Verification of the RMC-TotalRisk Software

C. Haden Smith, USACE Risk Management Center

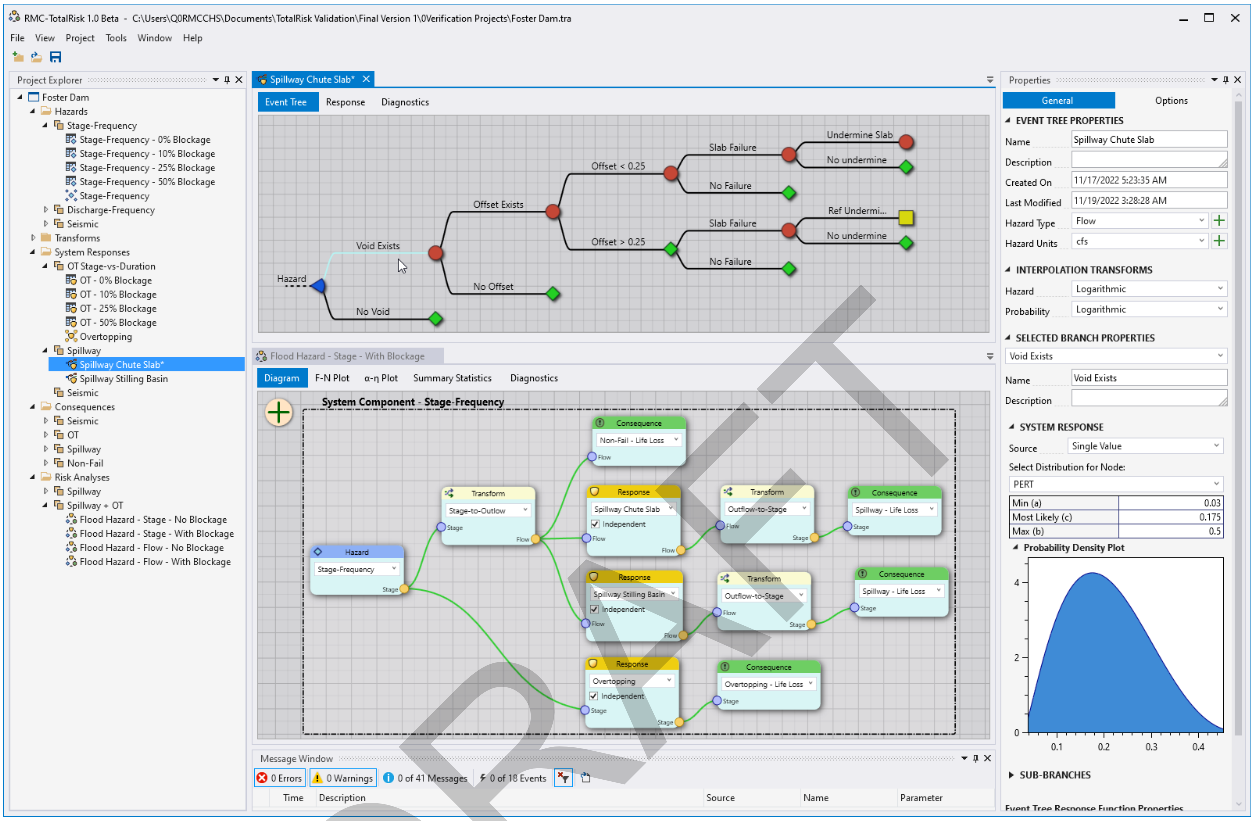

Table of Contents

Table of Figures

Table of Tables

Abstract

The purpose of this document is to provide verification and validation of key computations in RMC-TotalRisk. Software verification involves comparison of the numerical solution generated by the code with theoretical and analytical solutions, or with other known numerical solutions. Verification ensures that the software accurately solves the equations that constitute the mathematical model. RMC-TotalRisk has three main components that required verification: 1) general numerical methods; 2) input functions; and 3) quantitative risk analysis. The numerical components were verified against theoretical solutions and Monte Carlo simulation, as well as several prominent software used in industry. In all cases, the computations in RMC-TotalRisk performed as intended.

