SAND2004-2439
Unlimited Release
Printed July 2004

UC-505

A User's Guide to Sandia's Latin Hypercube Sampling Software: LHS UNIX Library/Standalone Version

Laura P. Swiler
Optimization and Uncertainty Estimation Department
Gregory D. Wyss
Systems and Vulnerability Analysis Department
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185-0847

Abstract

This document is a reference guide for the UNIX Library/Standalone version of the Latin Hypercube Sampling Software. This software has been developed to generate Latin hypercube multivariate samples. This version runs on Linux or UNIX platforms. This manual covers the use of the LHS code in a UNIX environment, run either as a standalone program or as a callable library. The underlying code in the UNIX Library/Standalone version of LHS is almost identical to the updated Windows version of LHS released in 1998 (SAND98-0210). However, some modifications were made to customize it for a UNIX environment and as a library that is called from the DAKOTA environment. This manual covers the use of the LHS code as a library and in the standalone mode under UNIX.

Acknowledgments

Ronald L. Iman, Michael J. Shortencarier, J. M. Davenport, and D. K. Ziegler developed the original LHS package at Sandia during the late 1970s and early 1980s (SAND83-2365). The authors are indebted to them for their pioneering work in the area of Latin hypercube sampling. Gregory Wyss and Sharon Daniel implemented a major upgrade to the software in the mid-1990s, converting it from Fortran 77 to Fortran 90, adding more than 25 new distributions, and including functionality that made the code much more portable (SAND98-0210).

Michael Eldred, Sharon Daniel, Laura Swiler, and Shannon Brown have been involved in porting the 1998 version of LHS to a Linux/UNIX environment.

Contents

Figures

Tables

1. Introduction

For more than twenty years, the Latin hypercube sampling (LHS) program has been successfully used to generate multivariate samples of statistical distributions. Its ability to use either Latin hypercube sampling or pure Monte Carlo sampling with both random and restricted pairing methods has made it an important part of uncertainty analyses in areas ranging from probabilistic risk assessment (PRA) to complex simulation modeling.

The original version of LHS developed at Sandia National Laboratories was documented in SAND83-2365 (Iman and Shortencarier). This code was substantially revised, extended, and upgraded in the mid-1990s. Gregory Wyss, Sharon Daniel, and Kelly Jorgensen designed and implemented much of this upgrade to the LHS software, converting it from Fortran 77 to Fortran 90, adding more than 25 new distributions, and including functionality that made the code much more portable. The revised version also included development of a Windows-based user interface to assist the user with input preparation as well as a graphical output system to support plotting of distributions generated by LHS. The documentation of the capabilities of the revised LHS code is presented in SAND98-0210 (Wyss and Jorgensen, 1998).

Michael Eldred, Sharon Daniel, Laura Swiler, and Shannon Brown ported the 1998 version of LHS (which was primarily designed for a Windows platform) to a Linux/UNIX environment in 2003-2004. This process involved writing some additional functionality to allow the LHS code to be called as a library from within the DAKOTA software environment ("input-by-call" vs. input by file), as well as some changes to modernize the code and make it more compatible with the needs of advanced simulators (e.g., converting single precision variables to double precision). The version of LHS that runs under a Linux or UNIX operating system can be compiled to run in two ways: called as a library or as a standalone LHS code run with file input.

This report documents the capabilities of the LHS UNIX Library/Standalone version. There is significant overlap in this report with SAND98-0210, however we wanted to create a separate User's Guide specifically for the UNIX version of LHS.

This manual is divided into six chapters. The next chapter is an introduction that describes the concepts involved in the Latin hypercube sampling and restricted pairing methods employed by the software. Chapter 3 describes how to run the LHS software in standalone mode and specifies the input file format needed, while Chapter 4 contains detailed descriptions of the distributions supported by LHS. Chapter 5 describes how to run LHS as a callable library from within DAKOTA, while Chapter 6 is a summary of the report.

Note: If you are familiar with LHS and just want the "quick guide" to the keywords that control LHS execution in standalone mode, jump to Section 3.2.

2. Latin Hypercube Sampling Theory

Latin hypercube sampling was developed to address the need for uncertainty assessment for a particular class of problems. Consider a variable Y that is a function of other variables $X_1, X_2, \ldots, X_k$. This function may be very complicated, for example, a computer model. A question to be investigated is "How does Y vary when the Xs vary according to some assumed joint probability distribution?" Related questions are "What is the expected value of Y?" and "What is the 99th percentile of Y?"

A conventional approach to these questions is to apply Monte Carlo sampling. By sampling repeatedly from the assumed joint probability density function of the Xs and evaluating Y for each sample, the distribution of Y, along with its mean and other characteristics, can be estimated. The LHS software supports this approach for generating samples of the Xs. The program output, for n Monte Carlo repetitions, is a set of n vectors of input variables (each such vector is k-dimensional). Each input vector can then be evaluated by the function or program to generate n values of the result Y (Y may be a scalar or a vector whose dimensionality is determined by the function or program of interest). This approach yields reasonable estimates for the distribution of Y if the value of n is quite large. However, since a large value of n requires a large number of computations from the function or program of interest, which is potentially a very large computational expense, additional methods of uncertainty estimation were sought.

2.1 Sampling

An alternative approach, which can yield more precise estimates, is to use a constrained Monte Carlo sampling scheme. One such scheme, developed by McKay, Conover, and Beckman (1979), is Latin hypercube sampling. Latin hypercube sampling selects n different values from each of k variables $X_1, \ldots X_k$ in the following manner. The range of each variable is divided into n nonoverlapping intervals on the basis of equal probability. One value from each interval is selected at random with respect to the probability density in the interval. The n values thus obtained for X_1 are paired in a random manner (equally likely combinations) with the n values of X_2. These n pairs are combined in a random manner with the n values of X_3 to form n triplets, and so on, until n k-tuplets are formed. These n k-tuplets are the same as the n k-dimensional input vectors described in the previous paragraph. This is the Latin hypercube sample. It is convenient to think of this sample (or any random sample of size n) as forming an ($n \times k$) matrix of input where the ith row contains specific values of each of the k input variables to be used on the i^{th} run of the computer model.

The Latin hypercube sampling technique has been applied to many different computer models since 1975. A more detailed description of Latin hypercube sampling with application to sensitivity analysis techniques can be found in Iman, Helton, and Campbell (1981a, b). A tutorial on Latin hypercube sampling may be found in Iman and Conover (1982b). A recent comparison of Latin hypercube sampling with other techniques is given in Helton and Davis (2001).

To help clarify how intervals are determined in the Latin hypercube sample, consider a simple example where it is desired to generate a Latin hypercube sample of size $n = 5$ with two input variables. Let us assume that the first random variable X_1 has a normal distribution with a mean value of μ and a variance of σ^2. The endpoints of the intervals are easily determined based on the parameters μ and σ^2. The intervals for $n = 5$ are illustrated in Figure 2-1 in terms of both the density function and the more easily used cumulative distribution function (CDF). The intervals in Figure 2-1 satisfy

$$P(-\infty \le X_1 \le A) = P(A \le X_1 \le B) = P(B \le X_1 \le C)$$

$$= P(C \le X_1 \le D) = P(D \le X_1 \le \infty) = 0.2$$

Thus each of the five intervals corresponds to a 20% probability.

We will assume in this example that the second random variable, X_2, has a uniform distribution on the interval from G to L. The corresponding intervals used in the Latin hypercube sample for X_2 are given in Figure 2-2 in terms of both the density function and the CDF.

The next step in obtaining the Latin hypercube sample is to pick specific values of X_1 and X_2 in each of their five respective intervals. This selection must be done in a random manner with respect to the density in each interval; that is, the selection must reflect the height of the density across the interval. For example, in the $(-\infty, A)$ interval for X_1, values close to A will have a higher probability of selection than those values in the tail of the distribution that extends to $-\infty$. Next, the selected values of X_1 and X_2 are randomly paired to form the five required two-dimensional input vectors. In the original concept of Latin hypercube sampling as outlined in McKay, Conover, and Beckman (1979), the pairing was done by associating a random permutation of the first n integers with each input variable. For illustration, in the present example consider two random permutations of the integers (1, 2, 3, 4, 5) as follows:

Permutation Set No. 1: (3, 1, 5, 2, 4)
Permutation Set No. 2: (2, 4, 1, 3, 5)

By using the respective position within these permutation sets as interval numbers for X_1 (Set 1) and X_2 (Set 2), the following pairing of intervals would be formed:

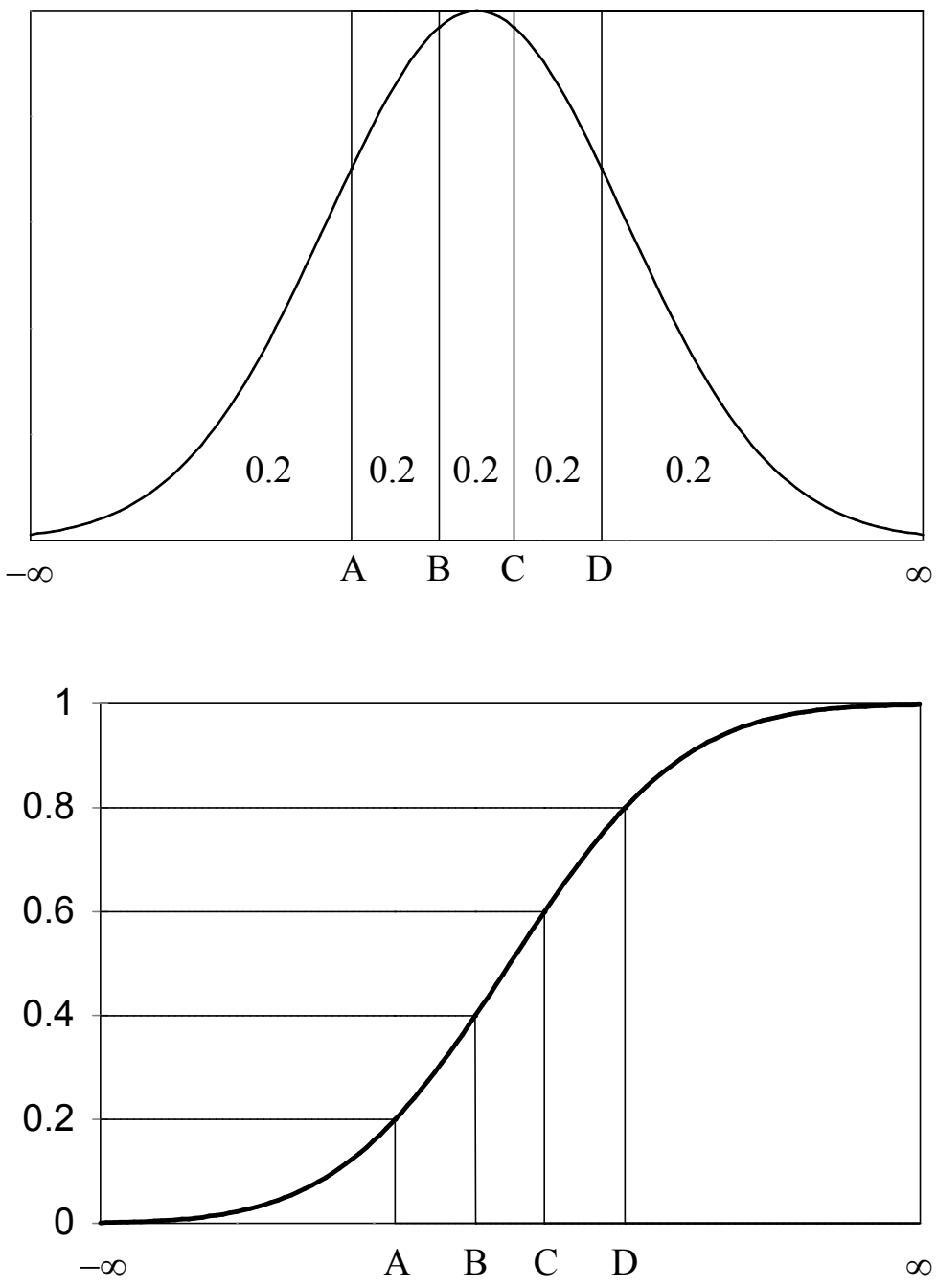

Figure 2-1: Intervals Used with a Latin Hypercube Sample of Size n = 5 in Terms of the Density Function and Cumulative Distribution Function for a Normal Random Variable

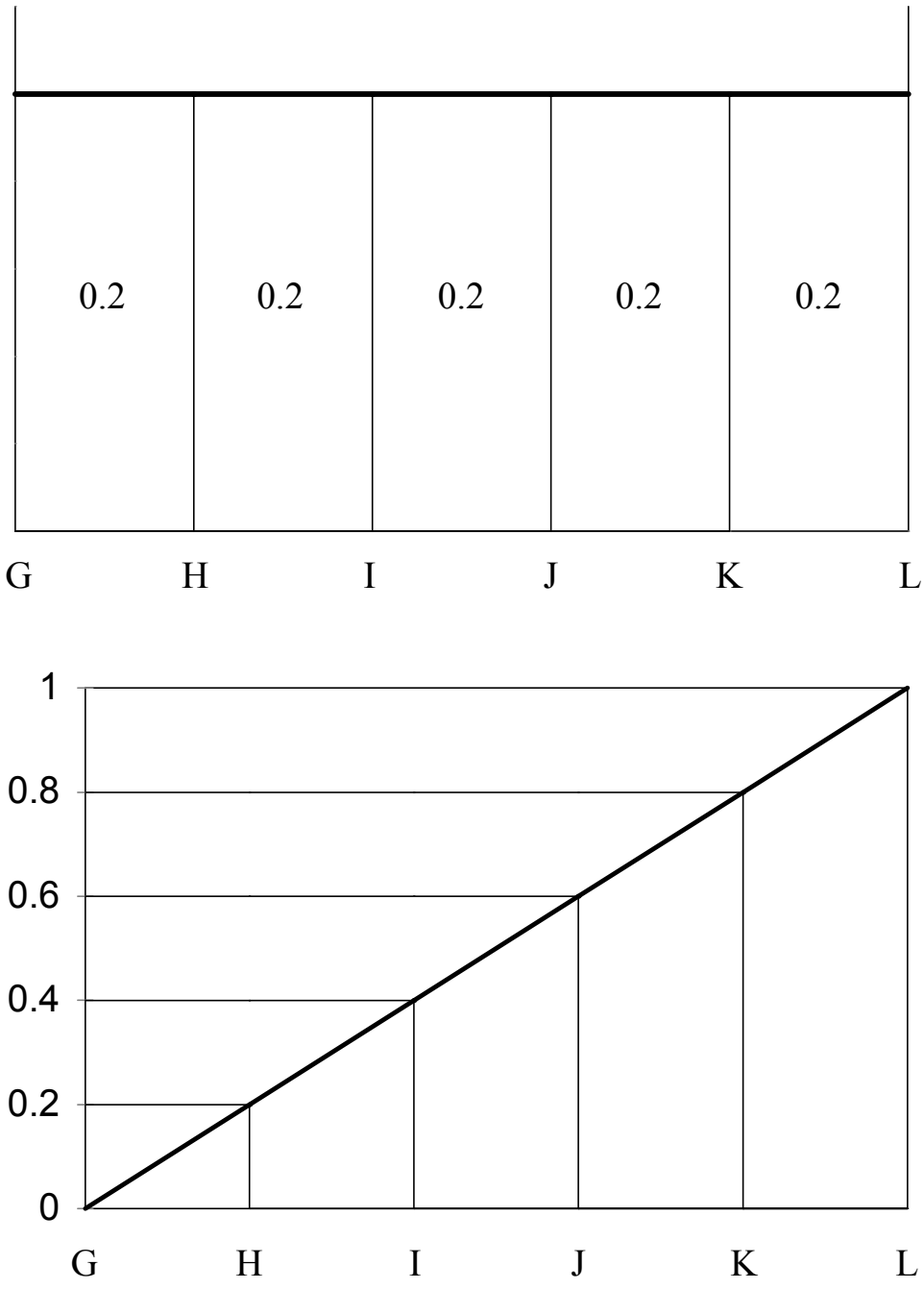

Figure 2-2: Intervals Used with a Latin Hypercube Sample of Size n = 5 in Terms of the Density Function and Cumulative Distribution Function for a Uniform Random Variable

Computer Run No.	Interval No. Used for X_1	Interval No. Used for X_2
1	3	2
2	1	4
3	5	1
4	2	3
5	4	5

Thus, on computer run number 1, the input vector is formed by selecting the specific value of X_1 from the interval number 3 (B to C) and pairing this value with the specific value of X_2 selected from interval number 2 (H to I). The vectors for the second and subsequent runs are constructed in a similar manner.

Once the specific values of each variable are obtained to form the five input vectors, a two-dimensional representation of the Latin hypercube sample such as that given in Figure 2-3 can be made. Note in Figure 2-3 that all of the intervals for X_1 have been sampled, and the same is true of X_2. In general, a set of n Latin hypercube sample points in k-dimensional Euclidean space contains one point in each of the intervals for each of the k variables.

To illustrate how the specific values of a variable are obtained in a Latin hypercube sample, consider the following example. Suppose it is desired to obtain a Latin hypercube sample of size n = 5 from a normal distribution with a mean of 5.0 and a variance of 2.618 as indicated in Figure 2-4. The density characteristics of the normal distribution allow for the definition of the equal probability intervals. These intervals are shown in Figure 2-4 in terms of a density function. The next step is to randomly select an observation within each of the intervals. This selection is not done uniformly within the intervals shown in Figure 2-4, but rather it is done relative to the probability density function distribution being sampled (in this case, the normal distribution). This is equivalent to uniformly sampling from the *quantiles* of the distribution (equivalent to sampling the vertical axis of the CDF) and then "inverting" the CDF to obtain the actual distribution values that those quantiles represent. This process is illustrated in Figure 2-4.

Therefore to get the specific values, n = 5 numbers are randomly selected from the standard uniform distribution (uniformly distributed between 0 and 1). Let these be denoted as U_m, where m = 1, 2, 3, 4, 5. These values will be used to select distribution values randomly from within each of the n = 5 intervals. To accomplish this, each of the random numbers U_m is scaled to obtain a corresponding cumulative probability, P_m, so that each P_m lies within the m^{th} interval. Thus, for this example with n = 5,

$$P_m = \left(\frac{1}{5}\right)U_m + \left(\frac{m-1}{5}\right)$$

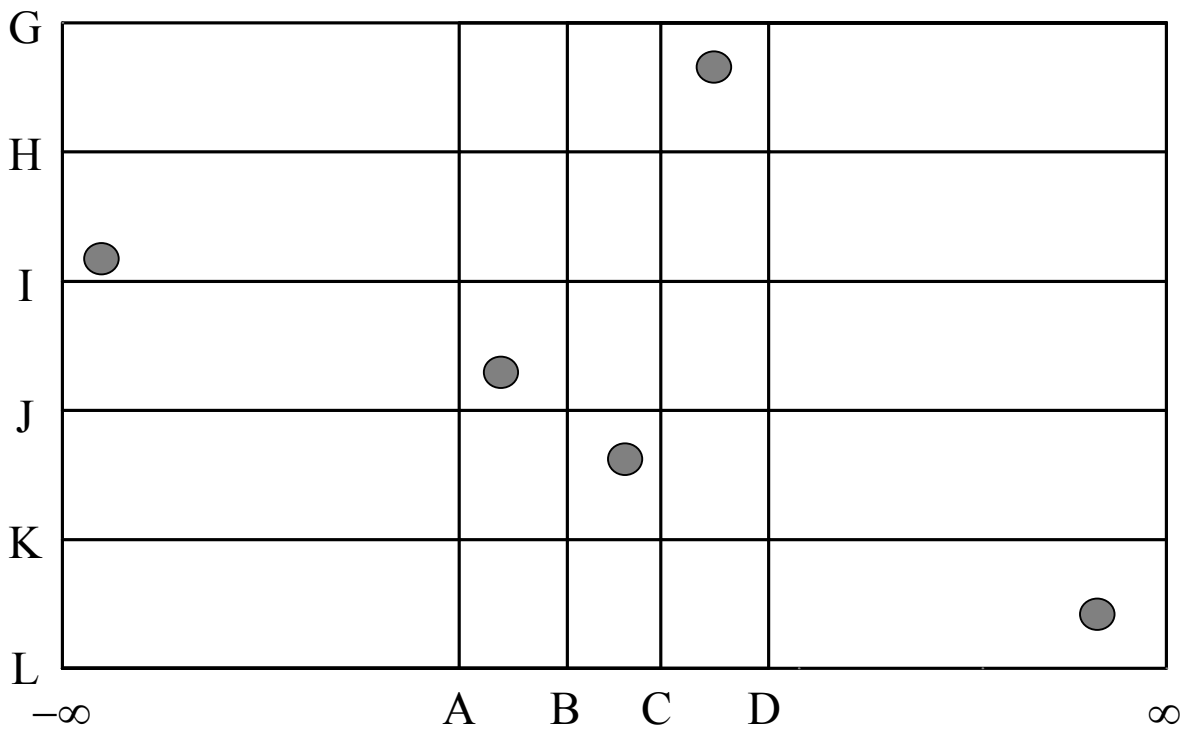

Figure 2-3: A Two-Dimensional Representation of One Possible Latin Hypercube Sample of Size 5 Utilizing X_1 and X_2

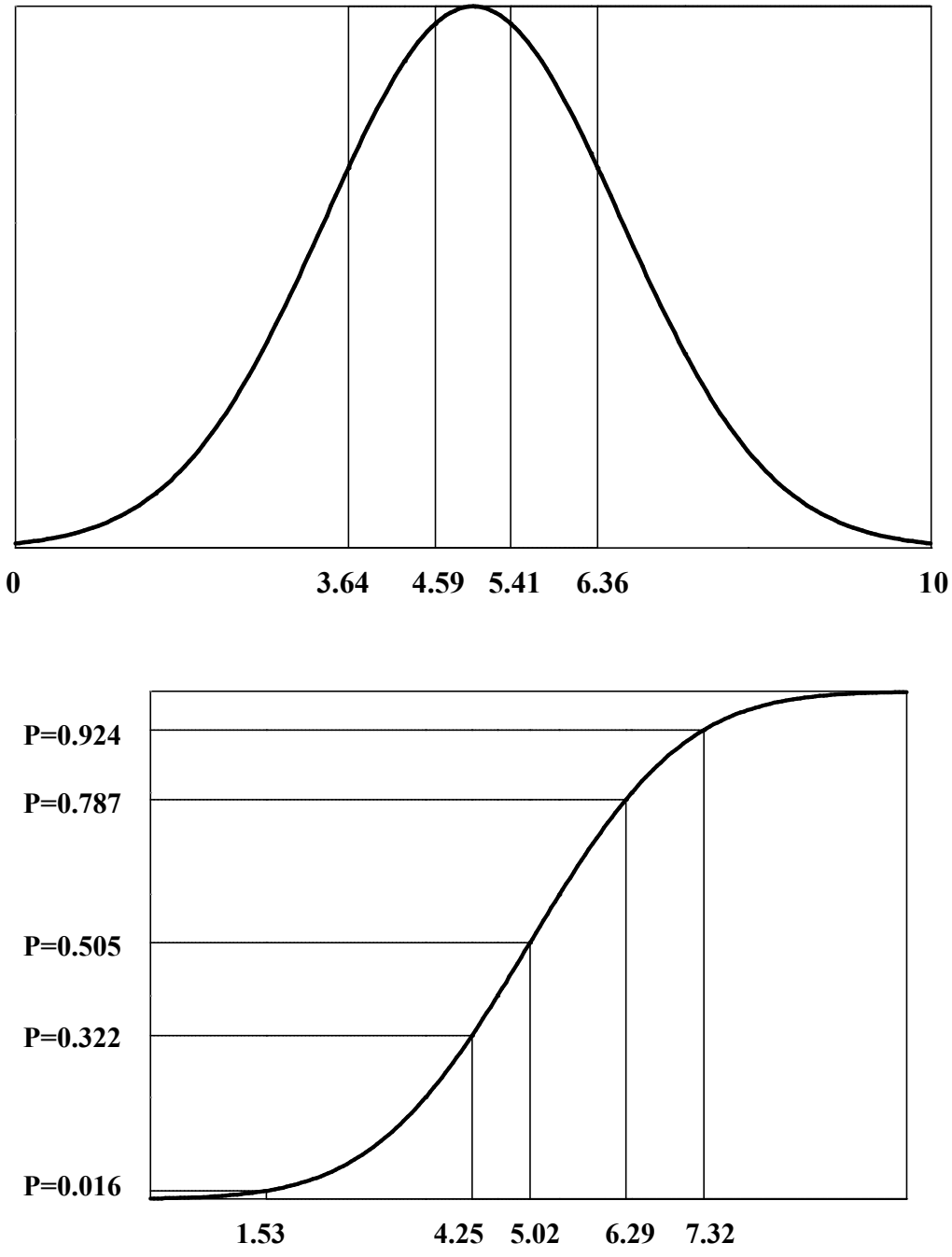

Figure 2-4: Interval Endpoints Used with a Latin hypercube sample of Size 5 (top) and Specific Values of X Selected Through the Inverse of the Distribution Function (bottom)

This ensures that exactly one probability, P_m, will fall within each of the five intervals (0, 0.2), (0.2, 0.4), (0.4, 0.6), (0.6, 0.8) and (0.8, 1). The values P_m are used with the inverse normal distribution function to produce the specific values to be used in the final Latin hypercube sample. Note that exactly one observation is taken from each interval shown in Figure 2-4. The entire process is shown in Table 2-1. Figure 2-4 makes it clear that when obtaining a Latin hypercube sample, it is easier to work with the CDF for each variable. This is the approach used in the computer program, rather than defining the endpoints of the intervals on the x-axis.

