Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

COMPARISON OF NEVADA SIMULATION TO MONTE CARLO SIMULATION

David J. Bryg

AlliedSignal Engines
PO 32272
Phoenix, AZ 85064-2272

ABSTRACT

A Monte Carlo simulation on a business jet engine thermodynamic performance was repeated using NEVADA Simulation in order to compare the accuracy and computing requirements of the different techniques.

NEVADA Simulation is a quadrature technique for calculating functions of random variables. In NEVADA (NumErical integration of Variance And probabilistic Dependence Analyzer) Simulation, variables are modeled with mixtures of 4-parameter random variables, called "Continuous Trees". Functions of random variables are calculated using gaussian quadrature. NEVADA Simulation can take advantage of the probabilistic independence in a decision problem while allowing for probabilistic dependence to achieve close to polynomial computational time complexity.

The results show that NEVADA Simulation was superior to Monte Carlo Simulation in terms of computational speed and accuracy. To achieve the accuracy of the NEVADA Simulation on specific fuel consumption, which took 3.9 *seconds* of 80486 time for NEVADA Simulation to solve, 187,000 Monte Carlo runs would be required. These 187,000 runs would require about 5.2 *days* of Cyber mainframe computer time.

1 INTRODUCTION

A Monte Carlo simulation was conducted on the TFE731 business jet engine thermodynamic performance manufactured by AlliedSignal Engines. We repeated this analysis using NEVADA Simulation to evaluate the relative merits of the two techniques.

NEVADA Simulation employs numerical integration to calculate functions of random variables. This technique contrasts to Monte Carlo simulation, which uses sampling to calculate functions of random variables.

NEVADA Simulation models uncertainty using mixtures of 4-parameter random variables, from the Johnson family of distributions. 4-parameter distributions can model a wide variety of skews and kurtoses, and thus can model, not only complex distributions of complex functions of distributions.

To repeat the Monte Carlo, the output of the Monte Carlo was taken and modeled using regression analysis. The input distributions were identical to that used in the original study.

2 GAS TURBINE FUNDAMENTALS

Business jet engines are called gas turbines, the "gas" term refering to the fact that air is the engine working fluid. In a gas turbine, air is compressed to high pressure in a high speed "Compressor", heated up to high temperatures in "Combustor", has work extracted in a "Turbine" and propels the plane by flowing through a high pressure "Nozzle". Important operating characteristics of a jet engine are its "Thrust" and its "Specific Fuel Consumption(sfc)", which is the fuel flow rate required to produce one pound of thrust.

Many variables effect these outputs. Uncertainty in compressor and turbine efficiencies and flows, uncertainties in pressure drops, and uncertainties in areas all effect "Thrust" and "sfc". Monte Carlo simulations have routinely been performed in support of risk analyses and component design.

Since NEVADA Simulation is a new technique, it is necessary to verify the accuracy and quantify the time savings, if any, it has to offer.

3 NEVADA SIMULATION

NEVADA Simulation employs the "Continuous Tree" algorithm developed by Bryg (1992). Figure 1 gives the "road map" of the "Continuous Tree" algorithm used by "Continuous Risk". Notice that the primary components are "fitting distributions" and "determining distributions for functions of random variables". The discussion that follows will explain the importance of each section. "Fitting Distributions" also includes mixtures of discrete and continuous distributions.

The following two sections go into more depth on each of these subsections. First, we will discuss how random variables are modeled in the "Continuous Tree" algorithm. Second, we discuss how we calculate and model functions of these random variables.