Introduction

The U.S. Army Corps of Engineers (USACE) Risk Management Center (RMC) developed the quantitative risk analysis software (RMC-TotalRisk) to enhance and expedite risk assessments within the Flood Risk Management, Planning, and Dam and Levee Safety communities of practice.

RMC-TotalRisk is a menu-driven software, which performs risk analysis from user defined hazard, system response, and consequence functions. The software features a fully integrated modelling platform, including a modern graphical user interface, data entry capabilities, report quality charts, and diagnostics. TotalRisk can perform multi-failure risk analysis for a single dam or levee or for a complex system with multiple components.

The purpose of this document is to provide verification of critical RMC-TotalRisk computations. Software verification involves comparison of the numerical solution generated by the code with one or more analytical solutions, or other numerical solutions. Verification ensures that the software accurately solves the equations that constitute the mathematical model.

The RMC-TotalRisk software uses two dynamic link libraries (dll) for performing numerical analyses: *Numerics.dll* and *RMC.TotalRisk.dll*. *Numerics* is a numerical library for .NET, which provides methods and algorithms for numerical computations in science and engineering. *Numerics* includes routines for special functions, interpolation, statistics, random numbers, probability distributions, uncertainty analysis, integration, optimization, root finding, and more. *RMC.TotalRisk* is a model library for the RMC-TotalRisk software, written in the .NET framework, which contains all remaining necessary functionality for input functions and quantitative risk analysis. Both libraries were developed internally by the RMC and, as such, the numerical methods contained within these libraries need to be verified.

The RMC-TotalRisk software has three main components that required verification: 1) general numerical methods; 2) input functions; and 3) quantitative risk analysis. Numerical verification for each component is detailed in the remaining chapters of this report.

Performance Metrics

Every verification test provided in this report is assessed using the following percent difference formula:

$$\%\,Difference = \left|\frac{x_2 - x_1}{x_1}\right| \cdot 100 \qquad \text{Equation 1}$$

where x_2 is the value computed from *Numerics* or RMC-TotalRisk; and x_1 is the actual value (or "true" value) from either an analytical or numerical solution. All values in *Numerics* and TotalRisk are computed with double precision, but results are reported with varying levels of precision depending on the test.

Target performance metrics depend on the test type as shown in Table 1. Verification tests for general numerical methods have the strictest performance requirements. A percent difference greater than 1% is considered unsatisfactory. The general numerical methods should produce exact results, with the exception being numerical integration and differentiation methods since these are approximate in nature.

Table 1 - Performance Ratings for different verification test types.

Test Type	Very Good	Satisfactory	Unsatisfactory
Verification of Numerical Methods	$\% Diff = 0$	$0 < \% Diff \leq 1$	$\%Diff > 1$
Verification of Input Functions	$0 < \% Diff \leq 1$	$1 < \% Diff \leq 5$	$\%Diff > 5$
Verification of Risk Analysis	$0 < \% Diff \leq 1$	$1 < \% Diff \leq 5$	$\%Diff > 5$
Comparison with other Risk Software	$0 < \% Diff \leq 1$	$1 < \% Diff \leq 5$	$\%Diff > 5$

The remaining verification test types have more relaxed performance requirements. For these remaining tests, a percent difference greater than 5% is considered unsatisfactory. The input function and risk analysis numerical methods have either a Monte Carlo component or an approximate numerical method component. These methods are approximate in nature and can produce different results depending on pseudo-random number generators, sample size, and software design choices.

In general, the goal of this verification effort was for all tests to have less than 1% difference. However, percent differences less than five percent are considered satisfactory, particularly when comparing risk analysis results from different software. Differences greater than five percent were generally considered unsatisfactory. All verification results and comparisons with differences greater than five percent required additional analysis and justification.

In the report, the verification test results are provided in tables, and the percent difference cells are colored based on the performance ranges in Table 1. An example of the conditional formatting is shown in Figure 1. Perfect agreement, or zero percent difference, is colored green. One percent difference is colored white. Percent differences greater than or equal to five are colored red.

