Abstract

LASSITER, MARY ELIZABETH. Modeling and Simulating Random Vectors with Given Marginals and Covariance Structure. (Under the direction of James R. Wilson and Stephen D. Roberts)

This research concerns the NORTA (NORmal To Anything) method for constructing a multivariate random vector having prespecified marginal distributions and prespecified correlations by applying an appropriate transformation of a standard normal random vector; and the main objective of this thesis is to develop an algorithm for estimating the normal correlations that are required to yield the desired correlations between the coordinates of the transformed random vector. A stochastic root finding method was implemented to approximate the Normal correlations of interest with a prespecified level of accuracy. Examples are shown of the experimentation performed with bivariate distributions having various marginals. Results are also shown for a bivariate distribution with two Uniform distributions as marginals, in which the functional relationship between the Normal correlation and the prespecified correlation is known exactly. The experimentation shows that this stochastic root finding algorithm is an excellent method for rapidly approximating the required Normal correlations to a reasonable degree of accuracy.

MODELING AND SIMULATING RANDOM VECTORS WITH GIVEN MARGINALS AND COVARIANCE STRUCTURE

by

MARY ELIZABETH LASSITER

A thesis submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Master of Science

INDUSTRIAL ENGINEERING

Raleigh

2000

APPROVED BY:

Co-Chair of Advisory Committee

Co-Chair of Advisory Committee

Dedication

To Mom and Dad for your love and encouragement.

Biography

Mary Elizabeth Lassiter was born on March 23, 1976, in Trenton, NC, where she was reared by her parents Joseph and Laura Lassiter, along with her sisters Beverly and Anne. She graduated from the North Carolina School of Science and Mathematics in Durham, NC, in May 1994.

In the fall of 1994, she entered North Carolina State University in Raleigh, NC. She studied mathematics and mathematics education. While pursuing these degrees she studied at the University of Leeds in Leeds, England. She also was a member of several honor societies, including Pi Mu Epsilon and Kappa Delta Pi. While attending North Carolina State University, she was a North Carolina Teaching Fellow, as well as a University Scholar. She graduated *magna cum laude* in both mathematics and mathematics education with a Bachelor of Science degree.

After completion of her Master of Science in Industrial Engineering, she plans to work with Andersen Consulting in Birmingham, Alabama.

Acknowledgements

I thank Dr. James Wilson and Dr. Stephen Roberts, Co-Chairs of my advisory committee, for the unending time and effort they have spent on my research and in the preparation of this thesis. They both have given me an incredible amount of support and encouragement, and it has truly been an honor to work with them. I also thank Dr. Thomas Johnson for serving on my committee.

Table of Contents

1 Introduction 1
1.1 Problem Statement and Research Objectives 1
1.2 Organization of Thesis 2
2 Literature Review 3
3 Methodology 9
3.1 Background 9
3.2 Method of Antithetic Variates 11
3.2.1 Proof of Method of Antithetic Variates with Positive Correlation 13
3.2.2 Results with and without Using Antithetic Variates 15
Case 18
3.3 Investigation of Quadratic Fit 19
3.4 Stochastic Root Finding Algorithm 20
3.4.1 Stochastic Root Finding Algorithm Details 26
4 Experimentation 33
4.1 Gamma Example 33
4.1.1 Two Gammas with Correlation = -0.6 33
4.1.2 Two Gammas with correlation = -0.4 38
4.1.3 Two Gammas with correlation = 0.1 45
4.1.4 Two Gammas with correlation = 0.6 49
4.1.5 Two Gammas with correlation = 0.9 51
4.2 Lognormal and Weibull 55
4.2.1 Lognormal and Weibull with correlation = -0.3 55
4.2.2 Lognormal and Weibull with correlation = -0.1 61
4.2.3 Weibull and Lognormal with correlation = 0.1 64
4.2.4 Lognormal and Weibull with correlation = 0.4 68
4.2.5 Lognormal and Weibull with correlation = 0.9 70
4.3 Uniform Example 72
5 Conclusions and Recommendations 74
5.1 Conclusions 74
5.2 Recommendations for Future Work 74
Bibliography 76
Appendix A. Excel Macro Code 78

List of Tables

Table 3.1: Parameters used for specified distributions 16
Table 3.2: Comparison of variance when ρ_Z = 0.25 16
Table 3.3: Comparison of variance when ρ_Z = 0.5 17
Table 3.4: Comparison of variance when ρ_Z = 0.75 17
Table 3.5: Comparison of variance when ρ_Z = -0.5 18
Table 3.6: Actual values and approximated values for two Gamma distributions 20
Table 4.1: Initial values with negative correlation for 2 Gammas 33
Table 4.2: First Iteration with correlation = -0.6 for 2 Gammas 35
Table 4.3: Second iteration with correlation = -0.6 for 2 Gammas 36
Table 4.4: Third iteration with correlation = -0.6 for 2 Gammas 37
Table 4.5: Final iteration with correlation = -0.6 for 2 Gammas 38
Table 4.6: First iteration with correlation = -0.4 for 2 Gammas 39
Table 4.7: Second iteration with correlation = -0.4 for 2 Gammas 40
Table 4.8: Third iteration with correlation = -0.4 for 2 Gammas 41
Table 4.9: Final iteration with correlation = -0.4 for 2 Gammas 41
Table 4.10: First iteration with correlation = -0.1 for 2 Gammas 42
Table 4.11: Second iteration with correlation = -0.1 for 2 Gammas 43
Table 4.12: Third iteration with correlation = -0.1 for 2 Gammas 44
Table 4.13: Final iteration with correlation = -0.1 for 2 Gammas 44
Table 4.14: Initial values with positive correlation for 2 Gammas 45
Table 4.15: First iteration with correlation = 0.1 for 2 Gammas 46
Table 4.16: Second iteration with correlation = 0.1 for 2 Gammas 47
Table 4.17: Third iteration with correlation = 0.1 for 2 Gammas 48
Table 4.18: Final iteration with correlation = 0.1 for 2 Gammas 48
Table 4.19: First iteration with correlation = 0.6 for 2 Gammas 49
Table 4.20: Second iteration with correlation = 0.6 for 2 Gammas 50
Table 4.21: Final iteration with correlation = 0.6 for 2 Gammas 51
Table 4.22: First iteration with correlation = 0.9 for 2 Gammas 51
Table 4.23: Second iteration with correlation = 0.9 for 2 Gammas 52
Table 4.24: Third iteration with correlation = 0.9 for 2 Gammas 53
Table 4.25: Final iteration with correlation = 0.9 for 2 Gammas 54
Table 4.26: Initial values with negative correlation for Lognormal and Weibull 55
Table 4.27: First iteration with correlation = -0.3 for Lognormal and Weibull 56
Table 4.28: Second iteration with correlation = -0.3 for Lognormal and Weibull 57
Table 4.29: Third iteration with correlation = -0.3 for Lognormal and Weibull 58
Table 4.30: Fourth iteration with correlation = -0.3 for Lognormal and Weibull 59
Table 4.31: Final iteration with correlation = -0.3 for Lognormal and Weibull 60
Table 4.32: First iteration with correlation = -0.1 for Lognormal and Weibull 61
Table 4.33: Second iteration with correlation = -0.1 for Lognormal and Weibull 62
Table 4.34: Third iteration with correlation = -0.1 for Lognormal and Weibull 63
Table 4.35: Final iteration with correlation = -0.1 for Lognormal and Weibull 63
Table 4.36: Initial values with positive correlation for Lognormal and Weibull 64
Table 4.37: First iteration with correlation = 0.1 for Lognormal and Weibull 65
Table 4.38: Second iteration with correlation = 0.1 for Lognormal and Weibull 66
Table 4.39: Third iteration with correlation = 0.1 for Lognormal and Weibull 66
Table 4.40: Final iteration with correlation = 0.1 for Lognormal and Weibull 67
Table 4.41: First iteration with correlation = 0.4 for Lognormal and Weibull 68
Table 4.42: Second iteration with correlation = 0.4 for Lognormal and Weibull 69
Table 4.43: Final iteration with correlation = 0.4 for Lognormal and Weibull 70
Table 4.44: First iteration with correlation = 0.9 for Lognormal and Weibull 70
Table 4.45: Final iteration with correlation = 0.9 for Lognormal and Weibull 71
Table 4.46: Results from Uniform example 72

List of Figures

Figure 3.1: Graph of two Gamma distributions using a quadratic fit 20
Figure 3.2: Two Gamma distributions approximated with various correlations 23
Figure 3.3: Initial values of L and U given a correlation of 0.6 24
Figure 3.4: Second set of approximated values for L and U 25
Figure 3.5: Flowchart of stochastic root-finding technique 26
Figure 3.6: Code for stochastic root finding technique 27
Figure 4.1: Initial correlation bands for two Gammas with negative correlation 34
Figure 4.2: First iteration with correlation = -0.6 for 2 Gammas 35
Figure 4.3: Second iteration with correlation = -0.6 for 2 Gammas 36
Figure 4.4: Third iteration with correlation = -0.6 for 2 Gammas 37
Figure 4.5: First iteration with correlation = -0.4 for 2 Gammas 39
Figure 4.6: Second iteration with correlation = -0.4 for 2 Gammas 40
Figure 4.7: Third iteration with correlation = –0.4 for 2 Gammas 41
Figure 4.8: First iteration with correlation = –0.1 for 2 Gammas 42
Figure 4.9: Second iteration with correlation = –0.1 for 2 Gammas 43
Figure 4.10: Third iteration with correlation = –0.1 for 2 Gammas 44
Figure 4.11: Initial correlation bands for 2 Gammas with positive correlation 45
Figure 4.12: First iteration with correlation = 0.1 for 2 Gammas 46
Figure 4.13: Second iteration with correlation = 0.1 for 2 Gammas 47
Figure 4.14: Third iteration with correlation = 0.1 for 2 Gammas 48
Figure 4.15: First iteration with correlation = 0.6 for 2 Gammas 49
Figure 4.16: Second iteration with correlation = 0.6 for 2 Gammas 50
Figure 4.17: First iteration with correlation = 0.9 for 2 Gammas 52
Figure 4.18: Second iteration with correlation = 0.9 for 2 Gammas 53
Figure 4.19: Third iteration with correlation = 0.9 for 2 Gammas 54
Figure 4.20: Initial correlation bands for Lognormal and Weibull distributions with negative correlation 56
Figure 4.21: First iteration with correlation = -0.3 for Lognormal and Weibull 57
Figure 4.22: Second iteration with correlation = -0.3 for Lognormal and Weibull 58
Figure 4.23: Third iteration with correlation = -0.3 for Lognormal and Weibull 59
Figure 4.24: Fourth iteration with correlation = -0.3 for Lognormal and Weibull 60
Figure 4.25: First iteration with correlation = -0.1 for Lognormal and Weibull 61
Figure 4.26: Second iteration with correlation = -0.1 for Lognormal and Weibull 62
Figure 4.27: Third iteration with correlation = -0.1 for Lognormal and Weibull 63
Figure 4.28: Initial correlation bands for Lognormal and Weibull distributions with positive correlation 64
Figure 4.29: First iteration with correlation = 0.1 for Lognormal and Weibull 65
Figure 4.30: Second iteration with correlation = 0.1 for Lognormal and Weibull 66
Figure 4.31: Third iteration with correlation = 0.1 for Lognormal and Weibull 67
Figure 4.32: First iteration with correlation = 0.4 for Lognormal and Weibull 68
Figure 4.33: Second iteration with correlation = 0.4 for Lognormal and Weibull 69
Figure 4.34: First iteration with correlation = 0.9 for Lognormal and Weibull 71

1 Introduction

1.1 Problem Statement and Research Objectives

Many times simulations do not mirror real systems as desired. Often distributions are used as inputs, because they are available and not necessarily because they are accurate reflections of the process. It is common to have processes that are dependent. An example of the preceding scenario would be a workpiece flowing through various workstations. Each station might have a different distribution to describe its processing time, but these processing times may be dependent according to the workpiece flowing through them. This relationship between workstations would create a situation that calls for dependent models in which the marginal distributions may be different. There exist models that use dependent components and marginal distributions from a common family [Devroye 1996, Johnson 1987]. It is possible to ignore these dependencies, but the output would not represent the real system.

A model has been proposed by Cario et al. [2000] that allows the component random variables to have prespecified correlations as well as prespecified dependent marginal distributions. This model is referred to as a NORTA (NORmal To Anything) distribution. It is an extension of the previous work by Cario and Nelson [1996, 1997] on the ARTA (Autoregressive-to-Anything) process. It transforms a multivariate normal random vector into a desired random vector with the required distributional characteristics. Estimating and simulating such a process is performed in two steps. First, a $k \times 1$ multivariate normal vector **Z** will be transformed into a multivariate uniform vector **U** by applying the standard Normal cumulative distribution function to each component of **Z** separately to yield the corresponding component of **U**. Next, this vector will be transformed into the desired input vector **X** by applying the method of inversion to each component of **U** separately to yield the corresponding component of **X**. The correlation matrix of **Z** determines the correlation matrix of **X**. The fundamental problem is finding the values $\rho_Z(i, j) = \text{Corr}(Z_i, Z_j)$ for $1 \le i, j \le k$ that yield the target correlations $\rho_X(i, j) = \text{Corr}(X_i, X_j)$ as specified by the user for $1 \le i, j \le k$.

In this thesis we develop an algorithm for estimating this correlation with a prespecified precision using the variance reduction technique of antithetic variates [Law and Kelton 1991, Chapter 11]; and we implemented this algorithm in an Excel Macro. Once the specified correlations and marginals, along with their respective parameters, have been defined, then we approximate the

functional relationship $\rho_X(i, j) = c_{ij}[\rho_Z(i, j)]$. One way to create this approximation is to sample five points at $\rho_Z(i, j)$ = -1, -0.5, 0, 0.5, and 1. A quadratic function can then be passed through these points, generating the specified graph.

There are two sources of error using this approach, one of which is the estimation of ρ_X for given values of ρ_Z. This is an approximation from a simulation and not an exact number; therefore there will exist some sampling error in the estimated coefficients of the quadratic function. Moreover, there is the potential for systematic lack of fit in the quadratic that is passed through the five points. The possibility that $c_{ij}(\cdot)$ may not be exactly quadratic implies that function specification error may exist, as well. This intuitively leads to the idea that the quadratic may not be a good fit. This observation will be investigated further in this thesis.

After investigating the nature of the error in using the quadratic function as an approximation, we decided to investigate a different type of solution—a stochastic root finding problem, in which we actually approximate a specific value for ρ_Z given a specific value for ρ_X. The idea is to find the root, ρ_Z, of the equation $\rho_X = c_{ij}(\rho_Z)$. This root is obtained by inverting a piecewise linear confidence band for the function $c_{ij}(\cdot)$ in an appropriate subinterval. The width of the confidence band is iteratively reduced and the resulting confidence band is inverted to give us progressively smaller intervals of uncertainty for ρ_Z. Eventually this process will converge to a sufficiently accurate estimate of the root, ρ_Z. This method will answer a question that is more likely to be asked—that of finding a specific Normal correlation directly, instead of approximating the relationship between ρ_X and ρ_Z.

1.2 Organization of Thesis

This thesis is organized as follows. Chapter 2 reviews previous work with multivariate analysis. Chapter 3 discusses the methodology used to perform the simulation and stochastic root finding algorithm. Chapter 4 presents the experimentation performed and the results generated. Chapter 5 discusses conclusions and recommendations for further work.

2 Literature Review

One of the main foundations of this thesis is the knowledge of multivariate analysis. Multivariate analysis is concerned with the study of random variables. Interest in multivariate distributions did not arise until the nineteenth century. Initially there was more interest in multivariate Normal distributions, particularly the bivariate Normal. However, interest in other multivariate distributions was stimulated during the last quarter of the century. Correlation was not used until the second half of the nineteenth century. Much work has been performed since.

There are several multivariate distributions, with the most common being the multivariate Normal. This is due to the many applications of the Normal, along with the fact that the most research and literature available is on the Normal because it is tractable. These results are obtained mainly by showing that the multivariate Normal distribution belongs to certain large families of density functions. Therefore, results shared by these families hold for the multivariate Normal distribution. Tong (1990) gives the following fundamental properties for the multivariate Normal distribution:

1. It is a natural extension of the univariate Normal and is an appropriate model for many problems.
2. If the original data cannot be fitted with a multivariate Normal distribution, then by the central limit theorem the distribution of the sample mean vector is asymptotically Normal. This means that the multivariate Normal can be used for approximating the distribution of the same mean vector in the large-sample case.
3. The density function of multivariate Normal is uniquely determined by the mean vector and the covariance matrix of the random variable.
4. Zero correlations between marginals imply independence.
5. The family of multivariate Normal distributions is closed under linear transformations and linear combinations.
6. The marginal distribution of any subset of components of a multivariate Normal variable is also multivariate Normal.
7. The conditional distribution in a multivariate Normal distribution is multivariate Normal.
8. Positive and negative dependence properties of the components of a random vector are completely determined by the sign and the size of the correlation coefficient for the bivariate Normal distribution.

Press (1982) gives the following definition for the multivariate Normal distribution:
Let **X** be a $p\times1$ random vector with density function $f(\boldsymbol{x})$. The matrix **X** follows a nonsingular multivariate (p-variate) Normal distribution with $p\times1$ mean vector θ and $p\times p$ covariance matrix Σ if Σ is positive definite

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\theta})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\theta})\right] \qquad \text{for all } \mathbf{x}.$$

Another multivariate distribution that may be of interest is the multivariate gamma distribution (with p dimensions). If $\{X_j : j = 0,1,...\, p\}$ are independent standard gamma random variables such that X_j has shape parameter α_j, scale parameter 1, and p.d.f.

$$f_{X_j}(x_j) = \frac{1}{\Gamma(\alpha_j)} x_j^{\alpha_j - 1} e^{-x_j} \qquad \text{for } x_j > 0$$

and $j = 0, 1, \ldots, p$, then the random variables

$$Y_j = X_0 + X_j, \qquad j = 1,..., p$$

have a multivariate gamma distribution with joint p.d.f.

$$f(y_1,..., y_p) = \left[\prod_{j=1}^{p}\Gamma(\alpha_j)\right]^{-1} \exp\left(-\sum_{j=1}^{p} y_j\right) \int_0^{\min\{y_j\}} u^{\alpha_0 - 1}\left[\prod_{j=1}^{p}(y_j - u)^{\alpha_j - 1}\right] e^{-(p-1)u}\, du$$

for $y_j > 0$, $j = 1,\ldots,p$.

Another distribution of interest is the multivariate lognormal distribution. A univariate random variable is said to be lognormal if its logarithm follows the Normal distribution. This type of distribution has may applications in business and economics. Let the $p\times1$ random vector **y** follow $\boldsymbol{N}(\theta,\Sigma)$. Let $\mathbf{y}' \equiv (y_1,..., y_p)$ where $y_j = \log x_j$. Define the "vector logarithm" as the following:

$$\log \mathbf{x} \equiv \begin{pmatrix} \log x_1 \\ \vdots \\ \log x_p \end{pmatrix}, \qquad \mathbf{x}' \equiv (x_1,..., x_p).$$

Then **x** is said to follow the lognormal distribution with density given by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}\Pi_1^p x_j} \exp\left[-\frac{1}{2}(\log \mathbf{x} - \boldsymbol{\theta})'\boldsymbol{\Sigma}^{-1}(\log \mathbf{x} - \boldsymbol{\theta})\right]$$

for $x_j > 0,\ j = 1,..., p,$ and $p(\mathbf{x}) = 0,$ otherwise.

The multivariate distributions discussed above are basic textbook distributions, and not exhaustive. Although research has been performed extensively in these areas, there are cases when these distributions fail to model the data at hand. These distributions are often not flexible, and many times simply inappropriate for some data modeling. There has been other work performed on other multivariate distributions that allows for more extensive representation of the data.

One of these distributions is the Johnson distribution (Mirka, et al.). It is defined as follows. First a normalizing translation is sought for a random vector **X** whose distribution is to be estimated. The normalizing translation can be modeled as

$$\mathbf{Z} = \boldsymbol{\gamma} + \boldsymbol{\delta}\mathbf{g}[(\boldsymbol{\lambda})^{-1}(\mathbf{X} - \boldsymbol{\xi})] \sim N_p(\mathbf{0}, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\gamma} = [\gamma_1,..., \gamma_p]'$ and $\boldsymbol{\delta} = [\delta_1,..., \delta_p]'$ are vectors of shape parameters, $\boldsymbol{\lambda} = [\lambda_1,..., \lambda_p]'$ is a vector of scale parameters, $\boldsymbol{\xi} = [\xi_1,..., \xi_p]'$ is a vector of location parameters and the transformation, $g(\cdot)$, is then identified as

$$\mathbf{g}[(y_1,..., y_v)^T] \equiv [g_1(y_1),..., g_v(y_v)]^T,$$

where

$$g_i(y_i) = \begin{cases} \ln(y_i), & \text{for } S_L \text{ (lognormal) family,} \\ \ln\left[y_i + \sqrt{y_i^2 + 1}\right], & \text{for } S_U \text{ (unbounded) family,} \\ \ln\left[\frac{y_i}{1 - y_i}\right], & \text{for } S_B \text{ (bounded) family,} \\ y_i, & \text{for } S_N \text{ (Normal) family,} \end{cases}$$

for $i = 1\ldots,p,$ so that the marginal distribution of each X_i is a univariate Johnson distribution. Then the matrices $\boldsymbol{\gamma}$, $\boldsymbol{\delta}$, $\boldsymbol{\xi}$, and $\boldsymbol{\lambda}$ are estimated. Finally the correlation matrix, $\boldsymbol{\Sigma}$, is estimated by inserting $\{\mathbf{X}_j : j = 1,..., n\}$ into the normalizing translation above to obtain a standard multinormal sample $\{\mathbf{Z}_j : j = 1,..., n\}$, and by computing the sample correlation matrix of $\{\mathbf{Z}_j\}$ as the approximate moment-matching estimator of $\boldsymbol{\Sigma}$. This method has shown to be successful in modeling data, and its flexibility allows it to be a better fit than would be found with distributions

such as the Normal, gamma, or lognormal, as shown above. Note that with this setup, the conditional distributions of **X** are also Johnson.

Mirka et al. (2000) have presented a method based on the Johnson translation system of univariate probability distributions, which takes independent standardized Johnson variates to build a multivariate random vector. This extension uses the Johnson family for its marginals, but the conditional marginal distributions are not of the Johnson family. This method proves to have many applications but has its own limitations as well, one of which being the computational difficulty.

Another exploration into multivariate analysis brings us to the Bezier distribution. Again, this is a distribution that is flexible and can model various types of data. It has been shown by Wagner and Wilson (1996) to be an effective model for fitting bivariate data. Bezier curves and surfaces are used in this distribution. A Bezier curve is defined as

$$\mathbf{P}(t) = \sum_{i=0}^{n} B_{n,i}(t)\mathbf{p}_i \,, \qquad \text{for all t} \in [0,1],$$

given n+1 control points $\left\{\mathbf{p}_i \equiv (x_i, z_i)^{\mathrm{T}} : i = 0,1,...,n\right\}$, and

$$\mathbf{x} \equiv (x_0,...,x_n) \text{ and } \mathbf{z} \equiv (z_0,...,z_n)$$

are vectors of the given control points. The Bernstein polynomials $\left\{B_{n,i}(t) : i = 0,1,..,n\right\}$ are defined as

$$B_{n,i}(t) \equiv \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, \qquad \text{for all } t \in [0,1].$$

If we have $(n_x + 1)(n_y + 1)$ control points

$$\left\{\mathbf{q}_{i,j} \equiv (x_{i,j}, y_{i,j}, z_{i,j})^{\mathrm{T}} : i = 0,1,...n_x,\ j = 0,1,...,n_y\right\},$$

then the corresponding Bezier surface is defined as

$$\mathbf{Q}(t_x, t_y) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} B_{n_x,i}(t_x) B_{n_y,i}(t_y) \mathbf{q}_{i,j} \qquad \text{for } t_x \text{ and } t_y \in [0,1],$$

where

$$\mathbf{x} = \left[x_{i,j}\right],\ \mathbf{y} = \left[y_{i,j}\right], \text{ and } \mathbf{z} = \left[z_{i,j}\right]$$

are the *x*-, *y*-, and *z*- coordinates of the given control points. Bezier marginals and conditional distributions are used. For a bivariate case, if $(X,Y)^{\mathrm{T}}$ is a continuous random vector with unknown c.d.f. $F_{XY}(\cdot,\cdot)$, then this c.d.f. can be approximated with a Bezier surface, where

$$x(t_x) = \sum_{i=0}^{n_x} B_{n_x,i}(t_x)x_i,$$

$$y(t_y) = \sum_{j=0}^{n_y} B_{n_y,i}(t_y)y_j, \quad \text{and}$$

$$F_{XY}\left[x(t_x), y(t_y)\right] = \sum_{i=0}^{n_x}\sum_{j=0}^{n_y} B_{n_x,i}(t_x)B_{n_y,i}(t_y)z_{i,j} \qquad \text{for all } t_x, t_y \in [0,1].$$

The marginals of this function are represented below:

$$F_X(\cdot) = \{x(t_x), F_X[x(t_x)]\}^{\mathrm{T}} = \sum_{i=0}^{n_x} B_{n_x,i}(t_x)\left[x_i, z_i^{(X)}\right]^{\mathrm{T}} \qquad \text{for all } t_x \in [0,1], \quad \text{and}$$

$$F_Y(\cdot) = \left\{y(t_y), F_Y[y(t_y)]\right\}^{\mathrm{T}} = \sum_{i=0}^{n_y} B_{n_y,j}(t_y)\left[y_j, z_j^{(Y)}\right]^{\mathrm{T}} \qquad \text{for all } t_y \in [0,1]$$

where

$$\mathbf{x} = \begin{bmatrix} x_0 & x_0 & \cdots & x_0 \\ x_1 & x_1 & \cdots & x_1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n_x} & x_{n_x} & x_{n_x} & x_{n_x} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n_y} \\ y_0 & y_1 & \cdots & y_{n_y} \\ \vdots & \vdots & \vdots & \vdots \\ y_0 & y_1 & \cdots & y_{n_y} \end{bmatrix}, \text{ and } \mathbf{z} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & z_{1,1} & \cdots & z_{1,n_y-1} & z_1^{(X)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & z_{n_x-1,1} & \cdots & z_{n_x-1,n_y-1} & z_{n_x-1}^{(X)} \\ 0 & z_1^{(Y)} & \cdots & z_{n_y-1}^{(Y)} & 1 \end{bmatrix}.$$

This is only a bivariate Bezier, and it isn't clear how to model a multivariate Bezier distribution. Here again, is another limitation in modeling multivariate distributions.