Figure 2-4 shows how one input variable having a normal distribution is sampled with Latin hypercube sampling. This procedure is repeated for each input variable, each time working with the corresponding cumulative distribution function. If a random sample is desired, then it is not necessary to divide the vertical axis into *n* intervals of equal width. Rather, *n* random numbers between 0 and 1 are obtained and each is directly (i.e., without scaling) mapped through the inverse distribution function to obtain the specific values. The final step in the sampling process involves pairing the selected values.

Table 2-1: One Possible Selection of Values for a Latin Hypercube Sample of Size 5 from a Normal Distribution with a Mean of 5 and a Standard Deviation of 2.618

Interval Number m	Uniform (0,1) Random No. U_m	Scaled Probabilities Within the Interval $P_m = U_m(0.2) + (m-1)(0.2)$	Corresponding Standard Normal Value (Z-score) from the Inverse Distribution	Corresponding N(5, 2.618) Observation Within the Intervals
1	0.080	0.016	-2.144	1.529
2	0.610	0.322	-0.462	4.252
3	0.525	0.505	0.013	5.021
4	0.935	0.787	0.796	6.288
5	0.620	0.924	1.433	7.319

2.2 Pairing

It should be noted that even though two variables are sampled independently and paired randomly, the sample correlation coefficient of the *n* pairs of variables in either a random sample or a Latin hypercube sample will, in general, not equal zero due to sampling fluctuations. In order to obtain a sample in which the sample correlations more nearly match the assumed, or intended, correlations, Iman and Conover (1982a) proposed a method for restricting the way in which the variables are paired. The effect of this restriction on the statistical properties of the estimated distribution of Y, its mean and percentiles, is believed to be small. The LHS software supports both the random pairing of variables described in the previous section and the restricted pairing of variables that is explained in this section.

While a full description of the restricted pairing methodology is beyond the scope of this report, a heuristic description is appropriate in order to give the prospective LHS software user a greater level of comfort with the software. Recall that the basic method for LHS is to select a set of n observations (random samples) for each random variable, and then to randomly "pair" those selected observations. This yields a set of n vectors, where each vector contains one value for each random variable. Thus, if there are k random variables, each of the n vectors is k-dimensional.

In the process of sampling a single distribution (generating n observations of values for that distribution), there is no significance to the order in which those observations are obtained. What matters for the purposes of the uncertainty analysis is that the entire *collection* of observations faithfully preserve the properties of the distribution from which it was sampled.

We can now view the pairing process as follows. Imagine writing the value of each observation for distribution X_1 on a separate slip of paper and placing those slips into a hat. There would now be n separate slips of paper in that hat. Now imagine following the same process for distribution X_2, and placing the slips in a second hat. The same process could be followed for distributions X_3 through X_k, so that at the end of the process one would have k hats, each containing n slips of paper with distribution values written on them.

To accomplish the random pairing process described previously, one would simply draw one slip of paper at random from each of the k hats, and the group of values written on those k slips of paper would form the first observation of the output data set. In the language of the previous section, these values would form the k-dimensional input vector for the first computer run, or the first row in the $(n \times k)$ matrix of sampling results. The second set of values drawn randomly from the k hats would form the second observation, k-dimensional input vector, or matrix row, and so forth until the last, or n^{th}, set of values was drawn to form the last such observation. Since exactly n values were generated for each distribution, all of the hats are now empty. Thus, all of the generated distribution values were used, so each column in the $(n \times k)$ matrix faithfully represents the distribution from which its values were drawn.

One could, however, visualize a different pairing process in which the slips of paper from the k hats were poured out to make k separate piles. The slips of paper from those piles could then be arranged in columns to form the $(n \times k)$ matrix described previously. Since there is no significance to the *order* in which the individual slips of paper are arranged in these columns, one might imagine a person examining the entire $(n \times k)$ matrix and *deliberately* moving slips of paper around within columns to achieve some goal for the overall matrix. If one were clever, one could order each column so that the correlation between its values and those of every other column in the matrix is as small as possible. When this process is completed, the $(n \times k)$ matrix still contains the same values in the same columns, but each column has been intentionally reordered. Each column in the matrix faithfully represents the distribution from which its values were drawn because, once again, all of the values generated for the distribution were used. Also, each row still represents the k-dimensional input vector for a computer run because it contains one value from each of the distributions X_1 through X_k. Thus the $(n \times k)$ matrix is in every sense equivalent to that generated by the random pairing process described earlier. In fact, there is some small probability that the random pairing algorithm would generate this matrix if exactly the right sequence of random draws

were to occur during the process. This intentional selection of the pairing of the selected observations is directly analogous to the restricted pairing procedure developed by Iman and Conover and implemented within the LHS software.

It is logical to ask then whether this restricted pairing technique can be used to induce specified *nonzero* correlations between random variables since it is already being used to cause these same correlations to tend toward zero (or, more properly, become as small as possible). The answer is yes, with the limitation that such induced correlations are based on the nonparametric technique known as rank correlation. Such a measure is used since it remains meaningful in the presence of non-normal distributions on the input variables. The LHS software provides the user with input parameters by which these pairwise correlations can be specified.

While correlation is defined and measured in terms of pairs of random variables, it is possible to consider multivariate correlations. One might, for example, have a group of five random variables that are all strongly correlated with each other. This situation can be specified to the LHS software by defining correlations between every possible combination (pair) of random variables within that five-variable group. For a group of five random variables, this would require 10 correlation specification statements (all the unique combinations of five random variables taken two at a time).

There are mathematical limitations to the ways that random variables can be correlated with one another. For example, the following three-way correlation is statistically impossible:

Correlation (A, B) = –0.95
Correlation (A, C) = –0.95
Correlation (B, C) = –0.95

The first two statements imply that large values of A tend to be paired with small values of both B and C, while the last statement implies that small values of B tend to be paired with large values of C. This condition cannot be realized by any real sampling scheme. An impossible correlation could also be inadvertently specified by a user if he or she were to attempt to define the five-variable correlation described above, but one pairwise combination of variables was omitted from the program input. This would cause the LHS software to attempt to correlate all five variables with each other and yet maintain one pair of variables within that group as statistically independent. This is clearly a statistical impossibility.

When the LHS software encounters such a condition, it generates a warning for the user and makes the smallest adjustments possible to the requested correlations so that it can generate its results. However, such conditions generally indicate that the user has either made a mistake in specifying input to the software or does not thoroughly understand the correlation conditions that he or she is attempting to model since contradictory correlation information has been entered. When the software detects these conditions, it is important to step back and reevaluate the software input to ensure that these contradictions are eliminated.

As a final note, if a correlation structure is <u>not</u> specified by the user, then the program computes a measure for detecting large pairwise correlations. This measure is known as the variance inflation

factor (VIF) and is defined as the largest element on the diagonal of the inverse of the correlation matrix. As the VIF gets larger than 1, there may be some undesirably large pairwise correlations present. Marquardt and Snee (1975) deal with some very large VIFs ($> 2 \times 10^6$) and provide a readable explanation on reasonable sizes of VIFs. Marquardt (1970) indicates that there can be serious colinearity (i.e., large pairwise correlations present) for VIF > 10. Thus, there is certainly no problem as long as the VIF is close to 1. The VIF appears as part of the computer printout when the user requests the correlation matrix to be printed without specifying a correlation structure.

3. The LHS Standalone Mode

This chapter explains how the LHS Standalone mode runs, and how the input file must be specified. LHS is run by entering the following at the command line:

lhsdrv *filename* <enter>

where *filename* is the name of the keyword file. If the parameter *filename* is omitted, the program will prompt the user for the name of the keyword file. The keyword file contains all of the necessary information to specify an LHS run (such as random seed, number of sample observations, distribution names and associated parameters, etc.)

3.1 Keyword Input Description and File Structure

The LHS program requires certain parameters to be defined in order to perform its sampling calculations. The program recognizes 55 keywords that dictate the characteristics of the generated sample such as size and type of sample, number of samples desired, correlation structure for input variables, and type of distribution specified on each variable. Other keywords are used to control the program output.

The program input may be placed in a single file, or the global program commands and distribution-specific commands may be placed in separate files. However, all input files require the same basic input format. The format of the distribution-specific input commands will be described after the general commands.

3.1.1 Input File Characteristics

All files read by LHS are based on a common file format that provides the user with a great deal of latitude in generating easily read, self-documenting input. Each LHS input file is a space-delimited free-format ASCII text file. As the file is read, its contents are treated as case-insensitive. The format supports continuation lines as well as whole-line and trailing comments.

3.1.2 Keyword File Characteristics

LHS assumes that each input line will contain no more than 80 ASCII characters. If longer input lines are used, the code will ignore all input that is not contained within the first 80 columns. Each line is assumed to contain either a whole-line comment or single command (possibly followed by a trailing comment). Each command in the keyword file must be fully contained on a single line. Note that this is different from the distribution input file in that continuation lines are not permitted in the section of the file that is used to specify command keywords to control LHS execution.

The LHS input file format provides for both full-line and trailing comments. Under this format, all blank lines and all text following "$" characters (dollar sign) are considered comments. A full-line comment is any line that contains only blank spaces or contains a dollar sign as its first nonblank

character. A trailing comment consists of any text that follows a dollar sign on any line in the file. In a trailing comment, the dollar sign must be preceded by one or more blank spaces. Since LHS ignores all text that follows a dollar sign, it is not possible to place comments between items on a single line.

A command may begin and end at any point on the line (an arbitrary number of leading and trailing spaces may be used as desired). If a command keyword is composed of more than one word, those words must be separated by *one and only one blank space*. However, the user may include as many blank spaces as desired between a keyword and any required alphabetic or numeric values. For example,

```
LHSOPTS      Random Sampling
```

is a legal input line because the keywords LHSOPTS and Random Sampling may be preceded and followed by an arbitrary number of blank spaces, but only a single blank space is permitted within the keyword Random Sampling. It should be noted that LHS treats commas (",") and tab characters as being fully equivalent to and interchangeable with blank space characters.

The alphanumeric input to LHS has been designed to allow maximum user flexibility. All character input is case-insensitive, so a user can provide any combination of upper- and lower-case characters for LHS input. Thus, the keywords Normal, NORMAL, normal, and noRMaL are all considered equivalent by LHS.

All numeric input is read in Fortran list-directed (free-format) input. Thus a user may specify numeric input in any Fortran-standard format, including integer (146), floating point (15.643), and exponential notation (1.426E-3).

The LHS keyword file contains the keywords that control program execution. These keywords specify, for example, the run title, the number of observations to be generated, the types of point estimate calculations to be performed, the names of the files to be used, and the types of reports to be generated. *Continuation lines cannot be used in the keyword section of this file*. The keywords that can be used in this file are described in the following section.

3.1.3 Input File Structure

LHS was designed to allow two different forms of input file structure: single-file and double-file. Single-file input simply places the distribution information and the command keyword information in a single file. It is possible to freely intersperse command keywords and distribution definitions within this single input file as long as all command keywords occur prior to the Dataset: keyword. However, the preferred method of file development bunches the command keywords near the top of the input file (with appropriate comments), followed by the Dataset: keyword and the complete set of distribution definitions. The single-file input structure is assumed whenever the command keyword PRETRIN is not used in the input file. Note that continuation lines are allowed in the distribution section of this input file.

The double-file input structure groups the command keywords in one file and the distribution definitions in a second file. This file structure was developed to aid the analyst in quality assurance of the distribution definition information because it allows the analyst to change the parameters of the LHS run (e.g., change the number of observations requested or the output file name) without modifying the file that contains the distribution definitions. The name of the second input file (the distribution definition file) is specified using the PRETRIN keyword. If the PRETRIN keyword is present, LHS will read distribution definitions *only* from this second file and will ignore any distribution definition information contained in the command keyword file.

3.2 Keywords to Control Program Execution

Table 3-1 lists the keywords that set up the execution of the LHS program. Recall that LHS is run in standalone mode with the following command: lhsdrv *keywordfilename*, where the keyword file lists the keywords that control execution. Also note that LHS is designed to be run as one element in a large suite of codes—each of which has its own set of keywords. Therefore, LHS ignores keywords that it does not recognize. More details about the keywords are in the sections following Table 3-1. An example keyword file is given in Section 3.2.5. Table 3-3 in Section 3.3 specifies the distribution names and parameters.

Table 3-1: Keywords in LHS Input file

KEYWORD	Specification	Required	Value
LHSSEED *iseed*	Random Number Seed	YES	Integer n $(1 \le n \le 2^{31} - 1)$
LHSOBS *num*	Number of observations per entire LHS sample	YES	Integer number of observations
LHSOUT *filename*	Name of LHS sampled data output file	YES	Valid filename
LHSMSG *filename*	Name of output file that contains the LHS user output and messages from LHS execution	YES	Valid filename
LHSREPS *nreps*	Number of multiple LHS sample sets generated	NO	Default: *nreps* = 1
LHSPVAL *ipval*	Point value specification *ipval* = 0 takes the point values specified in the input file; *ipval* = 1 takes the mean value of the input distribution as a point value; *ipval* = 2 takes the median as a point value	NO	Default: *ipval* = 1

LHSOPTS *options*	How the sampling is done	NO	RANDOM SAMPLE is pure Monte Carlo; RANDOM PAIRING refers to random pairing for correlation. Omitting the LHSOPTS keyword results in restricted pairing.
LHSTITL *title*	Title of LHS Run	NO	User-defined name
LHSRPTS *options*	Reporting Options	NO	Default = None. HIST = prints histogram of each sampled random variable, CORR = prints achieved correlation matrices, DATA = prints raw and rank sample values. Any combination of HIST, CORR, and DATA may be used.
LHSSCOL	Forces output to be one value per line	NO	Default: False
PRETRIN *filename*	Allows the LHS program to open the specified secondary file that contains distribution information	NO	Default: None
CONSTANT *value*	Allows one to write a constant to the output file	NO	
SAME AS *old_variable*	Allows one to specify aliases for distributions (one random variable is the same as another)	NO	*old_variable* must be an explicitly defined distribution in the input file
CORRELATE *first_variable second_variable corr*	Allows the user to specify a correlation coefficient, *corr*, between two variables	NO	$-1 < corr < 1$
DATA:	Precedes a distribution name, type, and parameter values	YES	If DATASET: is used (see next row), then DATA: is not required on each line.
DATASET:	DATASET can be used in front of a whole set of distributions	YES	Note: either DATA: must be specified on each distribution line or DATASET: can be used in front of a group of distributions

DISTRIBUTION NAME *parameter values*	See Section 3.3.3	NO	Required parameters are specific to each distribution. For example: DATA: MyVar1 NORMAL 0 1

3.2.1 Sampling and Processing Control Keywords

The keywords described in this section are used to control the processing that occurs in LHS. They are used to determine the starting point of the random number sequence, the types of sampling and pairing to be performed, the number of observations to be generated, and the method to be used for computing the point estimate values for the sampled distributions.

LHSSEED ***iseed***
(Required)

The positive integer *iseed* specified with this keyword provides the "seed value" or starting point for the LHS random number generator. The value specified by this keyword must be within the range of representable positive integers ($1 \leq n \leq 2^{31} - 1$). There is anecdotal evidence to suggest that the randomness of the random number generator may be somewhat compromised for very small seed values, so it is recommended that random seed values less than approximately 1000 be avoided. The seed value is printed in the LHS message file at the beginning of each sample. If the keyword LHSREPS (described later in this section) specifies a number greater than 1, the current value of the random seed is retrieved and printed at the start of the generation of each new sample. This allows the user to regenerate by rerunning the program with the new seed and with the LHSREPS parameter omitted (or having LHSREPS 1).

Example: `LHSSEED 15964`

A run with this example would start the random number generator based on an integer seed value of 15964.

LHSOBS ***num***
(Required)

This keyword must be followed by a positive integer that specifies the number of observations that LHS will generate to be included in the sample for each distribution. The maximum number of observations is defined by a parameter in the SIPRA.INI initialization file.

Example: `LHSOBS 500`

An LHS run containing this keyword would produce 500 observations for each variable.

LHSREPS ***nreps***

(Optional, default: *nreps* = 1)

This keyword can be used to generate multiple samples (replications). When present, it must be followed by a positive integer to specify the number of complete sample sets of the data (each of the size LHSOBS). The samples are then written back to back in the output file. If LHSREPS is omitted, a single sample is generated.

Example: `LHSREPS 3`

This example would produce three repetitions of the specified distribution samples, each with the size LHSOBS.

LHSPVAL *ipval*
(Optional, default: *ipval* = 1)

This keyword allows the user to select which method LHS will use to calculate the representative point value found in the first block of the LHS sampled data output file. If this keyword is omitted, the point values printed will represent the mean. LHSPVAL, followed by an integer number *ipval*, specifies whether the point value in the LHS output file is to represent the mean (*ipval = 1*) or median (*ipval = 2*) of the distribution. If a zero is specified (*ipval = 0*), then the point value specified for the distribution in the input file is copied directly to the sample output file. Note that the "optional" distribution point values become required input if zero is specified for this keyword.

Example: `LHSPVAL 0`

This example would cause LHS to enter the user-specified point value for each distribution in the point estimates block of the LHS output file. Note that if the analyst needs to import an LHS *input* file for use in TEMAC3, then LHSPVAL *must* be set to 0. TEMAC3 can, however, use LHS output files that were created with LHSPVAL set to values other than 0.

LHSOPTS *options*
(Optional, default: no options selected)

This keyword allows the user to control the sampling and pairing methods that LHS will use in preparing its sampled data output file. If the keyword is omitted, LHS will use Latin hypercube sampling with restricted pairing. If the keyword is present, it must be followed by one or more of the following keywords.

RANDOM SAMPLE This option causes LHS to suppress Latin hypercube sampling and use purely random (Monte Carlo) sampling techniques in its place. If this option is omitted, LHS uses Latin hypercube sampling.

RANDOM PAIRING This option causes LHS to suppress the restricted pairing method and generate a sample with purely random pairing. If this option is omitted, LHS uses restricted pairing.

```
Example:  LHSOPTS   RANDOM PAIRING
Example:  LHSOPTS   RANDOM SAMPLE
Example:  LHSOPTS   RANDOM SAMPLE   RANDOM PAIRING
```

The first example would cause LHS to produce its sample using Latin hypercube sampling techniques, and to use random pairing of variables. The second example would cause LHS to use random sampling with restricted pairing. The third example would cause LHS to use both random sampling and random pairing.

3.2.2 Output and Reporting Keywords

The keywords described in this section are used to define how LHS generates its output. Using these keywords, the user can control the title to be placed in the output file, the types of reports to be generated, and the format of the sampled data output file.

LHSTITL ***title***
(Optional, default: *title* = blank)

This keyword can be used to specify the title of each LHS run. If specified, it must be followed with alphanumeric data (up to 70 characters long) to help describe the application of the sample. This information will be printed as a one-line header on each page of the output. If it is omitted, a blank header is generated at the top of each page.

```
Example:  LHSTITL  This is a test EVNTRE/LHS run
```

An LHS run using this example LHSTITL record would have "This is a test EVNTRE/LHS run" written as a header on each page of the message output file and as a comment in the sampled data output file (if the default format is used).

LHSRPTS ***options***
(Optional, default: *options* = none)

This keyword is used to specify which reports LHS will print in the message output file. If LHSRPTS is omitted, the message file will contain only the title, run specifications, and descriptions of the distributions sampled. If LHSRPTS is included, it must be followed by one or more of the following three additional keywords. These additional keywords can appear in any order (separated by blanks). They function as follows:

CORR Print both the achieved raw correlation matrix and the achieved rank correlation matrix associated with the actual sample.

HIST Print a text-based histogram for each random variable.

DATA Print the complete set of all data samples and their ranks in the message output file. Use of this option makes the individual sample input vectors available in the message file. It should be used with caution, however, because it can cause the generation of an extremely large message output file.

Example 1: `LHSRPTS CORR HIST`
Example 2: `LHSRPTS DATA`
Example 3: `LHSRPTS HIST DATA CORR`

If Example 1 were used, the message output file would contain both raw and rank correlation matrices for the sample and a histogram for each variable. If Example 2 were specified, the message file would contain all of the sample data and rank data for every observation of every variable sampled. If Example 3 were specified, all three types of reports would be written to the message output file.

LHSSCOL
(Optional, default: False)

This keyword specifies that LHS is to write its sampled data output file in a single column – that is, one value per line. This has been found helpful by certain postprocessing programs that have difficulty interpreting data files that contain more than one value per line.

Example: `LHSSCOL`

This example causes LHS to generate its sampled data file in a single-column format with one value per line.

3.2.3 File Specification Keywords

LHS and its associated codes make use of several files. The keywords described in this section are used to specify the names of those files. Currently, file names should be specified in the MS DOS 8.3 file format (including, at the user's option, the full file path name, but without long file names or spaces), although enhancements to the software that will remove these restrictions are planned for the near future.

LHSOUT *filename*
(Required)

This keyword, followed by a file name, allows the user to specify the name of the LHS sampled data output file. This output file is designed to be read by other programs and is not formatted for easy examination by the analyst. The file designed for the analyst to read is the message file, which is specified by the next keyword description.

Example: `LHSOUT TestRun.LSP`

Example: `LHSOUT C:\ARRAMIS\NewProb\TestRun.LSP`

An LHS run with this example would write the sampled output to a file named "TestRun.LSP."

LHSMSG *filename*
(Required)

This keyword, followed by a file name, allows the user to specify the name of the output file that contains the LHS user output and reports ("messages") from the LHS execution.

Example: `LHSMSG TestRun.LMO`
Example: `LHSMSG C:\ARRAMIS\NewProb\TestRun.LMO`

An LHS run with this example would write the message and user reports to a file named "TestRun.LMO."

PRETRIN *filename*
(Optional, default: none)

The keyword PRETRIN, followed by a file name, allows the LHS program to open the specified secondary file that contains distribution description information. If this keyword is omitted, LHS will read the distribution description information from the keyword file (see Section 4.1.2).

Example: `PRETRIN TestRun.LDI`
Example: `PRETRIN C:\ARRAMIS\NewProb\TestRun.LDI`

This example would cause LHS to open the file TestRun.LDI to read the distribution and correlation specifications.

When the PRETRIN keyword is used, LHS assumes that all distribution definitions and correlation information will be read from the file designated by this keyword. Any random variables defined in other input files will be ignored. Within that file, each random variable must be defined either on a line that begins with the keyword Data: or within a block that begins with the keyword Dataset: (the definition of correlation between variables must follow these same rules – see Chapter 4).

3.2.4 Keywords Not Related to Distributions

Constants

As originally structured, LHS produced a sampled output file that contained only those random variables that were sampled by the LHS program. Thus, if a user wished to use some point estimate values in a model while sampling other model parameters, he or she would be required to place the point estimate data in a separate file and provide a method for both the point estimate data and the

sampled data to be read by the model program. Alternatively, the user could build a translation program that reads the LHS output file and the point estimate data file, merges them into a single entity, and writes the data in a format familiar to the model program. Both of these options are unnecessarily cumbersome. For this reason, the CONSTANT distribution type was added to LHS. Recall that LHS always automatically generates a point estimate value for each distribution sampled. Depending on the sampling options selected, this point estimate value will be the mean of the samples generated, the median of the samples, or the user-specified value that precedes the distribution description. This point estimate value is placed in the LHS output file in order to enable programs that receive LHS results to determine at run time whether to use the sampled results or the point estimate value on a random variable-by-random variable basis. The CONSTANT distribution type simply enables the user to make additional entries in the point estimates (constant values) section of the LHS output file without placing additional values in the sampled data section or requiring LHS to perform additional sampling or pairing operations. It enables the user to maintain all program input data in a single file and minimize data file quality assurance requirements—whether the data are sampled or point value in nature.