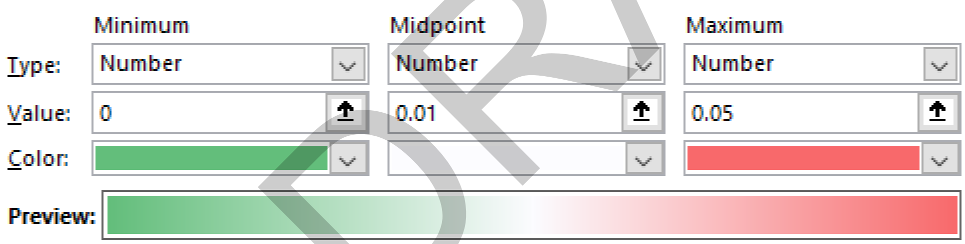

Figure 1 - Microsoft Excel© conditional formatting for percent difference.

Verification of General Numerical Methods

The *Numerics* library is used to perform a variety of general numerical methods in RMC-TotalRisk. *Numerics.dll* is a numerical library for .NET, which provides methods and algorithms for numerical computations in science and engineering, with a focus on statistical methods. *Numerics.dll* includes routines for special functions, interpolation, regression, statistics, probability distributions, bootstrap uncertainty analysis, Bayesian Markov Chain Monte Carlo, optimization, root finding, and more.

The *Numerics* library includes hundreds of individual verification tests (commonly referred to as *unit tests*) to ensure the software performs as intended. *Numerics* can be downloaded from GitHub[1] and is free to the public.

This report only provides verification of the general numerical methods considered to be important for RMC-TotalRisk. These include linear interpolation, probability distribution functions, numerical integration, numerical differentiation, and linear regression.

Linear Interpolation

In RMC-TotalRisk, the input functions can be defined with either parametric or nonparametric methods. All nonparametric function calculations are performed using linear interpolation. For example, a nonparametric (or empirical) probability distribution has the following distribution functions:

$$F(x) = p_i + (p_{i+1} - p_i)\left(\frac{x - x_i}{x_{i+1} - x_i}\right) \quad \textit{Equation 2}$$

$$F^{-1}(p) = x_i + (x_{i+1} - x_i)\left(\frac{p - p_i}{p_{i+1} - p_i}\right) \quad \textit{Equation 3}$$

where $F(x)$ is the cumulative distribution function (CDF) of the variable X; $F^{-1}(p)$ is the inverse CDF; and there is an array of continuous values $\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}$ with non-exceedance probabilities $\boldsymbol{p} = \{p_1, p_2, \dots, p_n\}$. The $\boldsymbol{x}$ values and non-exceedance probabilities $\boldsymbol{p}$ must be sorted in ascending order $x_i < x_{i+1}$ and $p_i \leq p_{i+1}$ with $0 \leq p_i \leq 1$.

There is often a benefit to applying a transform to the $\boldsymbol{x}$ and $\boldsymbol{p}$ values to improve the accuracy of the linear interpolation. For example, if the $\boldsymbol{x}$ values increase exponentially in real-space, then they will increase linearly in log-space. In this case, a log-transform will improve the accuracy of the linear interpolation of $\boldsymbol{x}$ values.

A log-transform can be applied to the $\boldsymbol{x}$ and/or $\boldsymbol{p}$ values. For example, when the exceedance probabilities are log-transformed, the inverse CDF becomes:

$$F^{-1}(p) = x_i + (x_{i+1} - x_i)\left(\frac{\log p - \log p_i}{\log p_{i+1} - \log p_i}\right) \quad \textit{Equation 4}$$

In addition, a Normal z transform can be applied to the $\boldsymbol{p}$ values as follows:

[1] https://github.com/USACE-RMC

$$F^{-1}(p) = x_i + (x_{i+1} - x_i)\left(\frac{\Phi^{-1}(p) - \Phi^{-1}(p_i)}{\Phi^{-1}(p_{i+1}) - \Phi^{-1}(p_i)}\right) \quad \text{Equation 5}$$

where $\Phi^{-1}(\cdot)$ is the inverse CDF of the standard Normal distribution.

Table 2 provides an example tabular response function. The $\boldsymbol{x}$ values represent hazard levels, such as river stage, and the $\boldsymbol{p}$ values represent conditional probabilities of failure at each hazard level. The response function is shown in Figure 2.

Table 2 - Example tabular response function data.