Another limitation in these models is that they do not necessarily allow for the representation of dependent processes in which the marginals are not necessarily from the same family of functions. One way of coping with this dependency was developed by Cario and Nelson (1996) through ARTA (AutoRegressive To Anything) processes. This allows for a stationary stochastic input process, $\{Y_t : t = 1,2,...\}$, that has the properties below:

1. A marginal c.d.f. $F_Y(y) = \Pr\{Y_t \le y\}$ for all *y* and for *t* = 1,2,…, with marginal mean, μ_Y, and marginal variance, σ_Y^2.
2. A given correlation structure out to a selected lag *p*, $\mathrm{Corr}(Y_t, Y_{t+h}) = \rho_h$ for *h* = 1,…,*p*.

An ARTA process is defined in terms of a stationary autoregressive process, $\{Z_t : t = 1,...\}$, of order *p*, called the base process, having the following properties:

1. The marginal distribution of Z_t is the standard Normal distribution.
2. The variates $\{Z_t\}$ satisfies the regression relation: $Z_t = \alpha_1 Z_{t-1} + ... + \alpha_p Z_{t-p} + \varepsilon_t$, *t* = *p*+1, *p*+2,….
3. The "white noise" process, given as $\{\varepsilon_t : t = 1,2,...\} \overset{\text{i.i.d}}{\sim} N(0, \sigma^2)$, where $\sigma^2 = 1 - \alpha_1 r_1 - \cdots - \alpha_p r_p$, and , $r_h = \text{Corr}(Z_t, Z_{t+h})$ for *h* =1,2,…, is the correlation structure.

Therefore the corresponding ARTA process , $\{Y_t : t = 1,2,...\}$, with desired marginal c.d.f. and covariance structure is given by

$$Y_t = F_Y^{-1}[\Phi(Z_t)] \qquad \text{for } t = 1,2,\ldots,$$

if we solve the following equations for the correlations of the base process:

$$\text{Corr}\left\{F_Y^{-1}[\Phi(Z_t)], F_Y^{-1}[\Phi(Z_{t+h})]\right\} = \rho_h \qquad \text{for } h = 1,\ldots, p.$$

Although ARTA processes deal with time series, this idea has been extended by Cario, Nelson, Roberts, and Wilson (2000) to transform a *k*×1 standard multivariate normal vector to obtain a *k*×1 input vector with given marginal c.d.f.'s and correlation matrix. This idea is extended as NORTA, and is discussed in Section 3. The basis of this thesis extends from this idea.

3 Methodology

3.1 Background

The simulation of these random vectors is performed as follows. A random vector **X** is defined that has the following properties:

1. $X_i \sim F_{X_i}$, *i=1,2,…,k,* where F_{X_i} is an arbitrary cumulative distribution function with finite mean, μ_i, and variance, σ_i^2; and
2. Cov[**X**] = $\left[\rho_X(i,j)\sigma_i\sigma_j\right]$ = Σ_X, where Σ_X is a given $k \times k$ covariance matrix.

The random vector, **X**, is a transformation of a *k*-dimensional, standard multivariate normal vector $\mathbf{Z} = (Z_1, Z_2,,\ldots,Z_k)'$ with correlation matrix Σ_Z = Cov(**Z**)= $\left[\rho_Z(i,j)\right]$. Another way to describe **X** is

$$\mathbf{X} = \begin{pmatrix} F_{X_1}^{-1}\left[\Phi(Z_1)\right] \\ F_{X_2}^{-1}\left[\Phi(Z_2)\right] \\ \vdots \\ F_{X_k}^{-1}\left[\Phi(Z_k)\right] \end{pmatrix},$$

where Φ is the standard normal cumulative distribution function (c.d.f.), and

$$F_{X_i}^{-1}(u) \equiv \inf\left[x : F_{X_i}(x) \geq u\right] \qquad \text{for all } u \in (0,1),$$

is the inverse c.d.f. This transformation ensures that X_i has the desired marginal distribution F_{X_i}. The problem, as stated above, is to select the matrix Σ_Z that yields Σ_X. Therefore, it is desired to estimate

$$\rho_X(i,j) = \text{Corr}(X_i, X_j) = \frac{E[X_i X_j] - \mu_i\mu_j}{\sigma_i\sigma_j} \qquad \text{for } i \neq j,\ 1 \leq i, j \leq k,$$

with absolute error δ or (relative) percentage error $100\gamma\%$. When $\rho_X(i,j) = 0$, the variables are independent; and it is no problem to solve. However, when $\rho_X(i,j) \neq 0$, the situation becomes more complicated.

Antithetic variates are used to estimate $\theta = E[X_1X_2]$ with absolute error $\delta' = \delta\sigma_1\sigma_2$. (See Section 3.2 for further details about the use of antithetic variates and their application in this algorithm).

For $i=1,2$ let

$$a_i = \begin{cases} \text{lower limit of } X_i, & \text{if this is finite,} \\ \mu_i - 10\sigma_i, & \text{otherwise.} \end{cases}$$

Let $Q_i = X_i - a_i$, for $i=1,2$. This will ensure that positive random variables are being used. Then,

$$\rho = \text{Corr}(X_1, X_2) \cong \text{Corr}(Q_1, Q_2) = \frac{\theta' - (\mu_1 - a_1)(\mu_2 - a_2)}{\sigma_1 \sigma_2},$$

where $\theta' = E[Q_1 Q_2]$. The quantity θ' is delivered once the number of runs, N, is computed to achieve the desired precision. The quantities N and θ' are computed by the following procedure:

1. Generate an initial sample $\{(U_{1j}, U_{2j}): j=1,2,\ldots,n_0\}$ of independent random numbers. Compute the following for $j = 1, 2, \ldots, n_0$:

$$Z_{1j} = \Phi^{-1}(U_{1j})$$

$$Z_{2j} = \rho_Z \Phi^{-1}(U_{1j}) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})$$

$$Q_{1j}^{+} = F_1^{-1}\left[\Phi\left(Z_{1j}\right)\right] - a_1 = F_1^{-1}(U_{1j}) - a_1$$

$$Q_{2j}^{+} = F_2^{-1}\left[\Phi\left(Z_{2j}\right)\right] - a_2 = F_2^{-1}\left\{\Phi\left[\rho_Z \Phi^{-1}\left(U_{1j}\right) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}\left(U_{2j}\right)\right]\right\} - a_2$$

$$Q_{1j}^{-} = F_1^{-1}\left(1 - U_{1j}\right) - a_1$$

$$Q_{2j}^{-} = F_2^{-1}\left\{\Phi\left[\rho_Z \Phi^{-1}\left(1-U_{1j}\right) + \sqrt{1-\rho_z^2}\,\Phi^{-1}\left(1-U_{2j}\right)\right]\right\} - a_2$$

$$P_j = \frac{1}{2}\left[Q_{1j}^{+} Q_{2j}^{+} + Q_{1j}^{-} Q_{2j}^{-}\right].$$

2. Compute

$$\overline{P}_{n_0} = \frac{1}{n_0}\sum_{j=1}^{n_0} P_j$$

$$s^2 = \frac{1}{n_0 - 1}\sum_{j=1}^{n_0} (P_j - \overline{P}_{n_0})^2$$

3. Compute

$$N = \left\lceil \frac{z_{1-\alpha/2}^2 \cdot s^2}{(\delta')^2} \right\rceil$$

4. If $N \le n_0$, then deliver $\hat{\theta}' = \overline{P}_{n_0}$, and $\hat{\rho} = \dfrac{\hat{\theta}' - (\mu_1 - a_1)(\mu_2 - a_2)}{\sigma_1 \sigma_2}$; otherwise collect the additional samples, $\{P_j: j = n_0+1,\ldots, N\}$. Take $n_0 \leftarrow N$, recompute $\overline{P}_{n_0}$, s^2 based on all data collected so far, and go to step 3.

This should yield an estimate of Σ_Z, the desired result. Once Σ_Z is determined, the simulation input is generated as follows [Cario et al. 2000]:

1. Set up: Determine a lower-triangular, nonsingular factorization **M** of Σ_Z so that $\mathbf{MM}'=\Sigma_Z$.
2. Generate $\mathbf{W} = (W_1, W_2, \ldots, W_k)'$, a kx1 random vector whose elements are i.i.d. standard normal random variables.
3. Set $\mathbf{Z} \leftarrow \mathbf{MW}$.
4. Return **X** where $X_i \leftarrow F_{X_i}^{-1}[\Phi(Z_i)]$, $i=1,2,\ldots, k$.
5. Repeat steps 2 - 4 for each vector $\boldsymbol{X}$ to be generated.

3.2 Method of Antithetic Variates

A method of computing the correlation matrix was developed in order to reduce the variance and thus the computational burden. Law and Kelton [1991, Chapter 11] describe the procedure of using antithetic variates as a way to induce correlation between separate runs, and to reduce the variance. The idea is to make pairs of runs where a "small" observation in one run is offset by a "large" observation in the corresponding run. This allows the average of the two observations in the pair to be a more precise estimate of μ than would have been obtained by the average of two independent observations of the same random variable.

Suppose we have Y_1 and Y_2 that have correct distributions but are not necessarily independent. We would like to estimate μ_Y, = E[Y_i], i = 1,2 by inducing Cov(Y_1, Y_2) < 0 and using $\overline{Y_2} = \frac{1}{2}(Y_1 + Y_2)$. We have

$$\text{var}(\overline{Y}_2) = \frac{1}{4}[\text{var}(Y_1) + \text{var}(Y_2) + 2\,\text{cov}(Y_1, Y_2)] = \frac{1}{2}[\sigma_Y^2 + \text{cov}(Y_1, Y_2)].$$

Obviously, if the covariance is less than 0 (or negative), then the variance will decrease. So, it is desired to induce correlation between pairs (X_1, X_2,), (X_3, X_4), etc. Across the pairs, let the observations be independent.

Bratley, Fox, and Schrage (1987) give the following basic result justifying the method of antithetic variates.

Theorem. *Suppose* Y_1 *and* Y_2 *are nonconstant, monotone functions.*

$$Y_1 = Y_1(U_1, U_2, \ldots),$$
$$Y_2 = Y_2(U_1^*, U_2^*, \ldots)$$

of each of their individual arguments. If $U_1, U_2, \ldots$ *are i.i.d. random numbers, and we take*

$$U_i^* = 1 - U_i \qquad \textit{for } i = 1, 2, \ldots,$$

then $\mathrm{Cov}(Y_1, Y_2) < 0$.

With Cov(Y_1,Y_2) < 0, $\overline{Y_2}$ is a more precise estimator of μ_Y than if Y_1,Y_2 were independent. Therefore in the estimator scheme of Section 3.1, it is desired to exploit random variables that meet this criteria. These quantities are given below:

$$Z_{1j} = \Phi^{-1}(U_{1j})$$
$$Z_{2j} = \rho_Z \Phi^{-1}(U_{1j}) + \sqrt{1 - \rho_Z{}^2}\,\Phi^{-1}(U_{2j})$$
$$Q_{1j}^+ = F_1^{-1}\left[\Phi(Z_{1j})\right] - a_1 = F_1^{-1}(U_{1j}) - a_1$$
$$Q_{2j}^+ = F_2^{-1}\left[\Phi(Z_{2j})\right] - a_2 = F_2^{-1}\left[\Phi\left(\rho_Z \Phi^{-1}(U_{1j}) + \sqrt{1 - \rho_Z{}^2}\,\Phi^{-1}(U_{2j})\right)\right] - a_2 ,$$

so that we can take

$$Y_{1j} = Y_{1j}(U_{1j}, U_{2j}) = Q_{1j}^+ Q_{2j}^+ \qquad \text{for } j = 1, 2, \ldots;$$

and similarly we take

$$Q_{1j}^- = F_1^{-1}(1 - U_{1j}) - a_1$$
$$Q_{2j}^- = F_2^{-1}\left\{\Phi\left[\rho_Z \Phi^{-1}(1 - U_{1j}) + \sqrt{1 - \rho_Z^2}\,\Phi^{-1}(1 - U_{2j})\right]\right\} - a_2 ,$$

so that we have

$$\begin{aligned} Y_{2j} &= Y_{2j}(U_{1j}^*, U_{2j}^*) \\ &= Y_{2j}(1 - U_{1j}, 1 - U_{2j}) \\ &= Q_{1j}^- Q_{2j}^- \qquad \text{for } j = 1, 2, \ldots . \end{aligned}$$

In the next section we establish that $Y_{1j}(U_{1j}, U_{2j})$ and $Y_{2j}(U_{1j}^*, U_{2j}^*)$ are nonconstant monotonically nondecreasing functions of each of their individual arguments.

When using antithetic variates, it is essential that the two observations be used for the same purpose. Drawbacks to the procedure include that the feasibility and efficacy of the procedure are model-dependent. This method is not guaranteed to work without falter. In this

situation, though, it appears to work well. For some examples of variance reduction observed, see Section 3.2.2.

3.2.1 Proof of Method of Antithetic Variates with Positive Correlation

Based on the theorem given in the above section, it can be shown that for a positive correlation the functions are monotonic.

For Q_{1j}^{+}, we see that:

$$Q_{1j}^{+} = F_1^{-1}(U_{1j}) - a_1 .$$

F_1^{-1} is monotonic increasing because F_1 is a cumulative distribution function, which is always monotonic increasing. Therefore, the inverse will also be monotonic. Subtracting a constant from F_1^{-1} will not change the monotonicity of F_1^{-1}. Therefore Q_{1j}^{+} will be a nonnegative, nonconstant monotonically nondecreasing function of U_{1j}, and Q_{1j}^{+} does not depend on U_{2j}.

For Q_{2j}^{+}, we see that:

$$Q_{2j}^{+} = F_2^{-1}\left[\Phi(Z_{2j})\right] - a_2 = F_2^{-1}\left\{\Phi\left[\rho_Z \Phi^{-1}(U_{1j}) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})\right]\right\} - a_2 .$$

Let U_{1j} be fixed. The inverse Normal cumulative distribution function is given as Φ^{-1}, so $\Phi^{-1}(U_{2j})$ is monotonic. We see that $\sqrt{1-\rho_Z^2}$ is a nonnegative constant, so $\sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})$ is a nonnegative monotonically nondecreasing function of U_{2j}. Since U_{1j} is fixed, then $\rho_Z \Phi^{-1}(U_{1j})$ is fixed. Adding a constant to a monotonic function does not affect the monotonicity, so $\Phi\left[\rho_Z \Phi^{-1}(U_{1j}) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})\right]$ is a nonnegative, nondecreasing function of U_{2j}. Applying F_2^{-1} to this gives a monotonic function. Subtracting a constant from the above term does not change the monotonicity of the function.

Now let U_{2j} be fixed. We see that $\Phi^{-1}(U_{1j})$ is monotonic because Φ^{-1} is the inverse Normal cumulative distribution function. Multiplying a nonnegative constant, ρ_Z, by

$\Phi^{-1}(U_{2j})$ will not change the monotonicity. Adding a constant, $\sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})$, will not change the monotonicity. Applying F_2^{-1} will be monotonic, as well.

Therefore Q_{2j}^{+} is a monotonically nondecreasing function of the arguments U_{1j} and U_{2j} individually when the other argument is fixed.

There are also special cases where $\rho_Z = 0, 1$. In these cases Q_{2j}^{+} behaves differently. Q_{1j}^{+} is not affected by ρ_Z, so the arguments presented above hold for these special cases, as well. For $\rho_Z = 0$, we see that:

$$\begin{aligned} Q_{2j}^{+} &= F_2^{-1}[\Phi(Z_{2j})] - a_2 = F_2^{-1}\{\Phi[0 * \Phi^{-1}(U_{1j}) + \sqrt{1-0} * \Phi^{-1}(U_{2j})]\} - a_2 \\ &= F_2^{-1}\{\Phi[\Phi^{-1}(U_{2j})]\} - a_2 \\ &= F_2^{-1}(U_{2j}) - a_2. \end{aligned}$$

The argument used for Q_{1j}^{+} above can also be applied here, so Q_{2j}^{+} will be a nonnegative, nonconstant monotonically nondecreasing function of U_{2j}, and Q_{2j}^{+} does not depend on U_{1j} in this case. For $\rho_Z = 1$, we see that:

$$\begin{aligned} Q_{2j}^{+} &= F_2^{-1}[\Phi(Z_{2j})] - a_2 = F_2^{-1}\{\Phi[1 * \Phi^{-1}(U_{1j}) + \sqrt{1-1} * \Phi^{-1}(U_{2j})]\} - a_2 \\ &= F_2^{-1}\{\Phi[\Phi^{-1}(U_{1j})]\} - a_2 \\ &= F_2^{-1}(U_{1j}) - a_2. \end{aligned}$$

The argument used for Q_{1j}^{+} above can also be applied here, so Q_{2j}^{+} will be a nonnegative, nonconstant monotonically nondecreasing function of U_{1j}, and Q_{2j}^{+} does not depend on U_{2j} in this case.

These are the only requirements for this method of antithetic variates to work. Antithetic variates are not used for negative correlation in this case, due to the fact that we have been unable to analytically prove that they work. This means that only Q_{1j}^{+} and Q_{2j}^{+} are used when calculating P_j for the case in which $\rho_Z < 0$. Incidentally, the method of antithetic variates "seems" to work for negative correlation. These results can be seen in Section 3.2.2.

3.2.2 Results with and without Using Antithetic Variates

The cases below show the actual variance reduction when using antithetic variates. Several random examples were used, and in every case, a significant variance reduction occurred. The following results were obtained by fixing ρ_Z for each pair of distributions. We sampled various values of ρ_Z at 0.25, 0.5, 0.75, and –0.5. The value for –0.5 was used simply to show that the technique of using antithetic variates seems to work, even though it cannot be proved. After fixing ρ_Z, then the corresponding ρ_X values were estimated using the algorithm found in Section 3.1. In these cases, 10,000 runs were used. The default parameters were also used for each case, which can be found in Table 3.1. Then the variance was recorded noting the difference between the variance when using the method of antithetic variates, and the variance when the method was not used. The following formula from the algorithm in Section 3.1 was used for this variance:

$$s^2 = \frac{1}{n_0 - 1} \sum_{j=1}^{n_0} (P_j - \overline{P}_{n_0})^2 ,$$

where

$$P_j = \begin{cases} \frac{1}{2}\left[Q_{1j}^{+} Q_{2j}^{+} + Q_{1j}^{-} Q_{2j}^{-}\right], & \text{when using antithetic variates, and} \\ Q_{1j}^{+} Q_{2j}^{+}, & \text{when not using antithetic variates} \end{cases}$$

$$\overline{P}_{n_0} = \frac{1}{n_0} \sum_{j=1}^{n_0} P_j$$

Table 3.1 shows the parameters used for various cases, while Tables 3.2 through 3.5 show the variance reduction found in various cases observed.

Table 3.1: Parameters used for specified distributions

Distribution	**Parameter(s)**
Gamma	$\alpha = 1$, $\beta = 1$ (where α is the shape parameter, and β is the scale parameter)
Uniform	minimum = 0, maximum = 1
Student's t	degrees of freedom = 3
Weibull	$\alpha = 1$, $\beta = 1$ (where α is the shape parameter, and β is the scale parameter)
Lognormal	mean = 0, standard deviation = 1
Chi-Square	degrees of freedom = 1
Exponential	mean = 1

Table 3.2: Comparison of variance when $\rho_Z = 0.25$

Case	**Variance**		**% Variance Reduction**
	without AV	With AV	
Gamma, Gamma	4.72210	1.87287	60.34%
Uniform, t	1.50862	0.43330	71.28%
Weibull, Lognormal	4.80396	2.46788	48.63%
Chi-square, Chi-square	4.41759	1.75471	60.28%
Exponential, Exponential	4.72196	1.87268	60.34%

The average percent reduction in variance when $\rho_Z = 0.25$ is approximately 60.17%.

Table 3.3: Comparison of variance when $\rho_Z = 0.5$

Case	Variance		% Variance Reduction
	without AV	with AV	
Gamma, Gamma	7.53703	3.15593	58.13%
Uniform, t	0.79219	0.20913	73.60%
Weibull, Lognormal	8.00602	4.34206	45.77%
Chi-square, Chi-square	8.63159	3.36892	60.97%
Exponential, Exponential	7.53699	3.15597	58.13%

The average percent reduction in the variance when $\rho_Z = 0.5$ is approximately 59.32%.

Table 3.4: Comparison of variance when $\rho_Z = 0.75$

Case	Variance		% Variance Reduction
	without AV	with AV	
Gamma, Gamma	11.66722	5.08547	56.41%
Uniform, t	0.310271	0.08194	73.59%
Weibull, Lognormal	12.78541	7.76593	39.26%
Chi-square, Chi-square	15.77950	6.14605	61.05%
Exponential, Exponential	11.66720	5.08568	56.41%

The average percent reduction in the variance when $\rho_Z = 0.75$ is approximately 57.34%.

The following table shows the variance reduction when the correlation is negative. As stated previously, it has not been proven that the method of antithetic variates works with negative correlation, but these results show a variance reduction, which is an interesting observation. The sampling scheme is changed slightly when using a negative correlation. The quantities Q_{2j}^{+} and Q_{2j}^{-} are now negative as seen below. Step 1 of the sampling scheme as seen in Section 3.1 is changed in the following manner:

1. Generate an initial sample $\{(U_{1j}, U_{2j}): j=1,2,\ldots,n_0\}$ of independent random numbers. Compute:

$$Z_{1j} = \Phi^{-1}(U_{1j})$$
$$Z_{2j} = \rho\Phi^{-1}(U_{1j}) + \sqrt{1-\rho^2}\,\Phi^{-1}(U_{2j})$$
$$Q_{1j}^{+} = F_1^{-1}[\Phi(Z_{1j})] - a_1 = F_1^{-1}(U_{1j}) - a_1$$
$$Q_{2j}^{+} = F_2^{-1}[\Phi(Z_{2j})] - a_2 = \left[F_2^{-1}\left\{\Phi\left[\rho_Z\Phi^{-1}(U_{1j}) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}(U_{2j})\right]\right\} - a_2\right]\mathrm{sgn}(\rho_Z)$$
$$Q_{1j}^{-} = F_1^{-1}(1-U_{1j}) - a_1$$
$$Q_{2j}^{-} = \left[F_2^{-1}\left\{\Phi\left[\rho_Z\Phi^{-1}(1-U_{1j}) + \sqrt{1-\rho_Z^2}\,\Phi^{-1}(1-U_{2j})\right]\right\} - a_2\right]\mathrm{sgn}(\rho_Z)$$
$$P_j = \frac{1}{2}\left[Q_{1j}^{+}Q_{2j}^{+} + Q_{1j}^{-}Q_{2j}^{-}\right],$$

where $\mathrm{sgn}(\rho_Z) = 1$ if $\rho_Z \geq 0$, and $\mathrm{sgn}(\rho_Z) = -1$ if $\rho_Z < 0$.

Changing the sign of Q_{2j}^{+} and Q_{2j}^{-} guarantees that these functions are monotonic increasing; but since $Q_{1j}^{+} \geq 0$, while $Q_{2j}^{+} \leq 0$ and $Q_{1j}^{-} \geq 0$ while $Q_{2j}^{-} \leq 0$, the products $Q_{1j}^{+}Q_{2j}^{+}$ and $Q_{1j}^{-}Q_{2j}^{-}$ are not necessarily monotonic functions of their random-number inputs. Using this scheme we find the following results.

Table 3.5: Comparison of variance when ρ_Z = -0.5

Case	Variance		% Variance Reduction
	without AV	with AV	
Gamma, Gamma	0.66107	0.20402	69.14%
Uniform, t	4.60772	1.75642	61.88%
Weibull, Lognormal	0.39002	0.13824	64.56%
Chi-square, Chi-square	0.27677	0.10713	61.29%
Exponential, Exponential	0.66107	0.20402	69.14%

The average percent reduction in variance when ρ_Z = -0.5 is approximately 65.20%. This is actually a better variance reduction than those shown for $\rho_Z > 0$. In practice, the method of antithetic variates is used often even when it has not been proven to work. This is a case where we cannot prove that it works all the time, but it seems to work well.