CONSTANT ***val***

The keyword CONSTANT places the distribution name and point estimate value *val* in the constant block section of the LHS output file.

First Example: `NewOne 150.0 CONSTANT 152.5`
Second Example: `Pi 3.14159 CONSTANT 3.14159`

The first example places the variable named "NewOne" into the constant block section of the LHS output file. If the user has requested that LHS compute the point estimate values for sampled distributions as either the mean or the median of the sampled data, the value associated with NewOne in the LHS output file will be 152.5. Otherwise, if the user has requested that LHS use the specified point estimate value, the value associated with NewOne in the LHS output file will be 150.0. While this example demonstrates the flexibility that can be used in specifying LHS input, it does not represent a typical application of the CONSTANT keyword. One usually specifies the same value for both the user-specified point estimate and the CONSTANT distribution's *val* parameter, as shown in the second example.

Distribution Aliases

It is not unusual in certain risk assessment applications to specify that two variables are "totally correlated." This might occur, for example, when the same valve model from the same manufacturer has been used in two different locations in a plant system. There is no reason to believe that these two seemingly identical valves will have different failure probabilities, but there is uncertainty as to exactly what that failure probability should be. Thus, the probability of valve failure is sampled from one distribution, but given two names—one for each physical valve—so that the risk model can readily identify failure probabilities for both components.

The keyword SAME AS is used to implement this condition. This keyword is used by first defining a random variable that defines the distribution to be sampled for one of the totally correlated variable names. The second totally correlated variable is then linked to the first using the SAME AS keyword so that its name becomes a valid alias for the same distribution samples. The LHS sampled data output file will contain only one set of values, and the file will contain a notation that indicates that this single set of values is to be used for both totally correlated variables.

SAME AS ***old_variable***

The keyword SAME AS defines a new name (alias) that is to be associated with an existing distribution. The *old_variable* parameter specifies the name of the existing random variable that is to be referred to by an additional new name. Note that *old_variable* must be the name of a random variable that is explicitly defined with a distribution description. The value of *old_variable* cannot point to the name of another variable that is defined using the SAME AS keyword.

Example:
```
First   0.0  NORMAL  0.0  1.0
Second  SAME AS  First
Third   SAME AS  First    $  This input is valid
$ Fourth SAME AS  Second  $  This is not valid
```

In this example, the random variable named "First" is explicitly defined as a normal distribution, and the random variables named "Second" and "Third" are set up to be totally correlated to First. In other words, Second and Third are simply different names (aliases) for the observations generated when the distribution First is sampled. The input for the variable named "Fourth" would be invalid because its *old_variable* parameter points to the variable Second, which was itself defined using the SAME AS keyword, instead of correctly pointing to the variable First as the object of its alias.

Controlling Correlation

Chapter 2 described how the LHS software, through the use of restricted pairing techniques, can intentionally pair samples from random variables in such a manner as to control correlation between those variables. By default, LHS assumes that the user desires the pairwise correlation between all pairs of random variables to approach zero (i.e., all random variables should be independent of one another to the degree possible). If the user wishes to specify other pairwise correlations between random variables, that is done using the CORRELATE keyword.

CORRELATE ***first_variable second_variable corr***

This keyword causes LHS to attempt to create a correlation of *corr* between the ranks of the sampled values for *first_variable* and the ranks of the sampled values for *second_variable*. The parameters *first_variable* and *second_variable* are the names of random variables that are explicitly defined with a distribution description and *not* using the SAME AS keyword. The specified

correlation value parameter *corr* must satisfy the condition (*–1 < corr < 1*). This statement will be evaluated by LHS to ensure that the data item names are legitimate prior to LHS processing.

Example:
```
X  Normal 0.0  1.0
Y  Uniform  1.7  3.5
Z  SAME AS  Y
CORRELATE  X  Y  0.5
$ CORRELATE  Y  X  0.5   $ Identical to above
$ CORRELATE  X  Z  0.7   $ This is not valid
```

In this example, the first Correlate statement will cause LHS to attempt to establish a rank correlation of 0.5 between the random variables *X* and *Y*. Note that the order in which the random variable names appear in the Correlate statement does not matter, so the second Correlate statement (seen in comments in this example) would, if used, produce results identical to the first Correlate statement.

The third Correlate statement (also seen in comments in the example) would not be valid because it involves the random variable *Z*, which is defined using a SAME AS statement. It could be corrected by replacing the variable name *Z* with the variable name Y so that it no longer points to a distribution that is defined using the SAME AS keyword.

There are two important points to remember about the use of correlation within LHS. First, LHS only considers CORRELATE statements when it operates in the restricted pairing mode. This is because the random pairing process is just that – random – and cannot control correlation between random variables. There are two ways that LHS can be forced into a random pairing mode: the user can (1) select Random Pairing in the user interface (see Chapter 3); or (2) set up a problem in which LHS is told to generate fewer observations than the number of random variables that are being sampled. In this second condition, LHS cannot accomplish restricted pairing because the mathematical algorithm that performs restricted pairing fails for these conditions. When the user attempts to run such a problem (more random variables than observations) with restricted pairing, LHS generates a warning message in its message output file to tell the user that restricted pairing could not be used, and a modified form of random pairing is used instead (see Section 3.3.5). This warning message, however, can be easily missed. Therefore it is important to bear this situation in mind when running problems with either a large number of random variables or a small number of observations. For best results, the user should ensure that the number of observations requested is at least $^4/_3$ the number of random variables being sampled (not including CONSTANT and SAME AS keywords).

The second point to remember about the CORRELATE keyword is that it truly specifies *pairwise* correlations between random variables. Thus, if more than two random variables are to be correlated with one another, the user must enter a CORRELATE record for *every pairwise combination* of variables to be so correlated. Consider four random variables named A, B, C and D that are all to be correlated with one another, with all pairs having a pairwise correlation coefficient of 0.2. This is accomplished using the following sample input:

Example:

```
A  Normal  0.0  1.0
B  Uniform  0.5  2.5
C  Lognormal  0.1  3.0
D  Uniform  -100.0  100.0
$
Correlate  A  B  0.2
Correlate  A  C  0.2
Correlate  A  D  0.2
Correlate  B  C  0.2
Correlate  B  D  0.2
Correlate  C  D  0.2
```

The user must be extra careful when attempting to induce negative correlation between more than two random variables. There are mathematical limitations to the ways that random variables can be correlated with one another. For example, the following three-way correlation is statistically impossible:

Example:

```
Correlate  A  B  -0.95
Correlate  A  C  -0.95
Correlate  B  C  -0.95
```

The first two statements imply that large values of A tend to be paired with small values of both B and C, while the last statement implies that small values of B tend to be paired with large values of C. This condition cannot be realized by any real sampling scheme. When the LHS software encounters such a condition, it generates a warning and makes the smallest adjustments possible to the requested correlations so that it can generate its results. However, such conditions generally indicate that the user has either made a mistake in specifying input to the software or does not thoroughly understand the correlation conditions that he or she is attempting to model since contradictory correlation information has been entered. It may be that the user actually intended that large values of A should be paired with small values of both B and C, so that B and C should in fact have a strong positive pairwise correlation, such as:

Example:

```
Correlate  A  B  -0.95
Correlate  A  C  -0.95
Correlate  B  C   0.95
```

When the software detects physically unrealizable correlation conditions, it is important for the user to step back and reevaluate the software input to ensure that these contradictions are eliminated.

3.2.5 Example Keyword and Distribution File

An example keyword input file is shown below. The distribution portion of the file is discussed in the next section and further distribution information is given in Chapter 4. Note that in this example listing, both the keywords and the distributions are given in the same keyword input file.

```
LHSTITL Sample Run for LHS Manual
LHSOBS  100
LHSSEED  56595857
LHSPVAL  0
LHSRPTS   CORR
LHSOUT   TESTMAN.LSP
LHSPOST  TESTMAN.MSP
LHSMSG   TESTMAN.LMO
DATASET:
VIN-FILT-MAINT    0.003     LOGNORMAL     1.1     2.0
VIN-FILT-PLUG     0.003     LOGNORMAL     1.1     2.0
VIN-PIPE-1     0.003     NORMAL     0.003     0.001
VIN-PIPE-2     0.003     NORMAL     0.003     0.001
VIN-PIPE-3     0.003     NORMAL     0.003     0.001
VIN-PUMP-F     0.003     LOGNORMAL     1.1     2.0
VIN-PUMP-PWR   0.003     LOGNORMAL     1.1     2.0
VIN-RES-BROKE  0.003     LOGNORMAL     1.1     2.0
VIN-RES-EMPTY  0.003     LOGNORMAL     1.1     2.0
CORRELATE  VIN-PIPE-1  VIN-PIPE-2  0.5
CORRELATE  VIN-PIPE-1  VIN-PIPE-3  0.5
```

Figure 3-1: Example Keyword File

Below is an example of just the distribution portion of the input. In this example, the keyword file would be given separately. The distribution information is outlined in detail in Section 3.3.

```
Data:    MOV-1A-FTC  1.0E-3  Lognormal  1.0E-3   3
$  Here is a full line comment
$  This distribution is entered using several continuation
$  lines to aid the analyst in documenting the data
Data:  MOV-2B-FTRO   1.8E-4   % $ Point value from data
                  Lognormal  %
                    1.0E-4   % $ Mean, From ref. xxxx
                      3      % $ Error Factor
Data:  Recover-Time   Uniform  38  51
                     $ Bounds from ref. wxyz
Dataset:
Velocity  Bounded Normal  950  74  500  1500
Test-Results  Binomial  0.01 847
Fire Temperature  Continuous Linear   #
           7               # $ 7 pairs of data
           300   0.0       # $ minimum value
           500   0.1       # $ 10 percentile
           650  0.25       # $ 25 percentile
           800  0.5        # $ median
           1000 0.75       # $ 75 percentile
           1400 0.95       # $ 95 percentile
           1900 1.0        # $ distribution maximum
Correlate  MOV-1A-FTC  MOV-2B-FTRO  0.4
```

Figure 3-2: Example Distribution File

3.3 Distribution Input File Summary

This section outlines the distribution information that is input to LHS. The distribution information may be specified as part of the keyword file, or may be specified in a separate file.

3.3.1 Names of Random Variables

In LHS, all random variables must be assigned a name. Since results from LHS are often used as input by other programs, LHS provides a way for those programs to properly identify the individual components of the sampled data in the LHS output file. LHS does this by requiring each specified distribution to be identified by a unique name. Each name must meet the following criteria:

- It may contain no more than 16 characters
- The "$" (dollar sign), "#" (number sign), "%" (percent sign), "," (comma), " " (blank space), and tab characters are not valid within names
- A name must not form a valid representation of a real number for an ANSII-standard Fortran compiler

Examples of valid and invalid distribution names are found in the following table.

Table 3-2: Valid and Invalid Distribution Names

Valid Distribution Names	Invalid Distribution Names	Reason Name is Invalid
Valve-Fails	Amount of Hydrogen	Includes spaces
Spark	%Hydrogen	Includes continuation character
PulseWidth	Repair$	Includes comment character
Hydrogen-Percent	1234567	A number
Acceleration	Loss%	Includes continuation character
	Valve#1-Fail	Includes continuation character
	Failure,Valve1	Includes a comma
	TheValveFailsWideOpen	Too many characters

Any name that is longer than 16 characters will be truncated to 16 characters. These names appear in the LHS output file and are used to identify which distribution corresponds to which set of observations in that file.

3.3.2 Distribution Specification

LHS will take as its distribution input any line from the file on which the first keyword is Data:.

LHS also recognizes an option where its input is concentrated in a block at the bottom of a file. This block must begin with the keyword Dataset: on a line by itself. All lines that follow this statement are assumed to belong to LHS input regardless of whether they begin with the keyword Data:. Note that continuation and comment lines are permitted for both types of distribution input statements.

The keywords that can be used to describe distribution input are described later in this chapter. However, they all follow a common format:

name point_value dist-name dist-parm dist-parm .....

In this format, *name* is the variable name the user assigns to this distribution (described in the previous section). For each named distribution, LHS will generate not only sampled data, but also a point estimate value. This value can be used by software models that cannot accept or effectively use the sampled results provided by LHS. At the user's discretion, this point value can be either the sample mean, the sample median, or a user-specified value. If the analyst wishes to use the user-specified value option, then he or she must enter that value using the *point_value* parameter. This field is required only when the analyst specifies that LHS is to use the "user-specified value" for the point estimate instead of computing the sample mean or median for this purpose. In all other instances, the *point_value* parameter is optional and can be omitted.

The remaining parameters in this format are used to specify which distribution LHS is to sample for this random variable, and what that distribution's characteristics are to be. The *dist-name* parameter specifies which type of distribution is to be sampled (e.g., normal, uniform, hypergeometric), and the *dist-parm* values are the distribution parameters required by LHS to describe the particular distribution. Note that each distribution type requires its own unique set of parameters, and that the number of parameters required can vary from one to four (for standard distribution types) or more (for user-defined distribution types). The user is free to enter the required parameters in any format that he or she finds meaningful as long as it remains consistent with the format requirements described in this section. Note that this distribution definition statement must either be preceded by a Data: qualifier or be part of the Dataset: block at the end of the input file if the LHS software is to recognize it.

Table 3-3 outlines the distributions defined in LHS. There are also other user-defined distributions recognized by LHS. These are outlined in Sections 3.3.4 and 3.3.5.

3.3.3 Defined Distributions in LHS

Table 3-3: LHS Distribution Names

Distribution Name	Parameters			
NORMAL	*mean*	*standard-deviation* *> 0.0*		
TRUNCATED NORMAL	*mean*	*standard_deviation* *> 0.0*	*lower* *0.0 ≤ lower < upper ≤ 1.0*	*upper*
BOUNDED NORMAL	*mean*	*standard_deviation* *> 0.0*	*lower_bound* *lower_bound < upper_bound*	*upper_bound*
NORMAL-B	*value_at_0.001*	*value_at_0.999*		
LOGNORMAL	*mean* *> 0.0*	*error_factor* *> 1.0*		
LOGNORMAL-N	*mean* *> 0.0*	*standard_deviation* *> 0.0*		
TRUNCATED LOGNORMAL	*mean* *> 0.0*	*error_factor* *> 1.0*	*lower* *0.0 ≤ lower < upper ≤ 1.0*	*upper*
TRUNCATED LOGNORMAL-N	*mean* *> 0.0*	*standard_deviation* *> 0.0*	*lower* *0.0 ≤ lower < upper ≤ 1.0*	*upper*
BOUNDED LOGNORMAL	*mean* *> 0.0*	*error_factor* *> 1.0*	*lower_bound* *lower_bound < upper_bound*	*upper_bound*
BOUNDED LOGNORMAL-N	*mean* *> 0.0*	*standard_deviation* *> 0.0*	*lower_bound* *lower_bound < upper_bound*	*upper_bound*
LOGNORMAL-B	*value_at_0.001* *> 0.0*	*value_at_0.999* *> 0.0*		
UNIFORM	*A* *A < B*	*B*		
LOGUNIFORM	*A* *0.0 < A < B*	*B*		
EXPONENTIAL	λ > 0			
MAXIMUM ENTROPY	*A* $0 \le A < \mu < B$	μ	*B*	
WEIBULL	α > 0	β > 0		
PARETO	α > 2.0	β > 0.0		
GAMMA	α	β		
BETA	*A* $0 \le A < B$	*B*	*p* ≥ 0.001	*q* ≥ 0.001
INVERSE GAUSSIAN	μ *> 0*	λ *> 0*		
TRIANGULAR	*a* *a < c*	*b* *a ≤ b ≤ c*	*c*	
POISSON	λ > 0			

Distribution Name	Parameters		
BINOMIAL	p $0 < p < 1$	n > 1	
NEGATIVE BINOMIAL	p $0 < p < 1$	n > 1	
GEOMETRIC	p $0 < p < 1$		
HYPERGEOMETRIC	N_N $N_1 < N_R < N_N$	N_1	N_R

3.3.4 User-Defined Continuous Distributions

CONTINUOUS LINEAR ***n ordered_pair$_1$ ordered_pair$_2$... ordered_ pair$_n$***
and
CONTINUOUS LOGARITHMIC ***n ordered_pair$_1$ ordered_pair$_2$... ordered_pair$_n$***

- n is an integer number of ordered pairs that will be used to represent the CDF ($n > 1$).
- Within each ordered pair, the first number is the value of the distribution; the second number is the cumulative probability (CDF value) associated with this distribution value.
- The probabilities must increase monotonically starting with 0.0 and ending with 1.0.
- The distribution values (the first entry in the ordered pair) must also increase monotonically.

CONTINUOUS FREQUENCY ***n ordered_pair$_1$ ordered_pair$_2$... ordered_pair$_n$***

- n is an integer number of ordered pairs that will be used to represent the CDF ($n > 1$).
- The first value in each ordered pair is the value of the distribution at a particular point.
- The second value in the ordered pair is a relative frequency associated with the value.
- The frequencies are relative only to each other, and must be strictly positive.
- The values must increase monotonically.

UNIFORM* ***n obs$_1$ obs$_2$... obs$_n$ first_point endpoint$_1$ endpoint$_2$... endpoint$_n$***
and
LOGUNIFORM* ***num obs$_1$ obs$_2$... obs$_n$ first_point endpoint$_1$ endpoint$_2$... endpoint$_n$***

- The first parameter specifies the number of intervals n in an n-part histogram.
- n is followed by a series of n integer values *obs*.
- Each value obs_i represents the number of observations that LHS will draw from the i^{th} interval. The sum of all obs_i must equal the number of observations requested from LHS.
- The obs_i are followed by n contiguous intervals that are to be sampled.
- The intervals are specified by entering a *first_point* (the distribution minimum) followed by n interval *endpoint* values.

3.3.5 User-Defined Discrete Distributions

DISCRETE CUMULATIVE ***n ordered_pair$_1$ order_pair$_2$... ordered_pair$_n$***

- *n* is an integer number of ordered pairs that will be used to represent the CDF (*n* > *1*).
- Within each ordered pair, the first number is the value of the distribution; the second number is the cumulative probability (CDF value) associated with this distribution value.
- Probabilities increase monotonically starting with a value greater than 0.0 and ending with 1.0.
- The distribution values (the first entry in the ordered pair) must increase monotonically.

DISCRETE HISTOGRAM *n ordered_pair$_1$ ordered_pair$_2$... ordered_pair$_n$*

- *n* is an integer number of ordered pairs that will be used to represent the CDF (*n* > *1*).
- The first value in each ordered pair is the value of the distribution at a particular point.
- The second value in the ordered pair is a relative frequency associated with the value.
- Frequencies are relative only to each other, and must be strictly positive.
- The values must increase monotonically.

3.4 Input File Specifications

This section summarizes the LHS Input File specifications.

3.4.1 File Characteristics

- Space-delimited case-insensitive free-format ASCII text file.
- Each input line must contain no more than 80 ASCII characters.
- Continuation character is either a "#" or a "%", preceded by one or more blanks, at the end of the line.
- All text following "$" characters (dollar sign) are considered comments.
- Each continuation line must end with a continuation character unless it terminates the command.
- A command may begin and end at any point on the line (an arbitrary number of leading and trailing spaces may be used as desired).
- If a command keyword is composed of more than one word, those words must be separated by *one and only one blank space*.
- Multiple blank spaces may be included between a keyword and any required alphabetic or numeric values.
- LHS treats commas (",") and tab characters as being fully equivalent to and interchangeable with blank space characters.
- Numeric input may be specified in any Fortran-standard format.
- In a trailing comment, the dollar sign must be preceded by one or more blank spaces.
- It is not possible to place comments between items on a single line.
- Full-line comments may be placed between a continued line and its continuation or between consecutive continuation lines.

3.4.2 Names of Random Variables

- Names may contain no more than 16 characters
- The "$" (dollar sign), "#" (number sign), "%" (percent sign), "," (comma), " " (blank space), and tab characters are not valid within names
- A name must not form a valid representation of a real number for an ANSII-standard Fortran compiler.

3.4.3 Distribution File Structure

- LHS will take as its input any line from the file on which the first keyword is Data:.
- All lines that follow the keyword Dataset: (on a line by itself) are assumed to belong to LHS input regardless of whether they begin with the keyword Data:.
- Keywords used to describe distribution input follow a common format:

name point_value dist-name dist-parm dist-parm .....

- The *point_value* parameter represents the value that the analyst wants used for this variable by models that cannot accept or effectively use the sampled results provided by LHS.
- The *point_value* field is required only when the analyst specifies that LHS is to use the user-specified value for the point estimate instead of computing the sample mean or median for this purpose. In all other instances, the *point_value* parameter is optional and can be omitted.

3.5 LHS Execution and Output

3.5.1 Running LHS

In Standalone mode, LHS is run by entering the following at the command line:

lhsdrv *filename* <enter>

where *filename* is the name of the keyword file. If the parameter *filename* is omitted, the program will prompt the user for the name of the keyword file. The keyword file contains all of the necessary information to specify an LHS run (such as random seed, number of sample observations, distribution names and associated parameters, etc.)

3.5.2 The Message Output File

The message file is the output file that the user will read. It consists of three parts: the header page, an input review, and an uncertainty data block.

Header Page

The header page contains a listing of the time and date the program was run, which files were accessed during execution, and which options were specified. Options include random or Latin hypercube sampling, random or restricted pairing, and output options such as correlation matrices and histograms.

Input Review

This section is simply an annotated copy of the input to LHS, including the distributions and correlation specifications that were performed, the starting point of the random number generator, the number of random variables, and the number of observations.

Uncertainty Data Block

This data block contains the information specified by the LHSRPTS keyword. If DATA was specified, then there will be a listing of each sampled distribution as well as the rank information from the sample. If CORR was specified, then the raw and rank correlation matrices associated with the actual sample generated are printed. If HIST was specified, then one text histogram will be printed for each random variable. Any combination of the above three options may be used.

3.5.3 The Sampled Data Output File

The file format consists of three blocks: the point estimate data block, the uncertainty header block, and the uncertainty data block. Since LHS is meant to be run in a suite of codes, this output will have characteristics similar to the input files of other programs so that other programs may easily use this output. All three blocks must be present in the file in the stated order. The point estimate data block starts at the beginning of the file and proceeds until the keyword "@UNCERTAINTY" is found as the first item on the line. This marks the beginning of the uncertainty header block. This header block ends with the keyword "@SAMPLEDATA". The uncertainty data block follows this keyword and occupies the rest of the file.

Common Characteristics of the Point Estimate and Uncertainty Header Blocks

1. All data fields are space-delimited. All processors should treat commas and tab characters as if they were blank spaces.

2. A data record may occupy multiple lines, but each line may have no more than 80 characters. Characters that occur in columns greater than 80 are ignored. If a data record is to be continued on the following line, it must contain a continuation character (either a "#" or a "%") as the last character prior to any trailing blanks or comments (see below). The continuation character must be preceded by one or more blank spaces.

3. All text following a "$" (dollar sign) is considered a comment. If a dollar sign is the first item on a data record, the entire record is treated as a comment. A dollar sign and comment may also follow all data items on a record or the continuation character (trailing comment). In these cases, the dollar sign must be preceded by one or more blank spaces. Comments may not come between data items on a single record (except following a continuation character on a continued line).

4. Blank lines are considered comment records and are ignored. Any number of blank lines and/or comment lines may be placed between a line and its continuation or between consecutive continuation lines.

5. All names used in the file must meet the following criteria: (a) each name must contain no more than 16 characters; (b) the "$" (dollar sign), "#" (number sign), "%" (percent sign), "," (comma), " " (blank space), and tab characters are not valid within names; and (c) a name must not form a valid representation of a real number for an ANSI-standard Fortran compiler. All names are to be treated as case-insensitive, so the names FRED, Fred, fred, and freD are all equivalent. Names that are longer than 16 characters may be truncated to 16 characters at the discretion of the processor.

6. Where numbers are specified, they may be in any format recognized by an ANSI-standard Fortran compiler.

7. Each name must be unique within each section. Duplicate definitions in the same section for a single data item are not valid. However, an item may be defined once in each section. If an item is defined once in the point estimate block and once in the uncertainty header block, both definitions are effective. The reading program would have to decide whether its calculations should use the point estimate value, the distribution, or both.