X Values	P Values
50	0.001
100	0.010
150	0.100
200	0.700
250	0.950
300	0.999

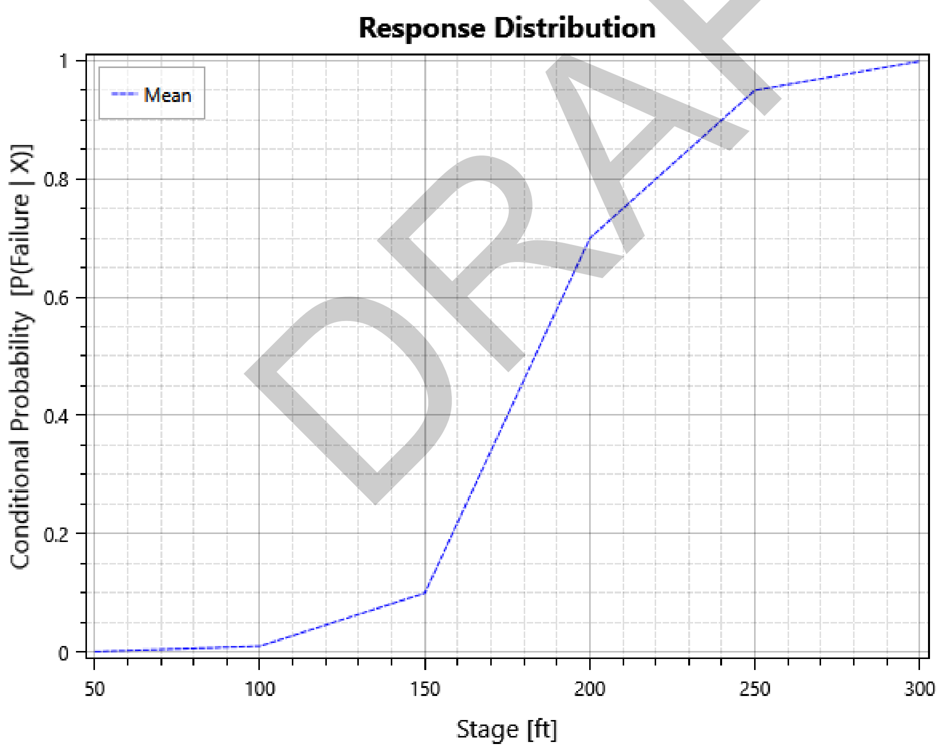

Figure 2 - Example tabular response function.

The *Numerics.dll* implements the smart table searching and interpolation algorithms described in Numerical Recipes [1]. In addition, *Numerics.dll* allows transforms on both variables. The *Numerics.dll* contains 21 tests for linear interpolation, which includes tests for the smart search routines and for all possible combinations of transforms. For the sake of brevity, this report only provides three of these tests. The other test results can be found in the *Numerics* download.

Numerical verification was performed using the *R 'stats'* package[2]. For the first verification test, interpolation was performed with no transforms on the $\boldsymbol{x}$ and $\boldsymbol{p}$ values. Results are provided in Table 3, where RMC-TotalRisk has perfect agreement with the *R 'stats'* package.

For the next test, the probability $\boldsymbol{p}$ values were log (base 10) transformed. Figure 3 shows the response function with the probabilities plotted on a logarithmic axis. Since the smaller probabilities plot in a straight line, the log-transform provides a more accurate linear interpolation in that range. Verification results are provided in Table 4, again showing perfect agreement.

Table 3 - Linear Interpolation results.

X Values	R 'stats'	Numerics	% Difference
76	0.00568	0.00568	0.0%
80	0.00640	0.00640	0.0%
96	0.00928	0.00928	0.0%
162	0.24400	0.24400	0.0%
170	0.34000	0.34000	0.0%
216	0.78000	0.78000	0.0%