3.3 Investigation of Quadratic Fit

As discussed previously, there are two sources of error when using the quadratic approximation. One source of error is the estimation of ρ_X for given values of ρ_Z. Another is the quadratic that passes through the five points. If the function $\rho_X = c_{ij}(\rho_Z)$ is not exactly a quadratic (second-degree) polynomial in ρ_Z, then there is systematic (nonrandom) error in our approximation to the desired functional relationship between ρ_Z and ρ_X. This intuitively leads to the idea that the quadratic may not be a good fit.

This idea was investigated further. It was decided to take five points that were known with certainty. These points were not simulated using the simulation discussed previously, but were found using a different program called ARTAFACTS by Cario and Nelson [1996]. A quadratic function was passed through these points, and then an error approximation was found. The method of least squares was employed to find the appropriate fit, which was performed by Excel. The following equation was used to derive this average error:

$$Average\ Error = \frac{1}{n}\sum_{i=1}^{n}\frac{|O_i - E_i|}{E_i}, \text{ where}$$

O_i = the approximated calculation at i,

E_i = the actual calculation at i.

A "good fit" would be assumed if this average error is less than 0.1.

One example was generated which compared two gamma functions. Table 3.6 shows the actual numerical values given by ARTAFACTS and the approximated values from the quadratic fit. Figure 3.1 shows the quadratic approximation of the correlations of the two gamma distributions.

Table 3.6: Actual values and approximated values for two Gamma distributions

Rhoz	Rhox (Actual)	Rhox(Quadratic)	Error
0.912	0.9	0.951317032	0.0570189244
0.72	0.7	0.769363749	0.09909107
0.328	0.3	0.371702643	0.23900881
-0.64	-0.5	-0.851158938	0.702317876

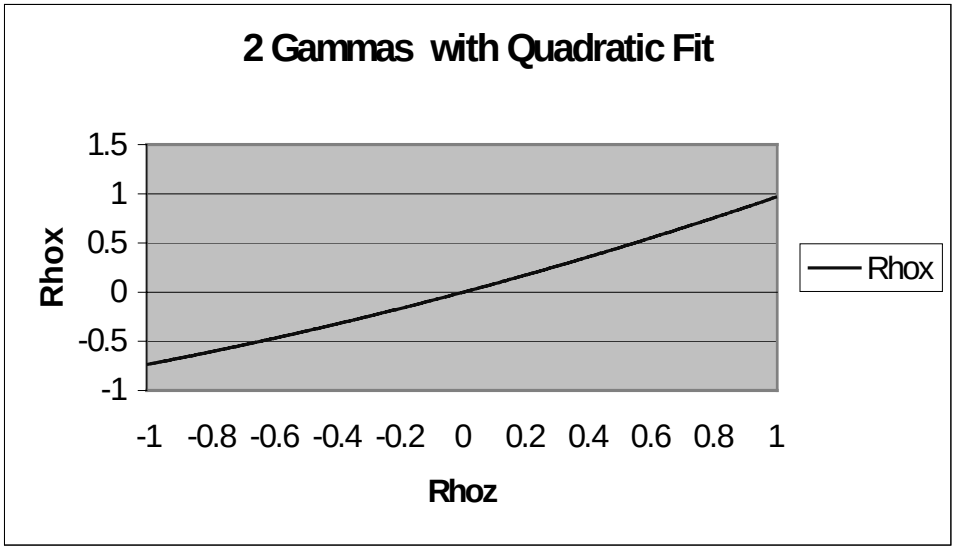

Figure 3.1: Graph of two Gamma distributions using a quadratic fit

The above values give a percent error of 0.2744, or 27.44%. This example shows that the quadratic is not a good fit, which was expected. It was not necessary to perform any other approximations after observing the nature of the quadratic fit in this case.

3.4 Stochastic Root Finding Algorithm

After observing that the quadratic fit was not an appropriate model for this data, we decided to develop another means to model the data. We noted that most users would not need an entire function to model their data. The average user would most likely have a specific correlation and

would simply want to transform that correlation into a Normal correlation. For instance, a user might have a production line in which one station has a process time having a Gamma distribution and another station with a process time having an Exponential distribution. The user will also have a target or desired correlation between these two processing times that may be based on sample data or subjective information (expert opinion). This user will be interested in the bivariate Normal correlation corresponding to the given correlation. This user will most likely not be interested in the entire functional relationship $\rho_X = c_{ij}(\rho_Z)$ for all values of ρ_Z, but simply the correlation ρ_Z that yields the desired correlation ρ_X between the two processing times. Using this logic, we decided to approach the problem from a different perspective.

The simulation discussed in Section 3.1 takes a Normal correlation and transforms it into a correlation of specified marginals. Using this information, a search method was implemented that approximates the correlation of interest, with a prespecified level of accuracy.

The basic idea is to find the root, ρ_Z, of the equation

$$\rho_X = c_{ij}(\rho_Z)$$

for a prespecified target value, ρ_X, by inverting a piecewise linear confidence band for the function $c_{ij}(\cdot)$ in an appropriate subinterval of [-1, +1]; and this yields an interval of uncertainty for the root of ρ_X. By iteratively reducing the width of the confidence band for $c_{ij}(\cdot)$ over the current interval of uncertainty for the root ρ_Z and inverting the resulting confidence band, we obtain progressively more narrow intervals of uncertainty for ρ_Z. We terminate this procedure when the latest interval of uncertainty has sufficiently small width, using linear interpolation to yield the final point estimate of the root ρ_Z. In this section we mainly use ρ_Z in the discussion. This is due to the fact that certain properties apply to ρ_Z that may or may not apply to ρ_X.

There were several things that were known about the function that allowed this approach to work:

1. A positive Normal correlation, ρ_Z, will always translate into a positive correlation of any other marginals, ρ_X.

2. A zero Normal correlation, ρ_Z, will always translate into a zero correlation of any other marginals, ρ_X.

3. A negative Normal correlation, ρ_Z, will always translate into a negative correlation of any other marginals, ρ_X.

4. The function that is of interest is very smooth, in fact $c_{ij}(\cdot)$ is infinitely differentiable.

Numbers 1, 2, and 3 are known via basic statistical laws. Item 4 is assumed based on observations of the function in past research.

The algorithm samples five points of Normal correlation based on whether the given correlation is positive or negative. If the given correlation, ρ_X, is positive, then the five Normal points that are sampled are ρ_Z = 0, 0.25, 0.5, 0.75, and 1. If the given correlation, ρ_X, is negative, the five Normal points that are sampled are ρ_Z = –1, -0.75, -0.5, -0.25, and 0. Each point is sampled with a confidence level of 99% and an absolute precision of δ = 0.05. This means, using a Bonferroni probability approach, that 1% of confidence is lost at each point simulated. This is based on the Bonferroni inequality, which is stated below:

Theorem. *Let A_i be the ith of n not necessarily independent events in a probability space. The Bonferroni inequality states the following:*

$$P(A_1 \cap A_2 \cap \ldots \cap A_n) \geq 1 - \sum_{i=1}^{n} (1 - P(A_i)).$$

After performing 5 sampling operations in our root-finding procedure such that the probability is at most 1% that the root-finding procedure will fail on any single sampling operation, we see the overall probability is at least 95% that the root-finding procedure has not failed—in other words, the probability is at least 95% that the latest uncertainty interval for the target value, ρ_Z, actually contains that value. This creates a lower and upper limit on each point sampled. A line connects every upper limit of every point, as well as every lower limit of every point. This creates a "band" which encapsulates the function. We can be confident that the band contains this function based on the smoothness of the curve, as well as sampling at five equally spaced points. See Figure 3.2 for an example of the band generated.

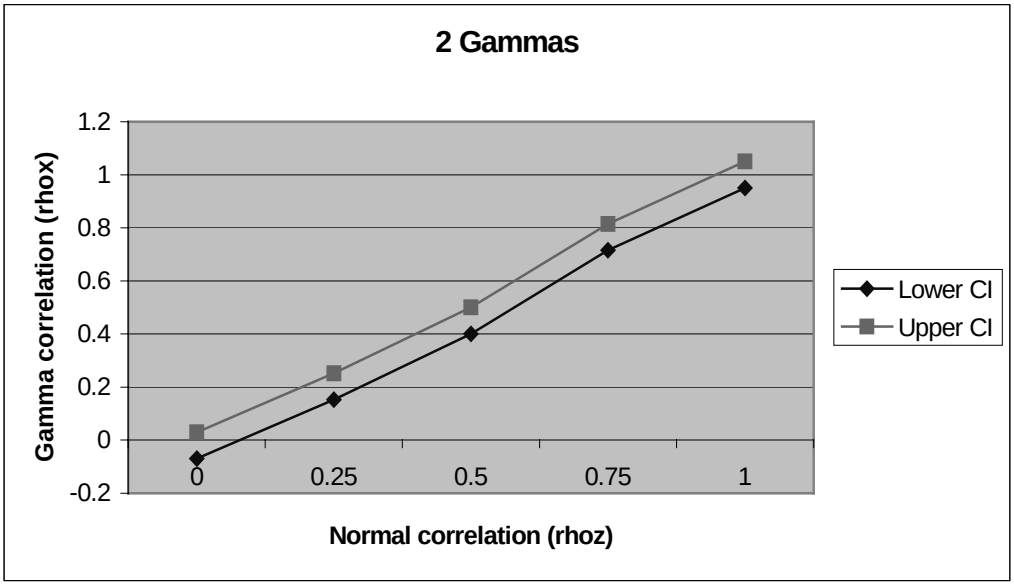

Figure 3.2: Two Gamma distributions approximated with various correlations

These band lines are used to make an upper and lower limit of the Normal correlation that is of interest. This is done by finding the location where the target correlation, ρ_X, intersects the band lines. These Normal values then become the new lower and upper limits for the correlation to be obtained. For instance, with a specified correlation of $\rho_X = 0.6$ in the above graph, the user would find that the new lower and upper limits on the Normal correlation would be L = 0.579055 and U = 0.658531 respectively. See Figure 3.3 for this graphical interpretation.

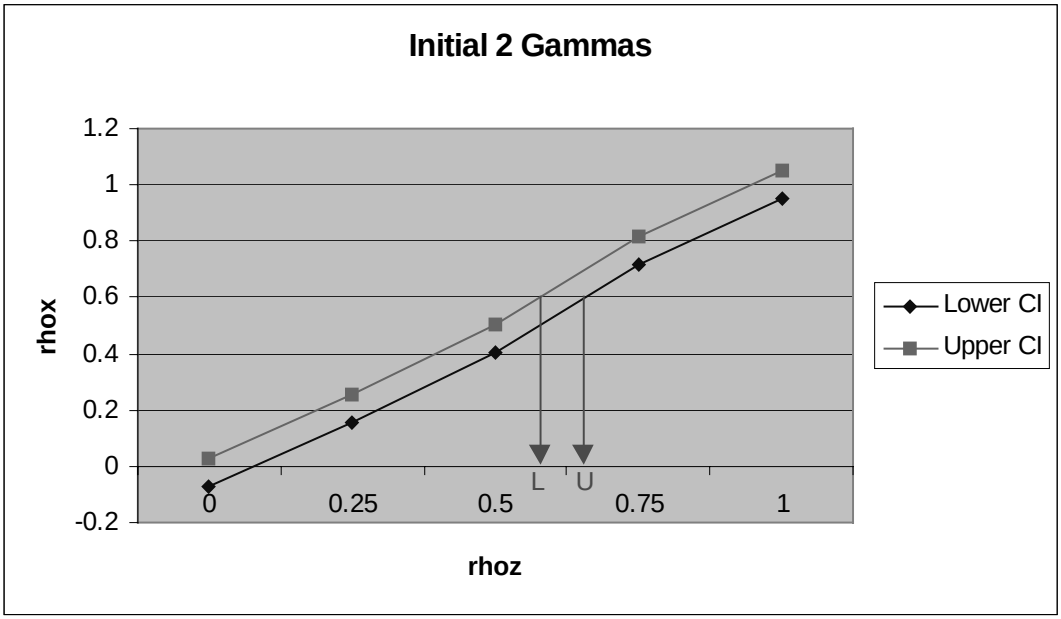

Figure 3.3: Initial values of L and U given a correlation of 0.6

New sample estimates of $c_{ij}(\rho_Z)$ are simulated at ρ_Z = L and ρ_Z = U to create a new band. The confidence interval is also reduced by half, thus making the bandwidth smaller from $\rho_X - \delta$ to $\rho_X + \delta$. This is done to ensure that the next iteration will continue to reduce the uncertainty of the values for the Normal correlations. It was also observed to be a useful technique by Chen and Schmeiser (2000). Using the same Bonferroni probability approach, 1% of confidence is lost at each of the points simulated. Now the probability that the correct value is in this band is 93%, since altogether 7 sampling operations have been performed so far to compute 99% confidence intervals on the ρ_X value corresponding to 7 different ρ_Z values. The intersection of these bands with the given correlation is found, and new upper and lower limits are found in the same way as before, yielding L = 0.608612 and U = 0.652239. See Figure 3.4 for another geometric interpretation.

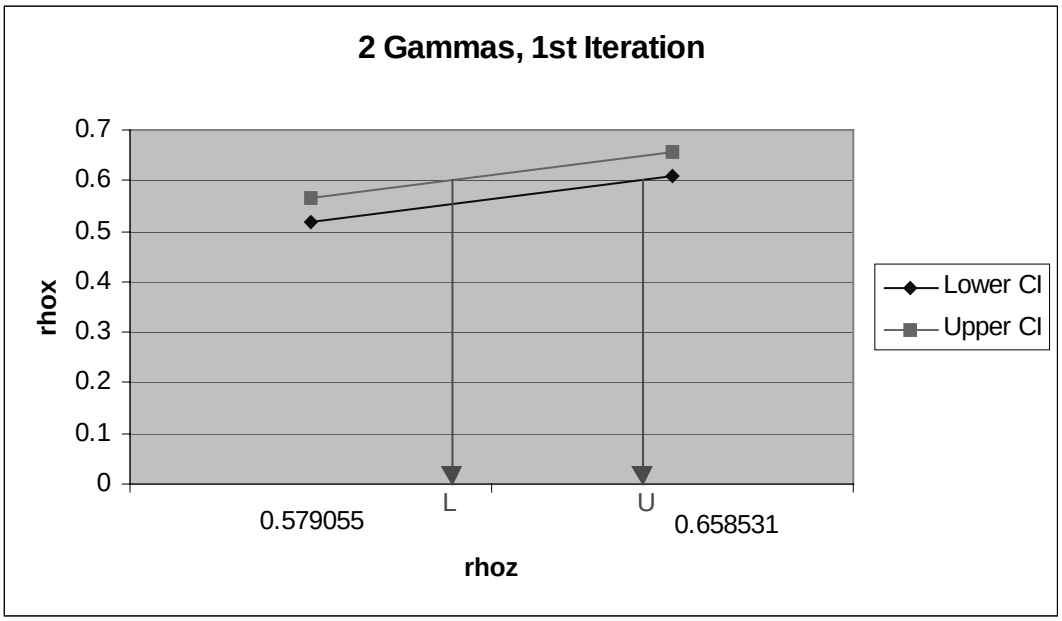

Figure 3.4: Second set of approximated values for L and U

It is possible that even though the confidence interval is reduced at each iteration, one or both limits will not change. In can be seen above that the upper limit does not change significantly. This iteration shows that the difference between the upper and lower limits is 0.043627, which does not satisfy the stopping rule, which is that this process continues until the difference in the upper and lower limits is less than 0.025Another iteration must be made. Once the difference is less than 0.025 then estimated ρ_X values are generated at L and U. The confidence interval stays the same width as the previous iteration. Then, linear interpolation is used to arrive at a final approximation of the normal correlation.

In practice, this algorithm has proven not to take more than 4 iterations, which means if we start with a 99% confidence interval, then the coverage probability for our final interval of uncertainty on the desired root, ρ_Z, has coverage probability of at least 85%.

3.4.1 Stochastic Root Finding Algorithm Details

In the previous section, a general look was taken at the stochastic root finding algorithm. This section will give a complete, concise look at the algorithm, and how it works. The details of each step of the algorithm will be discussed.

The following figure gives an overall idea of the algorithm:

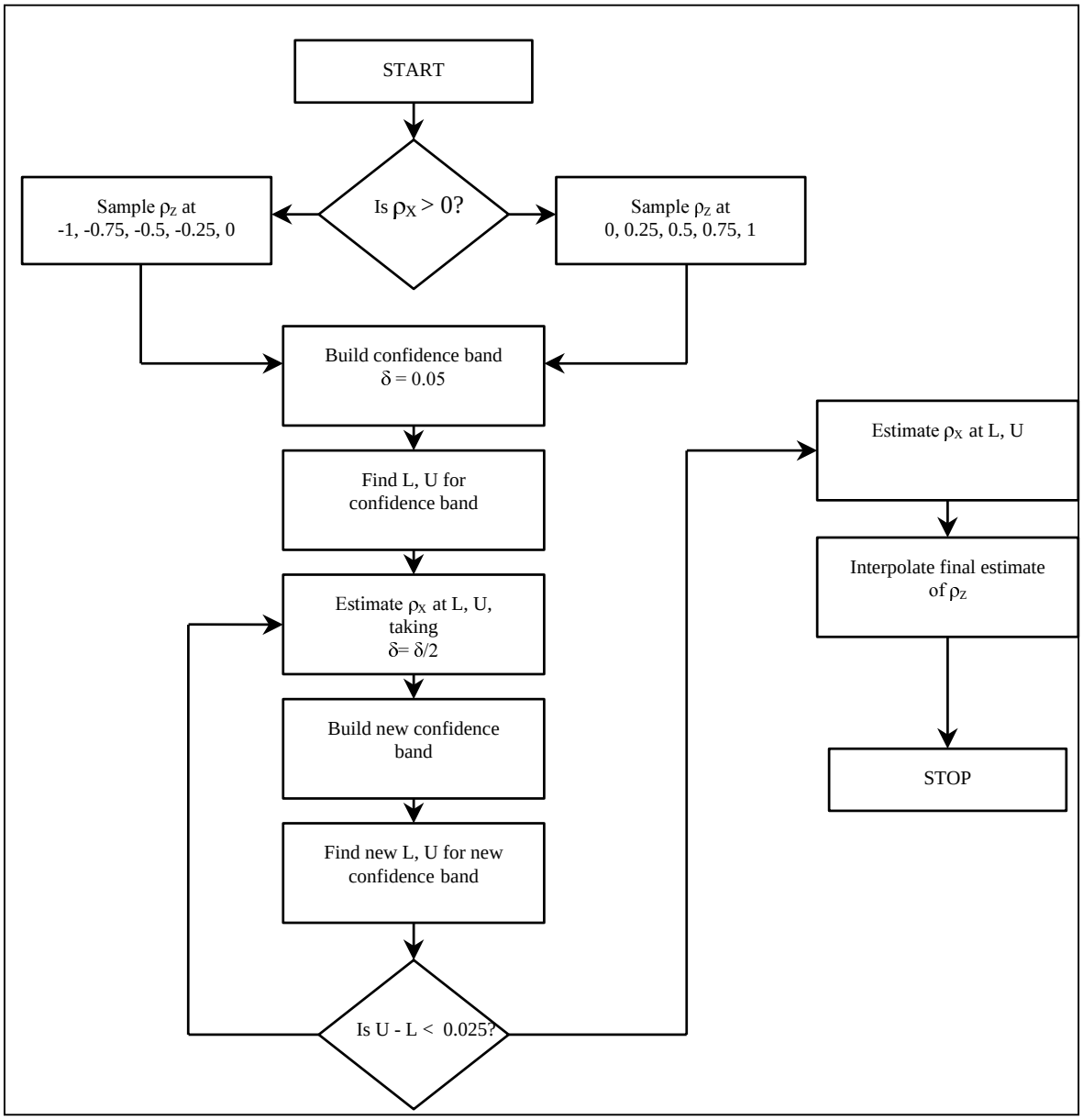

Figure 3.5: Flowchart of stochastic root finding technique

As stated previously, the code for this routine is contained in an Excel macro. A user of the code would run the macro, and through user interfaces, specify the number of marginals to be used (for this algorithm only two marginals can be specified), the marginals to be used, along with their parameters, and the correlation. Although bivariate cases can only be implemented here, multivariate cases simply use pair-wise comparison by using the bivariate case. So, this is not a limitation on the code. After the appropriate information is specified, the user will press "Run". After the root finding algorithm is called, the "code" seen in Figure 3.6 is invoked:

```
Let ρ̂_X be specified correlation from user.
If ρ̂_X > 0 Then          (Decide if ρ̂_X is in [0,1] or [-1,0]. If ρ̂_X is in
                          [-1,0], then set "bottom' = 1)
      Bottom = 0
Else
      Bottom = 1
End If
If Bottom = 0 Then
      ρ_Z1 = 0
      ρ_Z2 = 0.25
      ρ_Z3 = 0.5
      ρ_Z4 = 0.75
      ρ_Z5 = 1
Else
      ρ_Z1 = -1
      ρ_Z2 = -0.75
      ρ_Z3 = -0.5
      ρ_Z4 = -0.25
      ρ_Z5 = 0
End If
```

Figure 3.5: Code for stochastic root finding technique

Simulate at ρ_{Z1} (This is the simulation routine discussed in Section 3.1)

ρ_{X1} = returned value

Simulate at ρ_{Z2}

ρ_{X2} = returned value

Simulate at ρ_{Z3}

ρ_{X3} = returned value

Simulate at ρ_{Z4}

ρ_{X4} = returned value

Simulate at ρ_{Z5}

ρ_{X5} = returned value

(Now each band will be created. Each lower band will have i = 1, while an upper band will have i = 2. The index, j, will designate which ρ_Z value at which the band will start. For instance, if the combination is "14", then this indicates a lower band which starts at ρ_{Z4}.)

From i = 1 *to* 2

From j = 1 *to* 4

$$\text{slope}_{ij} = \frac{(\rho_{X\,j+1} - \delta) - (\rho_{X\,j} - \delta)}{0.25}$$

If i = 1 *Then*

$$\text{intercept}_{ij} = (\rho_{X\,j} - \delta) - \text{slope}_{ij} * \rho_{Z\,j}$$

Else

$$\text{intercept}_{ij} = (\rho_{X\,j} + \delta) - \text{slope}_{ij} * \rho_{Z\,j}$$

End If

$$X_{ij} = \frac{\hat{\rho}_X - \text{intercept}_{ij}}{\text{slope}_{ij}}$$

Step 1

Step 1

Figure 3.6, continued: Code for stochastic root finding technique

```
Count = 0            (This finds what iteration we are on.)

For i=1 to 2         (This loop finds which values of X_ij fall in which
                     intervals.)
     For j = 1 to 4
          If  (count < 2) and ( X_ij > ρ_Zj ) and ( X_ij < ρ_Z j+1 x(i,j) ) Then
                count = count + 1
                Z_count = X_ij



          Else If i =1 Then          (This will be used to tell if the number
                                     specified is an upper or lower limit.
                                     If Band =1, then we are dealing with an
                                     upper limit.)
                Band = 1
          Else
                Band = 2             (This will indicate an upper band, thus
                                     giving a lower limit)
          End If
     Step 1
Step 1

If count = 2 Then  (This means that U and L are both contained on the interval
                   specified.  Otherwise, we need to go outside of the
                   intervals to find where U or L lies.)
     U = max( Z_1 , Z_2 )
     L =  min( Z_1 , Z_2 )

 Else If count = 1 Then

     If bottom = 0 and Band = 1 Then  (If you're in the top half, and you've
                                      assigned one value, which falls on the
                                      lower interval, then this must be an
                                      upper limit.  Therefore, we designate
                                      L = 0 as the lower limit.)
          L = 0
          U = Z_1
     Else If bottom = 0 and Band =2 Then     (If you're in the top half, and
                                             you've assigned one value, which
                                             falls on the upper interval, then
                                             this must be a lower limit.
                                             Therefore, we designate U=1 as
                                             the upper limit.)
          L = Z_1
          U = 1
```

Figure 3.6, continued: Code for stochastic root finding technique

```
     Else If bottom = 1 and Band = 1 Then     (If you're in the lower half,and
                                               you've assigned one value, which
                                               falls on the lower interval,then
                                               this must be an upper limit.
                                               Therefore we designate L = -1 as
                                               the lower limit.)

           L = -1
           U = Z1

     Else If bottom = 1 and Band = 2 Then     (If you're in the lower half,and
                                               you've assigned one value, which
                                               falls on the upper limit, then
                                               this must be a lower limit.
                                               Therefore, we designate U = 0 as
                                               the upper limit.)
                 L = Z1
                 U = 0
     End If


Else                        (Count = 0, which means the lower limit must be the
                            normal bounds for ρ̂X .)
     If bottom = 1 Then

           If ρ̂X < (ρX1 − δ) Then      (If the user value is less than the min
                                        value possible)

                 Give error message: you're out of bounds.

           Else
                 L = -1
                 U = 0
           End If

     Else
           If ρ̂X > (ρX5 + δ) Then      (If the user value is more than the
                                        max value possible)

                 Give an error message: you're out of bounds.

           Else
                 L = 0
                 U = 1
           End If
     End If
End If

Counter = 0
```

Figure 3.6, continued: Code for stochastic root finding technique

Loop until (U-L) < 0 .025

Counter = counter + 1 (counter for iterations)

$\delta \leftarrow \frac{\delta}{2}$ (confidence interval width cut in half each iteration)

Simulate at L
XL = returned value

Simulate at U
XU = returned value

Slope = $\frac{\text{XL - XU}}{\text{L - U}}$

From i = 1 *to* 2 (These are the same i's from previously, where i =1 is a lower and i = 2 is an upper.)