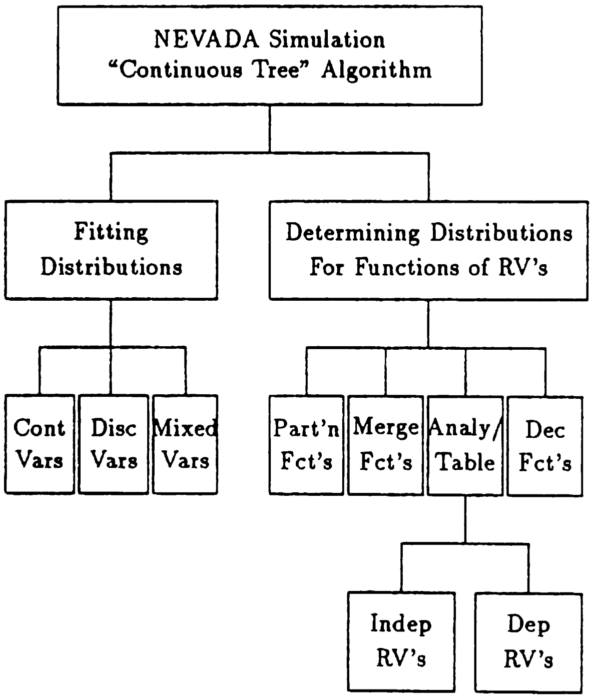


Figure 1: Outline of NEVADA Simulation - "Continuous Tree" Algorithm

4 FITTING DISTRIBUTIONS

4-parameter translational distributions are used to model random variables in the "Continuous Tree" algorithm. Translational distributions form a variety of skews and kurtoses by taking a simple distribution, such as a normal or a uniform and adding parameters to it to model skew and kurtosis. For a translational distribution, the ease of determining the percentiles of the distribution is the same as determining the percentiles of the underlying distribution, Elderton and Johnson (1969).

Of the non-uniform translational distributions, the Johnson S family has had more algorithms developed for fitting percentile and moment information to it than any other translational distribution. Bukac (1972), and Slifker and Shapiro (1980) give algorithms to fit Johnson distributions to percentile inputs.

The Johnson family of distributions is composed of three distributions: the S_B (logit-normal), the S_L (lognormal), and the S_U (sinh-normal). The S_B distributions model relatively flat, platykurtic, distributions. The S_L model bell-shaped, mesokurtic, distributions. The S_U distributions model peaked, leptokurtic, distributions.

The density functions for the three distributions are as follows:

$$S_B : P(x) = \frac{\delta}{\sqrt{2\pi}}\frac{1}{\lambda}\frac{1-\frac{x-\xi}{\lambda}}{\frac{x-\xi}{\lambda}}$$

$$\exp(-0.5(\gamma + \delta \ln(\frac{x-\xi}{\xi+\lambda-x}))^2), \xi \leq x \leq \xi + \lambda \quad (1)$$

$$S_L : P(x) = \frac{\delta}{\sqrt{2\pi}}\frac{\lambda}{\xi - x}$$

$$\exp(-0.5(\gamma + \lambda\delta \ln(\frac{\xi - x}{\lambda}))^2), \lambda\xi \leq \lambda x \qquad (2)$$

$$S_U : P(x) = \frac{\delta}{\sqrt{2\pi}}\frac{1}{\lambda}\sqrt{1 + (\frac{x-\xi}{\lambda})^2}$$

$$\exp(-0.5(\gamma + \delta \sinh^{-1}(\frac{x-\xi}{\xi+\lambda-x}))^2),$$

$$-\infty \leq x \leq \infty \qquad (3)$$

Since the Johnson is a 4-parameter distribution, it is necessary to model four moments when using it in the continuous decision tree method.

Sometimes, it is necessary to use a mixture of distributions to model distributions comprised of discretely conditioned events. For example, the distribution of fuel drop size at the combustor exit is a mixture of two distribution: one in which the fuel is completely evaporated and the other where is it not evaporated. The event "Combustor Exit Drop Size" could be modeled with the "Continuous Tree" in Figure 2 or the cumulative density function in Figure 3. It would more representative to model "Combustor Exit Drop Size" with a mixture of random variables rather than a single probability distribution, since it communicates the burned and unburned distributions.