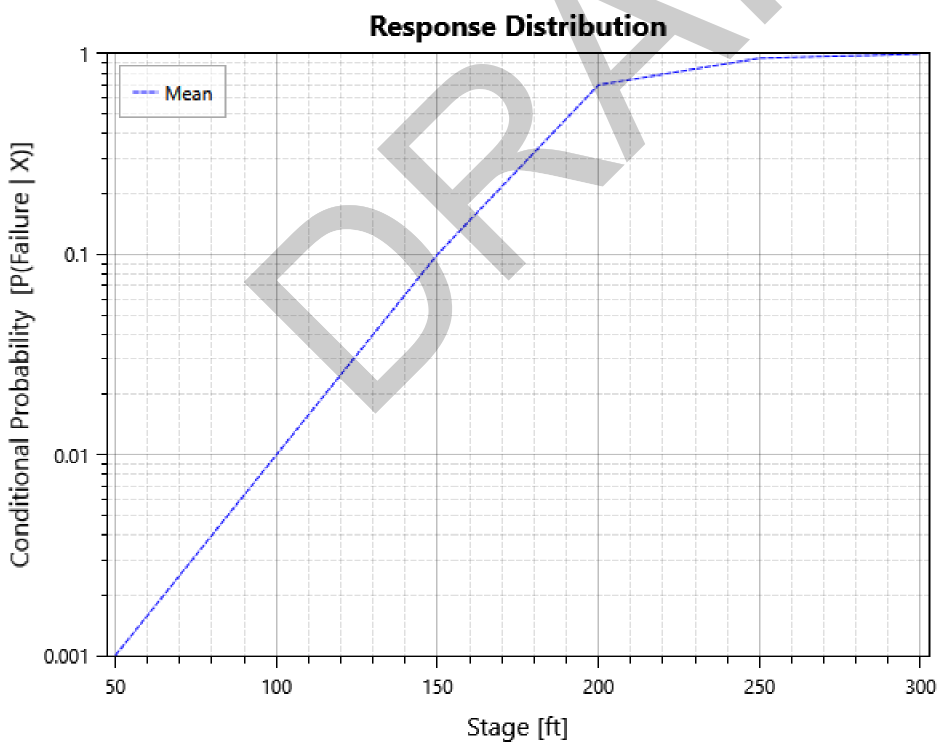

Figure 3 - Example tabular response function with probability plotted on logarithmic axis.

[2] https://www.rdocumentation.org/packages/stats/versions/3.6.2

Table 4 - Linear Interpolation results with logarithmic transform on probability values.

X Values	R 'stats'	Numerics	% Difference
76	0.00331	0.00331	0.0%
80	0.00398	0.00398	0.0%
96	0.00832	0.00832	0.0%
162	0.15952	0.15952	0.0%
170	0.21779	0.21779	0.0%
216	0.77186	0.77186	0.0%

In the final test, the probability p values were transformed using the Normal z-variates. Figure 4 shows the response function with the probabilities plotted on a Normal probability axis. Since the function plots in nearly a straight line, the Normal z-variate transform provides a more accurate linear interpolation across all probability values. Verification results are provided in Table 5, once again showing perfect agreement. Example code for replicating these linear interpolation problems with *R 'stats'* is provided in Figure 5 below.