If i = 1 *Then*

$\text{int}_i = (\text{XL} - \delta) - \text{slope} * \text{L}$

Else

$\text{int}_i = (\text{XL} + \delta) - \text{slope} * \text{L}$

End If

$X_i = \frac{\rho_X - \text{int}_i}{\text{slope}}$

Step 1

(The index notes the band we are using. So, if i = 2, we're on an upper band, which indicates the X value is a lower limit. If i = 1 we're on a lower limit, which indicates the value is an upper limit. The following finds what the new U and L should be based on where X_1 and X_2 fall with respect to L and U. "Change" is used for the final iteration, so that runs won't have to be unnecessarily replicated.)

If L < X_1 < U *and* L < X_2 < U *Then* (If both X_1 and X_2 fall between L and U, then they must be the new limits.)

L = X_2

U = X_1

Change = 2 (L and U have changed)

Else If L < X_1 < U *and* X_2 < L *Then* (If only X_1 falls between L and U, then it will be a new limit, and since X_1 is on a lower band, then it must be an upper limit. L stays the same as before.)

U = X_1

Change = 12 (L is same, U changed.)

Figure 3.6, continued: Code for stochastic root finding technique

```
      Else If L < X2 < U and X1 > U Then     (If only X2 falls between L and
                                              U, then it will be a new limit,
                                              and since X2 is on an upper
                                              band, the it must be a lower
                                              limit.  U stays the same.)
            L = X2
            Change = 11                      (L changed, U is same.)

      Else
            Change = 0                       (Any other case will mean that
                                              neither L nor U have changed.)
      End If

End Loop

If Change = 2
      Simulate at L
            XL = returned value

      Simulate at U
            XU = returned value

Else If Change = 11 Then
      Simulate at L
            XL = returned value

Else If Change = 12 Then
      Simulate at U
            XU = returned value

Else                                  (Nothing to simulate, because we'll use
                                      previous values for L and U.)

End If
```

$$\text{slope} = \frac{\text{XU - XL}}{\text{U - L}}$$ (Linearly interpolate for the final estimation.)

Intercept = XL – slope *L

Estimated ρ_Z = $\dfrac{\hat{\rho}_X - \text{intercept}}{\text{slope}}$ (Return this value to the user for ρ_Z.)

END

Figure 3.6, continued: Code for stochastic root finding technique

4 Experimentation

4.1 Gamma Example

Examples were generated with the gamma correlations at -0.6, -0.4, -0.1, 0.1, 0.6, and 0.9. Each subsection reviews the results found. The confidence level with every example is 99%. The relative percent error always starts with 5%. This means $\delta = .05$. Recall that δ is divided in half at every iteration. In the following Gamma examples, the number of runs required for each calculation is included to give the reader an idea of how long each iteration would take. The times listed are the exact time it took a Pentium II, 233 MHz computer with 64 MB of RAM to finish compute the run. In these examples, the calculations are performed stepwise by iteration. The stochastic root finding technique as implemented in the Excel macro will not iteratively show the user graphs and tables as presented in this section. These are given in this section to show the reader exactly how the algorithm works.

4.1.1 Two Gammas with Correlation = -0.6

In this example, each step of the algorithm will be outlined, so the reader can specifically see what is happening. In following examples this level of detail will not be provided.

Initial:

With a correlation of –0.6, the first five points to be simulated are –1, -0.5, -0.75, -0.5, -0.25, and 0. The following values were found:

Table 4.1: Initial values with negative correlation for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-1	-0.64262	-0.69262	-0.59262	40
-0.75	-0.54758	-0.59758	-0.49758	393
-0.5	-0.33169	-0.38169	-0.28169	1978
-0.25	-0.21117	-0.26117	-0.16117	4095
0	-0.02661	-0.07661	0.023393	7355

The "Rhoz" column gives the values of the normal correlations. The "Rhox" column gives the corresponding correlations for two gammas. "Lower CI" is the lower confidence interval bound on the "rhox" value, where

$$L = \hat{\rho}_X - \delta$$

"Upper CI" is the upper confidence interval bound on the "rhox" value, where

$$U = \hat{\rho}_X + \delta$$

"N" is the number of runs of the simulation routine required to achieve the rhox value given. The following graph is a visual display of the above data.

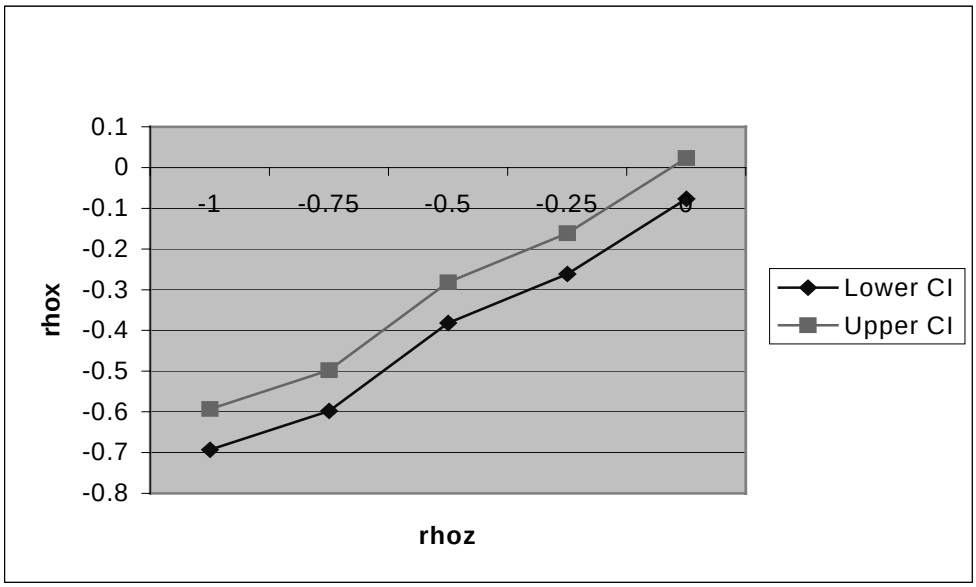

Figure 4.1: Initial correlation bands for two Gammas with negative correlation

The next step is to find where the correlation of –0.6 intersects the appropriate upper and lower confidence interval bands. It can be seen visually that –0.6 does not intersect any upper confidence interval band from –1 to 0. Since this is the case, then –1 is taken as the lower limit. It can be seen that –0.6 intersects the second lower confidence interval band, though, so it is necessary to find the corresponding point. This is found to be –0.75637. This gives the next approximations:

L = -1

U = -0.7400

U-L = 0.26

This does not meet the stopping rule.

First Iteration:

The upper and lower limits of rhoz are simulated for their corresponding rhox values. (Recall that delta is divided in half, so delta = 0.025 at this iteration.) The following values were found:

Table 4.2: First Iteration with correlation = -0.6 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-1	-0.64018	-0.66518	-0.61518	175
-0.75637	-0.50374	-0.52874	-0.47874	2258

This gives the following graph:

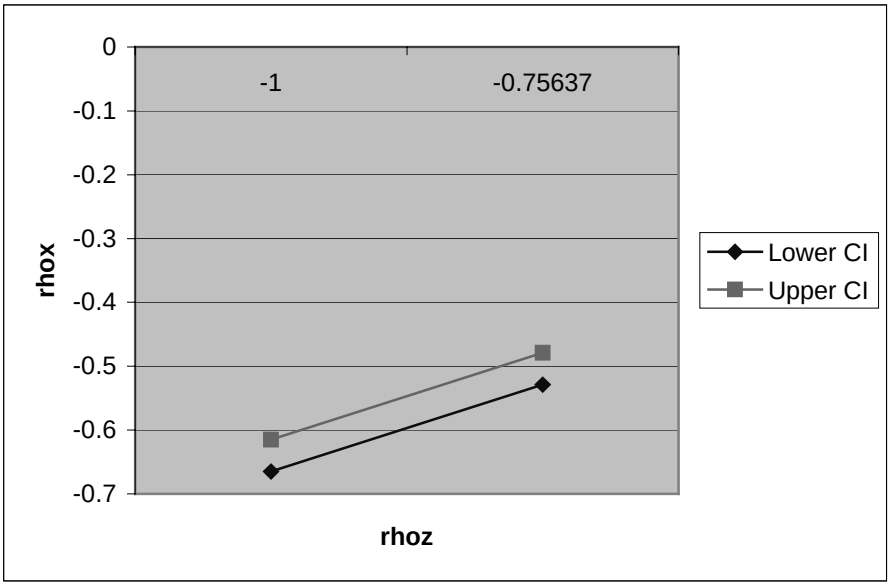

Figure 4.2: First iteration with correlation = -0.6 for 2 Gammas

Here –0.6 intersects the upper and lower confidence intervals at values of rhoz, -0.97289 and –0.88361 respectively.

L = -0.97289

U = -0.88361

U-L = 0.089281

This does not meet the stopping rule.

Second Iteration:

Now delta = 0.0125. Simulating at the above values for L and U, the following values are generated:

Table 4.3: Second iteration with correlation = -0.6 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.97289	-0.62546	-0.63796	-0.61296	1188
-0.88361	-0.58821	-0.60071	-0.57571	3638

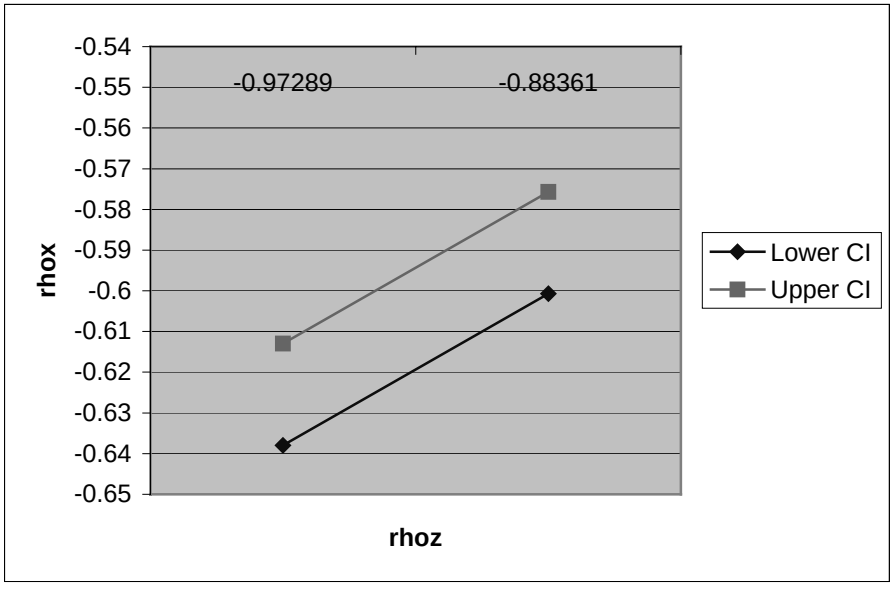

Figure 4.3: Second iteration with correlation = -0.6 for 2 Gammas

L = -0.94183

U = -0.8819

U-L = 0.059923

This does not satisfy the stopping rule.

Third Iteration:

Now delta = 0.0625. Simulating at L and U above:

Table 4.4: Third iteration with correlation = -0.6 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.94183	-0.61605	-0.6223	-0.6098	7471
-0.8819	-0.58467	-0.59092	-0.57842	14298

This gives the following graph:

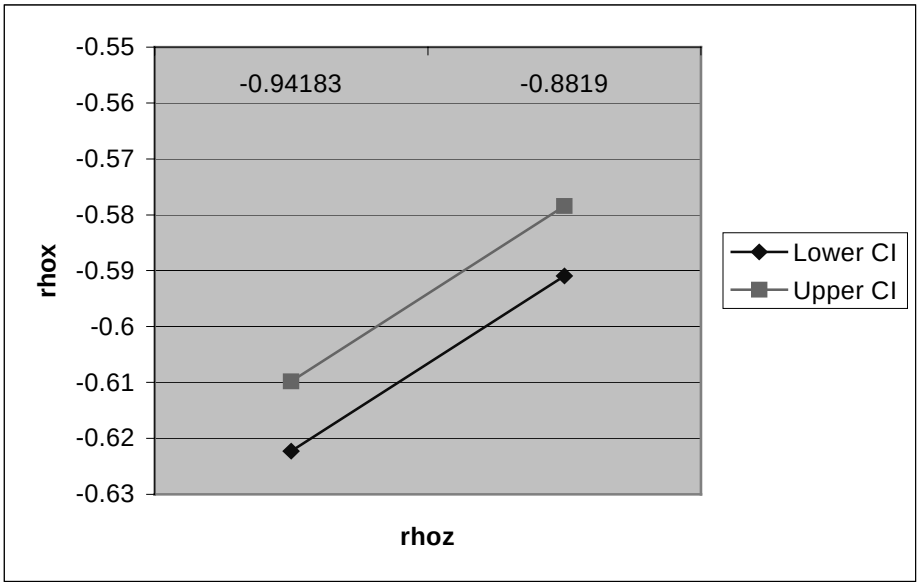

Figure 4.4: Third iteration with correlation = -0.6 for 2 Gammas

This gives the following new limits:

L = -0.92311

U = -0.89924

U-L = 0.02387

This does meet the stopping rule. So, the final step is to linearly interpolate between the lower and upper limits. Delta remains constant at 0.0625. This gives the following values:

Table 4.5: Final iteration with correlation = -0.6 for 2 Gammas

rhoz	Rhox	N
-0.92311	-0.60627	9409
-0.89924	-0.59415	12152

This gives the approximated rhoz value as –0.91076. It took 5 minutes and 39 seconds to find this value.

4.1.2 Two Gammas with correlation = -0.4

Initial:

These initial values were found in Section 4.1.2 and are used again, since the correlation is negative. For further detail see Table 4.1 and Figure 4.1. This gives the following initial limits:

L = -0.637

U = -0.5212

U-L = 0.115799

First Iteration:

Table 4.6: First iteration with correlation = -0.4 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.637	-0.44316	-0.46816	-0.41816	4370
-0.5212	-0.39185	-0.41685	-0.36685	6056

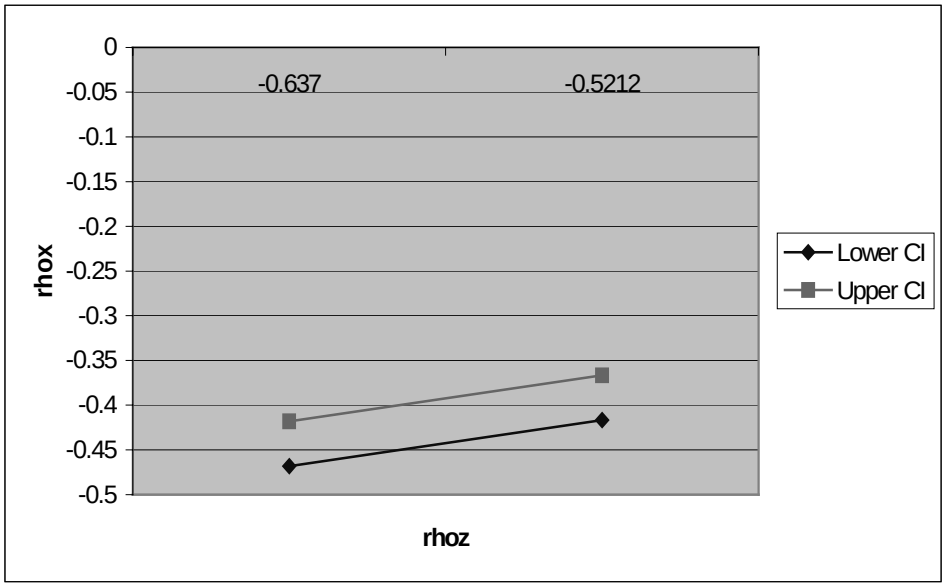

Figure 4.5: First iteration with correlation = -0.4 for 2 Gammas

Note: This is a case where the lower limit does not change. A lower limit was established in the previous iteration, which bounds the interval better than the lower limit found in this iteration. Therefore, the lower limit should not change.

L = -0.59602

U = -0.5212

U-L = 0.07482

Second Iteration:

Table 4.7: Second iteration with correlation = -0.4 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.59602	-0.42634	-0.43884	-0.41384	19369
-0.5212	-0.37248	-0.38498	-0.35998	27138

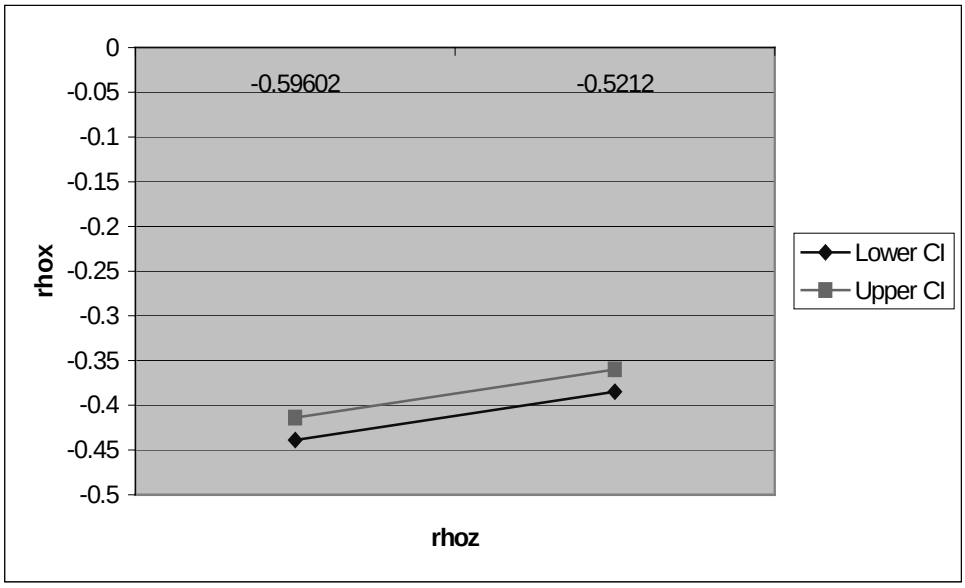

Figure 4.6: Second iteration with correlation = -0.4 for 2 Gammas

L = -0.719861

U = -0.54207

U-L = 0.034729

Third Iteration:

Table 4.8: Third iteration with correlation = -0.4 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.57679	-0.41338	-0.41963	-0.40713	83500
-0.54207	-0.31943	-0.32568	-0.31318	94036

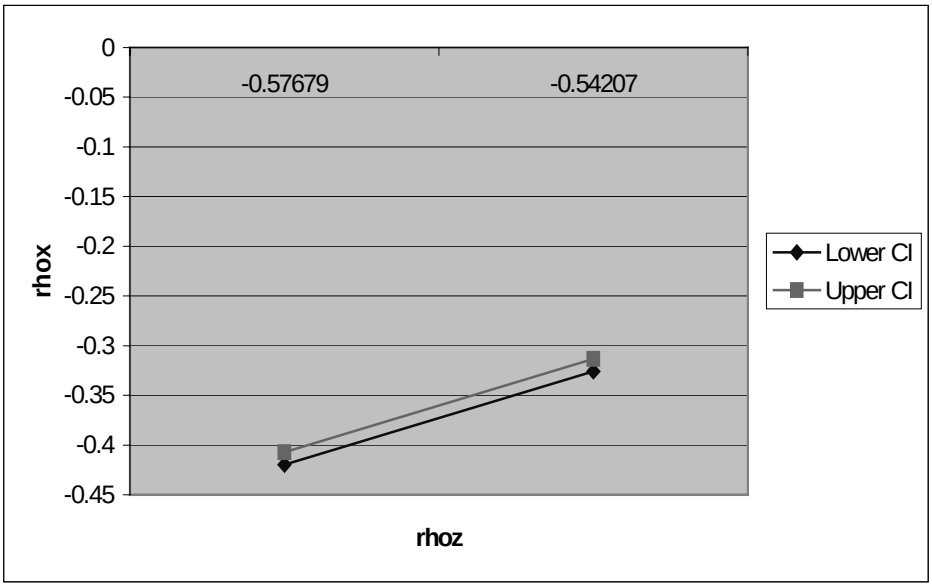

Figure 4.7: Third iteration with correlation = –0.4 for 2 Gammas

L = -0.57415

U = -0.56954

U-L = 0.00462

Final Iteration:

Table 4.9: Final iteration with correlation = -0.4 for 2 Gammas

Rhoz	Rhox	N
-0.57415	-0.41193	84259
-0.56954	-0.40861	84127

Estimated rhox = -0.55757

It took 23 minutes and 15 seconds to find this value.

4.1.3 Two Gamma Distributions with Correlation = -0.1

Initial:

These results are the same as the initial results found in Section 4.1.1. See Table 4.1 and Figure 4.1 for details.

First Iteration:

Table 4.10: First iteration with correlation = -0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.16714	-0.13774	-0.16274	-0.11274	20786
-0.03169	-0.01521	-0.04021	0.00979	29885

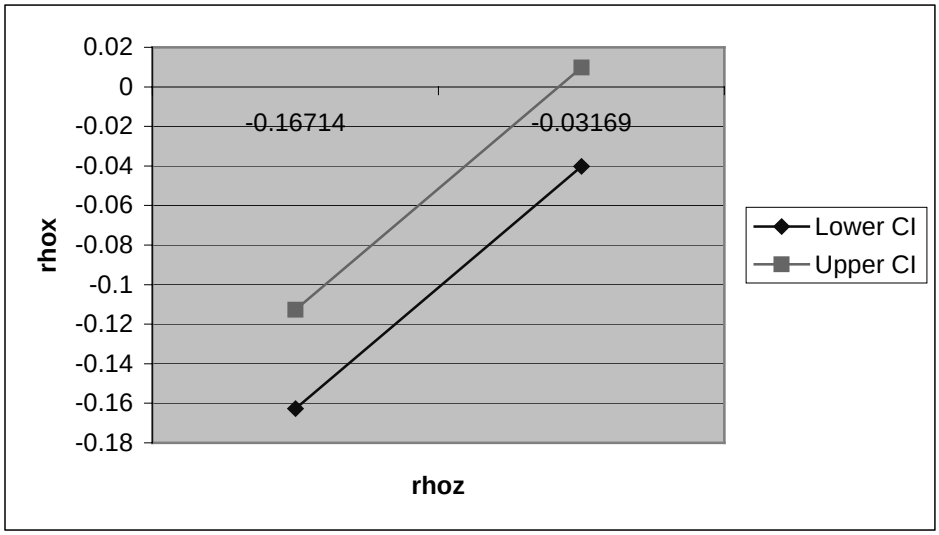

Figure 4.8: First iteration with correlation = –0.1 for 2 Gammas

L = -0.15306

U = -0.09779

U-L = 0.055273

Second Iteration:

Table 4.11: Second iteration with correlation = -0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.15306	-0.12173	-0.13423	-0.10923	85375
-0.09779	-0.07473	-0.08723	-0.06223	98781

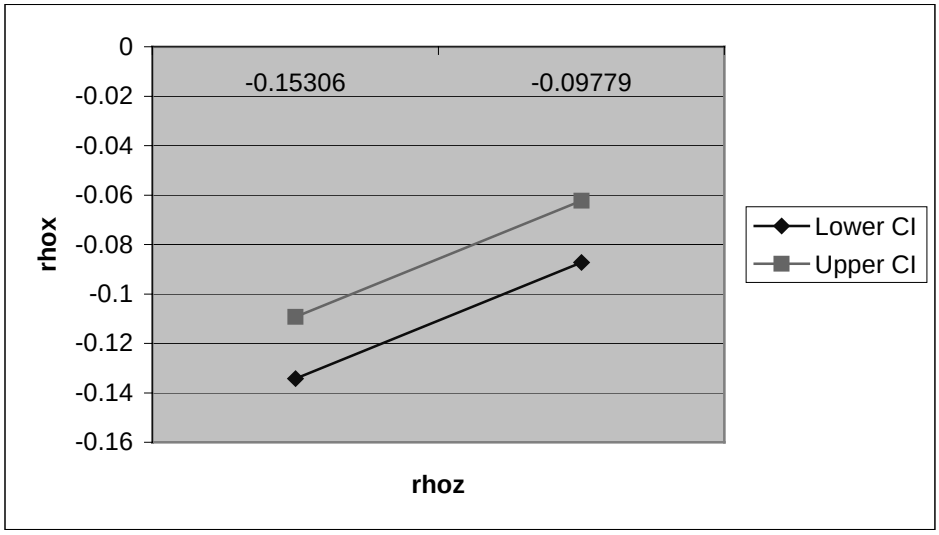

Figure 4.9: Second iteration with correlation = –0.1 for 2 Gammas

L = -0.14221

U = 0.020921

U – L = 0.029401

Third Iteration:

Table 4.12: Third iteration with correlation = -0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
-0.14221	-0.1103	-0.11655	-0.10405	356297
-0.11281	-0.09307	-0.09932	-0.08682	381404

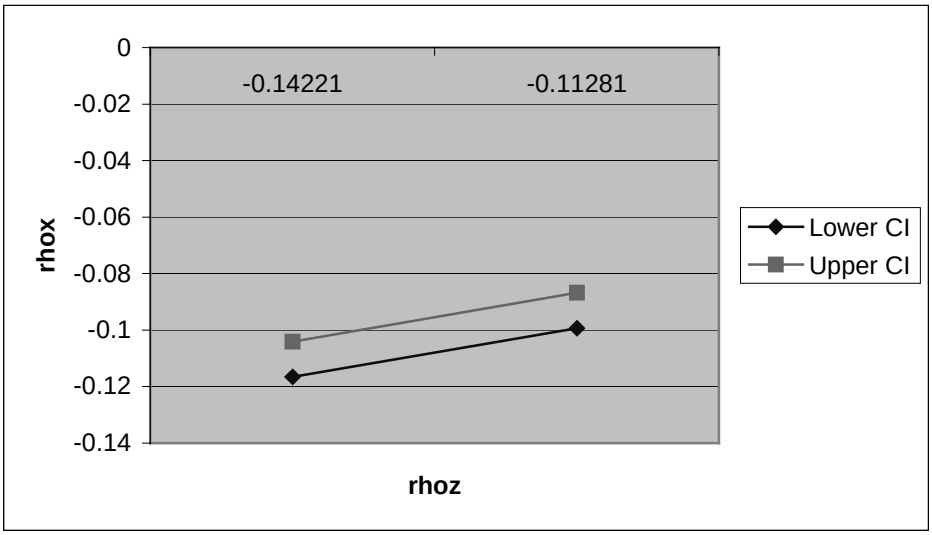

Figure 4.10: Third iteration with correlation = –0.1 for 2 Gammas

L = -0.1353

U = -0.11397

U-L = 0.021334

Final Iteration:

Table 4.13: Final iteration with correlation = -0.1 for 2 Gammas

Rhoz	Rhox	N
-0.1353	-0.10507	362532
-0.11397	-0.09392	380649

Estimated rhox = -0.1256

The time required for this run was 1 hour, 43 minutes, and 32 seconds.