8. A line and all of its continuation lines (with all trailing and intervening comments eliminated) must together consist of no more than 32,767 characters.

Point Estimate Block

This block starts the beginning of the file and proceeds until the keyword @UNCERTAINTY is found as the first item on a line. It begins with a single comment record (starting with a dollar sign) that documents the version of the LHS file format being used. The record should be exactly as follows:

```
$ LHS File Format Version 1.00
```

The record need not begin in the first column. Any program to read this record (in order to verify input file compatibility) may assume that only one blank character exists between each of the words, but should not make any assumptions regarding which characters will be in upper or lower case.

This initial comment record will be followed by several additional comment records that document the pedigree of the file, such as the name and version of the program, the data and time the program was run, and the list of files accessed by the program.

A data record in the point estimate data block consists of anything that is not discarded as a comment. Only one type of data record will be written out: a list of one or more names followed by one or two real number values. The first value represents the point value for each of the items found on the list of named data items. The second item, which is optional, is the standard deviation associated with the point estimate value. The point estimate value that appears at the end of the line is assigned to all named data items on that line (including all active continuation lines). The list is space delimited (see Section 2 above).

Any processor reading this data format should look for the following common errors: (a) a record consisting of a list of names that is not followed by a legal value is illegal; (b) a record consisting of a value without a corresponding list of names is illegal; (c) names that follow the value(s) on a record are illegal.

Any processor wishing to skip this section can simply read and discard all lines until the keyword "@UNCERTAINTY" is found as the first data item on a noncomment line (the keyword "@UNCERTAINTY" may be preceded by leading blank spaces).

The Uncertainty Header Block

The uncertainty header block contains two input sections that may occur in either possible order. The first section consists of a single line containing the keyword @OBSERVATIONS as its first data item, followed by a positive integer that represents the number of observations that are found in the uncertainty data block. It is critical that the value on this record accurately reflect the true number of observations present in the file. Any processor reading this data format must be sure that there are at least this number of observations present in the file. It may, however, ignore any additional observations that may be present.

The second section describes the number and names of the distributions that are found in the uncertainty data block. The section begins with a single line containing the keyword @VARIABLES as its first data item, followed by a positive integer that represents the number of distributions that are found in the uncertainty data block. It is critical that the value on this record accurately reflect the true number of variables present in the file. Any processor reading this data format must be sure that there are at least this number of distributions present in the file. It may, however, ignore any additional distributions that may be present.

The *@VARIABLES n* line must be followed by exactly *n* lines that provide names for the *n* distributions. Each of these lines begins with the primary name for the distribution, followed by a ":" (colon character). Following the colon character is an optional list of secondary names for the distribution. Thus, each of the *n* distributions is given exactly one primary name and an arbitrary number of secondary names. The list of names is space-delimited and may extend to one or more continuation lines. Comments may be present at any point in this block.

The order in which these records appear is significant. All names on the first record encountered in this section are assumed to correspond to the first distribution in the uncertainty data section of the file. The names on the second record are assumed to correspond to the second distribution, and so forth. For this reason, the number of records found in this section must correspond exactly to the number that follows the @VARIABLES keyword, and that value must exactly match the number of distributions found in the uncertainty data block. Anything else is an error condition.

The uncertainty header block is terminated by the @SAMPLEDATA keyword.

Uncertainty Data Block

The uncertainty data block is completely different from either of the preceding blocks. It contains only numeric data (no names). Comments lines are <u>not</u> supported in this block. Continuation lines exist only as described below. This block contains most of the numeric data generated by LHS.

It contains n_{obs} data records – one record for each observation. The data record for a file containing n_{var} variables consists of two integers followed by n_{var} real values. The first integer value contains the observation number. The second integer value contains the number of the distribution values that follow (n_{var}). These integers are followed by one value from each of the n_{var} distributions. In other words, there will be (n_{var}+2) columns of data – the first tells which observation is on that line,

the second contains the number of distributions, and the next n_{var} columns each correspond to one of the distributions performed.

The records in this section are generally written and read using a Fortran list-type format in an implied "Do Loop." Thus, one record may consist of several lines without explicit continuation characters. The Fortran program will read additional lines as needed. The following hypothetical code example would read one entire uncertainty data block (all variables and all observations):

```
Do i=1, NObs
     Read (1,*) i, NVar, (Values(i,j), j=1, NVar)
End Do
```

The block is typically written to the file using similar code. Note that the use of the Fortran "*" format for reading and writing does not allow for continuation characters or comment lines. Thus, these are not allowed in this data block.

Note that this block occupies the remainder of the file. Thus, an attempt to read this file following the completion of the above loop should result in an end-of-file error condition. Figure 3-3 contains a sample LHS output file.

```
$ LHS File Format Version 1.00
$
$ This LHS run was executed on  9/ 4/97 at 12:31:46.46
$ with LHS Version: 2.10 Release 2, Compiled 03/10/1997
$ The run title was:
$ LHS INPUT FOR NUCLEAR TECHNOLOGY TEST PROBLEM
$ Message output file for this run: LHS.MSG
$
$ Input file(s) for this run:
$
$ Point Values for the distributions follow:
$
$ All point values represent the optional point
$ values that were found in the input file.
$
    BA                    2.0000000E-01
    BB                    2.0000000E-01
    BC                    1.0000000E-01
    BD                    1.0000000E-01
    I1                    1.0000000E+00
    I2                    2.0000000E+00
$
@UNCERTAINTY
  @OBSERVATIONS     10
  @VARIABLES      6
    BA:
    BB:
    BC:
    BD:
    I1:
    I2:
@SAMPLEDATA
          1           6   0.174378       0.204163       0.203658
   0.712264E-01   0.175162        1.02023
          2           6   0.127138       0.135356       0.616336E-01
   0.994375E-01    5.18535       0.524736
          3           6   0.153722       0.247982       0.108648
   0.400652        1.12832        4.05337
          4           6   0.195077       0.189196       0.934008E-01
   0.280316E-01   0.809043        2.92969
          5           6   0.331527       0.151177       0.185418
   0.633564E-01    1.50779        1.50650
          6           6   0.131032       0.102713       0.117998
   0.845984E-01   0.371710        1.93410
          7           6   0.881318E-01   0.290377       0.511588E-01
   0.417372E-01   0.644589        1.72556
          8           6   0.258039       0.405209       0.679013E-01
   0.134097       0.603218       0.877367
          9           6   0.226539       0.122867       0.408741E-01
   0.145608       0.269958        1.20896
         10           6   0.285970       0.176833       0.293484E-01
   0.529115E-01   0.309393        2.55828
```

Figure 3-3: Sample Output File

4. Distribution Input

This chapter describes the distributions that can be sampled in LHS in more detail. The file specification for these distributions is found in Section 3.3.

4.1 Continuous Distributions Recognized by LHS

The keywords described in this section are used to specify which distributions are to be sampled by the LHS program. Users may select from among 21 traditional continuous distribution functions and 5 empirical continuous distribution functions in order to achieve the best possible representation of their data. Discrete distribution functions are described in the next section. The 21 continuous distributions are contained in 5 major groups: normal distributions, lognormal distributions, uniform and loguniform distributions, user-specified continuous distributions, and miscellaneous continuous distributions.

4.1.1 Normal Distributions

The LHS software provides the user with four different methods for sampling from the normal distribution. The normal distribution is defined by the density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \quad -\infty < x < \infty,$$

where the distribution mean and variance are μ and σ^2, respectively. The standard deviation of the distribution, which is required by LHS as an input parameter for several normal distribution sampling methods, is denoted by σ. The first sampling method for the normal distribution samples over all quantiles. The remaining methods sample normal distributions that have been truncated or bounded.

NORMAL ***mean standard-deviation***

The keyword NORMAL specifies that a normal distribution is to be performed. The function is sampled over all quantiles. The normal distribution is defined by two input parameters: the mean (μ) and the standard deviation (σ). The mean may be any real value; however, the standard deviation must be positive.

Example: `Item1 0.0 NORMAL 0.0 1.0`

This example defines a normal distribution named "Item1" with a point value of 0.0, a mean of 0.0, and a standard deviation of 1.0. Figure 4-1 shows the PDFs of the three normal distributions with varying μ and σ.

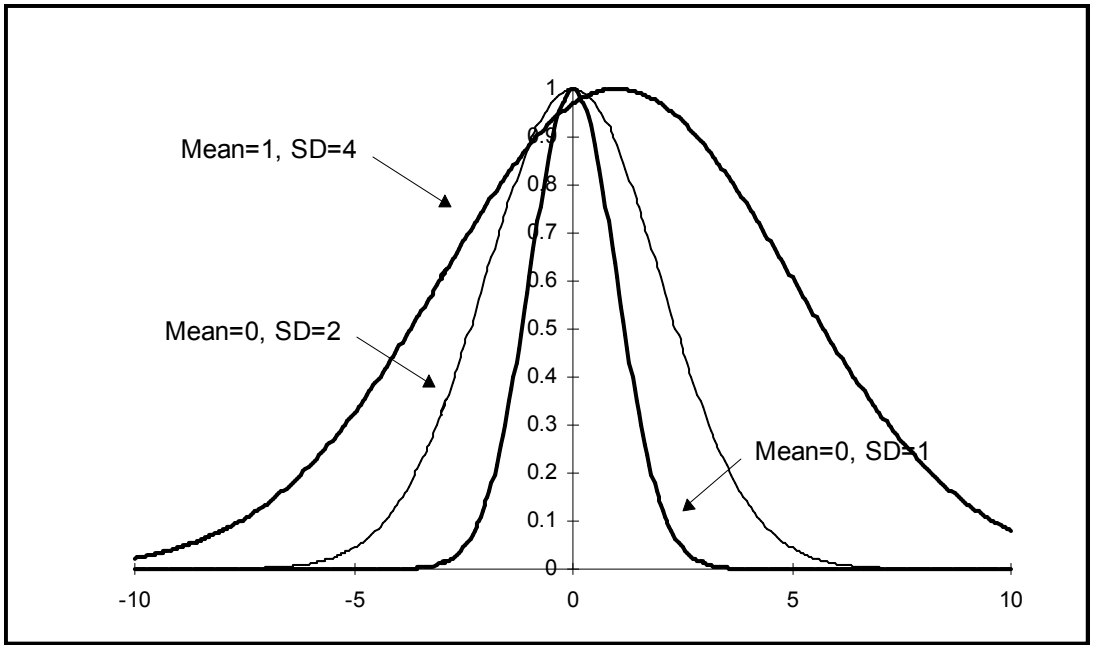

Figure 4-1: Normal Distribution PDF

TRUNCATED NORMAL ***mean standard_deviation lower upper***

A truncated normal distribution is a normal distribution that is only sampled between two user-specified quantiles. The TRUNCATED NORMAL distribution is defined by four input parameters: the mean and standard deviation of the normal distribution, the lower quantile beyond which sampling is not to occur, and the upper quantile beyond which sampling is not to occur. As above, the standard deviation must be positive. In addition, the lower and upper quantile values must satisfy the condition $0.0 \leq lower < upper \leq 1.0$.

Example: `Item2 0.5 TRUNCATED NORMAL 3.0 1.0 0.1 0.8`

This example defines a distribution with the name "Item2" and a point value of 0.5. The normal distribution with a mean of 3.0 and a standard deviation of 1.0 is to be sampled only between the 0.1 and 0.8 quantiles (the PDF for this distribution is shown in Figure 4-2).

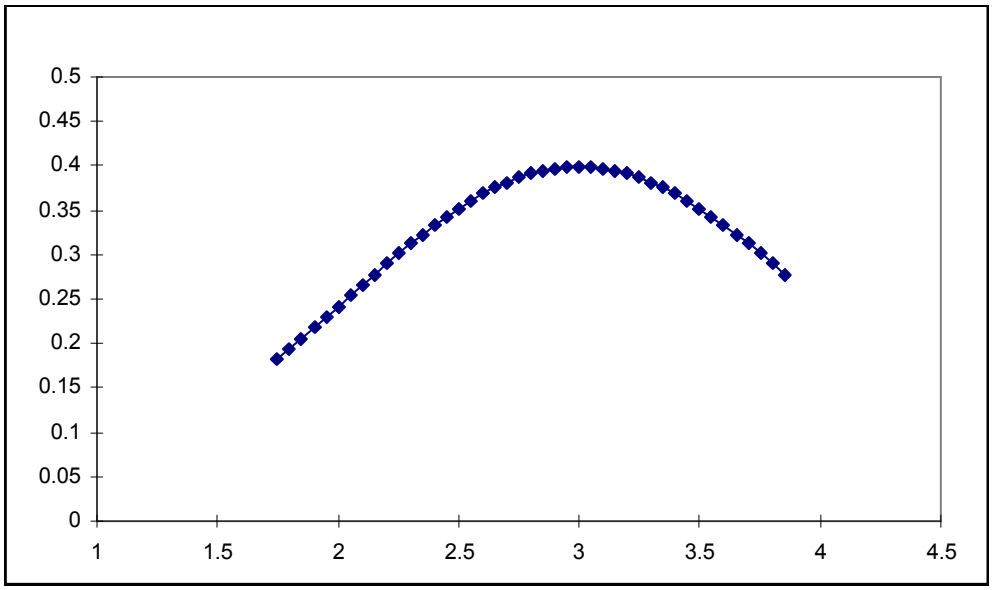

Figure 4-2: Truncated Normal Distribution PDF ($\mu = 3$, $\sigma = 1$, *lower* = 0.1, *upper* = 0.8)

BOUNDED NORMAL ***mean standard_deviation lower_bound upper_bound***

The bounded normal distribution is similar to the truncated normal distribution except that the user specifies the distribution values between which sampling is to occur, instead of the quantiles between which sampling is to occur. Four parameters are required: the mean and standard deviation of the sampled normal distribution, the lower function value beyond which sampling is not to occur, and the upper function value beyond which sampling is not to occur. The *lower_bound* parameter must be less than the *upper_bound*.

Example: `Item4 3.5 BOUNDED NORMAL 2.9 2.0 1.0 6.0`

This example defines a distribution named "Item4" with a mean of 2.9 and a standard deviation of 2.0 that is to be sampled only between the values 1.0 and 6.0. The point value of 3.5 is an optional input parameter if the user allows LHS to automatically compute a point value as the mean or median of the generated sample. Figure 4-3 shows the PDF of the normal distribution between 1.0 and 6.0.

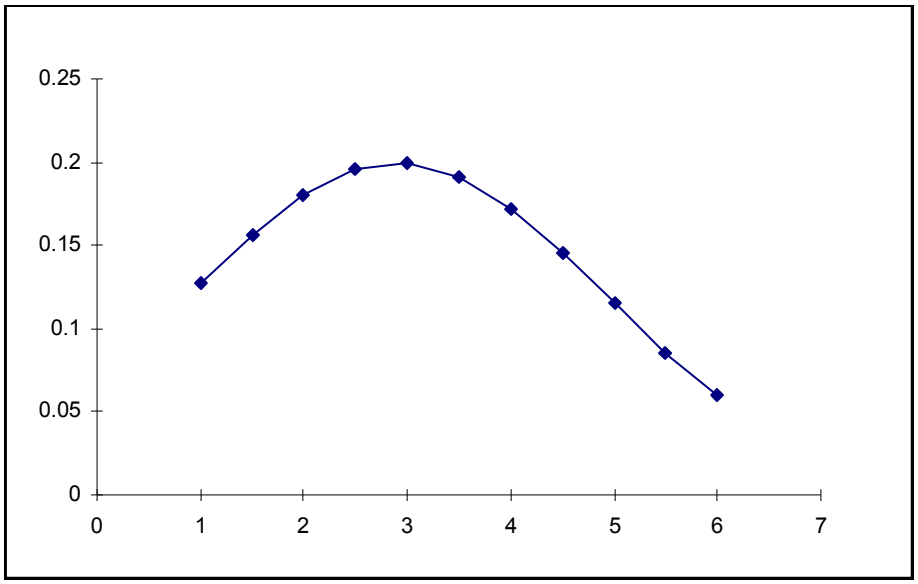

Figure 4-3: Bounded Normal Distribution PDF ($\mu = 2.9$, $\sigma = 2.0$, *lower_bound* = 1.0, *upper_bound* = 6.0)

NORMAL-B ***value_at_0.001 value_at_0.999***

This method of sampling a bounded normal distribution is provided mainly for backward compatibility with previous versions of LHS. It samples a normal distribution between the quantiles of 0.001 and 0.999. However, instead of specifying the mean and standard deviation of the normal distribution along with the bounds or quantiles to be sampled between, the user specifies only the range that is to be sampled. LHS *assumes* that the endpoints of this range are the values of the distribution at the 0.001 and 0.999 quantiles, respectively. These parameters can be related to the mean and the standard deviation of the distribution as follows:

$$V_{0.001} = \mu - 3.09023\sigma$$

$$V_{0.999} = \mu + 3.09023\sigma$$

Example: `Item3 5.85 NORMAL-B 2.0 9.7`

This example defines a normal distribution named "Item3" that has a range from 2.0 to 9.7. This range translates to a normal distribution with a mean of 5.85 and a standard deviation of 1.25 that is sampled only between the values of 2.0 and 9.7, or equivalently, between the quantiles of 0.001 and 0.999. In this example, the optional point value has been set to the mean value of the distribution. Figure 4-4 shows the normal-B distribution.

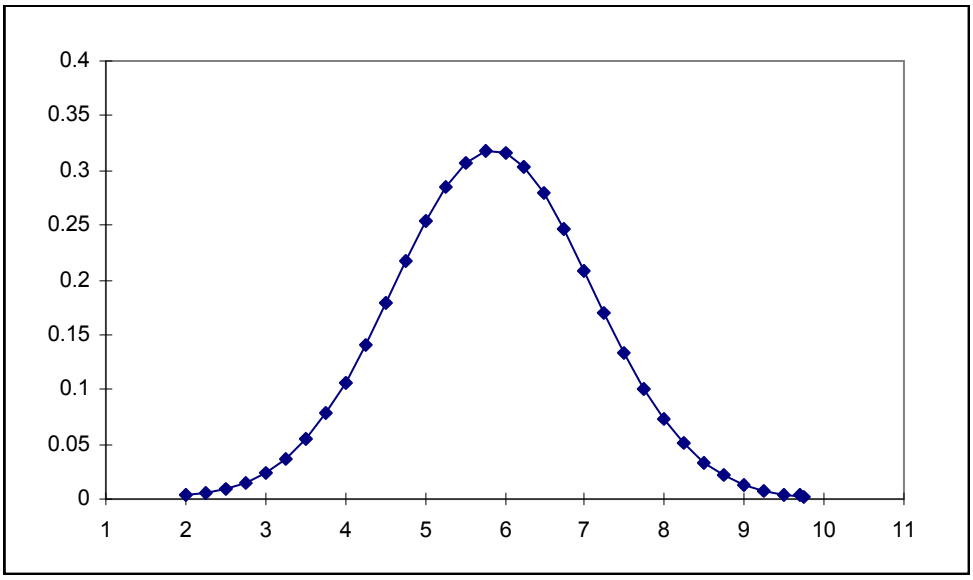

Figure 4-4: Normal-B Distribution PDF (*value_at_0.001* = 2.0, *value_at_0.999* = 9.7)

4.1.2 Lognormal Distributions

A lognormal distribution is simply a distribution whose logarithm is described by a normal distribution. A lognormal distribution is defined by a density function of

$$f(y) = \frac{1}{y\sigma\sqrt{2\pi}} \exp[\frac{-(\ln\ y - \mu)^2}{2\sigma^2}], \qquad y > 0\ 1$$

where μ and σ are the mean and standard deviations of the underlying normal distribution. The mean, variance, and median of the lognormal distribution can be computed from μ and σ using the following formulas:

$$E(y) = e^{\left(\mu + \frac{\sigma^2}{2}\right)}$$

$$V(y) = e^{\left(2\mu+\sigma^2\right)} \times \left(e^{\sigma^2} - 1\right)$$

$$\text{median} = e^{\mu}.$$

Lognormal distributions are typically specified in one of two ways throughout the literature. One way is to specify the mean and standard deviation of the underlying normal distribution (μ and σ) as described above. The other way is to specify the distribution using the mean of the lognormal distribution itself and a term called the "error factor." The error factor for a lognormal distribution is defined as the ratio of the 95th percentile to the median, or, equivalently, the ratio of the median to the 5th percentile. Physically, its square represents the width of a 90% confidence interval with

respect to the median. The mathematical relationship between the input mean and error factor, and the parameters of the underlying normal distribution (μ and σ) is shown by the following relations:

$$\sigma = \frac{\ln(\text{error factor})}{1.645}$$

$$\mu = \ln(\text{input mean}) - \frac{1}{2}\sigma^2 \ .$$

When the mean and error factor are used as input for the lognormal distribution, both input parameters must be positive, and the error factor must be greater than one. If μ and σ are specified, there is no restriction on μ, but σ must be positive.

LOGNORMAL ***mean error_factor***

The keyword LOGNORMAL is used to specify a lognormal distribution that is sampled over all quantiles using the mean and the error factor as input parameters.

Example: `Item7 0.32 LOGNORMAL 0.01 3.0`

This example represents a lognormal distribution with a mean of 0.1 and an error factor of 3.0. This translates to an underlying normal distribution with a mean of –4.82818 and a standard deviation of 0.667849. The 90% confidence interval for this distribution is a factor of 9.0. Figure 4-5 shows the PDF of a lognormal distribution.

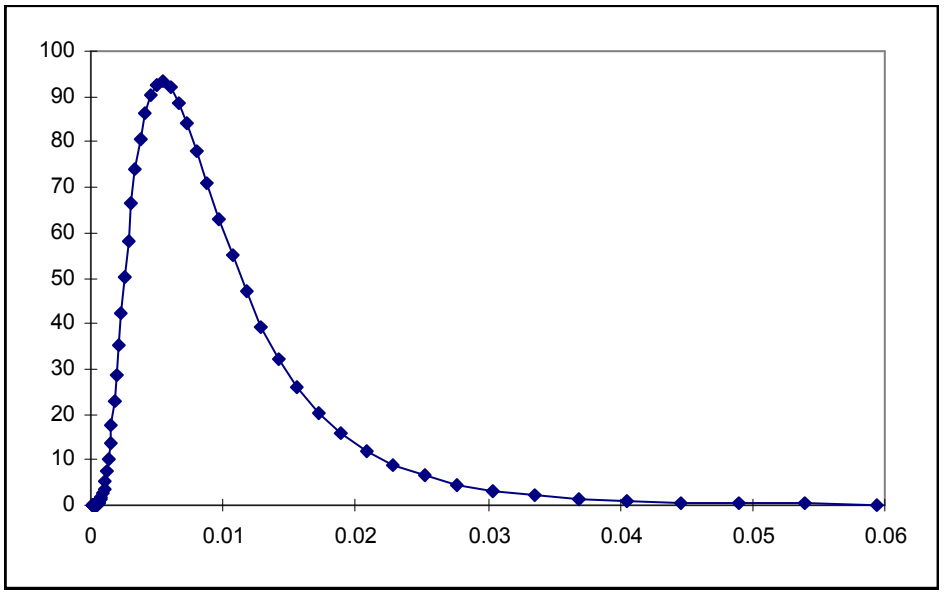

Figure 4-5: Lognormal Distribution PDF (*mean* = 0.01, *error_factor* = 3.0)

LOGNORMAL-N ***mean standard_deviation***

Use of the keyword LOGNORMAL-N will also produce a lognormal distribution sampled over all quantiles. With this keyword, however, the user must enter the distribution parameters as the mean and standard deviation of the underlying normal distribution (μ and σ).

Example: `Item6  0.2  LOGNORMAL-N  -4.82818  0.667849`

This example produces a lognormal distribution based on an underlying normal distribution with a mean of –4.82818 and a standard deviation of 0.667849. This lognormal distribution has a mean of 0.01 and an error factor of 3.0. This example is equivalent to the example shown for the lognormal distribution type above.

TRUNCATED LOGNORMAL ***mean error_factor lower upper***

The keyword TRUNCATED LOGNORMAL is used to describe a lognormal distribution that is sampled only between two user-specified quantiles using the mean and the error factor as input parameters. The *lower* and *upper* parameters represent the lower and upper quantiles beyond which sampling is not to occur. The lower and upper quantile values must satisfy the condition $0.0 \leq lower < upper \leq 1.0$.

Example: `Item10 TRUNCATED LOGNORMAL  2.0  2.0  0.13  0.86`

This example yields a lognormal distribution named "Item10" with a mean value of 2.0 and an error factor of 2.0. The distribution will only be sampled between the 13th and 86th percentiles (the 0.13 and 0.86 quantiles). In this example, the optional point value has been omitted.