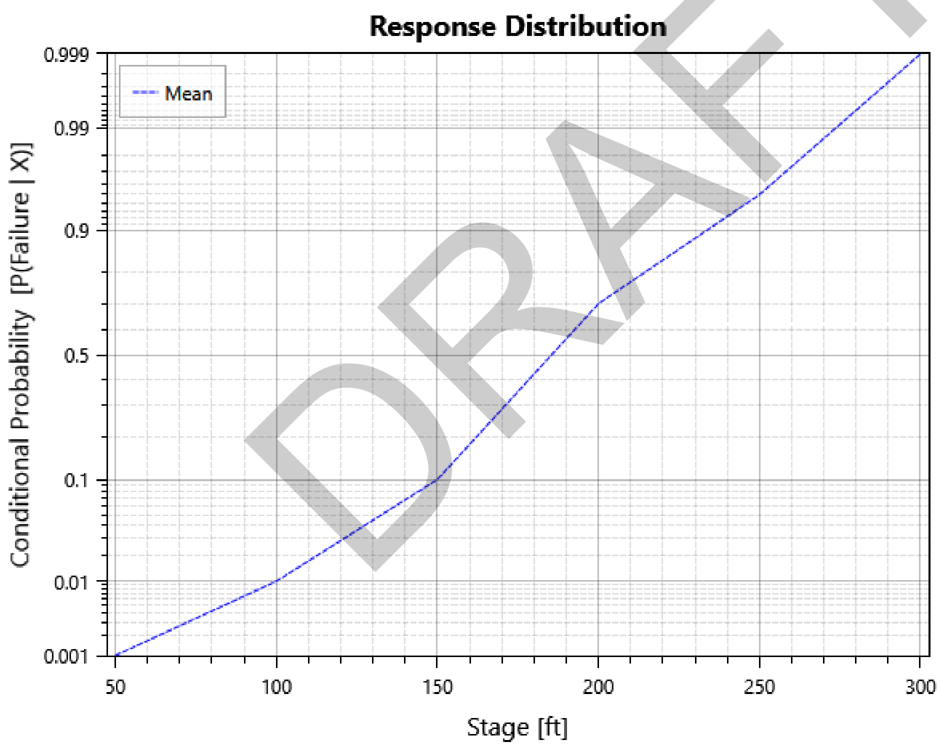

Figure 4 - Example tabular response function with probability plotted on Normal probability axis.

Table 5 - Linear Interpolation results with Normal z-variate transform on probability values.

X Values	R 'stats'	Numerics	% Difference
76	0.00354	0.00354	0.0%
80	0.00425	0.00425	0.0%
96	0.00848	0.00848	0.0%
162	0.19818	0.19818	0.0%
170	0.28802	0.28802	0.0%
216	0.81137	0.81137	0.0%

```
library(stats)

# The tabular x and y data
Xvalues = c(50, 100, 150, 200, 250, 300)
Pvalues = c(0.001, 0.01, 0.1, 0.7, 0.95, 0.999)

# The numeric values specifying where interpolation is to take place
Xout = c(76, 80, 96, 162, 170, 216)

# Perform linear interpolation
Pout = approx(x = Xvalues, y = Pvalues, xout = Xout)$y
print(Pout)
# [1] 0.00568 0.00640 0.00928 0.24400 0.34000 0.78000

# Perform linear interpolation with log10 transformed p-values
Pout = approx(x = Xvalues, y = log10(Pvalues), xout = Xout)$y
Pout = 10^Pout # transform back to real-space.
print(Pout)
# [1] 0.003311311 0.003981072 0.008317638 0.159523081 0.217790642 0.771859442

# Perform linear interpolation with Normal z-variate transformed p-values
Pout = approx(x = Xvalues, y = qnorm(p=Pvalues), xout = Xout)$y
Pout = pnorm(q=Pout) # transform back to real-space.
print(Pout)
# [1] 0.003540482 0.004245422 0.008482656 0.198184718 0.288022602 0.811367143
```
Figure 5 – Example code for performing linear interpolation with the R 'stats' package.

Probability Distributions

RMC-TotalRisk provides up to twenty different probability distributions for various input function options. The technical reference manual [2] provides a detailed description of each distribution and their typical applications.

All probability distribution functionality in RMC-TotalRisk is contained within *Numerics*. The *Numerics* library includes individual verification tests for all the probability distribution functions and methods. Comprehensive verification documentation for most these distributions is provided in the RMC-BestFit report [3].

Table 6 provides a listing of all the probability distributions and their verification sources. This report only provides verification of the CDF and inverse CDF of the remaining distributions (shown in blue in Table 6), since these are the only functions used by RMC-TotalRisk. Verification was performed using the *R* packages listed in Table 6.

Table 6 - Listing of probability distributions and verification sources.

Distribution	**Verification Source**
Exponential	RMC-BestFit Report
Gamma	RMC-BestFit Report
Generalized Beta	R 'mc2d'
Generalized Extreme Value	RMC-BestFit Report
Generalized Logistic	RMC-BestFit Report
Generalized Normal	R 'lmom'
Generalized Pareto	RMC-BestFit Report
Gumbel	RMC-BestFit Report
Kappa-4	R 'lmom'
Logistic	RMC-BestFit Report
Log-Normal	RMC-BestFit Report
Log-Pearson Type III	RMC-BestFit Report
Normal	RMC-BestFit Report
Nonparametric	RMC-BestFit Report
Pearson Type III	RMC-BestFit Report
PERT	R 'mc2d'
Triangular	R 'mc2d'
Truncated Normal	R 'truncnorm'
Uniform	R 'stats'
Weibull	RMC-BestFit Report