4.1.3 Two Gammas with correlation = 0.1

Initial:

Table 4.14: Initial values with positive correlation for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0	-0.01992	-0.06992	0.03008	7516
0.25	0.20212	0.15212	0.25212	5263
0.5	0.45053	0.40053	0.50053	8576
0.75	0.76509	0.71509	0.81509	15460
1	1	0.95	1.05	1

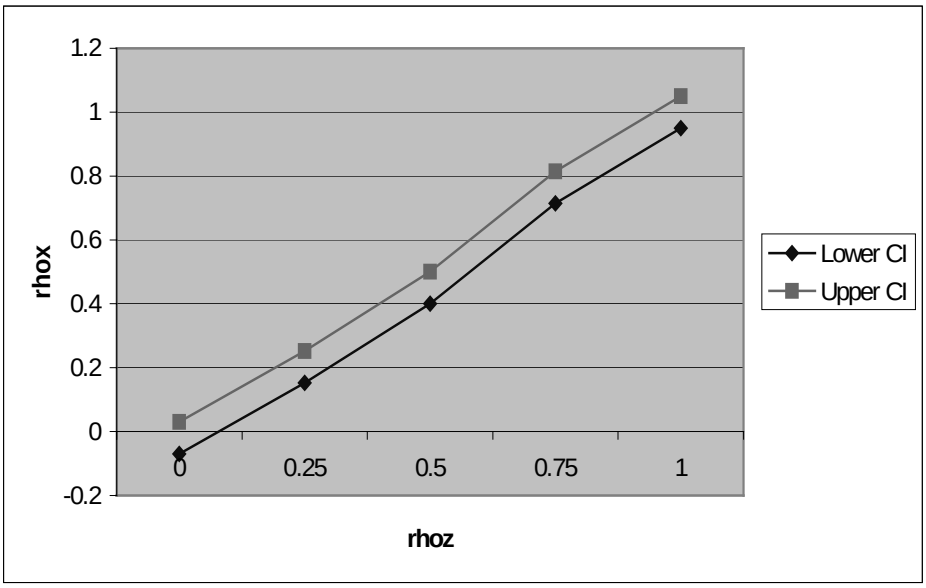

Figure 4.11: Initial correlation bands for 2 Gammas with positive correlation

L = 0.078725

U = 0.191317

U-L = 0.112592

First Iteration:

Table 4.15: First iteration with correlation = 0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.078725	0.061472	0.036472	0.086472	13704
0.191317	0.178281	0.153281	0.203281	19079

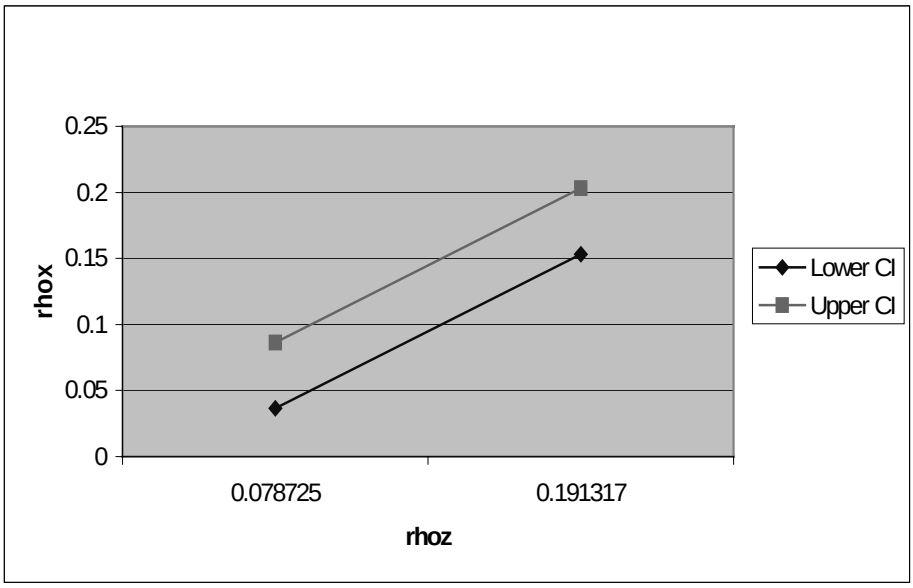

Figure 4.12: First iteration with correlation = 0.1 for 2 Gammas

L = 0.091765

U = 0.13996

U-L = 0.048195

Second Iteration:

Table 4.16: Second iteration with correlation = 0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.091765	0.080377	0.067877	0.092877	58435
0.13996	0.118918	0.106418	0.131418	63812

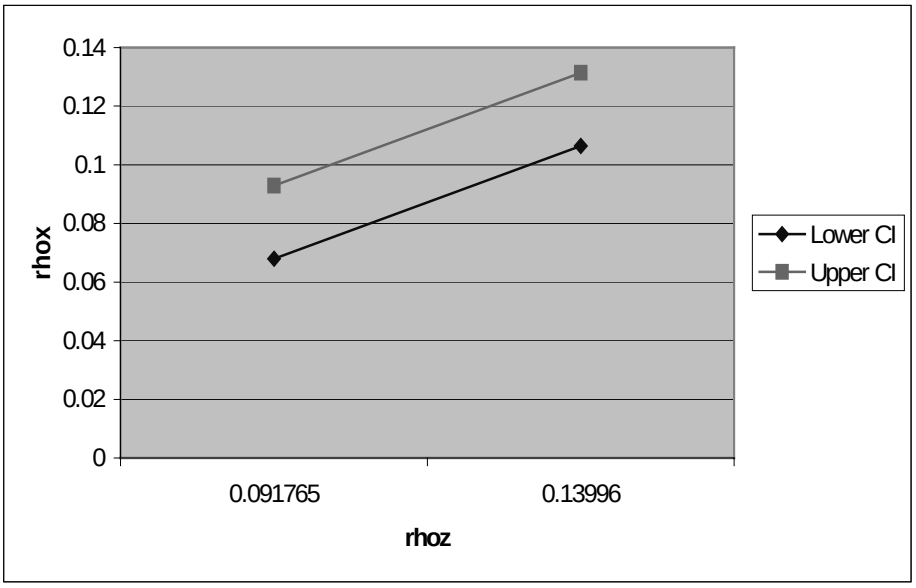

Figure 4.13: Second iteration with correlation = 0.1 for 2 Gammas

L = 0.100672

U = 0.131934

U-L = 0.031262

Third Iteration:

Table 4.17: Third iteration with correlation = 0.1 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.100672	0.083994	0.077744	0.090244	234061
0.131934	0.111768	0.105518	0.118018	255125

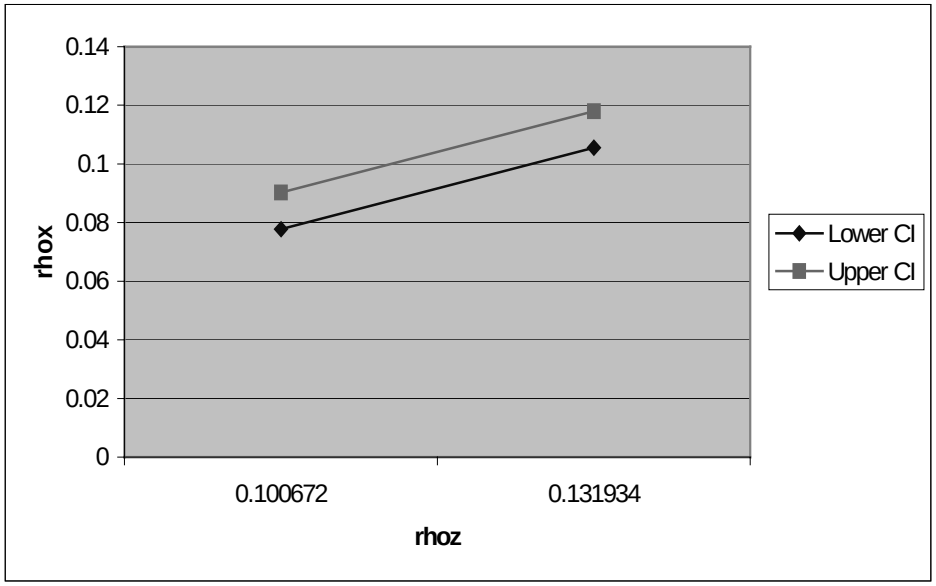

Figure 4.14: Third iteration with correlation = 0.1 for 2 Gammas

L = 0.111653

U = 0.125723

U-L = 0.01407

Final Iteration:

Table 4.18: Final iteration with correlation = 0.1 for 2 Gammas

Rhoz	Rhox	N
0.111653	0.093361	240595
0.125723	0.106814	251359

Estimated rhox = 0.118596

The time required to compute this correlation was 1 hour, 46 minutes, and 10 seconds.

4.1.4 Two Gammas with correlation = 0.6

Initial:

These results are the same as found in Section 4.1.4. See Table 4.14 and Figure 4.11 for details.

L = 0.579055

U = 0.658531

U-L = 0.079476

First Iteration:

Table 4.19: First iteration with correlation = 0.6 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.579055	0.541126	0.516126	0.566126	41513
0.658531	0.632211	0.607211	0.657211	49959

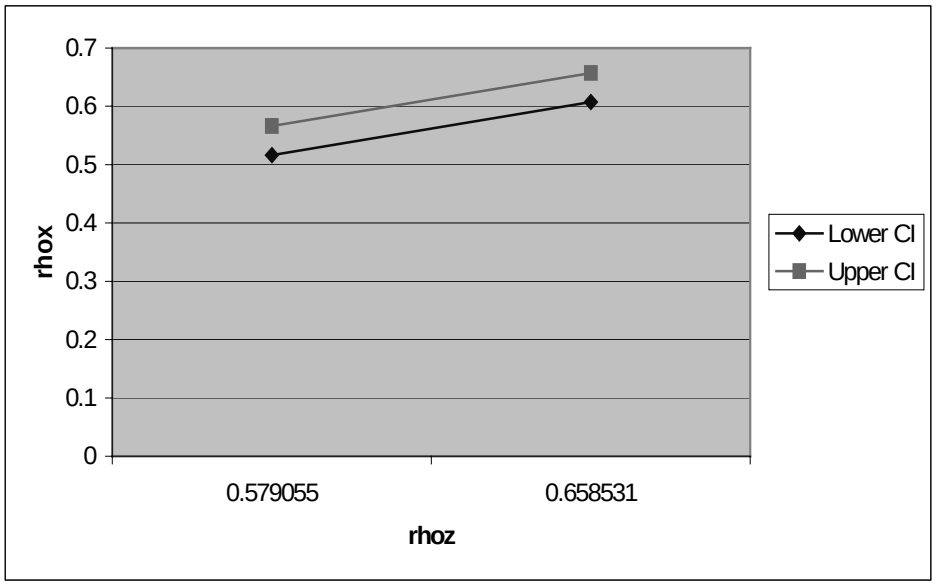

Figure 4.15: First iteration with correlation = 0.6 for 2 Gammas

L = 0.608612

U = 0.652239

U-L = 0.043627

Second Iteration:

Table 4.20: Second iteration with correlation = 0.6 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.608612	0.568569	0.556069	0.581069	175965
0.652239	0.614913	0.602413	0.627413	191711

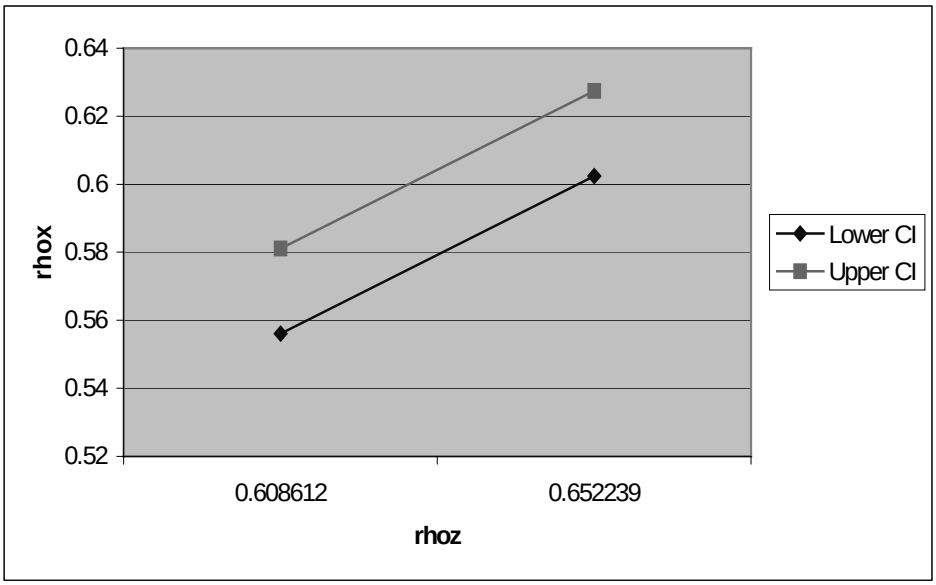

Figure 4.16: Second iteration with correlation = 0.6 for 2 Gammas

L = 0.626433

U = 0.649967

U-L = 0.023534

Final Iteration:

Table 4.21: Final iteration with correlation = 0.6 for 2 Gammas

Rhoz	Rhox	N
0.626433	0.587812	182353
0.649967	0.611456	190237

Estimated rhox = 0.638564

The time required to compute this value was 3 hours, 6 minutes, and 30 seconds.

4.1.5 Two Gammas with correlation = 0.9

Initial:

These results are the same as found in Section 4.1.4. See Table 4.14 and Figure 4.11 for details.

L = 0.840364

U= 0.946788

U-L = 0.106424

First Iteration:

Table 4.22: First iteration with correlation = 0.9 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.840364	0.826043	0.801043	0.851043	67566
0.946788	0.943137	0.918137	0.968137	79195

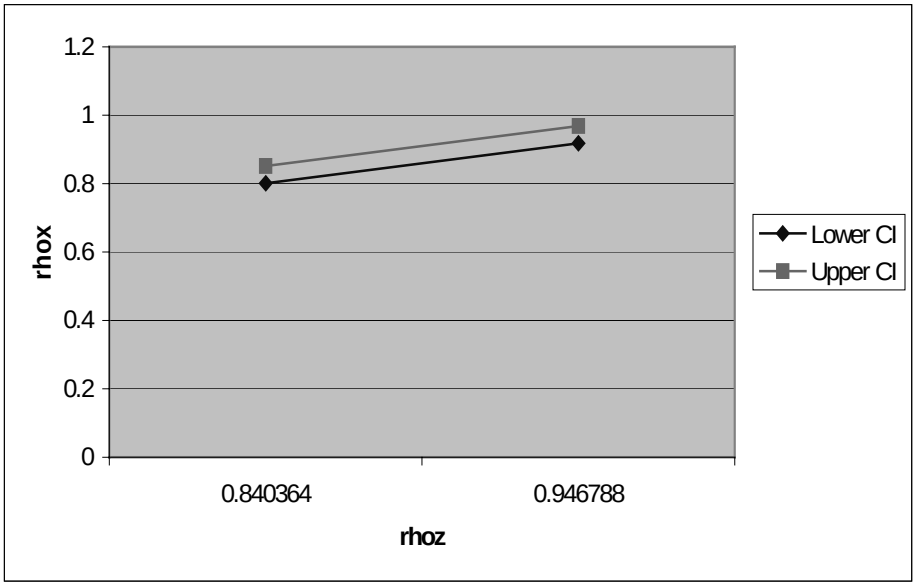

Figure 4.17: First iteration with correlation = 0.9 for 2 Gammas

L = 0.88286

U = 0.930304

U-L = 0.045444

Second Iteration:

Table 4.23: Second iteration with correlation = 0.9 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.88486	0.875483	0.862983	0.887983	289877
0.930304	0.914418	0.901918	0.926918	302139

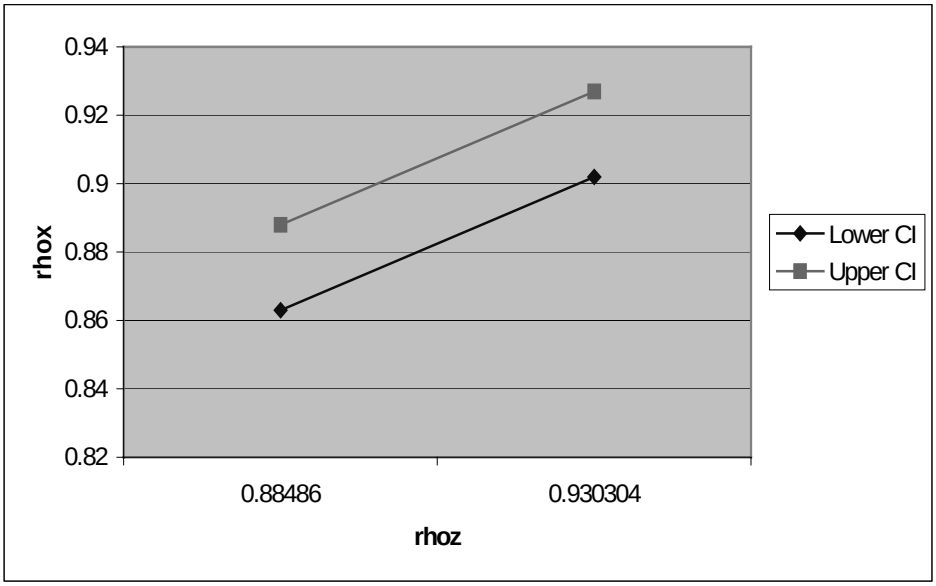

Figure 4.18: Second iteration with correlation = 0.9 for 2 Gammas

L = 0.898886

U = 0.928065

U-L = 0.029179

Third Iteration:

Table 4.24: Third iteration with correlation = 0.9 for 2 Gammas

Rhoz	Rhox	Lower CI	Upper CI	N
0.898886	0.881005	0.874755	0.887255	1168692
0.928065	0.915677	0.909427	0.921927	1225882

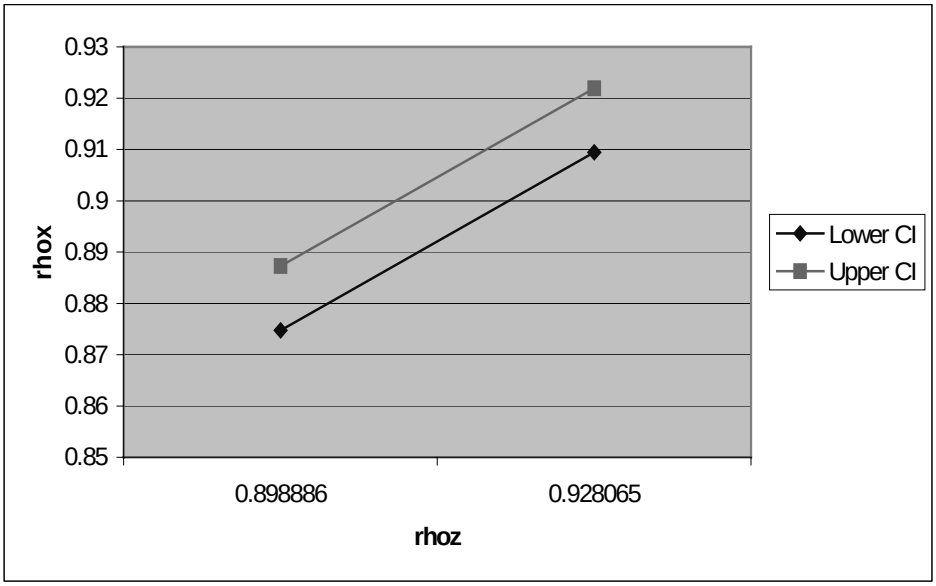

Figure 4.19: Third iteration with correlation = 0.9 for 2 Gammas

L = 0.909612

U = 0.920132

U-L = 0.01052

Final Iteration:

Table 4.25: Final iteration with correlation = 0.9 for 2 Gammas

Rhoz	Rhox	N
0.909612	0.893822	1189017
0.920132	0.9066	1210494

Estimated rhox = 0.914698

It took 2 hours, 4 minutes, and 52 seconds to compute this value.

4.2 Lognormal and Weibull

Examples in this section were generated with Lognormal and Weibull correlations at -0.3, -0.1, 0.1, 0.4, and 0.8. Each subsection reviews the results found. The confidence level with every example is 99%. The relative percent error always starts with 5%. This means $\delta = .05$. Recall that δ is divided in half at every iteration. In these examples, the calculations are performed stepwise by iteration. The stochastic root finding technique as implemented in the Excel macro will not iteratively show the user graphs and tables as presented in this section. These are given in this section to show the reader precisely how the algorithm works.

4.2.1 Lognormal and Weibull with correlation = -0.3

Initial:

Since the specified correlation is negative values or rhoz at –1. –0.75, -0.5, –0.25, and 0 are simulated.