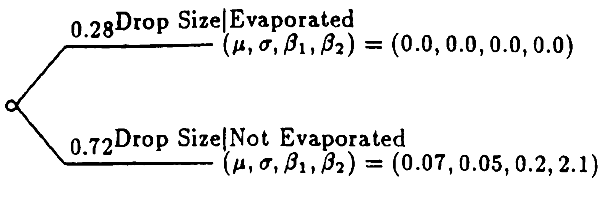

0.28 Drop Size|Evaporated
$(\mu, \sigma, \beta_1, \beta_2) = (0.0, 0.0, 0.0, 0.0)$
0.72 Drop Size|Not Evaporated
$(\mu, \sigma, \beta_1, \beta_2) = (0.07, 0.05, 0.2, 2.1)$

Figure 2: "Continuous Tree" of Combustor Exit Drop Size Distribution

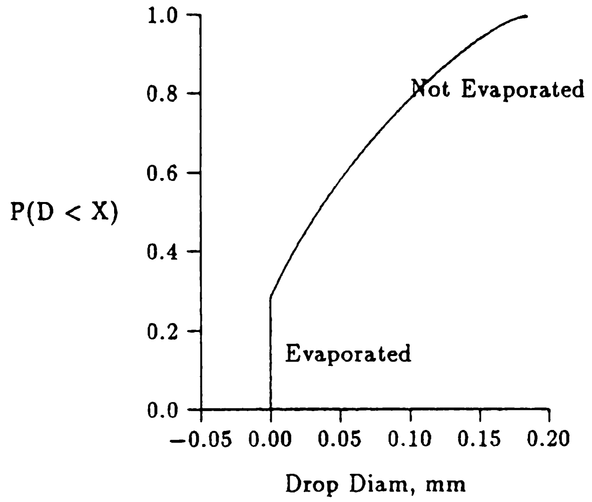


Figure 3: "Continuous Tree" Cumulative Density Function of Combustor Exit Drop Distribution

We will now illustrate how we "determine distributions for functions of random variables" in the "Continuous Tree" algorithm.

5 NEVADA SIMULATION

NEVADA Simulation is able to model probabilistic dependence using a concept called "Groups of Dependence". The idea of "Groups of Dependence" is to take advantage of the independence that exists in a decision problem, while allowing for one to model dependent variables. Figure 4 shows the process to calculate the combustor efficiency by combining P_{Fuel}, λ, D_{Fuel}, and τ. In Figure 4, ellipses denote variables, and circles denote operations. The operation "T" denotes an n-dimensional non-analytic, or table, function. The operation "f" denotes an algebraic function. Each of the random variables in Figure 4 could be a constant or a mixture of discrete and continuous variables.

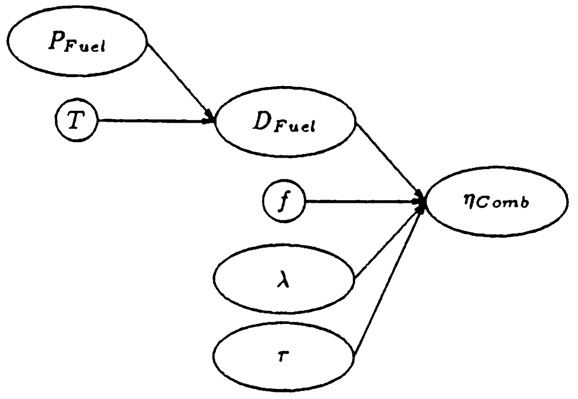
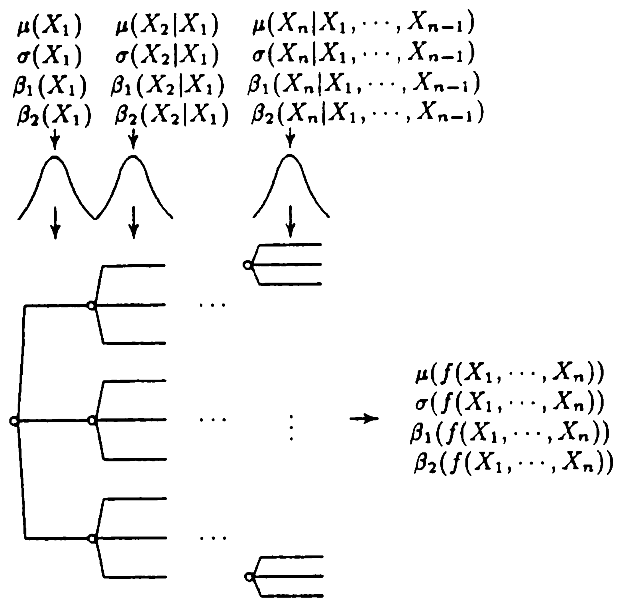


Figure 4: Example of "Groups of Dependence"

We look now at how functions of random variables are calculated. A "Continuous Tree" 'branch' is defined as one of the mixture distributions in an event. For each of the combinations of branches for each of the "Groups of Dependence", the following algorithm transforms the inputs of the "Group of Dependence" to the output continuous tree.