The Generalized Beta, PERT, and Triangular distributions were verified using the *R 'mc2d'* package[3]. Each of these distributions are bounded by lower and upper bounds. In RMC-TotalRisk, these distributions will commonly be used to represent uncertainty in a response probability, so they will be bounded between 0 and 1. The CDF was evaluated at five different x values. Then, the inverse CDF was evaluated by inputting the resulting probabilities from the CDF to ensure it returns the same x values. Verification

[3] https://cran.r-project.org/web/packages/mc2d/index.html

results for these three distributions are provided in Table 7 through Table 12. In each case, *Numerics* has perfect agreement with the *R 'mc2d'* package. Example code for replicating these distribution tests with *R 'mc2d'* is provided in Figure 6 below.

Table 7 - Verification of the CDF of the Generalized Beta distribution.

X Values	R 'mc2d'	Numerics	% Difference
0.10	0.271000	0.271000	0.0%
0.25	0.578125	0.578125	0.0%
0.50	0.875000	0.875000	0.0%
0.75	0.984375	0.984375	0.0%
0.90	0.999000	0.999000	0.0%

Table 8 - Verification of the inverse CDF of the Generalized Beta distribution.

P Values	R 'mc2d'	Numerics	% Difference
0.271000	0.10	0.10	0.0%
0.578125	0.25	0.25	0.0%
0.875000	0.50	0.50	0.0%
0.984375	0.75	0.75	0.0%
0.999000	0.90	0.90	0.0%

Table 9 - Verification of the CDF of the PERT distribution.

X Values	R 'mc2d'	Numerics	% Difference
0.10	0.0814600	0.0814600	0.0%
0.25	0.3671875	0.3671875	0.0%
0.50	0.8125000	0.8125000	0.0%
0.75	0.9843750	0.9843750	0.0%
0.90	0.9995400	0.9995400	0.0%

Table 10 - Verification of the inverse CDF of the PERT distribution.

P Values	R 'mc2d'	Numerics	% Difference
0.0814600	0.10	0.10	0.0%
0.3671875	0.25	0.25	0.0%
0.8125000	0.50	0.50	0.0%
0.9843750	0.75	0.75	0.0%
0.9995400	0.90	0.90	0.0%

Table 11 - Verification of the CDF of the Triangular distribution.

X Values	R 'mc2d'	Numerics	% Difference
0.10	0.0400000	0.0400000	0.0%
0.25	0.2500000	0.2500000	0.0%
0.50	0.6666667	0.6666667	0.0%
0.75	0.9166667	0.9166667	0.0%
0.90	0.9866667	0.9866667	0.0%

Table 12 - Verification of the inverse CDF of the Triangular distribution.

P Values	R 'mc2d'	Numerics	% Difference
0.0400000	0.10	0.10	0.0%
0.2500000	0.25	0.25	0.0%
0.6666667	0.50	0.50	0.0%
0.9166667	0.75	0.75	0.0%
0.9866667	0.90	0.90	0.0%


```
library(mc2d)

# CDF of the Generalized Beta distribution
x = c(0.1, 0.25, 0.5, 0.75, 0.9)
p = pbetagen(q = x, shape1 = 1, shape2 = 3, min = 0, max = 1)
# [1] 0.271000 0.578125 0.875000 0.984375 0.999000

# Inverse CDF of the Generalized Beta
qbetagen(p = p, shape1 = 1, shape2 = 3, min = 0, max = 1)
# [1] 0.10 0.25 0.50 0.75 0.90

# CDF of the PERT distribution
p = ppert(q = x, min = 0, mode = 0.25, max = 1)
# [1] 0.0814600 0.3671875 0.8125000 0.9843750 0.9995400

# Inverse CDF of the PERT
qpert(p = p, min = 0, mode = 0.25, max = 1)
# [1] 0.10 0.25 0.50 0.75 0.90

# CDF of the Triangular distribution
p = ptriang(q = x, min = 0, mode = 0.25, max = 1)
# [1] 0.0400000 0.2500000 0.6666667 0.9166667 0.9866667

# Inverse CDF of the Triangular
qtriang(p = p, min = 0, mode = 0.25, max = 1)
# [1] 0.10 0.25 0.50 0.75 0.90
```
Figure 6 – Example code for using probability distributions with the R 'mc2d' package.