Table 4.26: Initial values with negative correlation for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-1	-0.48705	-0.53705	-0.43705
-0.75	-0.46423	-0.51423	-0.41423
-0.5	-0.27419	-0.32419	-0.22419
-0.25	-0.15533	-0.20533	-0.10533
0	-0.10633	-0.15633	-0.05633

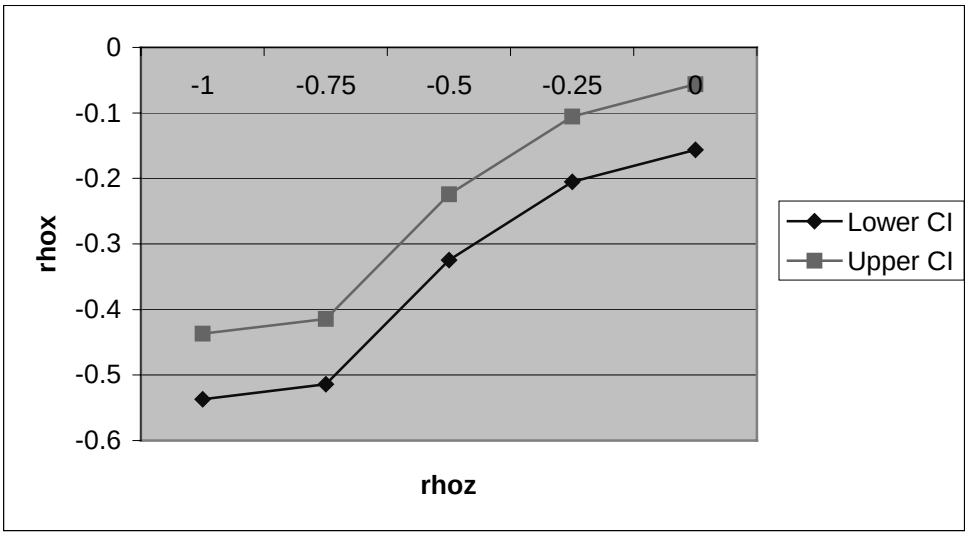

Figure 4.20: Initial correlation bands for Lognormal and Weibull distributions with negative correlation

L = -0.59973

U = -0.44912

U-L = 0.15061

First Iteration:

Table 4.27: First iteration with correlation = -0.3 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.59973	-0.33184	-0.35684	-0.30684
-0.44912	-0.27495	-0.29995	-0.24995

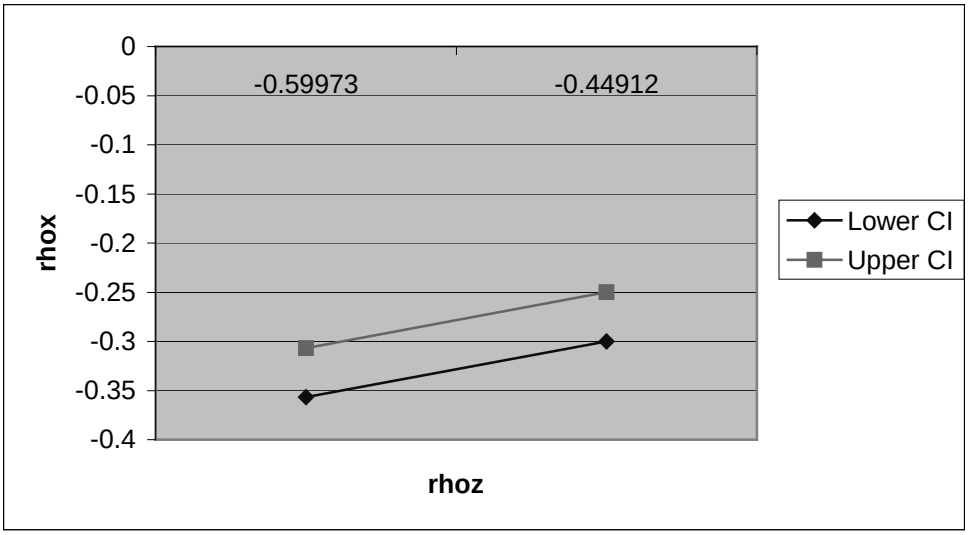

Figure 4.21: First iteration with correlation = -0.3 for Lognormal and Weibull

L = -0.58162

U = -0.44925

U-L = 0.13237

Second Iteration:

Table 4.28: Second iteration with correlation = -0.3 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.58162	-0.3309	-0.3434	-0.3184
-0.44925	-0.26339	-0.27589	-0.2509

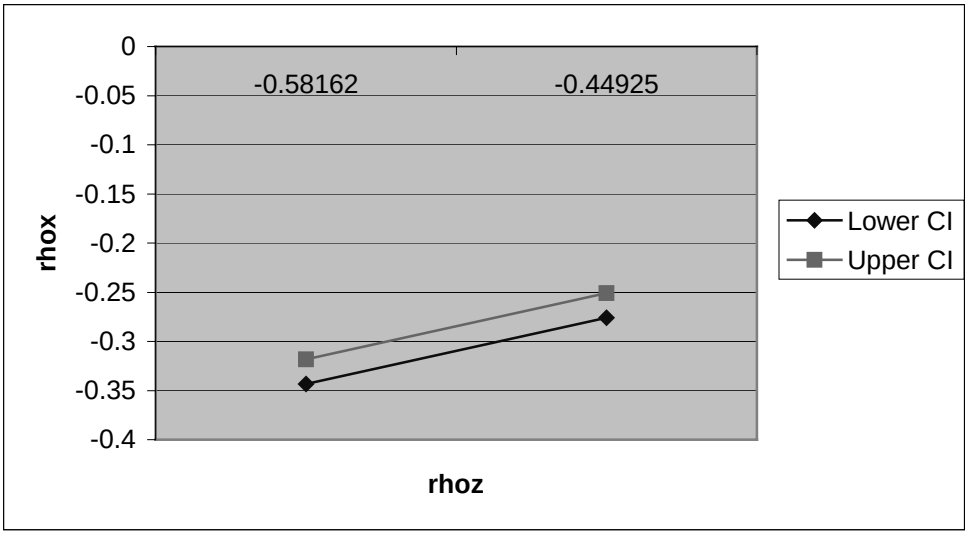

Figure 4.22: Second iteration with correlation = -0.3 for Lognormal and Weibull

L = -0.54554

U = -0.49682

U-L = 0.049019

Third Iteration:

Table 4.29: Third iteration with correlation = -0.3 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.54554	-0.31106	-0.31731	-0.30481
-0.49652	-0.29207	-0.29832	-0.28582

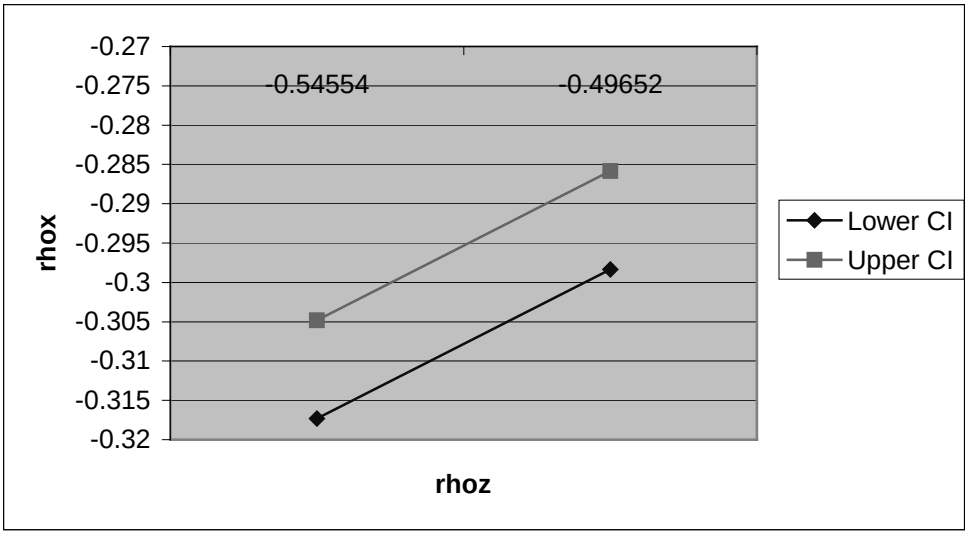

Figure 4.23: Third iteration with correlation = -0.3 for Lognormal and Weibull

L = -0.53312

U = -0.50086

U – L = 0.032267

This is the only case where it took more than three iterations to converge to a root. In this case, it will only take one more iteration.

Fourth Iteration:

Table 4.30: Fourth iteration with correlation = -0.3 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.53312	-0.3071	-0.31023	-0.30398
-0.50086	-0.29161	-0.29474	-0.28849

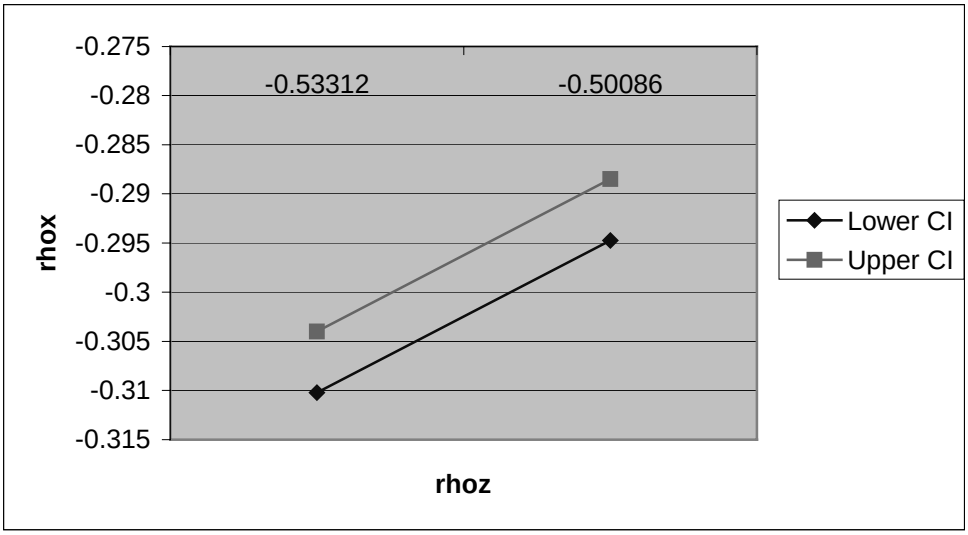

Figure 4.24: Fourth iteration with correlation = -0.3 for Lognormal and Weibull

L = -0.52484

U = -0.51183

U – L = 0.013016

Final Iteration:

Table 4.31: Final iteration with correlation = -0.3 for Lognormal and Weibull

Rhoz	Rhox
-0.52484	-0.30294
-0.51183	-0.29698

Estimated rhox = -0.51842

The time required for this run was 8 minutes and 37 seconds.

4.2.2 Lognormal and Weibull with correlation = -0.1

Initial:

The initial values with a correlation of -0.1 are the same as the initial values with a correlation of –0.3. For details see Table 4.26 and
Figure 4.20. The lower and upper limits produced are as follows:

$$L = -0.22281$$

$$U = 0$$

$$U - L = 0.22281$$

This is a case in which linear interpolation actually provides us with a value of 0.287398 for the upper limit, but since we know that a negative correlation was specified, we know that we cannot have a positive correlation for rhoz. Therefore 0 becomes the upper limit.
First Iteration:

Table 4.32: First iteration with correlation = -0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.22281	-0.15986	-0.18486	-0.13486
0	0.003	-0.022	0.028

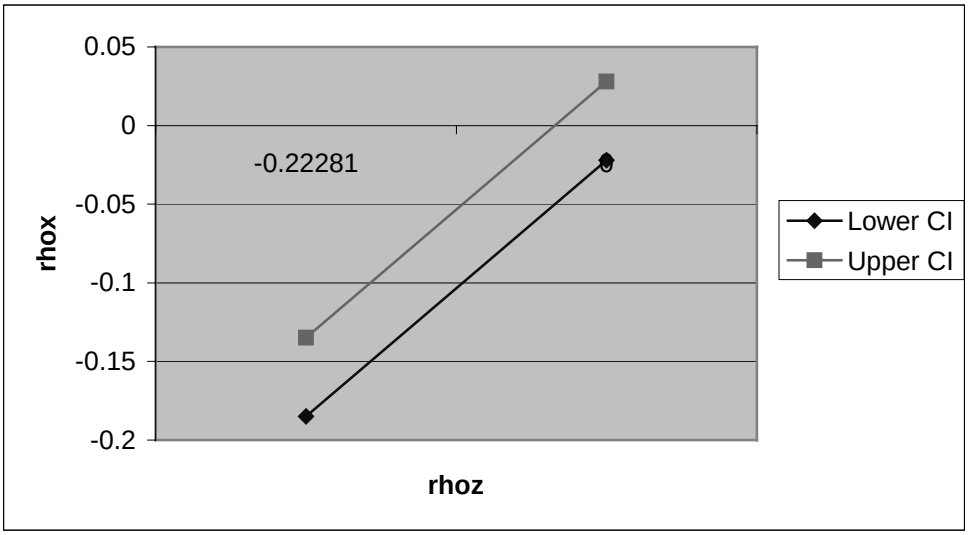

Figure 4.25: First iteration with correlation = -0.1 for Lognormal and Weibull

L = -0.17512

U = -0.10671

U – L = 0.068405

Second Iteration:

Table 4.33: Second iteration with correlation = -0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.17512	-0.11151	-0.12401	-0.09901
-0.10671	-0.07524	-0.08774	-0.06274

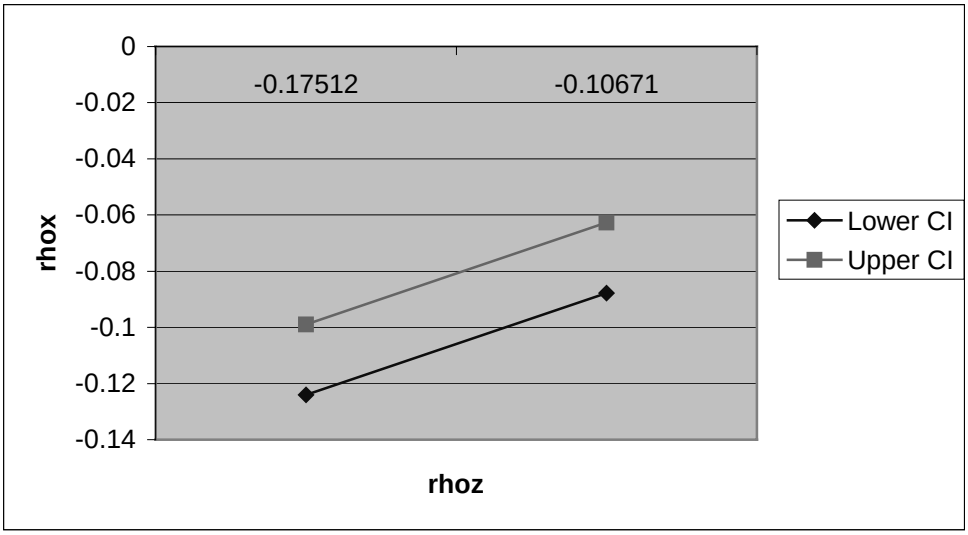

Figure 4.26: Second iteration with correlation = -0.1 for Lognormal and Weibull

L = -0.17512

U = -0.12983

U – L = 0.04529

Third Iteration:

Table 4.34: Third iteration with correlation = -0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
-0.17512	-0.11387	-0.12012	-0.10762
-0.12983	-0.08616	-0.09241	-0.07991

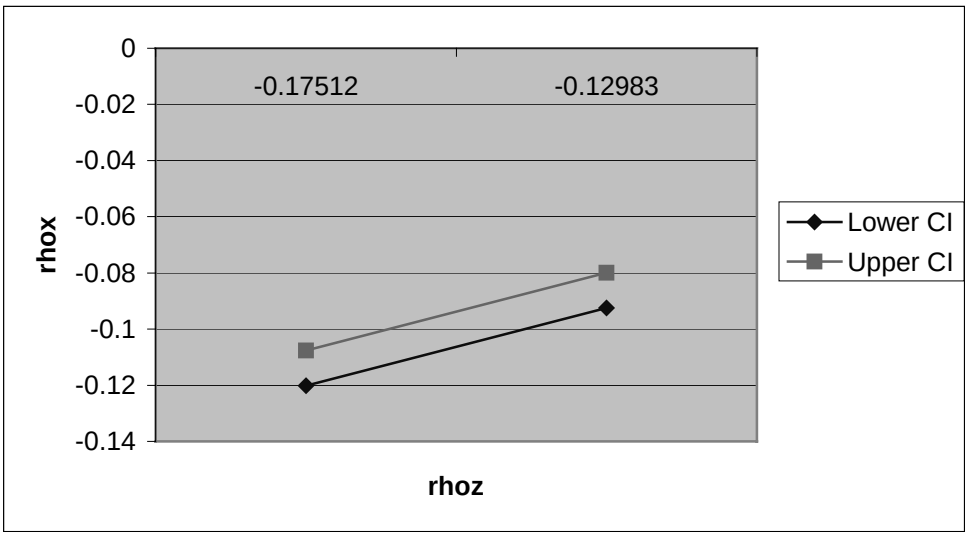

Figure 4.27: Third iteration with correlation = -0.1 for Lognormal and Weibull

L = -0.16267

U = -0.14224

U – L = 0.02043

Final Iteration:

Table 4.35: Final iteration with correlation = -0.1 for Lognormal and Weibull

Rhoz	Rhox
-0.16267	-0.10708
-0.14224	-0.09303

Estimated rhox = -0.15238

The time required to make this run was 34 minutes and 17 seconds.

4.2.3 Weibull and Lognormal with correlation = 0.1

Initial:

Since the specified correlation is positive, values of rhoz at 0, 0.25, 0.5, 0.75, and 1 are simulated.

Table 4.36: Initial values with positive correlation for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0	-0.1063	-0.1563	-0.0563
0.25	0.19743	0.14743	0.24743
0.5	0.40401	0.35401	0.45401
0.75	0.69	0.64	0.74
1	1	0.95	1.05

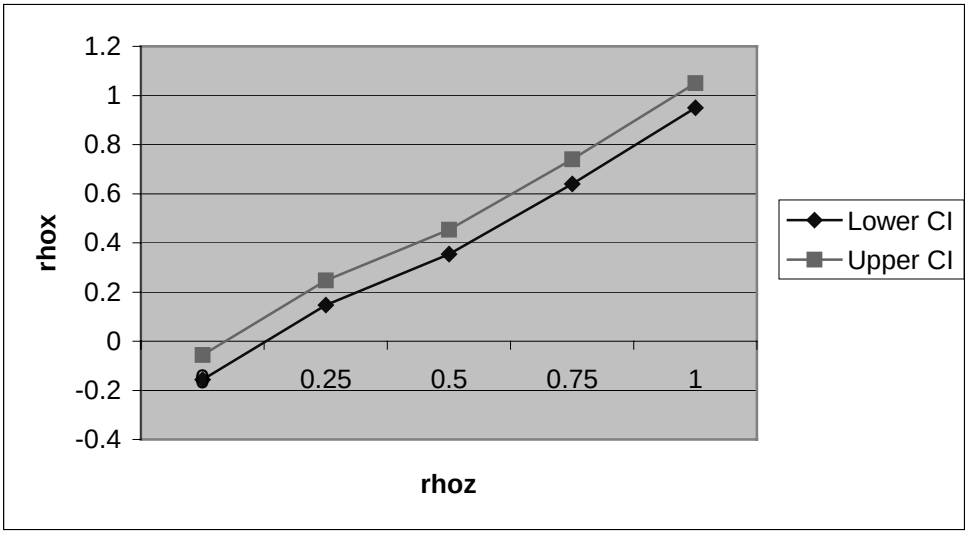

Figure 4.28: Initial correlation bands for Lognormal and Weibull distributions with positive correlation

L = 0.12865

U = 0.21096

U – L = 0.08231

First Iteration:

Table 4.37: First iteration with correlation = 0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.12865	0.09451	0.06951	0.11951
0.21096	0.16253	0.13753	0.18753

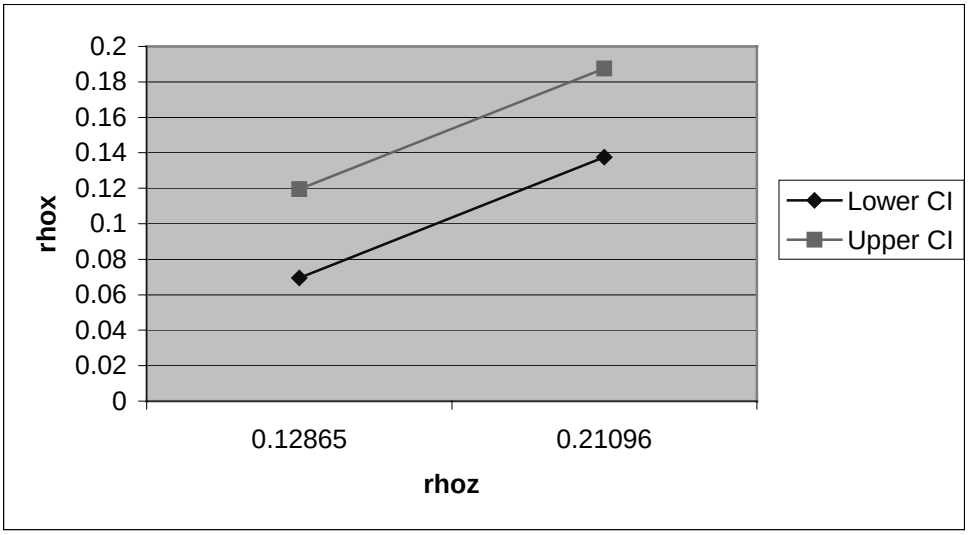

Figure 4.29: First iteration with correlation = 0.1 for Lognormal and Weibull

L = 0.12865

U = 0.165545

U – L = 0.036895

Second Iteration:

Table 4.38: Second iteration with correlation = 0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.12865	0.09729	0.08479	0.10979
0.165545	0.11246	0.09996	0.12496

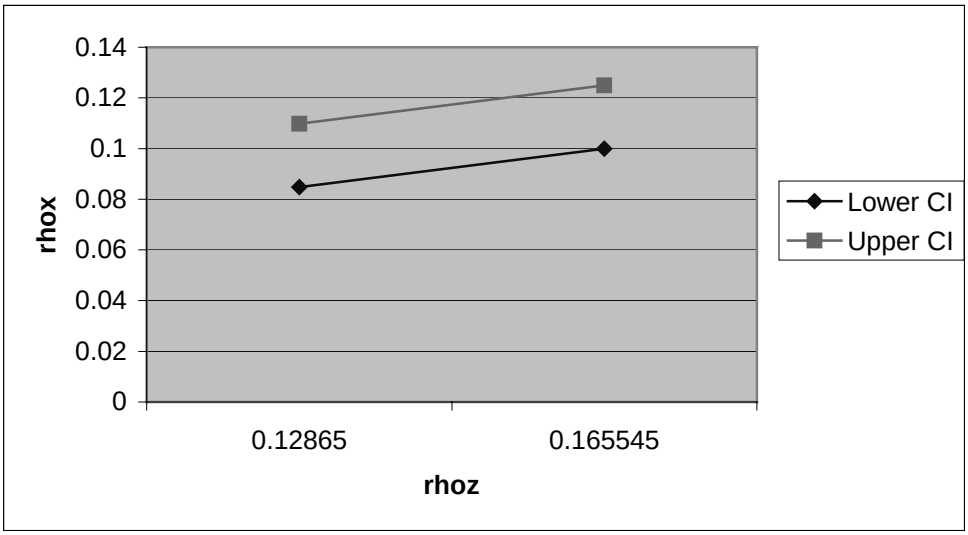

Figure 4.30: Second iteration with correlation = 0.1 for Lognormal and Weibull

L = 0.12865

U = 0.165642

U – L = 0.036992

Third Iteration:

Table 4.39: Third iteration with correlation = 0.1 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.12865	0.09289	0.08664	0.09914
0.165642	0.12265	0.1164	0.1289

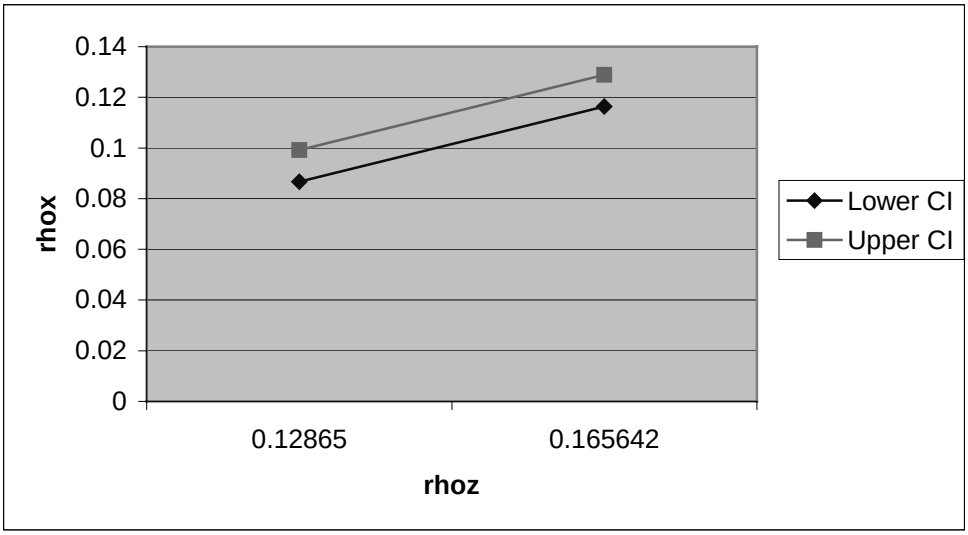

Figure 4.31: Third iteration with correlation = 0.1 for Lognormal and Weibull

L = 0.129719

U = 0.145257

U – L = 0.015538

Final Iteration:

Table 4.40: Final iteration with correlation = 0.1 for Lognormal and Weibull

Rhoz	Rhox
0.129719	0.09397
0.145257	0.10394

Estimated rhox = 0.139117

The time required to make this run was 53 minutes and 52 seconds.