1. For each random variable, fit a continuous distribution.
2. For each distribution, form a discrete approximation.
3. Build a fully dependent, n-dimensional probability tree with a discrete approximation of the n dependent input random variables. Calculate the endpoints of the probability tree according to the operation involved.
4. Calculate the moments of the output distribution. From these moments, another continuous distribution could be modeled.

Figure 5 shows how a "Group of Dependence" tree is formed, operated on, and reduced to a few central moments. The distributions are ordered by dependence characteristics and the probability tree is developed sequentially. Each variable is dependent, or possibly dependent on all variables to the left of it.

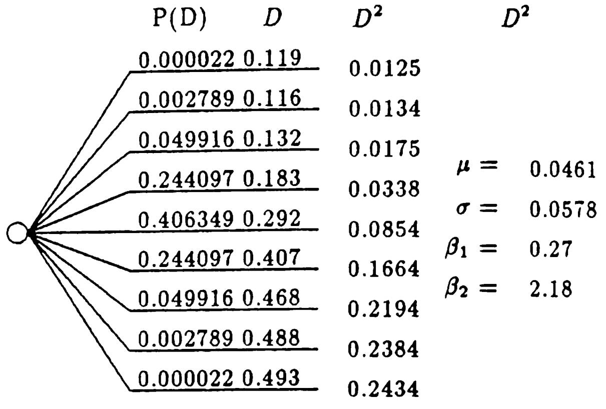


$\mu(X_1)$ $\sigma(X_1)$ $\beta_1(X_1)$ $\beta_2(X_1)$
$\mu(X_2|X_1)$ $\sigma(X_2|X_1)$ $\beta_1(X_2|X_1)$ $\beta_2(X_2|X_1)$
$\mu(X_n|X_1,\cdots,X_{n-1})$ $\sigma(X_n|X_1,\cdots,X_{n-1})$ $\beta_1(X_n|X_1,\cdots,X_{n-1})$ $\beta_2(X_n|X_1,\cdots,X_{n-1})$
$\mu(f(X_1,\cdots,X_n))$ $\sigma(f(X_1,\cdots,X_n))$ $\beta_1(f(X_1,\cdots,X_n))$ $\beta_2(f(X_1,\cdots,X_n))$

Figure 5: Combining "Groups of Dependence", General n-Dimensional Case

At this point we can illustrate how we use the "Groups of Dependence" concept to perform a simple, algebraic calculation. Figure 6 shows how we calculate the distribution of fuel spray diameter, D^2_{Fuel}, from the distribution of fuel spray diameter, D_{Fuel}. To accomplish this, it is necessary to approximate the distribution of D_{Fuel} with a discrete probability distribution, calculate the square of each of the discrete approximation legs, then calculate the first four central moments of the function D_{Fuel}. With these four moments, one can fit another 4-parameter continuous probability distribution to it.

The first column of numbers in Figure 6 shows the probability weights of the 9-point Gauss-Hermite approximation. The second column shows the 9-point Gauss-Hermite fractile of the drop diameter distribution. The third column is the square of the second column. The fourth column shows the first four central moments of the distribution of square roots of the value function.



P(D)	D	D^2	D^2
0.000022	0.119	0.0125	
0.002789	0.116	0.0134	
0.049916	0.132	0.0175	$\mu =$ 0.0461
0.244097	0.183	0.0338	$\sigma =$ 0.0578
0.406349	0.292	0.0854	$\beta_1 =$ 0.27
0.244097	0.407	0.1664	$\beta_2 =$ 2.18
0.049916	0.468	0.2194	
0.002789	0.488	0.2384	
0.000022	0.493	0.2434	

Figure 6: Performing a Unary Analytic Function on a Distribution

6 "CONTINUOUS RISK"

"Continuous Risk" is a 4GL probabilistic interpreted programming language that employs "Continuous Tree's" and "NEVADA Simulation" to model uncertainty and calculate functions of random variables. "Continuous Risk" is written in "C" for DOS