TRUNCATED LOGNORMAL-N ***mean standard_deviation lower upper***

The keyword TRUNCATED LOGNORMAL-N is equivalent in every way to the keyword TRUNCATED LOGNORMAL except that in this case the user must enter the lognormal distribution parameters as the mean and standard deviation of the underlying normal distribution (μ and σ) as was done for the LOGNORMAL-N keyword described previously. This distribution will be sampled only between the quantiles specified by the *lower* and *upper* input parameters. The lower and upper quantile values must satisfy the condition $0.0 \leq lower < upper \leq 1.0$.

Example: `Item11 TRUNCATED LOGNORMAL-N  2.0  3.0  0.13  0.86`

This example produces a lognormal distribution whose underlying normal distribution has a mean of 2.0 and a standard deviation of 3.0, and is only sampled between the 13th and 86th quantiles. The equivalent mean and error factor are 665.1 and 139.1. In this example, the optional point value has been omitted.

BOUNDED LOGNORMAL ***mean error_factor lower_bound upper_bound***

A bounded lognormal distribution is similar to the truncated lognormal distribution described previously except that the user specifies the distribution values between which sampling is to occur, instead of the quantiles between which sampling is to occur. The keyword BOUNDED LOGNORMAL must be accompanied by the mean, error factor, the lower function value beyond which sampling should not occur and the upper function value beyond which sampling should not occur. The *lower_bound* parameter must be less than the *upper_bound*, and both values must be positive.

Example: `Item7 0.22 BOUNDED LOGNORMAL 0.34 2.0 0.1 2.2`

This example produces a lognormal distribution named "Item7" with a mean of 0.34, an error factor of 2.0, and a point estimate value of 0.22. The lognormal distribution will not be sampled for values less than 0.1 nor for values greater than 2.2.

BOUNDED LOGNORMAL-N ***mean standard_deviation lower_bound upper_bound***

The keyword BOUNDED LOGNORMAL-N is equivalent in every way to the keyword BOUNDED LOGNORMAL except that in this case the user must enter the lognormal distribution parameters as the mean and standard deviation of the underlying normal distribution (μ and σ) as was done for the LOGNORMAL-N keyword described previously. This distribution will be sampled only between the lognormal distribution values specified by the *lower_bound* and *upper_bound* input parameters. The *lower_bound* parameter must be less than the *upper_bound*, and both values must be positive.

Example: `Item8 1.0 BOUNDED LOGNORMAL-N 0.1 0.2 1.0 2.0`

This example would produce a lognormal distribution based on an underlying normal distribution with the parameters $\mu = 0.1$ and $\sigma = 0.2$. The distribution will be sampled only between the lognormal distribution values 1.0 and 2.0.

LOGNORMAL-B ***value_at_0.001 value_at_0.999***

This method of sampling a bounded lognormal distribution is provided mainly for backward compatibility with previous versions of LHS. It samples a lognormal distribution between the quantiles of 0.001 and 0.999. However, instead of specifying the mean and error factor of the lognormal distribution along with the bounds or quantiles to be sampled between, the user specifies only the range that is to be sampled. LHS *assumes* that the endpoints of this range are the values of the lognormal distribution at the 0.001 and 0.999 quantiles, respectively. These parameters must be positive, and *value_at_0.001* must be less than *value_at_0.999*. These parameters can be related to the mean and the standard deviation of the underlying normal distribution, as well as the lognormal mean and error factor as follows:

$$V_{0.001} = e^{(\mu - 3.09023\sigma)} = e^{\left(\ln(M) - \frac{1}{2}\left(\frac{\ln(E)}{1.645}\right)^2 - 3.09023\left(\frac{\ln(E)}{1.645}\right)\right)}$$

$$V_{0.999} = e^{(\mu + 3.09023\sigma)} = e^{\left(\ln(M) - \frac{1}{2}\left(\frac{\ln(E)}{1.645}\right)^2 + 3.09023\left(\frac{\ln(E)}{1.645}\right)\right)},$$

where M and E are the mean and error factor of the desired lognormal distribution, and μ and σ are the mean and standard deviation of the underlying normal distribution.

Example: `Item12  4.55  LOGNORMAL-B  2.0  9.7`

This example defines a lognormal distribution named "Item3" that has a range from 2.0 to 9.7. This range translates to an underlying normal distribution with a mean of $\mu = 1.48$ and a standard deviation of $\sigma = 0.255$, or equivalently, a lognormal mean of $M = 4.55$ and an error factor of $E = 1.52$. The distribution is sampled only between the values of 2.0 and 9.7, or, equivalently, between the quantiles of 0.001 and 0.999. In this example, the optional point value has been omitted.

4.1.3 Uniform and Loguniform Distributions

A uniform distribution, specified over a particular bounded interval (A, B), has the property that all points within that interval are equally likely. A loguniform distribution is simply a distribution whose logarithm is described by a uniform distribution. A uniform distribution is defined by the following probability density and cumulative distribution functions:

$$f(x) = \frac{1}{B - A}, \quad A \le x \le B$$

$$F(x) = \frac{x - A}{B - A}, \quad A \le x \le B .$$

The mean $E(x)$, median $M(x)$, and variance $V(x)$ are

$$E(x) = M(x) = \frac{A + B}{2}$$

$$V(x) = \frac{(B - A)^2}{12}$$

Similarly, a loguniform distribution is defined by the following probability density and cumulative distribution functions:

$$f(x) = \frac{1}{x}(\ln B - \ln A) \quad A < x < B$$

$$F(x) = \frac{\ln x - \ln A}{\ln B - \ln A}, \quad A < x < B .$$

The mean $E(x)$, median $M(x)$, and variance $V(x)$ are

$$E(x) = \frac{B - A}{\ln B - \ln A}$$

$$M(x) = \exp\left(\frac{\ln B + \ln A}{2}\right) = \sqrt{AB}$$

$$V(x) = (B - A) \frac{(\ln B - \ln A)(B + A) - 2(B - A)}{2(\ln B - \ln A)^2} .$$

UNIFORM ***A B***

A uniform distribution samples values uniformly between two specified intervals. The keyword UNIFORM, followed with two endpoints, *A* and *B*, will sample a uniform distribution.

Example: `Item13 0.5 UNIFORM 0.0 1.0`

This example samples a uniform distribution named "Item13" over the interval from 0.0 and 1.0. Figure 4-6 shows the PDF of a uniform distribution.

LOGUNIFORM ***A B***

The keyword LOGUNIFORM allows the logarithm of the variable to be sampled uniformly over the logarithm of the range specified by the two endpoints supplied. The endpoints *A* and *B* must be positive.

An example may help the uninitiated user understand how a loguniform distribution is sampled in LHS. Consider an example distribution with $A = 10^{-3}$ and $B = 10$. The logarithm (base 10 is used here for convenience) of this distribution, which has distribution endpoints $A_{\log 10} = -3$ and $B_{\log 10} = 1$, is sampled as a uniform distribution. The individual sample results are then converted back to the original distribution by a base 10 exponentiation process. Thus, one-quarter of the samples will

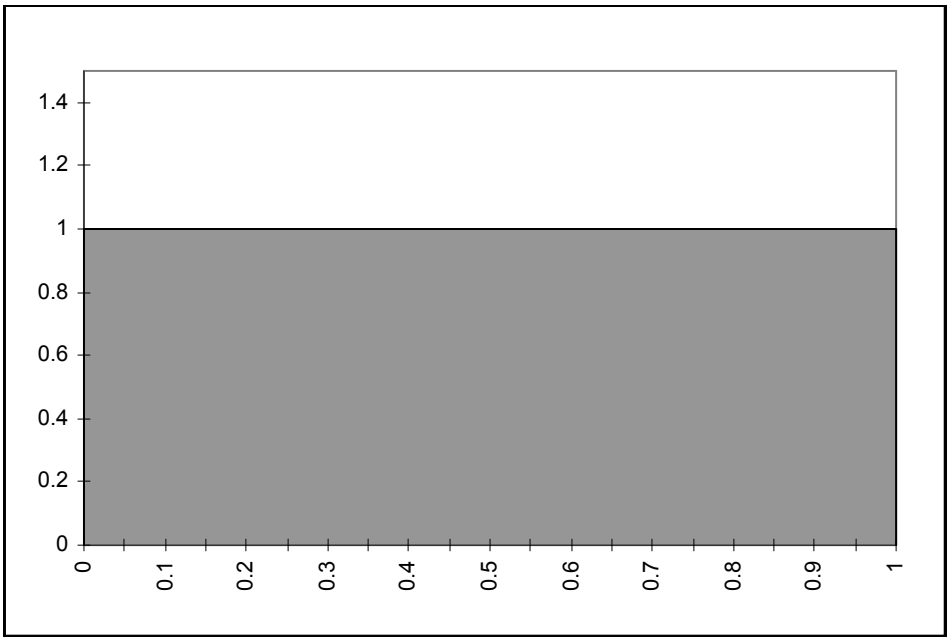

Figure 4-6: Uniform Distribution PDF ($A = 0.0$, $B = 1.0$)

be drawn between –3 and –2 for the underlying uniform distribution, another one-quarter will be drawn between –2 and –1, another one-quarter will be drawn between –1 and 0, and the final one-quarter will be drawn between 0 and 1. The lognormal variable will show one-quarter of its samples between 10^{-3} and 10^{-2}, another one-quarter between 10^{-2} and 10^{-1}, another one-quarter between 10^{-1} and 10^{0} (or 1) and the one-quarter between 1 and 10. Thus the variable is sampled uniformly on a logarithmic scale, and each decade is sampled with the same frequency.

Example: `Item14 0.375 LOGUNIFORM 0.001 10`

In this example, the variable named "Item14" with a point value of 0.375 will be sampled as a loguniform distribution over a range from 0.001 to 10. This input string implements the example distribution described in the previous paragraph. Figure 4-7 shows a Loguniform PDF.

4.1.4 User-Defined Continuous Distributions

The user-defined continuous distributions implemented within LHS represent various generalizations of the uniform and loguniform distributions described in the previous section. In each case, the user enters data that will be interpreted by LHS as the points for a multisection distribution that is, at the user's discretion, either piecewise-uniform or piecewise-loguniform. LHS provides five different methods by which these distributions can be entered. Three of these methods result in piecewise-uniform distributions (linear interpolation of the input data), while the other two result in piecewise-loguniform distributions (logarithmic interpolation of the input data).

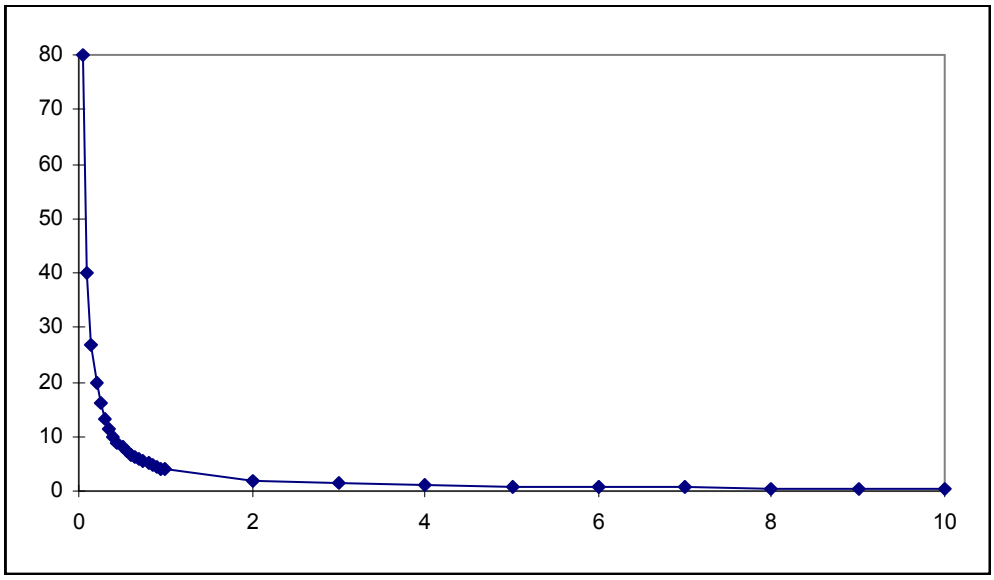

Figure 4-7: Loguniform PDF ($A = 0.001$, $B = 10$)

CONTINUOUS LINEAR ***n*** ***ordered_pair*$_1$** ***ordered_pair*$_2$ ... *ordered_pair*$_n$**

A continuous linear distribution is used to approximate an arbitrary distribution (either an empirical distribution or some other continuous distribution form not implemented within LHS) in terms of a set of piecewise uniform distributions. The user enters the CDF for this arbitrary distribution as a series of ordered pairs. The user must first specify *n*, an integer number of ordered pairs that will be used to represent the CDF ($n > 1$). The *n* ordered pairs that represent the CDF follow, forming the remainder of the input for this distribution. Within each ordered pair, the first number is the value of the distribution; the second number is the cumulative probability (CDF value) associated with this distribution value. *The probabilities in the ordered pairs (the second entry in the ordered pair) must increase monotonically starting with 0.0 and ending with 1.0.* The distribution values (the first entry in the ordered pair) must also increase monotonically. LHS uses these points to generate a piecewise-uniform distribution with the quantiles specified by the user. In other words, linear interpolation is used between the user-specified quantiles to define the CDF for this distribution. If $n = 2$, a uniform distribution is generated between the two points.

Example:
```
Item19    4.5     CONTINUOUS LINEAR    3   #
          5.0    0.0    #
          7.0    0.72   #
         10.0    1.0
```

This example constructs an empirical CDF named "Item19" with a minimum at $y = 5.0$, a maximum at $y = 10.0$, and its 72nd percentile at $y = 7.0$. Note the use of the continuation characters (#) to place the input requirements on more than one line to facilitate user interpretation. This distribution will be sampled as a piecewise uniform distribution with 72% of the samples drawn uniformly over the interval (5.0, 7.0) and 28% of the samples drawn uniformly over the interval

(7.0, 10.0). Figure 4-8 shows the PDF of a continuous linear distribution corresponding to the example with "Item19".

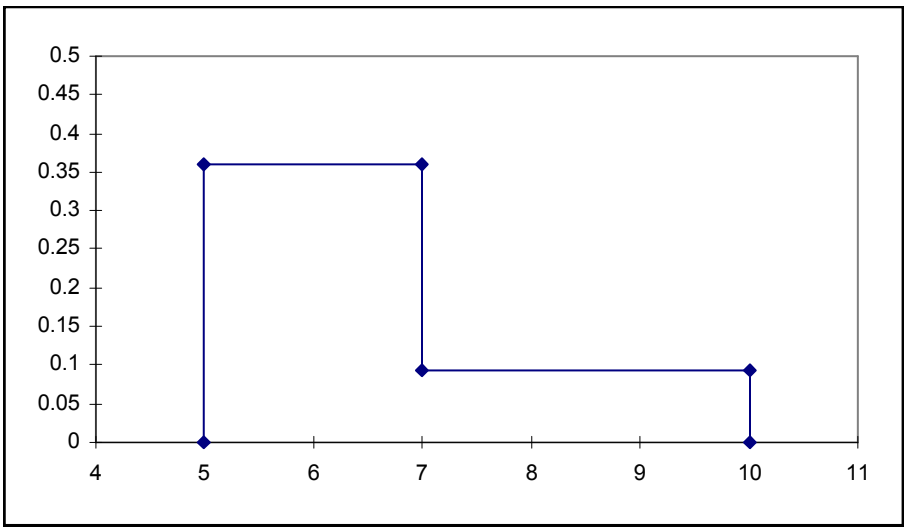

Figure 4-8: Continuous Linear PDF

CONTINUOUS LOGARITHMIC ***n ordered_pair*** $_1$ ***ordered_pair*** $_2$ **...** ***ordered_pair*** $_n$

A continuous logarithmic distribution is used to approximate an arbitrary distribution (either an empirical distribution or some other continuous distribution form not implemented within LHS) in terms of a set of piecewise loguniform distributions. Its input format and interpretation of data are identical to that of the continuous linear distribution above. LHS uses the specified ordered pairs to generate a piecewise-loguniform distribution with the quantiles specified by the user. In other words, logarithmic interpolation is used between the user-specified quantiles to define the CDF for this distribution. If $n = 2$, a loguniform distribution is generated between the two points.

Example:
```
Item20    5.4    CONTINUOUS LOGARITHMIC  4    #
                 0.01    0.0    #
                 0.05    0.35   #
                 0.1     0.79   #
                 1.0     1.0
```

This example constructs an empirical CDF named "Item20" with a minimum at $y = 0.1$, a maximum at $y = 1.0$, and its 35^{th} and 79^{th} percentiles at $y = 0.05$ and $y = 0.1$, respectively. This distribution will be sampled as a piecewise loguniform distribution with 35% of the samples based on a loguniform distribution over the interval (0.01, 0.05), 44% drawn similarly over the interval (0.05, 0.1), and the remaining 21% drawn from a loguniform distribution over the interval (0.1, 1.0). Note that both the distribution values (the first column in this example) and the quantiles (the second column in this example) increase monotonically. Figure 4-9 shows the PDF of a continuous logarithmic distribution corresponding to the example with "Item20".

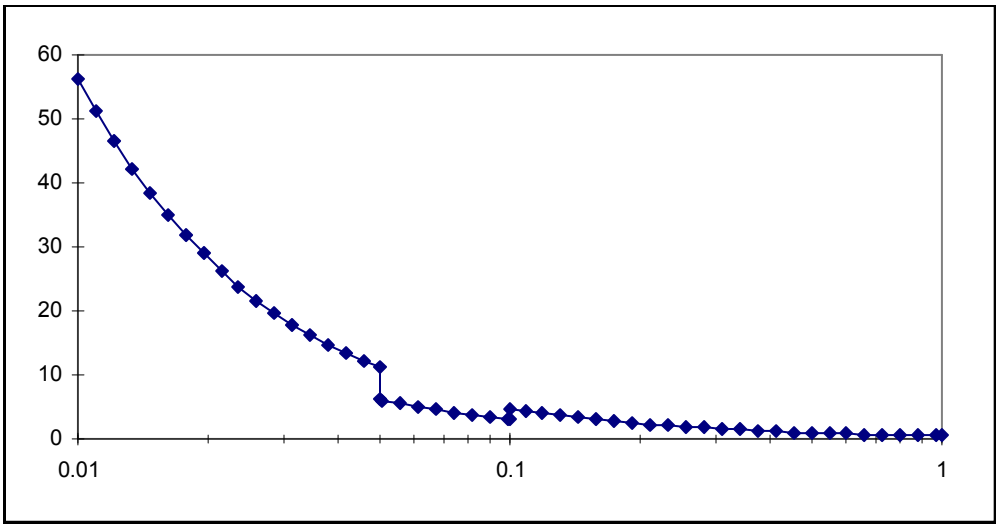

Figure 4-9: Continuous Logarithmic PDF

CONTINUOUS FREQUENCY ***n ordered_pair*** $_1$ ***ordered_pair*** $_2$ ***... ordered_pair*** $_n$

The keyword CONTINUOUS FREQUENCY provides an alternative method for specifying a piecewise uniform continuous distribution. It is similar in format to the continuous linear distribution described previously in that the user specifies the number of ordered pairs to be read, followed by a series of ordered pairs. Once again, the first value in each ordered pair is the value of the distribution at a particular point. However, instead of specifying the cumulative probability associated with that value as the second item in the ordered pair, as was done for the continuous linear distribution, the user specifies a relative frequency associated with the value. *The frequencies in the ordered pairs are relative only to each other, and must be positive.* The frequencies need not increase monotonically, although values must still increase monotonically.

LHS internally converts the continuous frequency distribution into a cumulative distribution function (continuous linear distribution) by assuming that between any two user-specified points, the relative probability density is the average of the input frequencies at the two points. The first and last points entered by the user are assumed to represent the minimum and maximum values of the distribution, respectively.

Example:
```
Item21   0.5   CONTINUOUS FREQUENCY   4   #
     11.0   1.0   #
     23.0  18.6   #
     30.0   7.2   #
     38.6   2.4
```

This example specifies a continuous distribution function that is piecewise uniform over three intervals. The relative probability density of the three intervals is 9.8, 12.9, and 4.8, so the total *relative* probability density over the entire distribution is 27.5. The relative interval probability densities are then normalized by the total relative probability density and aggregated to form the

cumulative distribution function that is sampled by LHS. Based on these values, the equivalent continuous linear representation for this example is:

```
Item21   0.5   CONTINUOUS LINEAR   4   #
     11.0   0.0     #
     23.0   0.356   #
     30.0   0.825   #
     38.6   1.0
```

If all input frequencies are identical, a uniform distribution is generated. If only two points are input, LHS adds an imaginary third point halfway between the two input points with a frequency of zero before converting the frequencies into a cumulative function. The continuous frequency distribution always uses linear interpolation. Figure 4-10 shows the PDF of the continuous frequency distribution corresponding to the "Item21" example.

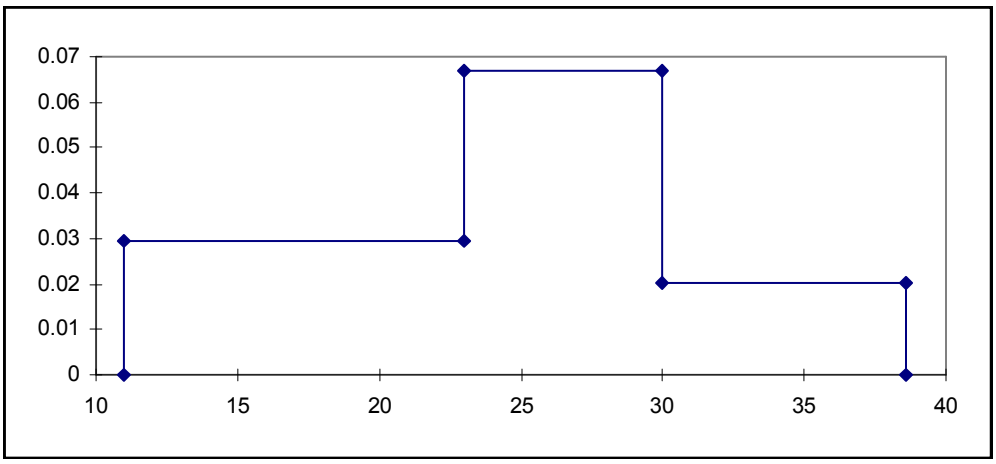

Figure 4-10: Continuous Frequency (Continuous Linear) PDF

UNIFORM* ***n*** ***obs$_1$*** ***obs$_2$ ... obs$_n$*** ***first_point*** ***endpoint$_1$*** ***endpoint$_2$ ... endpoint$_n$***

The user can enter a piecewise uniform distribution using an alternative format by specifying the UNIFORM* keyword. The results of sampling this distribution are similar to those generated by the continuous linear distribution type. This distribution enables the user to specify a piecewise uniform distribution in the form of an *n*-part histogram. The first parameter for this keyword specifies the number of intervals *n* for which piecewise uniform distributions will be generated. Next, the user specifies a series of *n* integer values, *obs*. Each value obs_i represents the number of observations that LHS will draw from the i^{th} interval. Therefore, *the sum of all obs_i must be exactly equal to the number of observations LHS has been requested to produce during this particular execution.* If the user wishes to run a second calculation with a different number of observations, the values of obs_i must be adjusted so that they sum to the new number of observations requested. Furthermore, each value of obs_i must be greater than or equal to zero.

Following the *n* values of obs_i, the user must specify the endpoints of the *n* intervals that are to be sampled. These intervals are assumed to be contiguous, so the endpoints of the *n* intervals are specified by entering a *first_point* (the distribution minimum) followed by *n* interval *endpoint* values. The value of *first_point* must be less than that of all *endpoint* values, and the *endpoint* values must be specified in increasing order. Note that the subintervals do not have to be of equal width.

Example:
```
Item13 0.29 UNIFORM*  3     #
     302   693    5         #
  -1.0   1.0   7.0   8.3
```

This example produces three piecewise uniform distributions. The first uniform distribution will select exactly 302 samples from the interval (–1.0, 1.0). The second uniform distribution will select exactly 693 samples from the interval (1.0, 7.0). The third uniform distribution will select exactly 5 samples from the interval (7.0, 8.3). This example would only be valid for problems that request LHS to generate 1000 observations from all distributions. For other requested numbers of observations, the second line of the example input would have to be changed so that it sums to the new requested number of observations. Figure 4-11 shows the PDF of a piecewise uniform distribution corresponding to the example "Item13".