4.2.4 Lognormal and Weibull with correlation = 0.4

Initial:

The initial values with a correlation of 0.4 are the same as with a correlation of 0.1. For details see Table 4.36 and Figure 4.28. The lower and upper limits produced are as follows:

$$L = 0.434638$$

$$U = 0.540202$$

$$U - L = 0.105564$$

First Iteration:

Table 4.41: First iteration with correlation = 0.4 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.434638	0.35198	0.32698	0.37698
0.540202	0.44499	0.41999	0.46999

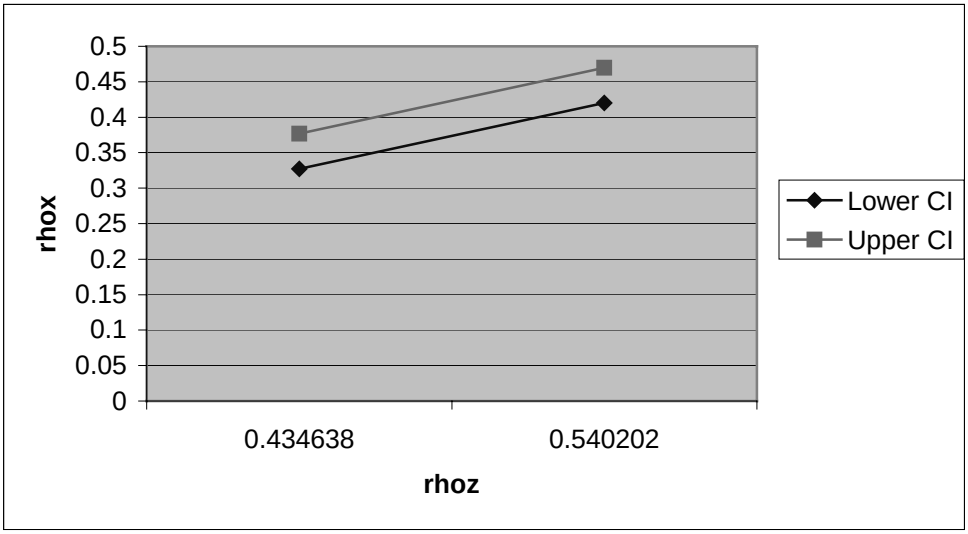

Figure 4.32: First iteration with correlation = 0.4 for Lognormal and Weibull

$$L = 0.460765$$

$$U = 0.517514$$

$$U - L = 0.056749$$

Second Iteration:

Table 4.42: Second iteration with correlation = 0.4 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.460765	0.369368	0.356868	0.381686
0.517514	0.424831	0.412331	0.424831

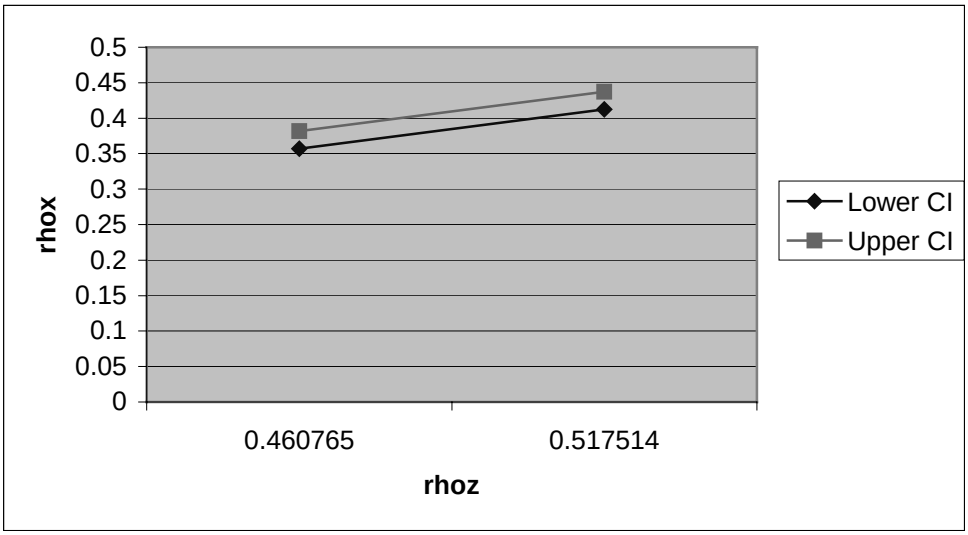

Figure 4.33: Second iteration with correlation = 0.4 for Lognormal and Weibull

L = 0.492107

U = 0.504897

U – L = 0.01279

Final Iteration:

Table 4.43: Final iteration with correlation = 0.4 for Lognormal and Weibull

Rhoz	Rhox
0.492107	0.397145
0.504897	0.413337

Estimated rhox = 0.494362

The time required to make this run was 1 hour, 11 minutes, and 17 seconds.

4.2.5 Lognormal and Weibull with correlation = 0.9

Initial:

The initial values are the same as the initial values for the other positive correlations. For details see Table 4.36 and Figure 4.28. The lower and upper limits generated are as follows:

$$L = 0.879032$$

$$U = 0.959677$$

$$U - L = 0.080645$$

We can see in this example that from the initial estimation we have a fairly close estimate, so it should not take many iterations to converge.

First Iteration:

Table 4.44: First iteration with correlation = 0.9 for Lognormal and Weibull

Rhoz	Rhox	Lower CI	Upper CI
0.879032	0.805398	0.780398	0.830398
0.959677	0.899956	0.874956	0.924956

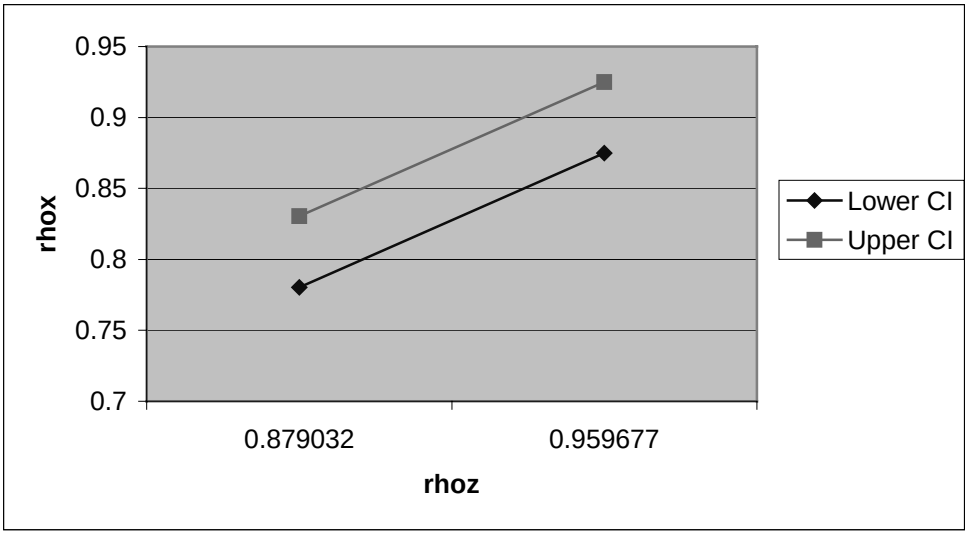

Figure 4.34: First iteration with correlation = 0.9 for Lognormal and Weibull

L = 0.938393

U = 0.959677

U – L = 0.021284

Final Iteration:

Table 4.45: Final iteration with correlation = 0.9 for Lognormal and Weibull

Rhoz	Rhox
0.938393	0.874259
0.959677	0.899596

Estimated rhox = 0.960016

The time required to make this run was 18 minutes and 22 seconds.

4.3 Uniform Example

This uniform example does not go through the various iterations performed, as above. It shows the results from running the stochastic root finding algorithm with various values for rhox. In the case of the uniform distribution, the actual relationship between $\rho_Z(i,j)$ and $\rho_X(i,j)$ is known and is given by the following equation:

$$\rho_X(i,j) = \frac{6}{\pi}\sin^{-1}\left[\frac{1}{2}\rho_Z(i,j)\right], \text{ so } \rho_Z(i,j) = 2*\sin\left[\frac{\pi}{6}\rho_X(i,j)\right]$$

The results presented in this section compare the output from the stochastic root finding algorithm to the actual known correlation, as given above. The results follow.

Table 4.46: Results from Uniform example

Rhox	Actual Rhoz	Estimated Rhoz	Absolute Error	Relative Error
-0.9	-0.90798	-0.90769	0.00029	0.00032
-0.75	-0.765366	-0.756034	0.009332	0.01219
-0.5	-0.517638	-0.51451	0.003128	0.00604
-0.25	-0.261052	-0.263399	0.002347	0.00899
-0.1	-0.104672	-0.108533	0.003861	0.03689
0.1	0.1046718	0.10645	0.001778	0.01699
0.25	0.2610522	0.26244	0.001388	0.00532
0.5	0.5176377	0.51914	0.001502	0.0029
0.75	0.7653663	0.7678	0.002434	0.00318
0.9	0.9079803	0.90976	0.00178	0.00196

"Actual Rhoz" above is the corresponding value of ρ_Z given ρ_X and using the formula above. "Estimated Rhoz" is the approximated value using the stochastic root finding algorithm. "Error" is found using the formula that was given previously in Section 3.3. In this case, "Error" is as follows:

$$Error = \frac{\left|EstimatedRhoz - ActualRhoz\right|}{\left|ActualRhoz\right|}.$$

The average error is expressed as:

$$Average\ Error = \frac{1}{n}\sum_{i=1}^{n} Error\ .$$

We find here that the average error is 0.009478. Previously we decided that an average error less than 0.1 would be a good estimate. In this case, we have an exceptional estimate. The remarkable observation is that we are comparing an estimate to a known relationship and achieving incredible results. This speaks highly for the simulation, as well as the stochastic root finding algorithm.

5 Conclusions and Recommendations

5.1 Conclusions

The main findings of the research can be summarized as follows:

1. A stochastic root finding technique has been developed that finds the root of the equation $\hat{\rho}_X = c_{i,j}(\hat{\rho}_Z)$ quickly. An efficient algorithm has been developed that uses antithetic variates to establish values of this equation.

2. The stochastic root finding technique is implemented in an Excel Macro, which means that this algorithm will be widely available. Using Excel also adds ease to coding by providing c.d.f.'s and their inverses, which would not be trivial to code in some cases.

3. We have experimented with this technique and found it to be accurate and practical.

5.2 Recommendations for Future Work

This work can be extended in various directions.

1. An investigation into the numerical stability of this technique as implemented by Excel would provide some insight to abnormal results occasionally found. Currently there is no concept of the numerical stability of the objects being reference in the code, such as inverse cumulative distributions. Although double precision was used, there is still room for error if these approximations used by Excel are not numerically stable.

2. A different sampling scheme could be implemented, in which the same random numbers are used in calculations. Currently for each ρ_Z value that is simulated, different random numbers are sampled, and their antithetic partners are generated. There is no reason to generate these random numbers over again, but we could generate these numbers simultaneously. This might reduce the run-time, as well as increasing the accuracy.

3. Instead of using a piecewise linear function to pass through the points of interest, a spline or some other nonlinear function could be used. The Gamma examples showed that linear bands were adequate, but the Lognormal and Weibull example, seen in Figure 4.20 illustrating the initial bands for negative correlations, shows that a better estimate might be obtained by using a spline.

4. The time it takes to run the program could be reduced in various ways. One way is to store the sum of values of P_j as the simulation is performed (as seen in Section 3.1). This could be easily implemented. Other ways could be explored.

5. An implementation using a stochastic root finding technique that does not use confidence bands could be explored.

6. Although the results shown here give the indication that the method of antithetic variates works when given a negative correlation, finding a proof for this case would be of interest.

Bibliography

1. P. Bratley, B. L. Fox, and L. E. Schrage. *A Guide to Simulation*, Second Edition, New York: Springer-Verlag, 1987.

2. M.C. Cario and B. L. Nelson. Autoregressive to Anything: Time Series Input Processes for Simulation, *Operation Research Letters* **19**, 51-58, 1996.

3. M.C. Cario and B. L. Nelson. Numerical Methods for Fitting and Simulating Autoregressive-to-Anything Processes, *INFORMS Journal of Computing* **10**, 72-81, 1997.

4. M.C. Cario, B. L. Nelson, S. D. Roberts, and J. R. Wilson. Modeling and Generating Random Vectors with Arbitrary Marginal Distributions and Correlation Matrix, 2000.

5. H. Chen and B.W. Schmeiser. Stochastic Root Finding via Retrospective Approximation, *IIE Transactions* (to appear), 2000.

6. R. D. Cook and M. E. Johnson. A Family of Distributions for Modeling Non-elliptically Symmetric Multivariate Data, *J. R. Statist. Soc.*, **43**, 210-218, 1981.

7. L. Devroye. *Non-Uniform Random Variate Generation*, New York: Springer-Verlag, 1986.

8. N. Draper and H. Smith. *Applied Regression Analysis*, New York: John Wiley, 1998.

9. J. Galambos and I. Simonelli, *Bonferroni-type Inequalities with Applications*, New York: Springer, 1996.

10. M. E. Johnson. *Multivariate Statistical Simulation*, New York: John Wiley, 1987.

11. N. L. Johnson and S. Kotz. *Distributions in Statistics: Continuous Multivariate Distributions*, New York: John Wiley and Sons, Inc., 1972.

12. W. D. Kelton and A. M. Law. *Simulation Modeling and Analysis*, Second Edition, New York: McGraw-Hill, 1991.

13. G. A. Mirka, N. F. Glasscock, P. M. Stanfield, and J. R. Wilson. "An empirical approach to characterizing trunk muscle coactivation using simulation input modeling techniques", *Journal of Biomechanics* (to appear), 2000.

14. S. J. Press. *Applied Multivariate Analysis: Using Bayesian and Frequentist Methods of Inference*, Second Edition, Malabar, Florida: Robert E. Krieger Publishing Company, 1982..

15. Y. L. Tong, *The Multivariate Normal Distribution*, New York: Springer-Verlag, 1990.

16. M. A. Wagner and J. R. Wilson. Recent Developments in Input Modeling with Bezier Distributions, *Proceedings of the 1996 Winder Simulation Conference*, 1996.