Each variable in "Continuous Risk" can be a mixture of random variables. `Arrays` and `Structures` can be modeled. "Continuous Risk" can model probabilistically dependent and independent random variables and supports programming language features such as `Functions`, `For` and `While` loops, and `If` statements. It can fit distributions by moments, data, percentiles, mixtures of distributions, or from named distributions. It has a wide range of analytical functions, including a Bayesian analyzer of arbitrary continuous distributions, an n-dimensional table interpolator of mixtures of continuous variables, a non-linear optimization, a non-linear solver, the ability to model of Markov processes with non-stationary transition matrices, and linear and logistic regression. It has flexible I/O options using "C"-like I/O commands, and a customized windowed text user interface generator.

7 ANALYSIS

Table 1 summarizes the input distributions. The original model used mixtures of two semi-normal distributions, one for the negative side of the mean, and one for the positive side. Since the output distributions are the sum of many distributions, it was reasonable to merge this mixture distribution into a single 4-parameter distribution. Table 1 gives the first four standardized moments of the input distributions.

Table 1: Moments of Input Distributions

Var	μ	σ	β_1	β_2
1	0.8455	0.0014	0	3
2	1.0000	0.0068	0	3
3	-0.0665	0.2550	-0.58	3.45
4	1.0918	0.0036	0	3
5	0.0000	0.6667	0	3
6	1.0000	0.0000	0	3
7	-0.1330	0.3481	-1.15	3.92
8	0.9865	0.0038	-2.69	5.41
9	0.1936	0.2155	-1.76	4.45
10	-0.0997	0.2215	-1.52	4.24
11	1.0199	0.0602	0.9	3.71
12	1.0000	0.0017	0	3

The NEVADA Simulation was run assuming the Monte Carlo was the gold standard. In reality, the Monte Carlo results will be distributed around some true value. To achieve a particular tolerance on the mean of a simulated variable, the number of runs required can be calculated by the following equation from Pfeifer, Bodily, and Frey (1991):

$$n = \frac{z^2}{(\frac{toler}{\sigma})^2} \qquad (4)$$

where z is the normal z parameter relating to the probability of being within the tolerance($z = 0.674$ for 50% chance of achieving tolerance) and σ is the standard deviation of the output parameter.

The simplest model we could use was a linear additive model. In this, each of the output parameters was calculated with the following equation:

$$y_j = \sum_{i=1}^{n}(x_{j_0}) + \frac{\partial x_j}{\partial y_i}(y_i - y_{i_0}) \qquad (5)$$

If a quadratic model were required, it would require and additional term:

$$y_j = \sum_{i=1}^{n}(x_{j_0}) + \frac{\partial x_j}{\partial y_i}(y_i - y_{i_0}) + \frac{\partial^2 x_j}{\partial y_i^2}(y_i - y_{i_0})^2 \quad (6)$$

For this particular study, the results of the linear model obviated the need for a quadratic model. Table 2 shows how accurate the NEVADA Simulation results were to the Monte Carlo. In all cases except the SURGELPC, the NEVADA Simulation results were within 0.01% (0.0001 fraction) of the Monte Carlo answer. The differences in the standard deviation were on the order of 1%.

The NEVADA Simulation results were achieved in 4 seconds of 80486 PC time. The 300 run Monte Carlo required 15 minutes.

Table 2: Comparison of NEVADA Simulation Results to Monte Carlo

	PCT ERROR	
Parameter	MEAN	STD DEV
SFC	-0.001	-0.8
FANSPEED	0.001	1.3
NMETER	0.001	1.4
HPSHAFT	-0.001	2.3
T45	0.007	-1.5
T41	0.004	-1.6
T30	0.004	1.9
SURGELPC	-0.139	-1.4

We can use Equation 4 to calculate how many Monte Carlo runs would be required to give Monte Carlo a 50% of being more accurate than the NEVADA Simulation. The Cyber run times were assuming that the 300 run Monte Carlo took 15 minutes to run. Since this model was linear, the more nonlinear a parameter was, the better Monte Carlo simulation was. However, even the best parameter for Monte Carlo, SURGELPC, would require 1.5 hours of Cyber time to achieve equal accuracy to NEVADA Simulation.