LOGUNIFORM* ***num*** ***obs_1*** ***obs_2 ... obs_n*** ***first_point*** ***$endpoint_1$*** ***$endpoint_2$ ... $endpoint_n$***

The keyword LOGUNIFORM* generates a series of piecewise continuous LOGUNIFORM distributions in exactly the same way that UNIFORM* generates a series of piecewise UNIFORM distributions. All of the restrictions from the UNIFORM* distribution apply to this distribution as well. Furthermore, for the LOGUNIFORM* distribution, *first_point* and all *endpoint* values must be positive.

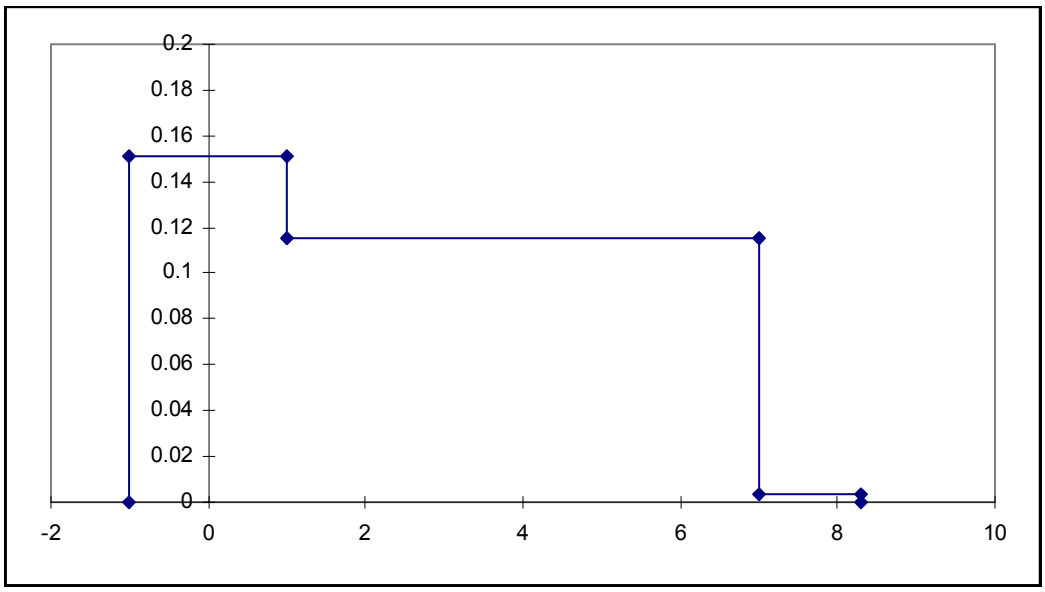

Figure 4-11: Piecewise Uniform PDF

```
Example:  Item28   4.7   LOGUNIFORM*   3#
               182    723     95         #
            0.4    1.0    7.0    14.6
```

This example produces three piecewise loguniform distributions. The first loguniform distribution will select exactly 182 samples from the interval (0.4, 1.0). The second loguniform distribution will select exactly 723 samples from the interval (1.0, 7.0). The third uniform distribution will select exactly 95 samples from the interval (7.0, 14.6). This example would only be valid for problems that request LHS to generate 1000 observations from all distributions. For other requested numbers of observations, the second line of the example input would have to be changed so that it sums to the new requested number of observations. Figure 4-12 shows the PDF of a piecewise loguniform distribution corresponding to "Item28".

4.1.5 Miscellaneous Continuous Distributions

LHS recognizes a number of additional distributions that may be needed for specialized applications. This section describes how the exponential distribution, the maximum entropy distribution, the Weibull distribution, the Pareto distribution, the gamma distribution, the beta distribution, the inverse Gaussian distribution, and the triangular distribution can be sampled using LHS.

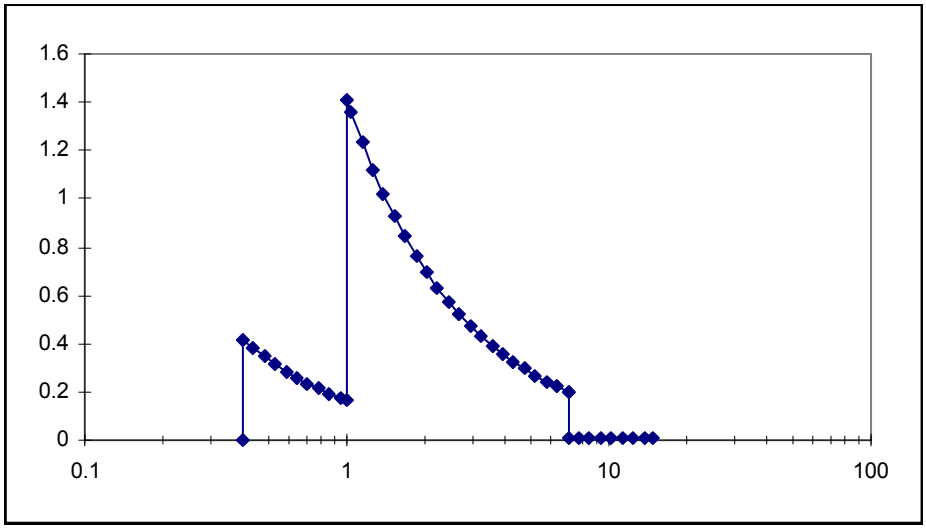

Figure 4-12: Piecewise Loguniform PDF

EXPONENTIAL λ

An exponential distribution is used to predict the time between events (such as radioactive decay), or a lifetime with a constant probability of failure. An exponential distribution is defined by the following probability density and cumulative distribution functions:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

$$F(x) = 1 - e^{-\lambda x}, \quad x \geq 0 .$$

The user must specify $\lambda > 0$.

Example: `Item28 0.36 EXPONENTIAL 2.0`

This example produces an exponential distribution named "Item28" with a point value of 0.36 and a distribution parameter $\lambda = 2.0$. Figure 4-13 shows the corresponding PDF of this exponential distribution.

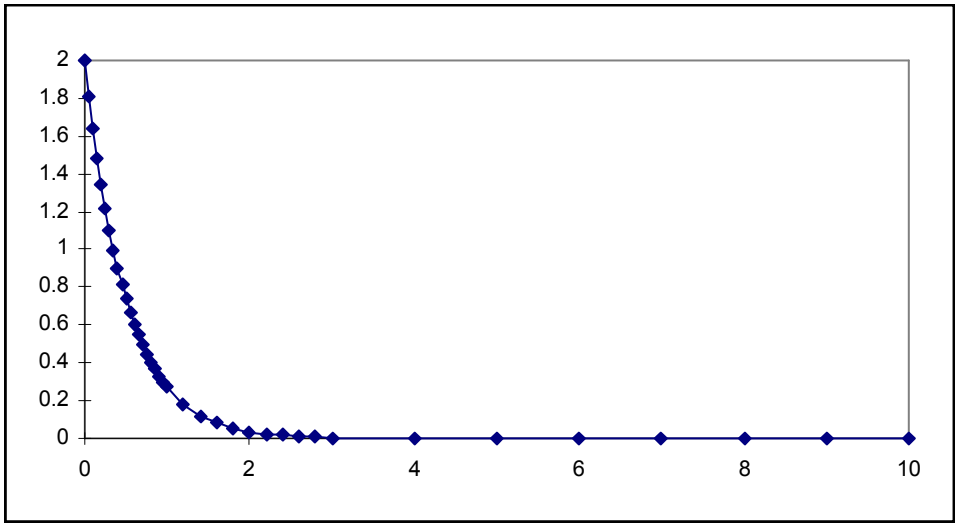

Figure 4-13: Exponential Distribution PDF ($\lambda = 2$)

MAXIMUM ENTROPY *A* μ *B*

The maximum entropy distribution implemented in LHS is a truncated exponential distribution, where A and B are the values of the distribution at its lower and upper bounds, respectively, and μ is the mean of the distribution. The user must specify $0 \leq A < \mu < B$. LHS then computes a distribution parameter λ for a bounded exponential distribution that provides the appropriate mean value.

Example: `Item33 0.25 MAXIMUM ENTROPY 0.0 1.0 3.0`

This example produces a truncated exponential distribution meeting maximum entropy criteria with a mean of 1.0, sampled between 0.0 and 3.0.

Some other types of distributions that satisfy maximum entropy criteria are implemented in LHS under other names. A maximum entropy distribution in which only the bounds (but no mean) are specified is equivalent to a uniform distribution. A maximum entropy distribution in which distribution bounds and quantiles are specified (but no mean) is equivalent to a cumulative linear distribution. The distribution form implemented with the MAXIMUM ENTROPY keyword is used when the analyst wishes to specify only the bounds and the mean of the distribution. Other maximum entropy distributions in which bounds, a mean, and a variance are specified, or in which bounds, quantiles, and a mean are specified, are not implemented in LHS. Figure 4-14 shows several PDFs for the maximum entropy distribution with A = 0.0, B = 3.0, and varying μ.

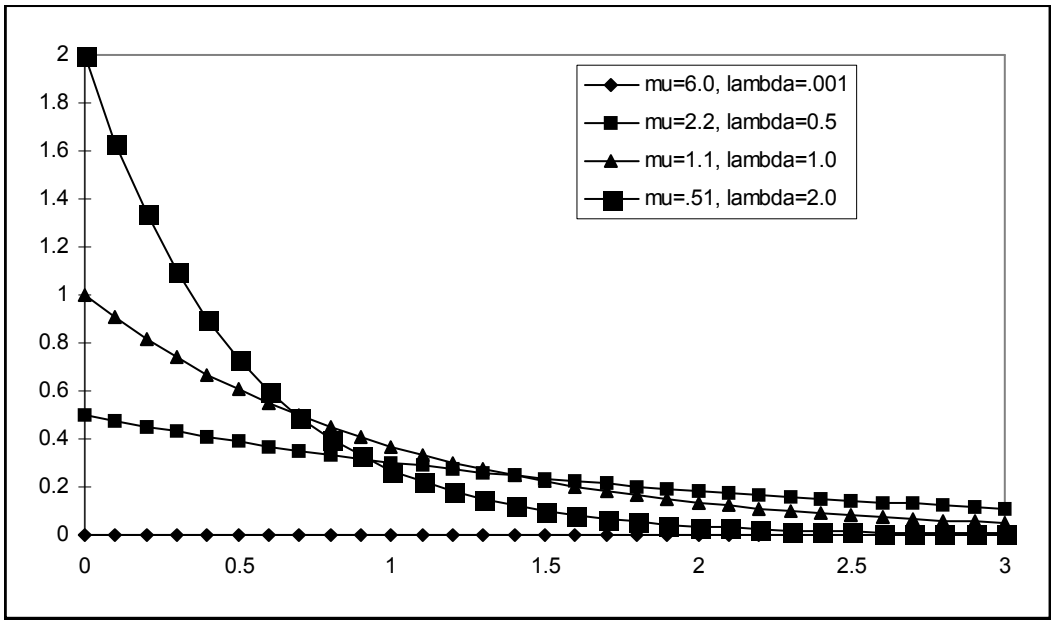

Figure 4-14: Maximum Entropy PDF (A = 0.0, B = 3.0, varying μ)

WEIBULL α β

A Weibull distribution is most commonly used in reliability studies to help predict the lifetime of a device. It is defined by the following density and cumulative distribution functions:

$$f(\omega) = \left(\frac{\alpha}{\beta}\right)\left(\frac{\omega}{\beta}\right)^{(\alpha-1)} e^{-\left(\frac{\omega}{\beta}\right)^{\alpha}}$$

$$F(\omega) = 1 - e^{-\left(\frac{\omega}{\beta}\right)^{\alpha}}, \quad 0 \le \omega \le \infty.$$

The user must specify α and β, both greater than zero.

Example: `Item29  0.35  WEIBULL  0.2  0.4`

This input produces a Weibull distribution named "Item29" with the distribution parameters α = 0.2 and β = 0.4. Figure 4-15 shows the PDF of the Weibull distribution corresponding to "Item29".

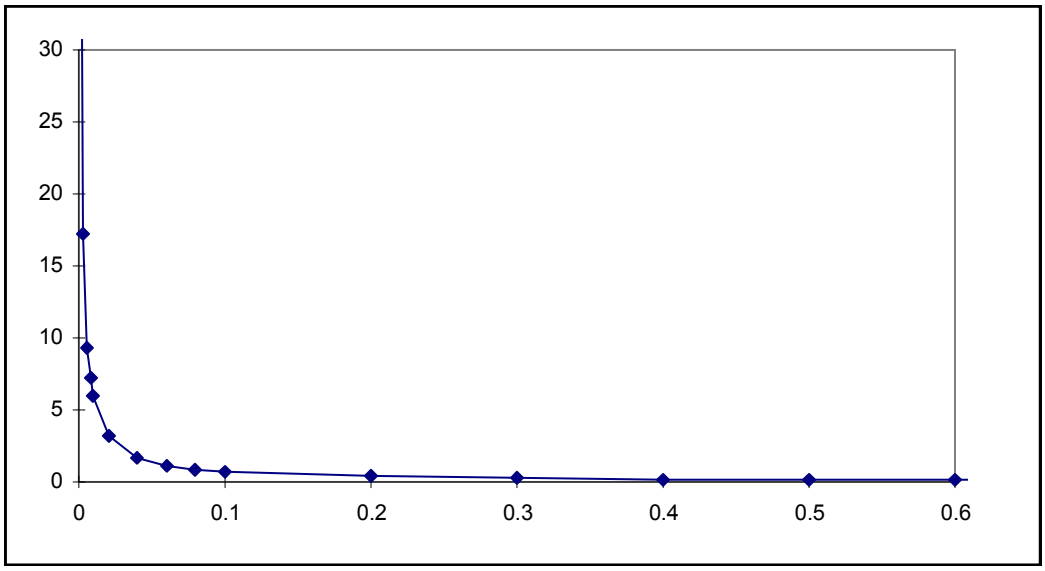

Figure 4-15: Weibull Distribution PDF ($\alpha = 0.2$, $\beta = 0.4$)

PARETO α β

A Pareto distribution is used to find a distribution of high numbers (such as middle/high incomes). It is defined by the following density and cumulative distribution functions:

$$p(x) = \frac{\alpha \beta^{\alpha}}{x^{\alpha+1}}$$

$$P(x) = 1 - \left(\frac{\beta}{x}\right)^{\alpha}, \quad \text{where} \quad \beta \le x \le \infty.$$

The user must specify $\alpha > 2.0$ and $\beta > 0.0$.

Example: `Item30 0.35 PARETO 2.4 0.5`

This example will produce a Pareto distribution named "Item30" with distribution parameters $\alpha = 2.4$ and $\beta = 0.5$. Figure 4-16 shows the PDF of the Pareto distribution corresponding to the "Item30" example.

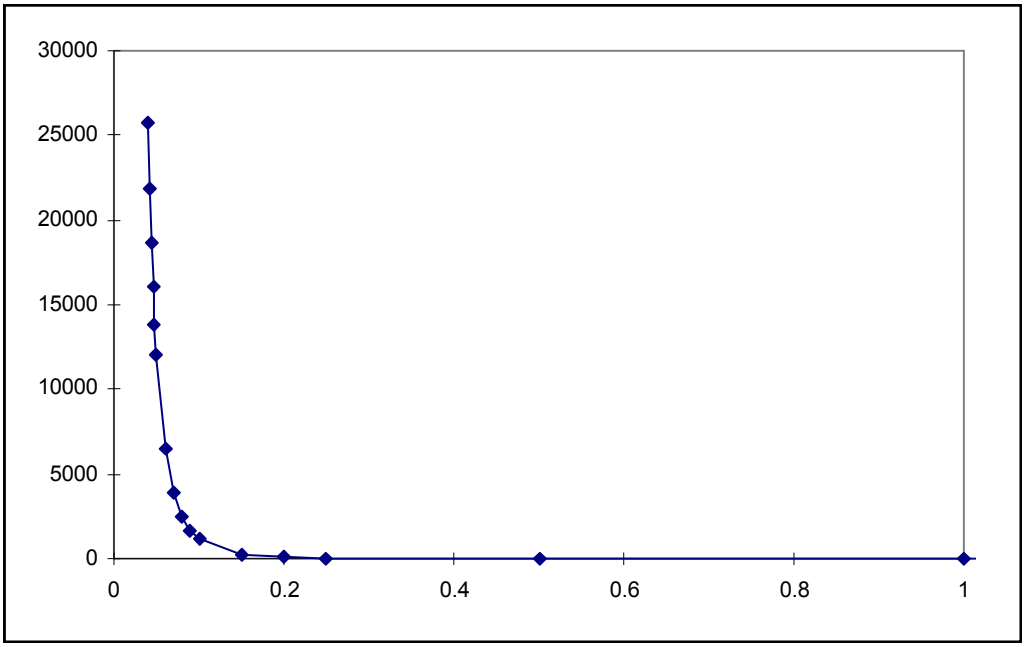

Figure 4-16: Pareto Distribution PDF ($\alpha = 2.4$, $\beta = 0.5$)

GAMMA α β

A gamma distribution is used to predict the time to finish a specific task. It has a density function defined by

$$f(x) = \frac{\beta^{\alpha} x^{(\alpha-1)} e^{-\beta x}}{\Gamma(\alpha)} \qquad \text{where} \qquad \Gamma(\alpha) = \int_0^{\infty} y^{(\alpha-1)} e^{-y}\, dy.$$

The user must specify α, the parameter of the gamma function, and β, a scaling factor, both of which are real numbers. The mean provided by this distribution is α/β. Note that the gamma distribution is found in the literature in two different forms. These forms differ only in the definition of β. In the other form, β is defined as the reciprocal of its definition in this form. Thus the other form of the gamma distribution results in a mean value of $\alpha\beta$. The user should be careful in selecting the parameters for a gamma distribution to ensure that the parameters entered are based on the definition of β used in this program.

Example: `Item31 0.35 GAMMA 2.0 3.0`

The preceding example will produce a gamma distribution named "Item31" in which the gamma function parameter is 2.0 and the result is scaled by 3.0. Figure 4-17 shows the PDF of a gamma distribution.

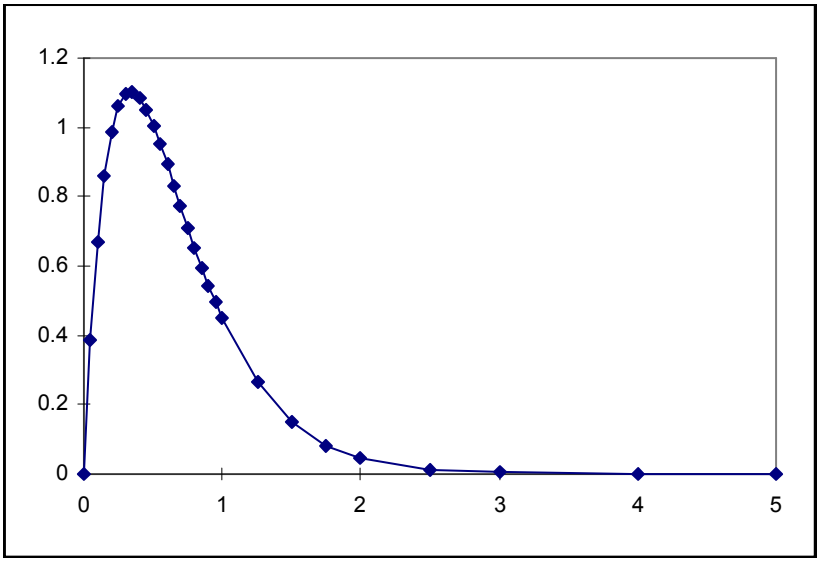

Figure 4-17: Gamma Distribution PDF ($\alpha = 2.0$, $\beta = 3.0$)

BETA ***A B p q***

A beta distribution is used as a rough model in the absence of sufficient data. The distribution is normalized to occur over a range from *A* to *B*, and is based on two shape factors, *p* and *q*. A beta distribution is defined with a density function of

$$f(\beta) = \frac{\beta}{\int_A^B \beta} \qquad \text{where} \qquad \beta = x^{(p-1)}(1-x)^{(q-1)},$$

p and *q* are shape parameters, and *A* and *B* are the endpoints of the distribution. The following conditions must be satisfied: $p, q \geq 0.001$, and $0 \leq A < B$. Figure 4-18 has been provided to help the user see the effect of various choices of *p* and *q*. As an example, this figure can be used to see the influence of changing *p* with *q* fixed at different values, or changing *q* with *p* fixed at different values, or letting $p = q$ as both increase.

Example: `Item18  0.2  BETA  2.0  4.0  2.0  2.0`

This example defines a beta distribution named "Item18" that will generate sample values between 2.0 and 4.0 based on the shape factors $p = 2.0$ and $q = 2.0$.

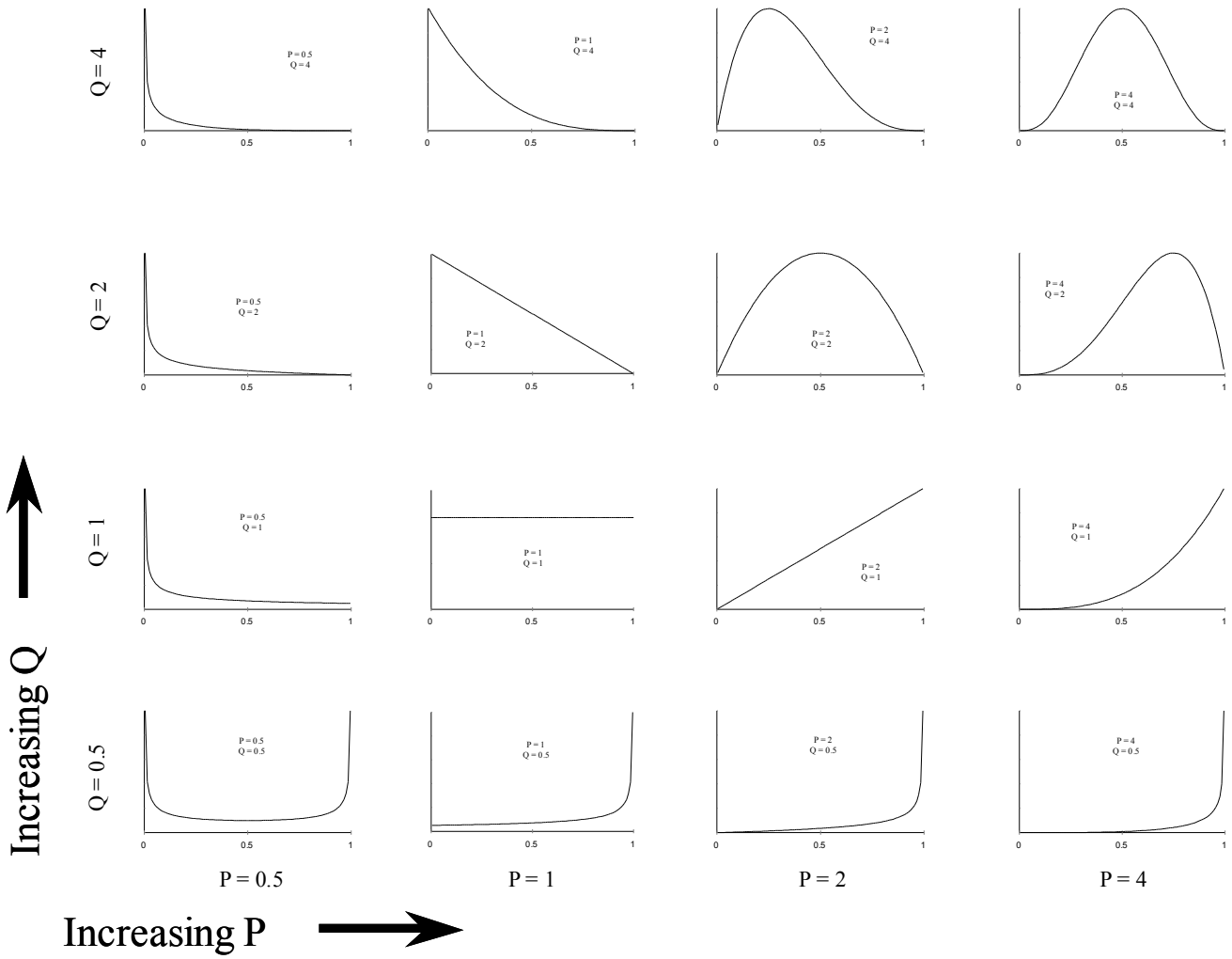

Figure 4-18: Effect of Varying P, Q for Beta Distribution

INVERSE GAUSSIAN μ λ

An inverse Gaussian distribution is characterized by the density function

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}}\, e^{-\left(\frac{\lambda\cdot(x-\mu)^2}{2\mu^2 x}\right)} .$$

The user must specify the distribution parameters μ and λ, both greater than zero.