Appendix A. Excel Macro Code

```
'Known Problems
'  Parameters need to be checked for inverses according to Excel
Option Explicit
Dim marginal() As frmMarginal
Dim row As Integer 'used for spreadsheet output
Dim used As Integer 'number of outputs within a simulation

Private Sub cmdExit_Click()
   End
End Sub

Private Sub cmdSpecifyCorrelations_Click()
    frmCorrelation.Show
    cmdSpecifyCorrelations.Visible = False
    cmdRun.Visible = True
    fraMethod.Visible = True
    lblMessage.Caption = "Choose Method and Click Run"
End Sub

Private Sub cmdSpecifyMarginals_Click()
   frmNORTA.Hide
   ReDim Preserve marginal(1 To Val(txtNumberMarginals))
   Dim i As Integer
   For i = 1 To Val(txtNumberMarginals) Step 1
      Set marginal(i) = New frmMarginal
      marginal(i).Caption = "Marginal Distribution " + VBA.Str(i)
      marginal(i).Show
   Next

   cmdSpecifyMarginals.Visible = False
   cmdSpecifyCorrelations.Visible = True
   txtNumberMarginals.Enabled = False
   lblMessage.Caption = "Click Specify Correlations"
   frmNORTA.Show
End Sub

Private Sub cmdStartOver_Click()
   cmdStartOver.Visible = False
   cmdRun.Visible = False
   lblNumberMarginals.Visible = True
   txtNumberMarginals.Visible = True
   txtNumberMarginals.Enabled = True
   cmdSpecifyMarginals.Visible = True
   lblMessage.Caption = "Enter Number of Marginals and Click Specify Marginals"
   used = 0
End Sub

Private Sub optQuadratic_Click()
   lblMessage.Caption = "Quadratic provides an approximate corresponding rho z."
```

```
End Sub

Private Sub optRootFind_Click()
   lblMessage.Caption = "Root Finding provides an almost exact corresponding rho z."
End Sub

Private Sub txtNumberMarginals_Change()
   If Val(txtNumberMarginals.Text) < 2 Then
      lblMessage.Caption = "Number of Marginals must be >=2"
   End If
End Sub

Private Sub cmdRun_Click()

If used = 0 Then
   'Worksheets.Add
   'ActiveSheet.Name = "NORTA"
   ActiveSheet.Columns("A:O").Clear
End If
ActiveSheet.Columns("A:I").ColumnWidth = 15
Range("A1").value = "Observaton"
Range("B1").value = "Marg 1 Q Plus"
Range("C1").value = "Marg 2 Q Plus"
Range("D1").value = "Marg 1 Q Minus"
Range("E1").value = "Marg 2 Q Minus"
Range("F1").value = "Rho i"
Range("G1").value = "Rhoz Mean"
Range("H1").value = "Rhoz Variance"
Range("I1").value = "Est Rhoz"

   Dim i As Integer
   Dim j As Integer
   For i = 1 To Val(txtNumberMarginals) - 1 Step 1
      For j = (i + 1) To Val(txtNumberMarginals)
         EstimateRhoz marginal(i), marginal(j), i, j
      Next j
   Next i

lblNumberMarginals.Visible = False
txtNumberMarginals.Visible = False
cmdRun.Visible = False
fraMethod.Visible = False
cmdStartOver.Visible = True
lblMessage.Caption = "Either Click Start Over or Quit"
End Sub

Private Sub EstimateRhoz(ByRef marginal1 As frmMarginal, ByRef marginal2 As frmMarginal, first As
Integer, second As Integer)

Dim desiredRhox As Double
desiredRhox = frmCorrelation.correlations(first, second)

Dim marginalPairName1 As String
Dim marginalPairName2 As String
marginalPairName1 = getMarginalPairName(marginal1)
```

```
marginalPairName2 = getMarginalPairName(marginal2)

row = 2 + used
Cells(row, 1).value = "Marginal" + VBA.Str(first) + ": " + marginalPairName1 + _
                " with Marginal" + VBA.Str(second) + ": " + marginalPairName2 + _
                " having Correlation = " + VBA.Str(desiredRhox)
used = used + 1

If optQuadratic = True Then
   frmQuadratic.Show
   ViaQuadraticMethod marginal1, marginal2, desiredRhox
Else
   ViaRootFind marginal1, marginal2, desiredRhox
End If

End Sub


Public Function getMarginalPairName(ByRef marginalPair As frmMarginal) As String
'Get the names of each of the marginals and their specified parameters
   Select Case marginalPair.fraMarginal.ActiveControl.Name
      Case "optNormal"
         getMarginalPairName = "Normal(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optExponential"
         getMarginalPairName = "Exponential(" + marginalPair.txtParam1.Text + ")"
      Case "optUniform"
         getMarginalPairName = "Uniform(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optBeta"
         getMarginalPairName = "Beta(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + _
                        marginalPair.txtParam3.Text + ", " + marginalPair.txtParam4.Text + ")"
      Case "optChiSquare"
         getMarginalPairName = "Chi-Square(" + marginalPair.txtParam1.Text + ")"
      Case "optF"
         getMarginalPairName = "F distribution(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optGamma"
         getMarginalPairName = "Gamma(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optLognormal"
         getMarginalPairName = "LogNormal(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optT"
         getMarginalPairName = "t distribution(" + marginalPair.txtParam1.Text + ")"
      Case "optTriangular"
         getMarginalPairName = "Triangular(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + _
                        marginalPair.txtParam3.Text + ")"
      Case "optWeibul"
         getMarginalPairName = "Weibul(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optErlang"
```

```
        getMarginalPairName = "Erlang(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optBoundedJohnson"
        getMarginalPairName = "Bounded Johnson(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + _
                    marginalPair.txtParam3.Text + ", " + marginalPair.txtParam4.Text + ")"
      Case "optUnboundedJohnson"
        getMarginalPairName = "Unbounded Johnson(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + _
                    marginalPair.txtParam3.Text + ", " + marginalPair.txtParam4.Text + ")"
      Case "optEmpiricalDiscrete"
        getMarginalPairName = "Empirical Discrete at " + marginalPair.txtEmpiricalRange.Text
      Case "optEmpiricalContinuous"
        getMarginalPairName = "Empirical Continuous at " + marginalPair.txtEmpiricalRange.Text
      Case "optDiscreteUniform"
        getMarginalPairName = "Discrete Uniform(" + marginalPair.txtParam1.Text + " to " +
marginalPair.txtParam2.Text + ")"
      Case "optBinomial"
        getMarginalPairName = "Binomial(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optNegativeBinomial"
        getMarginalPairName = "Negative Binomial(" + marginalPair.txtParam1.Text + ", " +
marginalPair.txtParam2.Text + ")"
      Case "optGeometric"
        getMarginalPairName = "Geometric(" + marginalPair.txtParam1.Text + ")"
      Case "optPoisson"
        getMarginalPairName = "Poisson(" + marginalPair.txtParam1.Text + ")"
      Case Else
        MsgBox "An error in obtaining the marginal pair name."
   End Select
End Function
Public Sub ViaRootFind(ByRef marginal1 As frmMarginal, ByRef marginal2 As frmMarginal, ByRef
desiredRhox As Double)

Dim bottom As Integer
Dim rhoz(1 To 5) As Double
Dim rhox(1 To 5) As Double
Dim delta As Double
Dim stdz As Double
Dim specifiedRhoz As Double
Dim estimatedRhox As Double


If desiredRhox >= 0 Then    'decide if desiredrhox is in bottom or top interval
                'if desiredrhox in [-1,0] then bottom = 1.  else desiredrhox
                'in [0,1], so bottom = 0.
   bottom = 0

   Else
      bottom = 1
End If

If bottom = 0 Then
   rhoz(1) = 0
   rhoz(2) = 0.25
```

```
    rhoz(3) = 0.5
    rhoz(4) = 0.75
    rhoz(5) = 1

    Else
        rhoz(1) = -1
        rhoz(2) = -0.75
        rhoz(3) = -0.5
        rhoz(4) = -0.25
        rhoz(5) = 0
End If


delta = 0.05    'set width of CI
            'let this be a public var b/c pass into simulate function

stdz = 2.58

'simulate at rhoz(1)
specifiedRhoz = rhoz(1)
Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
rhox(1) = estimatedRhox

'simulate at rhoz(2)
specifiedRhoz = rhoz(2)
Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
rhox(2) = estimatedRhox

'simulate at rhoz(3)
specifiedRhoz = rhoz(3)
Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
rhox(3) = estimatedRhox

'simulate at rhoz(4)
specifiedRhoz = rhoz(4)
Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
rhox(4) = estimatedRhox

'simulate at rhoz(5)
specifiedRhoz = rhoz(5)
Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
rhox(5) = estimatedRhox

'Now each band will be created.  Each lower band will have i = 1, while a
'upper band will have i=1.  j will desginate which rhoz value the band will start.
'For instance, if the combination is "14" then this indicates a lower band
'which starts at rhoz4.


Dim i As Integer
Dim j As Integer
Dim slope(1 To 2, 1 To 4) As Double
Dim intercept(1 To 2, 1 To 4) As Double
Dim x(1 To 2, 1 To 4) As Double
Dim count As Integer
```

```
Dim z(1 To 2) As Double
Dim U As Double
Dim L As Double


For i = 1 To 2 Step 1
   For j = 1 To 4 Step 1

   slope(i, j) = ((rhox(j + 1) - delta) - (rhox(j) - delta)) / 0.25

   If i = 1 Then
      intercept(i, j) = (rhox(j) - delta) - slope(i, j) * rhoz(j)

      Else
      intercept(i, j) = (rhox(j) + delta) - slope(i, j) * rhoz(j)

   End If

   x(i, j) = (desiredRhox - intercept(i, j)) / slope(i, j)

   Next
Next


count = 0   'This counts to see if U and L have been designated.  If count =1,
          'then a lower limit or an upper limit has been designated.

Dim Band As Integer

For i = 1 To 2              'This loop finds which values of x(i,j) fall in which intervals
   For j = 1 To 4
      If count < 2 And rhoz(j) < x(i, j) And x(i, j) < rhoz(j + 1) Then

         count = count + 1
         z(count) = x(i, j)
            If i = 1 Then
               Band = 1
            Else
               Band = 2
            End If

      End If
   Next
Next

Dim int_L As Double
Dim guess As Double
Dim int_U As Double
Dim counter As Integer

If count = 2 Then       'This means that U and L are both contained on the interval
                  'Otherwise, we need to go outside of the intervals to find
                  'where U and L lie.
   If z(2) > z(1) Then
```

```
        U = z(2)
        L = z(1)

        Else

        U = z(1)
        L = z(2)

    End If

    ElseIf count = 1 Then

        If bottom = 0 And Band = 1 Then     'If you're in the top half and you've assigned one value, which
                                            'falls on the lower interval (band = 1) then this must be an upper
                                            'limit.  Therefore, we designate L = 0.
            L = 0
            U = z(1)

        ElseIf bottom = 0 And Band = 2 Then     'If you're in the top half and you've assigned one value
which
                                                'falls on the upper interval (band = 2) then this must be a
                                                'lower limit.  So, U = 1
                L = z(1)
                U = 1

        ElseIf bottom = 1 And Band = 1 Then     'If you're in the lower half, and you've assigned one value
which
                                                'falls on the lower interval, then this must be an upper limit.
                                                'So, L = -1.
                L = -1
                U = z(1)

        ElseIf bottom = 1 And Band = 2 Then     'If you're in the lower half, and you've assigned one value
which
                                                'falls on the upper limit, then this must be a lower limit.  So,
                                                'U = 0.
                L = z(1)
                U = 0
        End If

    Else        'count = 0, so lower and upper limits must be the normal bounds for rhox
        If bottom = 1 Then
            If desiredRhox < (rhox(1) - delta) Then     'If the user value is less than the min val possible
                'Give an error message: you're out of bounds

            Else
                L = -1
                U = 0

            End If

        Else
            If desiredRhox > (rhox(5) + delta) Then     'If the user value is more than the max val possible
                'Give an error message: you're out of bounds
```

```
            Else
               L = 0
               U = 1

            End If

        End If
End If


Dim xL As Double
Dim xU As Double
Dim next_slope As Double
Dim integ(1 To 2) As Double
Dim next_x(1 To 2) As Double

Dim change As Integer

counter = 0

Do Until (U - L) < 0.025

   counter = counter + 1       'counter for number of iterations

   delta = delta / 2

   'simulate at L
   specifiedRhoz = L
   Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
   xL = estimatedRhox

   'simulate at U
   specifiedRhoz = U
   Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
   xU = estimatedRhox

   next_slope = (xL - xU) / (L - U)

   For i = 1 To 2
      If i = 1 Then
         integ(i) = (xL - delta) - next_slope * L

      Else
         integ(i) = (xL + delta) - next_slope * L

      End If

      next_x(i) = (desiredRhox - integ(i)) / next_slope

   Next i

   'The index notes the band we are using.  So if i = 2, wer'e on an upper band, which indicates
   'the x value is a lwoer limit.  If i = 1 we're on a lower limit, which indicates the value is
   'an upper limit.  The following finds what the new U and L should  be based on shere x(1) and
   'x(2) fall with respect to L and U.  "Change" is used for the final iteration, so that runs
```

```
    'won't have to be unnessarily replicated.

    If L < next_x(1) < U And L < next_x(2) < U Then    'If both x(1) and x(2) fall between L and U then
                                                       'they must be the new limits

        L = next_x(2)
        U = next_x(1)
        change = 2

        ElseIf L < next_x(1) < U And next_x(2) < L Then 'If only x(1) falls between L and U, then it will be a
                                                        'new limit, and since x(1) is on a lower band, then it
                                                        'must be an upper limit.  L stays the same.
            U = next_x(1)
            change = 12                         'L is same, U changed

        ElseIf L < next_x(2) < U And next_x(1) > U Then 'If only x(2) falls between L and U, then it will be
a new
                                                        'limit, and since x(2) is on an upper band, then it must
                                                        'be a lower limit.  U stays the same.
            L = next_x(2)
            change = 11                 'L changed, U is same

        Else

            change = 0                  'Any other case will mean that neither L nor U have changed

    End If

Loop


If change = 2 Then
    'simulate at L
    specifiedRhoz = L
    Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
    xL = estimatedRhox

    'simulate at U
    specifiedRhoz = U
    Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
    xU = estimatedRhox

    ElseIf change = 11 Then
        'simulate at L
        specifiedRhoz = L
        Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
        xL = estimatedRhox

        ElseIf change = 12 Then
            'simulate at U
            specifiedRhoz = U
            Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz
            xU = estimatedRhox

            Else
```

```
            'nothing
End If

Dim estrho As Double
Dim sloper As Double
Dim intercepter As Double



sloper = (xU - xL) / (U - L)

intercepter = xL - sloper * L

estrho = (desiredRhox - intercepter) / sloper

frmNORTA.Hide

Cells(row, 1).value = "Desired Rho(rhox) ="
Cells(row, 2).value = desiredRhox
Cells(row, 3).value = "NORTA Rho(rhoz) = "
Cells(row, 4).value = estrho

End


End Sub

Public Sub ViaQuadraticMethod(ByRef marginal1 As frmMarginal, ByRef marginal2 As frmMarginal,
ByRef desiredRhox As Double)

Dim confidenceLevel As Double
Dim stdz As Double
Dim delta As Double
Dim deltaprime As Double
confidenceLevel = Val(frmQuadratic.txtConfidence.Text)
stdz = Excel.WorksheetFunction.NormSInv(confidenceLevel)
delta = (0.5) * (0.1 ^ Val(frmQuadratic.cmbAccuracy.SelText))
'deltaprime = delta / (Stddev1 * Stddev2)
'delta = 0.00625

'set up values for rohz
Dim zRho(1 To 4) As Double
zRho(1) = 1
zRho(2) = 0.5
zRho(3) = -0.5
zRho(4) = -1

Dim estimatedRhox As Double

Dim SumY As Double
SumY = 0
Dim SumYX As Double
SumYX = 0
Dim SumYX2 As Double
SumYX2 = 0
```

```
Dim index As Integer
For index = 1 To 4 Step 1 'Four iteration are needed for rhox = 1, .5, -.5, -1

   Dim specifiedRhoz As Double
   specifiedRhoz = zRho(index)

   Simulate specifiedRhoz, estimatedRhox, marginal1, marginal2, delta, stdz

   'sums needed to fit the polynomial on rhox
   SumY = SumY + estimatedRhox
   SumYX = SumYX + (estimatedRhox * zRho(index))
   SumYX2 = SumYX2 + (estimatedRhox * zRho(index) * zRho(index))

Next index

row = row + 1

'Compute the A, B, C in A*x^2 + B*x + C where x is the rhox and the value of the polynomial is rhoz
Dim aCoeff As Double
Dim bCoeff As Double
Dim cCoeff As Double

Cells(row, 1).value = "A = "
aCoeff = (2 / 1.75) * SumYX2 - (1 / 1.75) * SumY
Cells(row, 2).value = aCoeff
Cells(row, 3).value = "B = "
bCoeff = SumYX / 2.5
Cells(row, 4).value = bCoeff
Cells(row, 5).value = "C = "
cCoeff = (4.25 / 8.75) * SumY - (5 / 8.75) * SumYX2
cCoeff = cCoeff - desiredRhox 'so the RHS of quadratic form is zero when solving with specified Rho
Cells(row, 6).value = cCoeff

row = row + 1

'use the polynomial to compute rhoz from the desired rho
Cells(row, 1).value = "Desired Rho(rhox) ="
Cells(row, 2).value = desiredRhox
Cells(row, 3).value = "NORTA Rho(rhoz) = "
Cells(row, 4).value = (-bCoeff + Sqr(bCoeff ^ 2 - 4# * aCoeff * cCoeff)) / (2# * aCoeff)
Cells(row, 5).value = (-bCoeff - Sqr(bCoeff ^ 2 - 4# * aCoeff * cCoeff)) / (2# * aCoeff)

End Sub

Public Sub Simulate(ByRef specifiedRhoz As Double, ByRef estimatedRhox As Double, ByRef marginal1
As frmMarginal, _
      ByRef marginal2 As frmMarginal, ByRef delta As Double, ByRef stdz As Double)

'Obtain pairs of correlated marginals through Monte Carlo samples

Dim marginal(1 To 2) As frmMarginal
Set marginal(1) = marginal1
Set marginal(2) = marginal2
```

```
Dim BigSign As Double
If specifiedRhoz < 0 Then
   BigSign = -1
Else
   BigSign = 1
End If

Dim output As Long
output = 1000 'prints out information every "output" iterations

Dim Mean1 As Double
Dim Mean2 As Double
Dim Stddev1 As Double
Dim Stddev2 As Double
Mean1 = marginal1.m
Stddev1 = marginal1.s
Mean2 = marginal2.m
Stddev2 = marginal2.s

Dim MarginalMin(1 To 2) As Double
MarginalMin(1) = marginal1.Mn
MarginalMin(2) = marginal2.Mn

Dim Mean1Mean2 As Double
Mean1Mean2 = (Mean1 - MarginalMin(1)) * (Mean2 - MarginalMin(2))
Dim Stddev1Stddev2 As Double
Stddev1Stddev2 = Stddev1 * Stddev2

Dim rhoi As Double
'Dim rhoMean As Double
estimatedRhox = 0
Dim rhoVariance As Double
Dim inc As Double
Dim Tot As Double

'rhoMean = 0
Tot = 0

Dim n0 As Long
Dim n As Double
n0 = 1      'Begin with first generation
'If frmCorrelation.cmbAccuracy.value = 1 Then
n = 10 'if accuracy = 1 digit, start with 10 iterations before testing stopping rule
'Else
'n = 5000 'if accuracy = 2 digits, start with 5000 before testing stopping rule
'End If

Dim counter As Long
counter = 0
Do  'Do until stopping rule completes

      counter = counter + 1

      'Get two random numbers
      Dim U1 As Double
```

```
        Dim U2 As Double
        Do
        U1 = Rnd
        Loop Until U1 < > 0
        Do
        U2 = Rnd
        Loop Until U2 < > 0


        'Compute two standard normals
        Dim stdZ1 As Double
        Dim stdZ2 As Double
        stdZ1 = Excel.WorksheetFunction.NormSInv(U1)
        stdZ2 = Excel.WorksheetFunction.NormSInv(U2)


        'Compute the two correlated bivarate normals from the standard normals and the target correlation
        'Compute the arguments for the original and anithetic arguments
        Dim yo(1 To 2) As Double
        Dim ya(1 To 2) As Double
        yo(1) = U1
        ya(1) = 1 - U1
        If specifiedRhoz > 0.99999 Then
           yo(2) = U1
           ya(2) = Excel.WorksheetFunction.NormSDist(-stdZ1)
        ElseIf specifiedRhoz < -0.99999 Then
           yo(2) = Excel.WorksheetFunction.NormSDist(-stdZ1)
           ya(2) = U1
        Else
           yo(2) = Excel.WorksheetFunction.NormSDist(specifiedRhoz * stdZ1 + ((1 - specifiedRhoz *
specifiedRhoz) ^ 0.5) * stdZ2)
           ya(2) = Excel.WorksheetFunction.NormSDist(specifiedRhoz * -stdZ1 + ((1 - specifiedRhoz *
specifiedRhoz) ^ 0.5) * -stdZ2)
        End If

        'Define some variables for use in computing various inverses
        Dim a As Double
        Dim b As Double
        Dim c As Double
        Dim d As Double
        Dim U As Double
        Dim z(1 To 2) As Double
        Dim x(1 To 2) As Double

        Dim sign As Integer
        Dim Qplus(1 To 2) As Double
        Dim Qminus(1 To 2) As Double
        Dim i As Integer

    'Compute the two correlated marginals from inverses, depending on distribution of marginals.
    'Compute Qplus for any case of desiredrhox.

    For i = 1 To 2 Step 1

            Select Case marginal(i).fraMarginal.ActiveControl.Name
```

```
        Case "optNormal"
            Qplus(i) = (Excel.WorksheetFunction.NormInv(yo(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optExponential"
            Qplus(i) = ((-Val(marginal(i).txtParam1.Text) * Excel.WorksheetFunction.Ln(1 - yo(i))) -
MarginalMin(i))

        Case "optUniform"
            a = Val(marginal(i).txtParam1.Text)
            b = Val(marginal(i).txtParam2.Text)
            Qplus(i) = (a + (b - a) * yo(i) - MarginalMin(i))

        Case "optBeta"
            Qplus(i) = (Excel.WorksheetFunction.BetaInv(yo(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text),
Val(marginal(i).txtParam3.Text), _
                Val(marginal(i).txtParam4.Text)) - MarginalMin(i))

        Case "optChiSquare"
            Qplus(i) = (Excel.WorksheetFunction.ChiInv(yo(i), _
                Val(marginal(i).txtParam1.Text)) - MarginalMin(i))

        Case "optF"
            Qplus(i) = (Excel.WorksheetFunction.FInv(yo(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optGamma"
            Qplus(i) = (Excel.WorksheetFunction.GammaInv(yo(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optLognormal"
            Qplus(i) = (Excel.WorksheetFunction.LogInv(yo(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optT"
            Qplus(i) = (Excel.WorksheetFunction.TInv(yo(i), _
                Val(marginal(i).txtParam1.Text)) - MarginalMin(i))

        Case "optTriangular"
            a = Val(marginal(i).txtParam1.Text)
            b = Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam3.Text)
            d = (b - a) / (c - a)
            U = Excel.WorksheetFunction.NormSDist(z(i))
            If U <= d Then
                x(i) = a + ((b - a) * (c - a) * U) ^ 0.5
            Else
                x(i) = c - ((c - b) * (c - a) * (1 - U)) ^ 0.5
            End If
            Qplus(i) = x(yo(i)) - MarginalMin(i)

        Case "optWeibul"
            Qplus(i) = (Val(marginal(i).txtParam2.Text) * _
```

```
            ((-Excel.WorksheetFunction.Ln(1# - yo(i))) ^ (1# / Val(marginal(i).txtParam1.Text))) -
MarginalMin(i))


        Case "optErlang"
            Qplus(i) = (Excel.WorksheetFunction.GammaInv(yo(i), _
                Val(marginal(i).txtParam2.Text), Val(marginal(i).txtParam1.Text)) - MarginalMin(i))


        'Johnson inverses use the standard normals directly
        Case "optBoundedJohnson"
            d = (z(i) - Val(marginal(i).txtParam1.Text)) / Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam4.Text) - Val(marginal(i).txtParam3.Text)
            x(i) = Val(marginal(i).txtParam3.Text) + c * (1 / (1 + VBA.Math.Exp(-d)))
            Qplus(i) = x(yo(i)) - MarginalMin(i)

        Case "optUnboundedJohnson"
            d = (z(i) - Val(marginal(i).txtParam1.Text)) / Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam4.Text) - Val(marginal(i).txtParam3.Text)
            x(i) = Val(marginal(i).txtParam3.Text) + c * ((VBA.Math.Exp(d) - VBA.Math.Exp(-d)) / 2)
            Qplus(i) = x(yo(i)) - MarginalMin(i)

        Case "optEmpiricalDiscrete"
            Dim tempRange As Range
            Set tempRange = marginal(i).DataRange
            U = Excel.WorksheetFunction.NormSDist(z(i))
            Dim iRow As Integer
            iRow = 0
            Dim iRows As Integer
            iRows = Val(marginal(i).txtNumber.Text)
            Dim cumProb As Double
            cumProb = 0#
            Do
                iRow = iRow + 1
                If iRow > iRows Then
                    MsgBox "An error occurred in computing the empirical discrete probabilities"
                End If
                cumProb = cumProb + tempRange.Cells(iRow, 2)
            Loop Until U <= cumProb
            x(i) = tempRange.Cells(iRow, 1)
            Qplus(i) = x(yo(i)) - MarginalMin(i)

        Case "optEmpiricalContinuous"    'this doesn't work
            'Dim tempRange As Range
            Set tempRange = marginal(i).DataRange
            U = Excel.WorksheetFunction.NormSDist(z(i))
            'Dim iRow As Integer
            iRow = 0
            'Dim iRows As Integer
            iRows = Val(marginal(i).txtNumber.Text)
            'Dim cumProb As Double
            cumProb = 0#
            Dim oldCumProb As Double
            Do
                iRow = iRow + 1
```

```
            If iRow > iRows Then
               MsgBox "An error occurred in computing the empirical discrete probabilities"
            End If
            oldCumProb = cumProb
            cumProb = cumProb + tempRange.Cells(iRow, 3)
         Loop Until U <= cumProb
         x(i) = tempRange.Cells(iRow, 1) + (U - oldCumProb) * (tempRange.Cells(iRow, 2) -
tempRange.Cells(iRow, 1)) / (cumProb - oldCumProb)

      Case "optDiscreteUniform"
         U = Excel.WorksheetFunction.NormSDist(z(i))
         x(i) = Val(marginal(i).txtParam1.Text) + Int(U * (Val(marginal(i).txtParam2.Text) - _
               Val(marginal(i).txtParam1.Text) + 1))
         Qplus(i) = x(yo(i)) - MarginalMin(i)

      Case "optBinomial"
         U = Excel.WorksheetFunction.NormSDist(z(i))
         Dim nSuccess As Integer
         nSuccess = 0
         Do While U > Excel.WorksheetFunction.BinomDist(nSuccess, Val(marginal(i).txtParam1.Text),
Val(marginal(i).txtParam2.Text), True)
            nSuccess = nSuccess + 1
         Loop
         x(i) = nSuccess
         Qplus(i) = x(yo(i)) - MarginalMin(i)

      Case "optNegativeBinomial"
         U = Excel.WorksheetFunction.NormSDist(z(i))
         Dim nFailures As Integer
         nFailures = 0
         cumProb = Excel.WorksheetFunction.NegBinomDist(0, Val(marginal(i).txtParam1.Text),
Val(marginal(i).txtParam2.Text))
         Do While U > cumProb
            nFailures = nFailures + 1
            cumProb = cumProb + Excel.WorksheetFunction.NegBinomDist(nFailures,
Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text))
         Loop
         x(i) = nFailures
         Qplus(i) = x(yo(i)) - MarginalMin(i)

      Case "optGeometric"
         U = Excel.WorksheetFunction.NormSDist(z(i))
         x(i) = Int(Excel.WorksheetFunction.Ln(1 - U) / Excel.WorksheetFunction.Ln(1 -
Val(marginal(i).txtParam1.Text)))
         Qplus(i) = x(yo(i)) - MarginalMin(i)

      Case "optPoisson"
         U = Excel.WorksheetFunction.NormSDist(z(i))
         Dim nEvents As Integer
         nEvents = 0
         Do While U > Excel.WorksheetFunction.Poisson(nEvents, Val(marginal(i).txtParam1.Text),
True)
            nEvents = nEvents + 1
         Loop
         x(i) = nEvents
```

```
            Qplus(i) = x(yo(i)) - MarginalMin(i)

        Case Else
            MsgBox "An error in computing the marginal variate"

        End Select
    Next i


If specifiedRhoz > 0 Then     'only compute Qminus if desiredrhox > 0.  This means using antithetic.

    'Compute the two correlated marginals from inverses, depending on distribution of marginals

    For i = 1 To 2 Step 1

        Select Case marginal(i).fraMarginal.ActiveControl.Name
        Case "optNormal"
            Qminus(i) = (Excel.WorksheetFunction.NormInv(ya(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optExponential"
            Qminus(i) = ((-Val(marginal(i).txtParam1.Text) * Excel.WorksheetFunction.Ln(1 - ya(i))) -
MarginalMin(i))

        Case "optUniform"
            a = Val(marginal(i).txtParam1.Text)
            b = Val(marginal(i).txtParam2.Text)
            Qminus(i) = (a + (b - a) * ya(i) - MarginalMin(i))

        Case "optBeta"
            Qminus(i) = (Excel.WorksheetFunction.BetaInv(ya(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text),
Val(marginal(i).txtParam3.Text), _
                Val(marginal(i).txtParam4.Text)) - MarginalMin(i))

        Case "optChiSquare"
            Qminus(i) = (Excel.WorksheetFunction.ChiInv(ya(i), _
                Val(marginal(i).txtParam1.Text)) - MarginalMin(i))

        Case "optF"
            Qminus(i) = (Excel.WorksheetFunction.FInv(ya(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optGamma"
            Qminus(i) = (Excel.WorksheetFunction.GammaInv(ya(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optLognormal"
            Qminus(i) = (Excel.WorksheetFunction.LogInv(ya(i), _
                Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text)) - MarginalMin(i))

        Case "optT"
            Qminus(i) = (Excel.WorksheetFunction.TInv(ya(i), _
                Val(marginal(i).txtParam1.Text)) - MarginalMin(i))
```

```
        Case "optTriangular"
            a = Val(marginal(i).txtParam1.Text)
            b = Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam3.Text)
            d = (b - a) / (c - a)
            U = Excel.WorksheetFunction.NormSDist(z(i))
            If U <= d Then
                x(i) = a + ((b - a) * (c - a) * U) ^ 0.5
            Else
                x(i) = c - ((c - b) * (c - a) * (1 - U)) ^ 0.5
            End If
            Qminus(i) = x(ya(i)) - MarginalMin(i)

        Case "optWeibul"
            Qminus(i) = (Val(marginal(i).txtParam2.Text) * _
                ((-Excel.WorksheetFunction.Ln(1# - ya(i))) ^ (1# / Val(marginal(i).txtParam1.Text))) -
MarginalMin(i))

        Case "optErlang"
            Qminus(i) = (Excel.WorksheetFunction.GammaInv(ya(i), _
                Val(marginal(i).txtParam2.Text), Val(marginal(i).txtParam1.Text)) - MarginalMin(i))

        'Johnson inverses use the standard normals directly
        Case "optBoundedJohnson"
            d = (z(i) - Val(marginal(i).txtParam1.Text)) / Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam4.Text) - Val(marginal(i).txtParam3.Text)
            x(i) = Val(marginal(i).txtParam3.Text) + c * (1 / (1 + VBA.Math.Exp(-d)))
            Qminus(i) = x(ya(i)) - MarginalMin(i)

        Case "optUnboundedJohnson"
            d = (z(i) - Val(marginal(i).txtParam1.Text)) / Val(marginal(i).txtParam2.Text)
            c = Val(marginal(i).txtParam4.Text) - Val(marginal(i).txtParam3.Text)
            x(i) = Val(marginal(i).txtParam3.Text) + c * ((VBA.Math.Exp(d) - VBA.Math.Exp(-d)) / 2)
            Qminus(i) = x(ya(i)) - MarginalMin(i)

        Case "optEmpiricalDiscrete"
            Set tempRange = marginal(i).DataRange
            U = Excel.WorksheetFunction.NormSDist(z(i))
            iRow = 0
            iRows = Val(marginal(i).txtNumber.Text)
            cumProb = 0#
            Do
                iRow = iRow + 1
                If iRow > iRows Then
                    MsgBox "An error occurred in computing the empirical discrete probabilities"
                End If
                cumProb = cumProb + tempRange.Cells(iRow, 2)
            Loop Until U <= cumProb
            x(i) = tempRange.Cells(iRow, 1)
            Qminus(i) = x(ya(i)) - MarginalMin(i)

        Case "optEmpiricalContinuous"    'this doesn't work
            'Dim tempRange As Range
            Set tempRange = marginal(i).DataRange
            U = Excel.WorksheetFunction.NormSDist(z(i))
```

```
            'Dim iRow As Integer
            iRow = 0
            'Dim iRows As Integer
            iRows = Val(marginal(i).txtNumber.Text)
            'Dim cumProb As Double
            cumProb = 0#
            Do
               iRow = iRow + 1
               If iRow > iRows Then
                  MsgBox "An error occurred in computing the empirical discrete probabilities"
               End If
               oldCumProb = cumProb
               cumProb = cumProb + tempRange.Cells(iRow, 3)
            Loop Until U <= cumProb
            x(i) = tempRange.Cells(iRow, 1) + (U - oldCumProb) * (tempRange.Cells(iRow, 2) -
tempRange.Cells(iRow, 1)) / (cumProb - oldCumProb)

         Case "optDiscreteUniform"
            U = Excel.WorksheetFunction.NormSDist(z(i))
            x(i) = Val(marginal(i).txtParam1.Text) + Int(U * (Val(marginal(i).txtParam2.Text) - _
                  Val(marginal(i).txtParam1.Text) + 1))
            Qminus(i) = x(ya(i)) - MarginalMin(i)

         Case "optBinomial"
            U = Excel.WorksheetFunction.NormSDist(z(i))
            nSuccess = 0
            Do While U > Excel.WorksheetFunction.BinomDist(nSuccess, Val(marginal(i).txtParam1.Text),
Val(marginal(i).txtParam2.Text), True)
               nSuccess = nSuccess + 1
            Loop
            x(i) = nSuccess
            Qminus(i) = x(ya(i)) - MarginalMin(i)

         Case "optNegativeBinomial"
            U = Excel.WorksheetFunction.NormSDist(z(i))
            nFailures = 0
            cumProb = Excel.WorksheetFunction.NegBinomDist(0, Val(marginal(i).txtParam1.Text),
Val(marginal(i).txtParam2.Text))
            Do While U > cumProb
               nFailures = nFailures + 1
               cumProb = cumProb + Excel.WorksheetFunction.NegBinomDist(nFailures,
Val(marginal(i).txtParam1.Text), Val(marginal(i).txtParam2.Text))
            Loop
            x(i) = nFailures
            Qminus(i) = x(ya(i)) - MarginalMin(i)

         Case "optGeometric"
            U = Excel.WorksheetFunction.NormSDist(z(i))
            x(i) = Int(Excel.WorksheetFunction.Ln(1 - U) / Excel.WorksheetFunction.Ln(1 -
Val(marginal(i).txtParam1.Text)))
            Qminus(i) = x(ya(i)) - MarginalMin(i)

         Case "optPoisson"
            U = Excel.WorksheetFunction.NormSDist(z(i))
            nEvents = 0
```

```
            Do While U > Excel.WorksheetFunction.Poisson(nEvents, Val(marginal(i).txtParam1.Text),
True)
               nEvents = nEvents + 1
            Loop
            x(i) = nEvents
            Qminus(i) = x(ya(i)) - MarginalMin(i)

         Case Else
            MsgBox "An error in computing the marginal variate"

         End Select
      Next i

      rhoi = (((Qplus(1) * Qplus(2) + Qminus(1) * Qminus(2)) / 2) - Mean1Mean2) / Stddev1Stddev2

      Else

         rhoi = (Qplus(1) * Qplus(2) - Mean1Mean2) / Stddev1Stddev2 'no antithetic

End If

      inc = rhoi - estimatedRhox
      estimatedRhox = estimatedRhox + inc / counter
      Tot = Tot + (counter - 1) * inc * (inc / counter)

      'estRho = rhoMean

      If (counter > 10) Then
         rhoVariance = Tot / (counter - 1)
         If (Abs(specifiedRhoz) >= 0.99999) And (Abs(estimatedRhox) >= 0.99999) Then
         'no need to do any more sampling -> have perfect correlation (floating point comparison >= .99999)
            If specifiedRhoz < 0 Then
               estimatedRhox = -1
            Else
               estimatedRhox = 1
            End If
            n = n0 - 1
         Else 'Get values to test Stopping Rule
            'n = rhoVariance * ((stdz / deltaprime) ^ 2) + 1
            n = rhoVariance * ((stdz / delta) ^ 2) + 1
            n0 = counter
         End If
      End If


      'Write out values to spreadsheet
      If (counter Mod output) = 0 Then

         row = row + 1

         Cells(row, 1).value = counter
         Cells(row, 2).value = Qplus(1)
         Cells(row, 3).value = Qplus(2)
         Cells(row, 4).value = Qminus(1)
         Cells(row, 5).value = Qminus(2)
```

```
        Cells(row, 6).value = rhoi
        Cells(row, 7).value = estimatedRhox
        Cells(row, 8).value = rhoVariance

        'If Rho < 0 Then
        '   sign = -1
        'Else
        '   sign = 1
        'End If
        'estRho = ((sign * rhoMean) - (Mean1 - MarginalMin(1)) * (Mean2 - MarginalMin(2))) / (Stddev1
* Stddev2)
        'Cells(row, 9).value = estRho


    End If


Loop Until n <= n0  'No more generations needed

row = row + 1

Cells(row, 1).value = "n = "
Cells(row, 2).value = n
Cells(row, 3).value = "n0 = "
Cells(row, 4).value = n0
Cells(row, 5).value = "Given Rhoz ="
Cells(row, 6).value = specifiedRhoz
Cells(row, 7).value = "Estimated Rhox ="
Cells(row, 8).value = estimatedRhox
'Cells(row, 9).value = "Adj. Rho = "
'Cells(row, 10).value = estRho

End Sub
```