each of which is itself a system. In addition, a system also contains a set of tests that can be run on this system. We can thus use this strategy to create arbitrarily complex (hierarchical) systems. The subsystems can be either specifically named systems or can be chosen from a specified set of functionally equivalent systems. For example, an engine component could be specified as an instance of an E123 engine, or chosen randomly from E121, E122, or E123.

Describing the tests is a key part of this model. Testing first requires a statement of the functional dependence on the subsystems and some specification of the errors that can be present. The errors could include both the random errors and the systematic errors. When a "realization" of this system is generated, the tests section will contain the actual values of the tests. When we have to take into account time-dependence, we will also have to have some way to include this information.

4.1 Generic Description of an Individual System

We first specify a way to designate a system. We have in mind using XML as the language. Here is the first example of a system description. It describes our simple "A-B-C" system where A is the top level system with subsystems B and C.

```
<SystemSet>
   <System name = "A"  >
      <SubSystemSet>
          <System name = "B" numberToGenerate = "1"/>
          <System name = "C" numberToGenerate = "1"/>
      </SubSystemSet>

   <TestSet>
      <Test name="test1" functForm = "linear" available_at_parent="no"
        isOutput = "yes">
         <coeff sub = "B" testName = "test1" outputNum="0" value = "1.0"/>
         <coeff sub = "C" testName = "test1" outputNum="0" value = "2.0"/>
         <coeff sub = "constant_term" value = "5.0"/>
```

```
       <error
              random_error_dist = "normal"
              random_error_mean = "0.0"
              random_error_var = ".04"
              systematic_error = ".005"
       />

       <bounds target = "7.0" upper = "7.5" lower = "6.8"/>

    </Test>
 </TestSet>

 </System>



<System name = "B"  >

    <SubSystemSet>
       <System name = "input" value = "3.14"
               numberToGenerate = "1">
       <error
              random_error_dist = "normal"
              random_error_mean = "0.0"
              random_error_var = ".02"
              systematic_error = ".003"
       />
       </System>
    </SubSystemSet>

   <TestSet>
    <Test name="test1" functForm = "linear" available_at_parent="no"
      isOutput = "yes">
       <coeff sub = "input" outputNum="0" value = "1.0"/>
       <coeff sub = "constant_term" value = "2.718"/>

       <error
              random_error_dist = "normal"
              random_error_mean = "0.0"
              random_error_var = ".02"
              systematic_error = ".003"/>


       <bounds target = "5.85" upper = "5.92" lower = "5.65"/>

    </Test>
   </TestSet>
```

```
  </System>

  <System name = "C"  >

      <SubSystemSet>
         <System name = "input" value = "1.6"
                  numberToGenerate = "1">
         <error
                random_error_dist = "normal"
                random_error_mean = "0.0"
                random_error_var = ".02"
                systematic_error = ".003"
         />
         </System>
      </SubSystemSet>

     <TestSet>
      <Test name="test1" functForm = "linear" available_at_parent="no"
        isOutput = "yes">
         <coeff sub = "input" outputNum="0" value = "2.0"/>
         <coeff sub = "constant_term" value = "1.414"/>

         <error
                random_error_dist = "normal"
                random_error_mean = "0.0"
                random_error_var = ".03"
                systematic_error = ".004"
         />

         <bounds target = "4.61" upper = "4.7" lower = "4.58"/>

      </Test>
     </TestSet>

  </System>



</SystemSet>
```

The hard part is the specification of the tests. We need to design this part carefully and be clear about the errors that we propose to address.

Remarks on describing the tests:

- For the linear case, we can simply supply a parameter for each subsystem and, pos-

sibly, a constant.

- Each subsystem can provide several values to the system in which it operates. Thus a test may involve only a subset of the inputs from a given subsystem and only a subset of the subsystems.
- A system described as above can be considered as the general "family" of such systems. Tests can be used to learn about the family and/or to learn about the actual system in which they are embedded.
- A test of a system gives information about the family and thus has an effect on other systems that contain systems from the same family.
- We have put a specification of a simple, normal error with a bias. This can clearly be expanded as necessary to create more varying situations.
- Our goal will be to construct a testing strategy that will yield information about performance of the systems at other levels in the overall system.

4.2 Specification of a Sample Set

Given the above descriptions of systems, we need to be able to specify a means to generate a sample. We suppose that there will be fielded systems of various ages along with new systems that have been recently manufactured. These systems may contain a variety of different versions of the components or subsystems. Here is our first cut at specifying a sample. The first section of this file simply specifies where the input comes from (sampleB.xml) and where to store the output (sampleB.xml).

```
<Generate>

<Files
     allSystemsInFile     = "systemsB.xml"
     sampleXML            = "sampleB.xml"

/>


<Sample>

     <System name = "A" numberToGenerate= "2">

          <SubSystemSet>
                 <System name="B" numberToGenerate = "1">
                      <SubSystemSet>
                         <System name = "input" numberToGenerate ="1"/>
```

```
                    </SubSystemSet>
                </System>
                <System name="C" numberToGenerate = "1">
                    <SubSystemSet>
                        <System name = "input" numberToGenerate ="1"/>
                    </SubSystemSet>
                </System>
            </SubSystemSet>
        </System>

        <System name="B" numberToGenerate="1">
            <SubSystemSet>
                <System name="input" numberToGenerate = "1"/>
            </SubSystemSet>

        </System>

        <System name="C" numberToGenerate="1">
            <SubSystemSet>
                <System name="input" numberToGenerate = "1"/>
            </SubSystemSet>

        </System>

</Sample>



</Generate>
```

Thus the specification of the sample needs to include the number of each system type required and, for each type, the specification of the particular subsystems. As noted above, we could allow for some randomness in this choice. Although this could require a fairly long XML description, it is relatively straightforward.