Example: `Item32 0.35 INVERSE GAUSSIAN 0.01 0.3`

This example produces an inverse Gaussian distribution named "Item32" with a point value of 0.35 and distribution parameters μ = 0.01 and λ = 0.3. Figure 4-19 shows the PDF of an inverse Gaussian distribution.

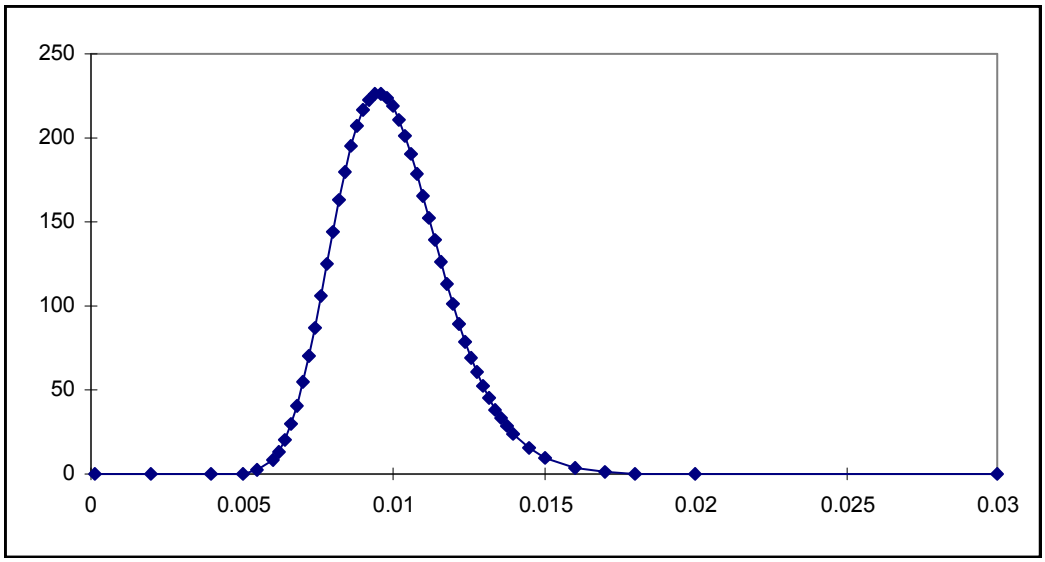

Figure 4-19: Inverse Gaussian PDF (μ = 0.01, λ = 0.3)

TRIANGULAR *a* *b* *c*

A triangular distribution can be used to model certain situations where specific data are lacking. The probability density function for this distribution is a triangle with a distribution minimum at *a*, a distribution maximum at *c*, and a most likely value at *b* (the apex of the triangle). In other words, all samples drawn from this distribution will come from the interval (*a*, *c*), and will be most densely populated in the area near *b*. No samples are drawn from the region where *x* is less than *a* or the region where *x* is greater than *c*. The values of the parameters must be such that $a < c$, and *b* must be located within the inclusive region [*a*, *c*]. In other words, *b* must satisfy the condition $a \le b \le c$.

For the most general case where $a < b < c$, the triangular distribution has the following probability density and cumulative distribution functions:

$$f(x) = \frac{2\,(x-a)}{(c-a)\,(b-a)} \qquad a \le x \le b$$

$$= \frac{2\,(c-x)}{(c-a)\,(c-b)} \qquad b \le x \le c$$

$$F(x) = \frac{(x-a)^2}{(c-a)\,(b-a)} \qquad a \le x \le b$$

$$= \frac{b-a}{c-a} - \frac{(x+b-2c)\,(x-b)}{(c-a)\,(c-b)} \qquad b \le x \le c\ .$$

In this general case, the mean, variance, and median of the distribution are

$$E(x) = \frac{a+b+c}{3}$$

$$V(x) = \frac{a\,(a-b) + b\,(b-c) + c\,(c-a)}{18}$$

$$\text{median} = a + \sqrt{\frac{(c-a)\,(b-a)}{2}} \qquad b \ge \frac{a+c}{2}$$

$$= c - \sqrt{\frac{(c-a)\,(c-b)}{2}} \qquad b \le \frac{a+c}{2}$$

The probability density function for this distribution is shown in Figure 4-20.

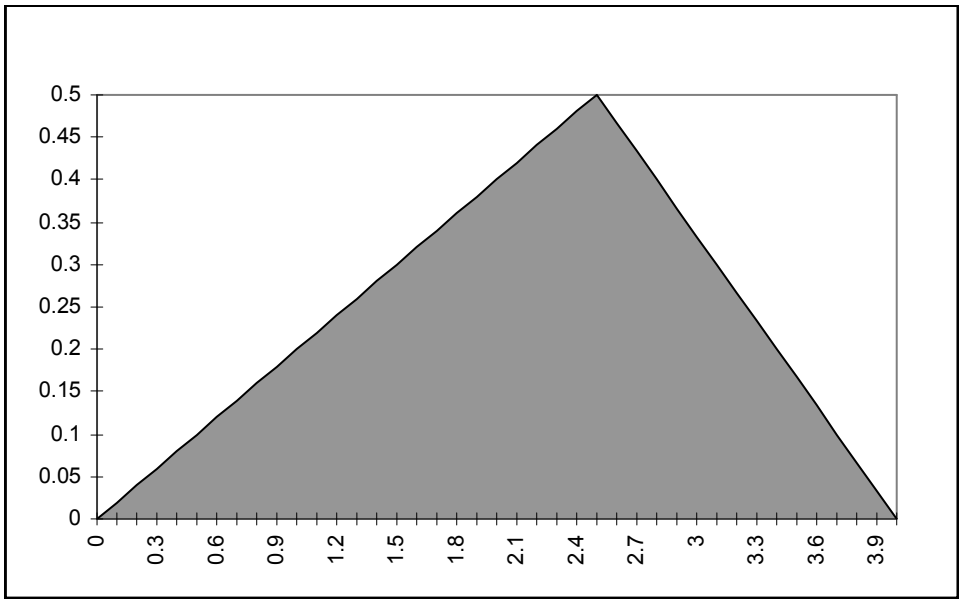

Figure 4-20: Triangular Distribution PDF ($a = 0.0$, $b = 2.5$, $c = 4.0$)

In the special case where $b = a$, the triangular distribution has the following probability density and cumulative distribution functions:

$$f(x) = \frac{2\,(c - x)}{(c - a)^2}$$

$$F(x) = \frac{(x - a)\,(2c - x - a)}{(c - a)^2} \qquad a \le x \le c$$

In this special case, the mean, variance, and median of the triangular distribution are

$$E(x) = \frac{2a + c}{3}$$

$$V(x) = \frac{(c - a)^2}{18}$$

$$\text{median} = c - \frac{c - a}{\sqrt{2}}.$$

The probability density function for this distribution is shown in Figure 4-21.

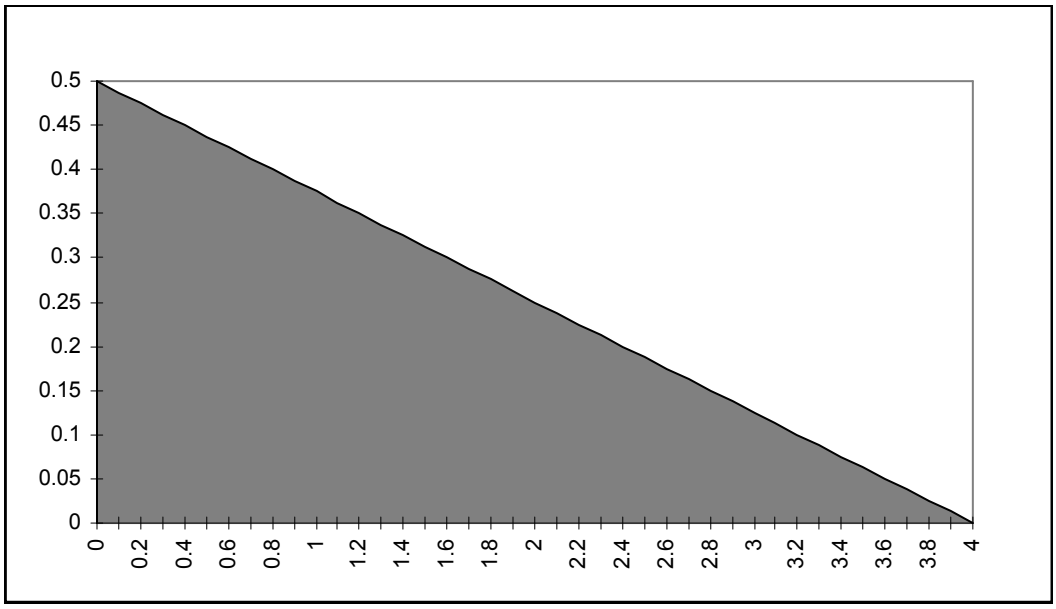

Figure 4-21: Triangular Distribution PDF ($a = b = 0.0$, $c = 4.0$)

Finally, in the special case where $b = c$, the triangular distribution has the following probability density and cumulative distribution functions:

$$f(x) = \frac{2\,(x - a)}{(c - a)^2}$$

$$F(x) = \frac{(x - a)^2}{(c - a)^2} \quad a \le x \le c$$

In this special case, the mean, variance, and median of the triangular distribution are

$$E(x) = \frac{a + 2c}{3}$$

$$V(x) = \frac{(c - a)^2}{18}$$

$$\text{median} = a - \frac{c - a}{\sqrt{2}}.$$

The probability density function for this distribution is shown in Figure 4-22.

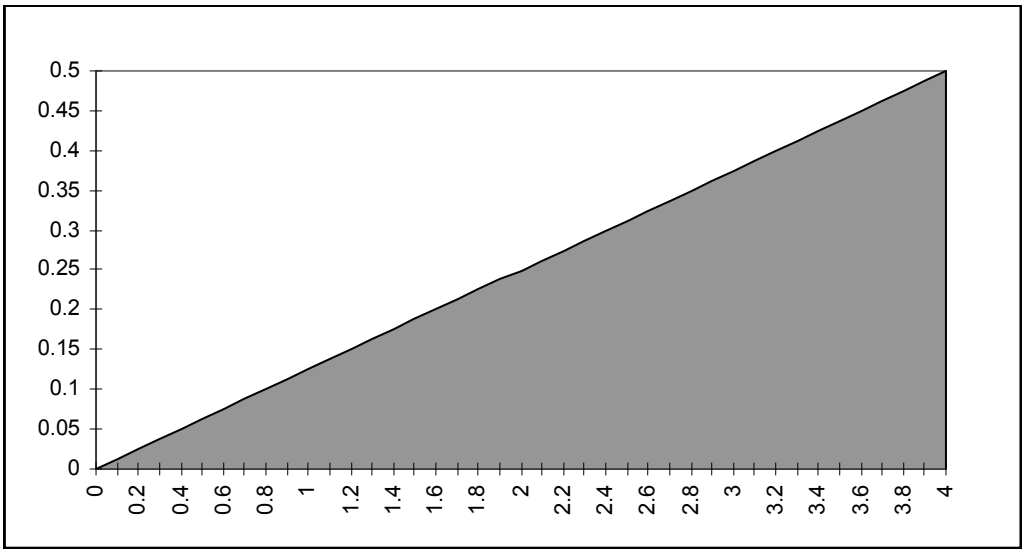

Figure 4-22: Triangular Distribution PDF ($a = 0.0,\ b = c = 4.0$)

The following three examples illustrate these three cases of triangular distribution.

General case example: `Item15 2.2 TRIANGULAR 0.0 2.5 4.0`

Special case $b = a$ example: `Item16 0.37 TRIANGULAR 0.0 0.0 4.0`

Special case $b = c$ example: `Item17 3.24 TRIANGULAR 0.0 4.0 4.0`

Each of these examples defines a triangular distribution over the interval (0.0, 4.0). The first example has a most likely value of 2.5, which is within the interval, while the second example has a most likely value equal to the distribution minimum, and the third example has a most likely value equal to the distribution maximum.

4.2 Discrete Distributions Recognized by LHS

The keywords described in this section are used to specify which discrete distributions are to be sampled by the LHS program. Users may select from among five traditional discrete distribution functions and two empirical discrete distribution functions in order to achieve the best possible representation of their data. The traditional and empirical discrete distributions are described in the following two sections.

4.2.1 Common Discrete Distributions

LHS recognizes a number of common discrete distributions that an analyst may find useful for statistical modeling applications. This section describes how the Poisson, binomial, negative binomial, geometric, and hypergeometric distributions can be sampled using LHS.

POISSON λ

A Poisson distribution is used to help predict the number of discrete events that happen in a given time interval. The random variable x has a Poisson distribution if it has a density function of

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad x=0, 1, 2, 3, \ldots$$

The user must specify a positive real value for the frequency λ. This is a discrete distribution that returns real number representations of sampled integer values. The code execution speed decreases significantly as the input parameter λ becomes large.

Example: `Item24  2.0  POISSON  3.0`

In this example, "Item24" specifies a Poisson distribution with a frequency of 3.0 and an input point estimate value of 2.0. Figure 4-23 shows the PDF for two sample Poisson distributions.

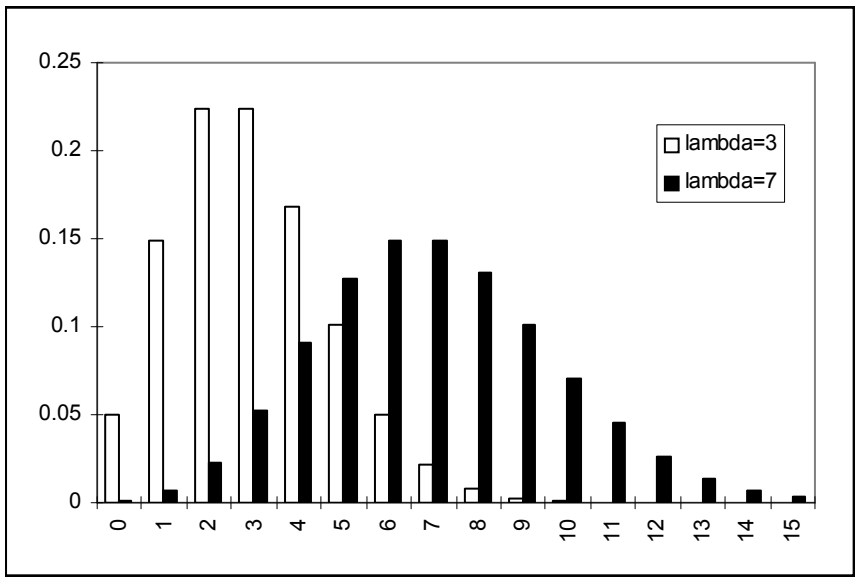

Figure 4-23: Poisson PDF with Varying λ

BINOMIAL ***p n***

A binomial distribution is used to predict the number of failures (or defective items) in a total of n tests (or a batch of n products) with a probability p of being a failure (or the probability of being defective). A binomial distribution is defined by the density function

$$f(y) = \binom{n}{y} p^{y} \left(1 - p\right)^{(n-y)} \qquad y=0, 1, 2, \ldots$$

The user must specify p, the failure probability ($0 < p < 1$), and n, the number of tests ($n > 1$). This is a discrete distribution that returns real number representations of sampled integer values. The code execution speed decreases as the integer input parameter n becomes large.

Example: `Item26 0.34 BINOMIAL 0.45 50`

This example specifies a binomial distribution in which each observation of "Item26" contains the number of failures in a hypothetical series of 50 tests, where each unit has a failure probability of 45%. Figure 4-24 shows the PDF of a binomial distribution.

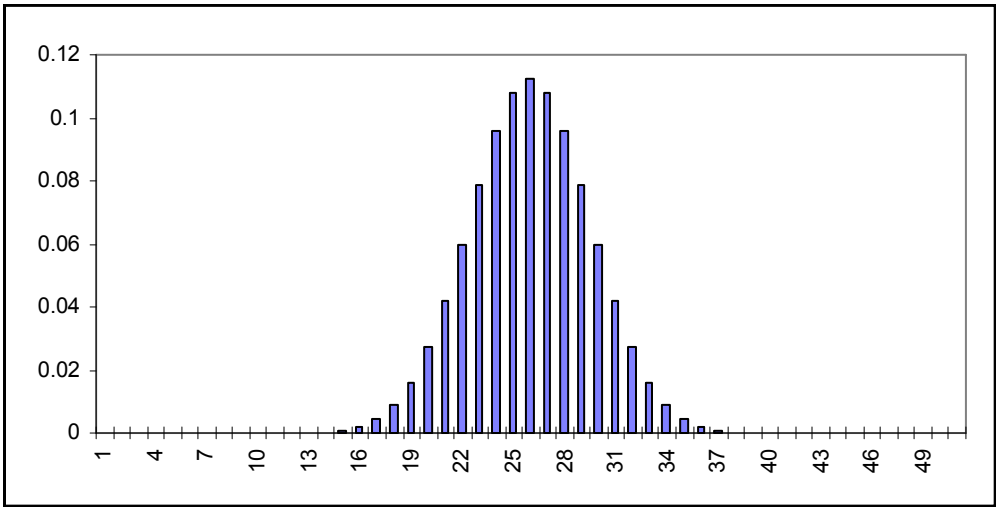

Figure 4-24: Binomial Distribution PDF ($p = 0.45$, $n = 50$)

NEGATIVE BINOMIAL *p n*

A negative binomial distribution is used to find the number of times to perform a test to have n successes, with a probability of success of p. A negative binomial distribution is defined by the density function

$$f(x) = \binom{n + x - 1}{x} p^{n} \left(1 - p\right)^{x}$$

The user must specify the probability of success p ($0 < p < 1$) and a number of tests n ($n > 1$). This is a discrete distribution that returns real number representations of sampled integer values.

Example: `Item27 67 NEGATIVE BINOMIAL 0.5 100`

In this example, the variable "Item27" is represented by a negative binomial distribution in which each observation contains the number of trials needed to achieve 100 successful trials with a probability of 50% to achieve success. Figure 4-25 shows the PDF of a negative binomial distribution.

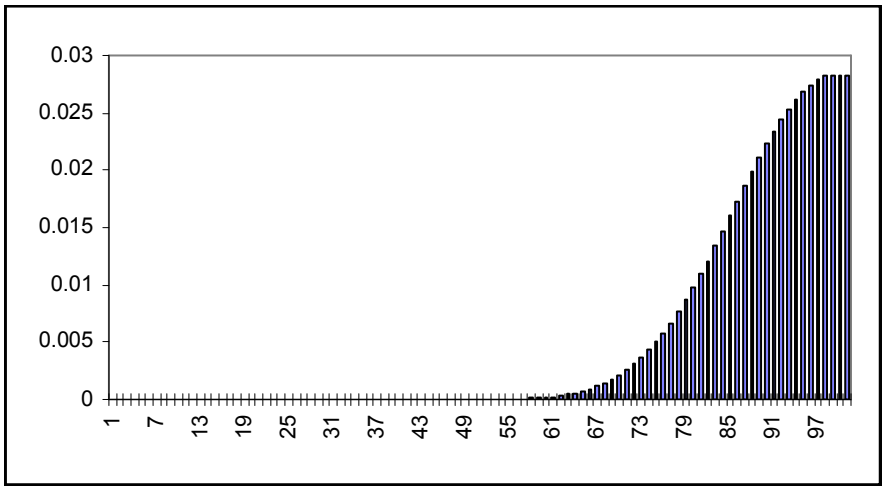

Figure 4-25: Negative Binomial PDF (p = 0. 5, n = 100)

GEOMETRIC *p*

A geometric distribution represents the number of successful trials that might be observed before a failure occurs. This is a discrete distribution that is defined by the density function

$$f(y) = (1 - p)^y \, p \qquad y=0, 1, 2, \ldots$$

This is a discrete distribution that returns real number representations of sampled integer values. The user must specify p, the probability ($0 < p < 1$) of success. The code execution speed decreases as the input parameter p becomes small.

Example: `Item25 0.43 GEOMETRIC 0.67`

This example produces a geometric distribution named "Item25" that represents the number of successful trials until a failure with a 67% chance of success. Figure 4-26 shows the PDF of a geometric distribution.

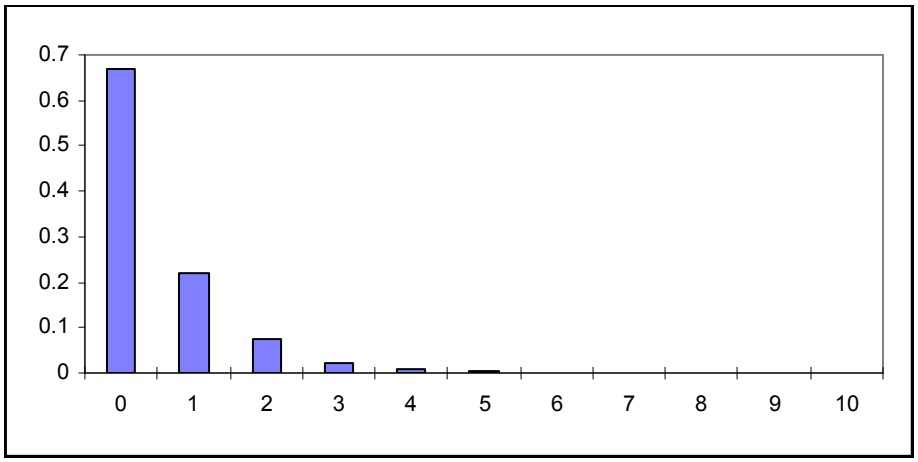

Figure 4-26: Geometric Distribution PDF ($p = 0.67$)

HYPERGEOMETRIC N_N N_1 N_R

A hypergeometric distribution is used to define the number of failures in a set of tests that has a known proportion of failures. A random variable x is said to have a hypergeometric distribution if it satisfies the following density function:

$$f(x) = \frac{\binom{N_1}{x}\binom{N_N - N_1}{N_R - x}}{\binom{N_N}{N_R}}$$

where N_N is the total number of tests, N_1 is the number of items in the subpopulation (number of tests that failed), and N_R is the number of selections. Note that $N_1 < N_R < N_N$. This is a discrete distribution that returns real representations of sampled integer values.

Example: `Item27 0.34 HYPERGEOMETRIC 110 30 45`

In this example, each observation of the variable "Item27" will contain the number of failed tests that were found in a randomly picked batch of 30 out of 110 total tests, where there are 45 total failed tests. Figure 4-27 shows the PDF of a hypergeometric distribution.

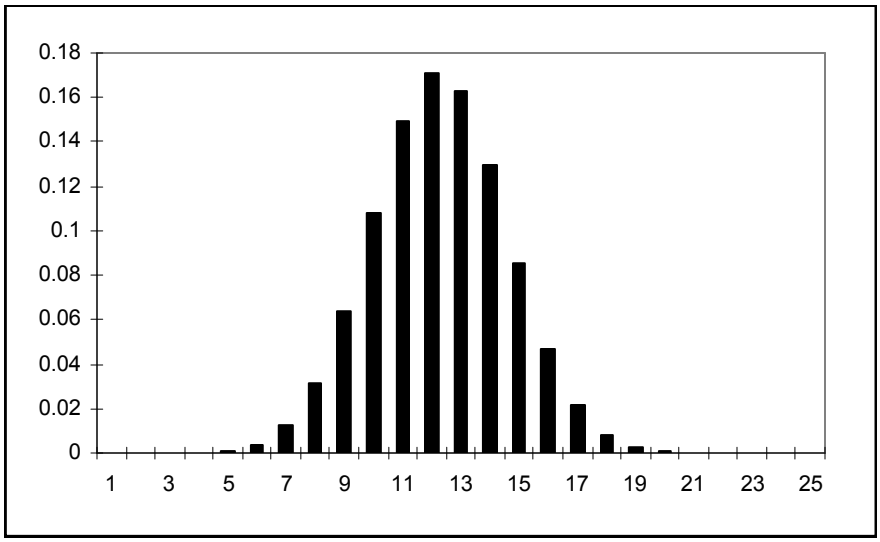

Figure 4-27: Hypergeometric Distribution PDF ($N_N = 110$, $N_I = 30$, $N_R = 45$)

4.2.2 User-Defined Discrete Distributions

The user-defined discrete distributions implemented within LHS provide an opportunity to specify the points of a discrete probability distribution function as either a cumulative distribution function or as a histogram (an unnormalized discrete density function). The discrete histogram function is converted internally by LHS into a discrete cumulative distribution function prior to sampling. The input format for these distributions is similar to their continuous counterparts described in Section 4.2.4.