Table 3: Monte Carlo vs NEVADA Simulation Time

Output Parameter	Number of Monte Carlo Simulations to Equal NEVADA Accuracy	Required Cyber Run Time to Equal NEVADA Accuracy
SFC	187,736	156.4 hr
FANSPEED	2,280	1.9 hr
NMETER	2,051	1.7 hr
HPSHAFT	500,338	416.9 hr
T45	12,063	10.1 hr
T41	10,602	8.8 hr
T30	3,950	3.3 hr
SURGELPC	1,807	1.5 hr
NEVADA Simulation Time = 3.9 seconds		

We can also show how well the NEVADA Simulation models the distribution of output values. Figure 7 shows the Monte Carlo output of specific fuel consumption, standardized by its mean and standard deviation. The dashed line is the NEVADA Simulation Johnson distribution calculated and normalized by the Monte Carlo results. Notice the two cumulative density functions are almost identical.

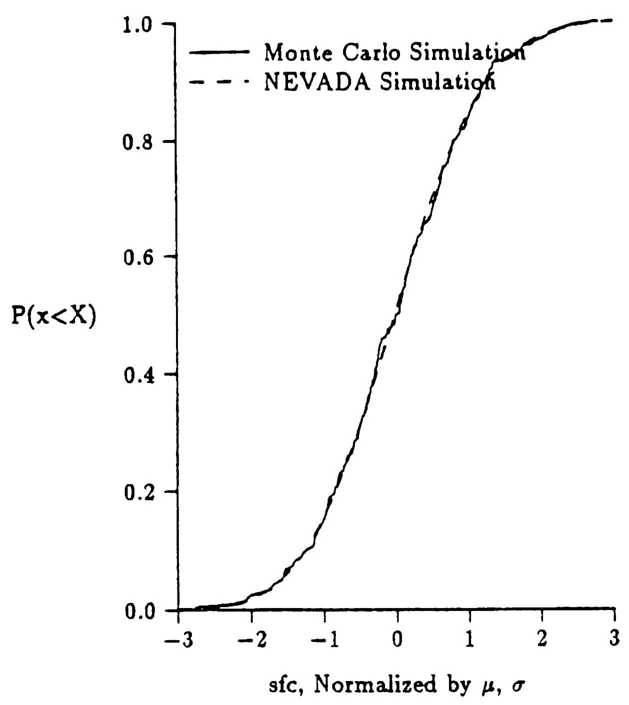


Figure 7: Comparison of sfc Distribution, Monte Carlo vs NEVADA Simulation

8 SUMMARY

For this simulation, NEVADA Simulation was superior in conducting engine thermodynamic risk analyses in terms of speed and accuracy. For this NEVADA Simulation, a linear additive model was sufficient to achieve a high accuracy. It is not known how great the input uncertainties would have to be to warrant a quadratic model. Also, one could make the claim that one could just perform a root-sum-square analysis and achieve the same accuracy with less computational effort. While a RSS analysis would give the same expected value, one could not necessarily assume the output distributions were gaussian. An assumption of gaussian output distributions would lead to large errors in other analyses, including calculations of engine reject rate.

Overall, it appears that NEVADA Simulation is very competitive compared to Monte Carlo simulation in terms of accuracy and speed.

REFERENCES

Bryg, D. J. 1992. Continuous Trees: A Quadrature Technique for Modeling Sequential Decision Problems with Dependent, Continuous Variables. Ph.D. Thesis. Decision and Information Systems Department. Arizona State University. Tempe, Arizona.

Pfeifer, P. E., S. E. Bodily, and S. C. Frey. 1991. Pearson-Tukey Three-Point Approximations Versus Monte Carlo Simulation, *Decision Sciences* 22: 74-90.

AUTHOR BIOGRAPHIES

DAVID J. BRYG is a Senior Engineer in the Performance and Operability Department at AlliedSignal Engines and a Clinical Assistant Professor of Internal Medicine at the University of Nevada, Reno. He is also President of NEVADA Simulations. His research interests lie in the development of decision aids involving combinations of empirical data and expert judgement in engineering, business, and medicine. He is a member of ASME, ORSA, TIMS, and SMDM.