A Python script has been written to use the information from the specification script (generateB.xml) and the system file (systemB.xml) and produce the output sample (contained in sampleB.xml). The script is called GenTotalB.xml and is in the src directory. The output of this first run is a sample with two instances of the system A and one instance each of systems B and C. Note that since A contains B and C, there are two instances of B and C that are contained in the two samples of A. Here is the output.

```
<SampleSet>
  <System sampleNumber='0' name='A'>
```

```
  <SubSystemSet>
    <System sampleNumber='1' name='B'>
      <SubSystemSet>
        <System outputNum='0' name='input' value='3.03834385956'>
          <SubSystemSet/>
          <TestSet/>
        </System>
      </SubSystemSet>
      <TestSet>
        <Test outputNum='0' name='test1' value='5.62580035036'/>
      </TestSet>
    </System>
    <System sampleNumber='2' name='C'>
      <SubSystemSet>
        <System outputNum='0' name='input' value='1.54628349191'>
          <SubSystemSet/>
          <TestSet/>
        </System>
      </SubSystemSet>
      <TestSet>
        <Test outputNum='0' name='test1' value='4.58812458191'/>
      </TestSet>
    </System>
  </SubSystemSet>
  <TestSet>
    <Test outputNum='0' name='test1' value='19.6007914937'/>
  </TestSet>
</System>
<System sampleNumber='3' name='A'>
  <SubSystemSet>
    <System sampleNumber='4' name='B'>
      <SubSystemSet>
        <System outputNum='0' name='input' value='3.05499334439'>
          <SubSystemSet/>
          <TestSet/>
        </System>
      </SubSystemSet>
      <TestSet>
        <Test outputNum='0' name='test1' value='5.83762072724'/>
      </TestSet>
    </System>
    <System sampleNumber='5' name='C'>
      <SubSystemSet>
        <System outputNum='0' name='input' value='1.54157915002'>
          <SubSystemSet/>
          <TestSet/>
        </System>
```

```
        </SubSystemSet>
        <TestSet>
          <Test outputNum='0' name='test1' value='4.51990787038'/>
        </TestSet>
      </System>
    </SubSystemSet>
    <TestSet>
      <Test outputNum='0' name='test1' value='20.0708486615'/>
    </TestSet>
  </System>
  <System sampleNumber='6' name='B'>
    <SubSystemSet>
      <System outputNum='0' name='input' value='3.02685978074'>
        <SubSystemSet/>
        <TestSet/>
      </System>
    </SubSystemSet>
    <TestSet>
      <Test outputNum='0' name='test1' value='5.85551711292'/>
    </TestSet>
  </System>
  <System sampleNumber='7' name='C'>
    <SubSystemSet>
      <System outputNum='0' name='input' value='1.33212402105'>
        <SubSystemSet/>
        <TestSet/>
      </System>
    </SubSystemSet>
    <TestSet>
      <Test outputNum='0' name='test1' value='3.91317795007'/>
    </TestSet>
  </System>
</SampleSet>
```

5 Conclusion

In this paper we have considered the problem of determining the reliability of complex hierarchical systems. In particular, we have considered "engineered" systems where we know the intended performance of each component and have functional relationships for how components are used at higher levels. We have described in detail the assumptions we make on the system, our definition of the reliability in this context, and our procedure for determining the reliability based on tests at various levels within the system. Finally, we have shown how to estimate confidence in the estimate for the reliability.

We recognize that there is much to do to complete even this early phase of the project. Several things are highlighted below:

- Complete the analysis of the suggested procedures to ensure that we are on a solid mathematical foundation.
- Work out detailed procedures and algorithms for carrying out the indicated computations.
- Pay careful attention to the retention of intermediate computational results in order to be able to re-use these results as necessary in later stages.
- Develop a suite of test examples and run these procedures to demonstrate proof-of-concept.

Other issues may arise as we complete these tasks; we will continue to update this document as necessary.

References

[1] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, Inc., New York, NY, 1972.

[2] R. G. Ghanem and A. Doostan. On the construction and analysis of stochastic models: Characterization and propagation of the errors associated with limited data. *Journal of Computational Physics*, 217:63–81, 2006.

[3] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover Publications, Inc, Mineola, NY, revised edition, 2003.

[4] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.

[5] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[6] M. I. Jordan. Graphical models. *Statistical Science*, 19:140–155, 2004.

[7] S. Kakutani. Spectral analysis of stationary Gaussian processes. In J. Neyman, editor, *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 2, pages 239–247. University of California Press, 1961.

[8] A. N. Kolmogorov and S. V. Fomin. *Elements of the Theory of Functions and Functional Analysis*. Dover Publications, Mineola, NY, 1957.

[9] H. F. Martz and R. A. Waller. Bayesian reliability analysis of complex series/parallel systems of binomial subsystems and components. *Technometrics*, 32:407–416, 1990.

[10] H. F. Martz, R. A. Waller, and E.T. Fickas. Bayesian reliability analysis of series systems of binomial subsystems and components. *Technometrics*, 30:143–154, 1988.

[11] A. Papoulis and S. U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 4th edition, 2002.

[12] Walter Rudin. *Real and Complex Analysis*. McGraw-Hill, 3rd edition, 1987.

[13] N. Wiener. The homogeneous chaos. *Amer. J. Math.*, 60:897–936, 1938.

[14] N. Young. *An Introduction to Hilbert Space*. Cambridge University Press, Cambridge, UK, 1997.

DISTRIBUTION:

3 MS 9018
Central Technical Files, 8940-1

1 MS 0899
Technical Library, 9616

1 MS 9021
Classification Office, 8511, for
Technical Library, MS 0899, 9616
DOE/OSTI via URL