DISCRETE CUMULATIVE ***n ordered_pair ordered_pair ordered_pair ...***

A discrete cumulative distribution is used to represent an arbitrary empirical distribution for which a limited number of discrete outcomes may occur. The user enters the CDF for this arbitrary distribution as a series of ordered pairs. The user must first specify *n*, an integer number of ordered pairs that will be used to represent the CDF ($n > 1$). The *n*-ordered pairs that represent the CDF follow, forming the remainder of the input for this distribution. Within each ordered pair, the first number is the value of the distribution; the second number is the cumulative probability (CDF value) associated with this distribution value. *The probabilities in the ordered pairs (the second entry in the ordered pair) must increase monotonically starting with a value greater than 0.0 and ending with 1.0.* The distribution values (the first entry in the ordered pair) must also increase monotonically. LHS uses these points to generate a discrete distribution in which the probability *density* associated with a particular value is the difference between its cumulative probability and that of its immediate predecessor (or, for the first value, the difference between its cumulative probability and zero).

```
Example:  Item22 7.5 DISCRETE CUMULATIVE          3 #
                      5.0 0.33333  #
                      7.0 0.66667  #
                     10.0 1.0
```

In this example, one-third of the observations for the random variable named "Item22" will be set to exactly 5.0, another third to exactly 7.0, and the remainder to exactly 10.0. The PDF for this distribution is shown in Figure 4-28.

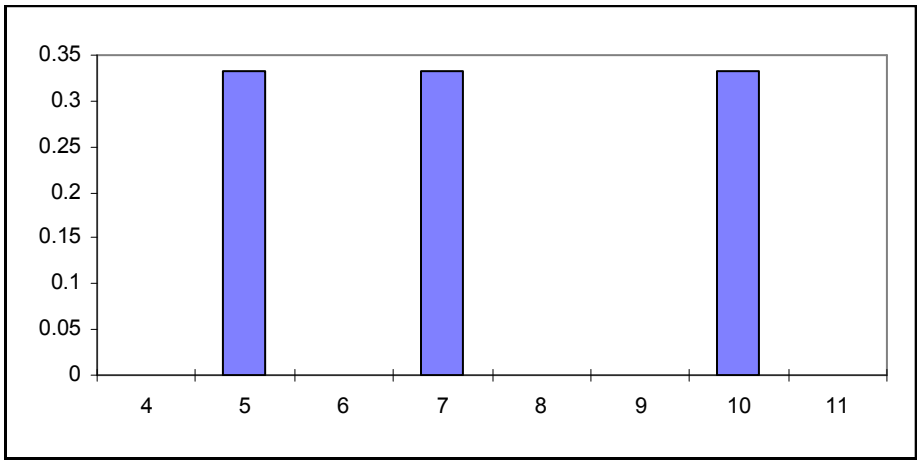

Figure 4-28: Discrete Cumulative PDF

DISCRETE HISTOGRAM ***n ordered_pair*** $_1$ ***ordered_pair*** $_2$ ***... ordered_pair*** $_n$

The keyword DISCRETE HISTOGRAM provides an alternative method for the user to specify an empirical discrete distribution. It is similar in format to the discrete cumulative distribution described previously, in that the user specifies the number of ordered pairs to be read, followed by a series of ordered pairs. Once again, the first value in each ordered pair is the value of the distribution at a particular point. However, instead of specifying the cumulative probability associated with that value as the second item in the ordered pair, as was done for the discrete cumulative distribution, the user specifies the relative frequency associated with the value (or the length of the "bar" that would be associated with that value on a histogram graph). *The frequencies in the ordered pairs are relative only to each other, and must be positive.* The frequencies need not increase monotonically, although values must still increase monotonically.

```
Example:  Item23   7.5   DISCRETE HISTOGRAM      3   #
                           5.0   17.0   #
                           7.0   17.0   #
                          10.0   17.0
```

This example generates a discrete distribution named "Item23" that is equivalent to the distribution generated in the example for the discrete cumulative distribution in that one-third of the observations for this random variable will be set to exactly 5.0, another third to exactly 7.0, and the

remainder to exactly 10.0. This occurs because all three discrete values are associated with the same relative frequency (the value 17.0). LHS internally converts this histogram into a discrete continuous distribution prior to sampling. This distribution is shown in Figure 4-29.

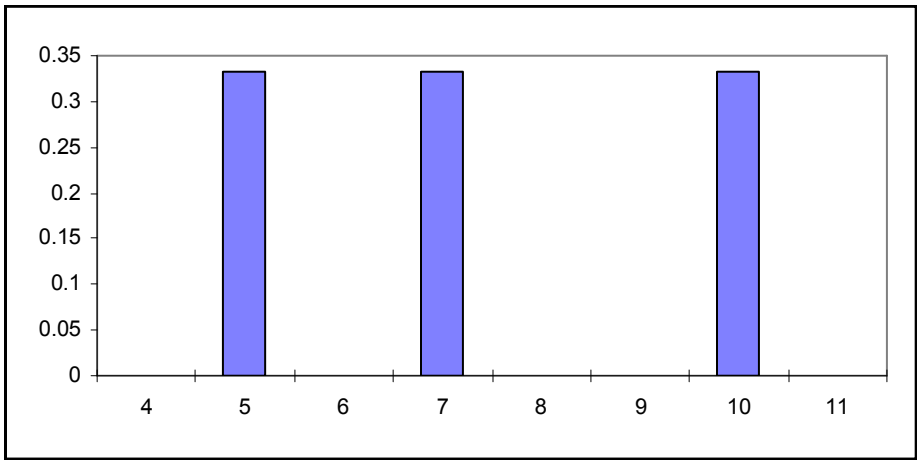

Figure 4-29: Discrete Histogram (Discrete Continuous) PDF

5. LHS and DAKOTA

This section outlines the use of LHS from within the DAKOTA software. DAKOTA (Design Analysis Toolkit for Optimization and Terascale Applications) is a sophisticated toolkit of iteration methods that "wrap around" engineering simulation models: optimization solvers, design of experiments, and sampling techniques. LHS is just one of the methods available from within DAKOTA. More information about DAKOTA can be found at: http://endo.sandia.gov/DAKOTA or in the set of DAKOTA manuals (User's Manual, Reference Manual, and Developers Manual (2001)).

The use of LHS from within DAKOTA is an example of using DAKOTA as a callable library. Section 5.1 describes the interface to LHS from within DAKOTA, and Section 5.2 has a description of the input-by-call format for people wanting to use LHS as a library from within their software.

5.1 DAKOTA implementation of LHS Library

To run LHS from within DAKOTA, one needs to specify some commands from within the DAKOTA input file. Specifically, DAKOTA is run with the command: dakota –i *InputFileName*. The input file name has many sections which are documented in the User's Manual. The use of LHS requires two things in the DAKOTA input file: the specification of LHS under the "methods" section, and the specification of uncertain variable(s) in the "variables" section.

5.1.1 Specification of LHS as a Nondeterministic Method

This document will not list the complete specification of the DAKOTA input file; that is done in the DAKOTA Reference Manual. However, the interface to LHS is done through the nondeterministic sampling method. This method has the following options:

```
( {nond_sampling}                                    \
            [seed = <INTEGER>] [fixed_seed] [samples = <INTEGER>] \
            [ {sample_type} {random} | {lhs} ] [all_variables]    \
            [ {distribution} {cumulative} | {complementary} ]     \
            [ {response_levels = <LISTof><REAL>}                  \
              [num_response_levels = <LISTof><INTEGER>]           \
              [ {compute} {probabilities} | {reliabilities} ] ]   \
            [ {probability_levels = <LISTof><REAL>}               \
              [num_probability_levels = <LISTof><INTEGER>] ]      \
            [ {reliability_levels = <LISTof><REAL>}               \
              [num_reliability_levels = <LISTof><INTEGER>] ] )    \
```

The way to read this specification is as follows: individual specifications (such as the nond_sampling method) are enclosed in {}, optional individual specifications are enclosed in [], and either/or relationships are denoted by the | symbol. Thus, a rather minimal set of commands to perform LHS sampling in DAKOTA might look like this:

```
method,
      nond_sampling
      sample_type lhs
      seed = 54997
      samples =100
```

The nondeterministic sampling method in DAKOTA has two mutually exclusive sampling options: random sampling or LHS sampling. Since the nondeterministic sampling method has been designed for the analysis of engineering reliability problems, the user can specify if he or she wants to see the output as a cumulative distribution function (CDF) or as a complementary cumulative distribution function (CCDF). Also, the last three main options in this specification have to do with the type of output. For example, if one wants to know the probability or reliability calculated at particular response levels of a code, then one can specify:

```
      response_levels 3.5 5.2 8.9
```

For example, if we ran a code with 100 LHS input samples, the above statement would calculate the probability (from the 100 outputs corresponding to the 100 input samples) of being less than or equal to 3.5, of being less than or equal to 5.2, and of being less than or equal to 8.9 assuming CDF was specified. If CCDF was specified, the probabilities would be greater than 3.5, 5.2, and 8.9. Likewise, one could specify the following probability levels:

```
      probability_levels .25 .5 .75 1
```

to obtain the response output at the four quartiles of the response output distribution, for example.

Note that there are several things within the standalone version of LHS that cannot be specified from within the DAKOTA version. The pairing option is one: the LHS default in DAKOTA always uses restricted pairing (not random pairing.) Not all distributions are supported (see the variable specification below). Also, the user cannot run replicated LHS samples in the current version. However, these things require minor changes to the DAKOTA interface that can be made if there is user demand for these features.

5.1.2 Specification of Uncertain Variables

The variables that the user wants to sample from and then pass these samples to an engineering model within the DAKOTA architecture are specified within the variables section of the DAKOTA input file. The specification of uncertain variables looks like this:

```
      [ {normal_uncertain = <INTEGER>}                    \
            {nuv_means = <LISTof><REAL>}                  \
            {nuv_std_deviations = <LISTof><REAL>}         \
            [nuv_dist_lower_bounds = <LISTof><REAL>]      \
            [nuv_dist_upper_bounds = <LISTof><REAL>]      \
            [nuv_descriptors = <LISTof><STRING>] ]        \
      [ {lognormal_uncertain = <INTEGER>}                 \
            {lnuv_means = <LISTof><REAL>}                 \
            {lnuv_std_deviations = <LISTof><REAL>}              \
```

```
      | {lnuv_error_factors = <LISTof><REAL>}                \
      [lnuv_dist_lower_bounds = <LISTof><REAL>]              \
      [lnuv_dist_upper_bounds = <LISTof><REAL>]              \
      [lnuv_descriptors = <LISTof><STRING>] ]                \
[ {uniform_uncertain = <INTEGER>}                            \
      {uuv_dist_lower_bounds = <LISTof><REAL>}               \
      {uuv_dist_upper_bounds = <LISTof><REAL>}               \
      [uuv_descriptors = <LISTof><STRING>] ]                 \
[ {loguniform_uncertain = <INTEGER>}                         \
      {luuv_dist_lower_bounds = <LISTof><REAL>}              \
      {luuv_dist_upper_bounds = <LISTof><REAL>}              \
      [luuv_descriptors = <LISTof><STRING>] ]                \
[ {weibull_uncertain = <INTEGER>}                            \
      {wuv_alphas = <LISTof><REAL>}                          \
      {wuv_betas = <LISTof><REAL>}                           \
      [wuv_dist_lower_bounds = <LISTof><REAL>]               \
      [wuv_dist_upper_bounds = <LISTof><REAL>]               \
      [wuv_descriptors = <LISTof><STRING>] ]                 \
[ {histogram_uncertain = <INTEGER>}                          \
      [ {huv_num_bin_pairs = <LISTof><INTEGER>}              \
        {huv_bin_pairs = <LISTof><REAL>} ]                   \
      [ {huv_num_point_pairs = <LISTof><INTEGER>}            \
        {huv_point_pairs = <LISTof><REAL>} ]                 \
      [huv_descriptors = <LISTof><STRING>] ]                 \
[uncertain_correlation_matrix = <LISTof><REAL>]              \
```

Thus, one can see that the DAKOTA implementation allows six distributions: normal, lognormal, uniform, loguniform, weibull, and histogram. Each uncertain variable specification contains descriptive tags and distribution lower and upper bounds. Distribution lower and upper bounds are explicit portions of the normal, lognormal, uniform, loguniform, and weibull specifications, whereas they are implicitly defined for histogram variables from the extreme values within the bin/point pairs specifications. In addition to tags and bounds specifications, normal variables include mean and standard deviation specifications, lognormal variables include mean and either standard deviation or error factor specifications, weibull variables include alpha and beta specifications, and histogram variables include bin pairs and point pairs specifications.

The inclusion of lower and upper distribution bounds for all uncertain variable types (either explicitly or implicitly) allows the use of these variables with methods that rely on a bounded region to define a set of function evaluations (i.e., design of experiments and some parameter study methods). In addition, distribution bounds can be used to truncate the tails of distributions for normal and lognormal uncertain variables (see "bounded normal", "bounded lognormal", and "bounded lognormal-n" distribution types in Section 4.)

These six distribution types are the distributions most commonly used in DAKOTA's engineering and reliability applications, however, we can expose other distribution types available in the standalone version of LHS depending on user demand for these distributions.

Below is an example of the variables section of the DAKOTA input file specifying uncertain variables:

```
variables,                                                          \
      normal_uncertain = 2                                          \
        nuv_means              =  248.89, 593.33                    \
        nuv_std_deviations     =   12.4,   29.7                     \
        nuv_descriptor         =  'TF1n'  'TF2n'                    \
      uniform_uncertain = 2                                         \
        uuv_dist_lower_bounds =  199.3,  474.63                     \
        uuv_dist_upper_bounds =  298.5,  712.0                      \
        uuv_descriptor         =  'TF1u'  'TF2u'                    \
      weibull_uncertain = 2                                         \
        wuv_alphas             =   12.,    30.                      \
        wuv_betas              =  250.,   590.                      \
        wuv_descriptor         =  'TF1w'  'TF2w'                    \
      histogram_uncertain = 3                                       \
        huv_num_bin_pairs   = 3 4                                   \
        huv_bin_pairs       = 5 17 8 21 10 0 .1 12 .2 24 .3 12 .4 0 \
        huv_num_point_pair  = 2                                     \
        huv_point_pairs     = 3 1 4 1                               \
        huv_descriptor      =  'TF1h'  'TF2h'  'TF3h'
```

According to this input file, there are two normal uncertain variables, two uniform, two weibull, and three histogram uncertain variables. For the histogram uncertain variables, two have "bin pairs" (three and four, respectively), and one has a point pair. For the histogram uncertain variable specification, the bin pairs and point pairs specifications provide sets of (x,y) pairs for each histogram variable. The distinction between the two types is that while the former specifies counts for bins of non-zero width, the latter specifies counts for individual point values (which can be thought of as bins with zero width.) In the terminology of LHS, the former is a "continuous linear histogram" and the latter is a "discrete histogram." To fully specify a bin-based histogram with *n* bins where the bins can be of unequal width, *n+1* (x,y) pairs must be specified. For example, the first histogram variable has three bin pairs: (5, 17), (8,21), and (10,0). This variable has two bins: one between 5 and 8 with a count of 17, and one between 8 and 10 with a count of 21. The last bin pair must have a 0 as the y-value to specify the end of the last bin. The histogram with point pair values simply has two values: 3 or 4, each with 1 count so each value will be sampled equally.

5.2 LHS as a callable library on Linux/UNIX platform

The LHS UNIX Library/Standalone can be run as a callable library on a Linux or UNIX platform. The interface specifications are given in Table 5-1. For more complete details about the meaning of some of these passed parameters, look at Section 3 of this manual and the source code.

Table 5-1. LHS Input-by-Call Routines

1. **or**	**LHS_INIT**(LHSOBS,LHSSEED,IError) **LHS_INIT_MEM**(LHSOBS,LHSSEED,LNMAX,LMAXNNV,LNVAR, NINTMX,LNCVAR,LMAXTB,LIPrint,LISamW,IError)	**Required**
2.	**LHS_OPTIONS**(LHSREPS,LHSPVAL,LHSOPTS,IError)	Optional
3.	**LHS_FILES**(LHSOUT,LHSMSG,LHSTITL,FLOPTS,IError)	Optional
4.	**LHS_DIST**(NAMVAR,IPTFLAG,PTVAL,DISTYPE,APRAMS, NUMPRMS,IError,IDISTNO,IPVNO)	**See Note**
5.	**LHS_SDIST**(NAMVAR,IPTFLAG,PTVAL,DISTYPE, NUMINTV, NOBSpI,ENDPNTS,IError,IDISTNO,IPVNO)	**See Note**
6.	**LHS_UDIST**(NAMVAR,IPTFLAG,PTVAL,DISTYPE, NUMPTS,XVAL,YVAL,IError,IDISTNO,IPVNO)	**See Note**
7.	**LHS_CONST**(NAMVAR,PTVAL,IError,IPVNO)	Optional
8.	**LHS_SAMEAS**(NEWNAM,ORGNAM,IError,IDISTREF,IPVNO)	Optional
9.	**LHS_CORR**(NAM1,NAM2,CORRVAL,IError)	Optional
10.	**LHS_PREP**(IError,NUMNAM,NUMVAR)	**Required**
11.	**LHS_RUN**(MAXVAR,MAXOBS,MAXNAM,IError, LSTDNAM, INDXNAM,PTVALST,NUMNAM,SMATX,NUMVAR)	**Required**
12.	**LHS_COROUT**(MAXNUMC,MAXVAR,IError,C1MATX,C2MATX, NumCor,NumVar)	Optional
13.	**LHS_RtvSEED**(Ierror,LastSeed) Optional	
14.	**LHS_CLOSE**(IError)	Optional*

Note: At least one call must be made to LHS_DIST, LHS_UDIST, or LHS_SDIST.
*Although not required, please call LHS_CLOSE as it makes for much more efficient use of the computer resources.

6. Summary

The objective of this document has been to describe both the theory and use of the Latin hypercube sampling software in the LHS UNIX Library/Standalone version. The theory of Latin hypercube sampling was presented, along with detailed descriptions of how to use LHS in a standalone mode on a Linux/UNIX operating system. The distribution types allowed in LHS UNIX Library/Standalone were described in detail. The DAKOTA implementation of LHS was also described.

LHS has been a powerful tool for sampling statistical distributions in uncertainty analyses for more than 20 years. The LHS UNIX Library/Standalone version represents an investment to modernize the code capabilities and allow this valuable uncertainty analysis capability to remain viable for large-scale simulation models running under a Linux or UNIX operation system.

7. References

Eldred, M.S., Giunta, A. A., van Bloeman Waanders, B.G., Wojtkiewicz, S.F., Hart, W.E., and Alleva, M.P. (2001). "DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis." Technical Reports SAND2001-3796 (User's Manual), SAND2001-3515 (Reference Manual), and SAND2001-3514 (Developer's Manual). Sandia National Laboratories, Albuquerque NM.

Helton, J. C. and Davis, F. J. (2001). "Latin Hypercube Sampling and the Propagation of Uncertainty in Analyses of Complex Systems." Technical Report SAND2001-0417, Sandia National Laboratories, Albuquerque, NM.

Iman, R.L., and Conover, W.J. (1982a). "A Distribution-Free Approach to Inducing Rank Correlation Among Input Variables," *Communications in Statistics*, B11(3), 311-334.

Iman, R.L., and Conover, W.J. (1982b). "Sensitivity Analysis Techniques: Self-Teaching Curriculum," Nuclear Regulatory Commission Report, NUREG/CR-2350, Technical Report SAND81-1978, Sandia National Laboratories, Albuquerque, NM.

Iman, R.L., and Davenport, J.M. (1982). "An Iterative Algorithm to Produce a Positive Definite Correlation Matrix from an Approximate Correlation Matrix (with a Program User's Guide)," Technical Report SAND81-1376, Sandia National Laboratories, Albuquerque, NM.

Iman, R.L., and Shortencarier, M.J. (1984). "A Fortran 77 Program and User's Guide for the Generation of Latin Hypercube and Random Samples for Use with Computer Models," NUREG/CR-3624, Technical Report SAND83-2365, Sandia National Laboratories, Albuquerque, NM.

Iman, R.L., Davenport, J.M., and Ziegler, D.K. (1980). "Latin Hypercube Sampling (Program User's Guide)," Technical Report SAND79-1473, Sandia National Laboratories, Albuquerque, NM.

Iman, R.L., Helton, J.C., and Campbell, J.E. (1981a). "An Approach to Sensitivity Analysis of Computer Models, Part 1. Introduction, Input Variable Selection and Preliminary Variable Assessment," *Journal of Quality Technology*, 13(3), 174-183.

Iman, R.L., Helton, J.C., and Campbell, J.E. (1981b). "An Approach to Sensitivity Analysis of Computer Models, Part 2. Ranking of Input Variables, Response Surface Validation, Distribution Effect and Technique Synopsis," *Journal of Quality Technology*, 13(4), 232-240.

Marquardt, D.W. and Snee, R.D. (1975). "Ridge Regression in Practice," *The American Statistician*, 29(1), 3-20.

Marquardt, D.W., (1970). "Generalized Inverses, Ridge Regression, Biased Linear Estimation, and Nonlinear Estimation," *Technometrics*, 12(3), 591-612.

McKay, M.D., Conover, W.J., and Beckman, R.J. (1979). "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," Technometrics, 221, 239-245.

Wyss, G. D., and Jorgensen, K. H. (1998). "A User's Guide to LHS: Sandia's Latin Hypercube Sampling Software." Technical Report SAND98-0210. Sandia National Laboratories, Albuquerque, NM.

DISTRIBUTION

Kenneth Comer
The Procter & Gamble Company
8256 Union Centre Boulevard
West Chester, OH 45069

Urmila Diwekar
University of Illinois at Chicago
Chemical Engineering
810 S. Clinton St.
209 CHB, M/C 110
Chicago, IL 60607

Donald Flaggs
Lockheed Martin Space Systems Co.
Missiles & Space Operations Advanced
Technology Center
O/ABCS, B/153
1111 Lockheed Martin Way
Sunnyvale, CA 94089

Roger Ghanem
Dept. of Civil Engineering
Johns Hopkins University
Baltimore, MD 21218

Raphael Haftka
Dept. of Aerospace/Mechanical Engr. and
Engineering Science
P.O. Box 116250
University of Florida
Gainesville, FL 32611-6250

Lawrence Livermore National
Laboratory (5)
Attn.: Evi Dube, MS L-095
Richard Klein, MS L-023
Roger Logan, MS L-125
Cynthia Nitta, MS L-096
Charles Tong, MS L-560
7000 East Ave.
P.O. Box 808
Livermore, CA 94550

Sankaran Mahadevan
Dept. of Civil & Environmental Engineering
Vanderbilt University
Box 6077, Station B
Nashville, TN 37235

Los Alamos National Laboratory (3)
Attn.: S. Keller-McNulty, MS F600
Ed Rodriguez, MS P946
Scott Doebling, MS T080
Mail Station 5000
P.O. Box 1663
Los Alamos, NM 87545

Bob Pelle
The Goodyear Tire & Rubber Co
Technical Center D/431A
1376 Tech Way Drive
Akron, Ohio 44316

John Renaud
Dept. of Aerospace & Mechanical
Engineering
University of Notre Dame
Notre Dame, IN 46556

Robert Secor
3M Center, Building 518-1-01
3M Engineering Systems and Technology
St. Paul, MN 55144

Sameer Talsania
PPG Industries, Inc.
P. O. Box 2844
Pittsburgh, PA 15230

Ben Thacker
Southwest Research Institute
6220 Culebra Road
Postal Drawer 28510
San Antonio, TX 78228-0510

MS0759 G. D. Wyss, 4145
MS0748 S. L. Daniel, 6862
MS0847 S. F. Wojtkciewicz, 9124
MS0847 A. A. Giunta, 9133
MS0770 J.C. Helton, 9133
MS0828 M. Pilch, 9133
MS0828 V. J. Romero, 9133
MS0828 W. L. Oberkampf, 9133
MS0370 S. L. Brown, 9211
MS0370 M. S. Eldred, 9211
MS0370 S. A. Mitchell, 9211
MS0370 L. P. Swiler, 9211 [5]
MS1176 J. E. Campbell, 15312
MS1176 R. M. Cranwell, 15312

MS9018 Central Technical Files, 8945-1
MS0899 Technical Library, 9616 [2]