The Generalized Normal and Kappa-4 distributions were verified using the R 'lmom' package[4]. These distributions are commonly used for flood frequency analysis [4]. In RMC-TotalRisk, they can be used when creating a parametric hazard function. The CDF was evaluated at five different x values. Then, the inverse CDF was evaluated by inputting the resulting probabilities from the CDF to ensure it returns the same x values. Verification results for these distributions are provided in Table 13 through Table 16. In each case, *Numerics* has perfect agreement with the *R 'lmom'* package. Example code for replicating these distribution tests with *R 'lmom'* is provided in Figure 7 below.

Table 13 - Verification of the CDF of the Generalized Normal distribution.

X Values	R 'lmom'	Numerics	% Difference
5	0.07465069	0.07465069	0.0%
10	0.53400804	0.53400804	0.0%
12	0.73775928	0.73775928	0.0%
15	0.92073519	0.92073519	0.0%
18	0.98333335	0.98333335	0.0%

Table 14 - Verification of the inverse CDF of the Generalized Normal distribution.

P Values	R 'lmom'	Numerics	% Difference
0.07465069	5	5	0.0%
0.53400804	10	10	0.0%
0.73775928	12	12	0.0%
0.92073519	15	15	0.0%
0.98333335	18	18	0.0%

Table 15 - Verification of the CDF of the Kappa-4 distribution.

X Values	R 'lmom'	Numerics	% Difference
5	0.07168831	0.07168831	0.0%
10	0.53317660	0.53317660	0.0%
12	0.73279234	0.73279234	0.0%
15	0.91293987	0.91293987	0.0%
18	0.97980084	0.97980084	0.0%

Table 16 - Verification of the inverse CDF of the Kapp-4 distribution.

P Values	R 'lmom'	Numerics	% Difference
0.07168831	5	5	0.0%
0.53317660	10	10	0.0%
0.73279234	12	12	0.0%
0.91293987	15	15	0.0%
0.97980084	18	18	0.0%

[4] https://cran.r-project.org/web/packages/lmom/index.html

```
library(lmom)

# CDF of the Generalized Normal (GNO) distribution
x = c(5, 10, 12, 15, 18)
p = cdfgno(x = x, para = c(9.7, 3.5, -0.1))
# [1] 0.07465069 0.53400804 0.73775928 0.92073519 0.98333335

# Inverse CDF of the GNO
# This should return the x values used with the CDF
quagno(f = p, para = c(9.7, 3.5, -0.1))
# [1]  5 10 12 15 18

# CDF of the Kappa-4 (K4) distribution
p = cdfkap(x = x, para = c(8.7, 3.1, 0.14, -0.1))
# 0.07168831 0.53317660 0.73279234 0.91293987 0.97980084

# Inverse CDF of the K4
quakap(f = p, para = c(8.7, 3.1, 0.14, -0.1))
# [1]  5 10 12 15 18
```

Figure 7 – Example code for using probability distributions with the R 'lmom' package.

Finally, the Truncated Normal and Uniform distributions were verified using the *R 'truncnorm'* and *R 'stats'* package[5], respectively. Both distributions are bounded by lower and upper bounds. In RMC-TotalRisk, these distributions will commonly be used to represent uncertainty in a response probability, so they will be bounded between 0 and 1. Verification results for these distributions are provided in Table 17 through Table 20. In each case, *Numerics* has perfect agreement with the *R* packages. Example code for replicating these distribution tests with *R is* provided in Figure 8 below.

Table 17 - Verification of the CDF of the Truncated Normal distribution.

X Values	R 'truncnorm'	Numerics	% Difference
0.10	0.1341936	0.1341936	0.0%
0.25	0.3761035	0.3761035	0.0%
0.50	0.7522070	0.7522070	0.0%
0.75	0.9474634	0.9474634	0.0%
0.90	0.9887291	0.9887291	0.0%

Table 18 - Verification of the inverse CDF of the Truncated Normal distribution.

P Values	R 'truncnorm'	Numerics	% Difference
0.1341936	0.10	0.10	0.0%
0.3761035	0.25	0.25	0.0%
0.7522070	0.50	0.50	0.0%
0.9474634	0.75	0.75	0.0%
0.9887291	0.90	0.90	0.0%

5 https://cran.r-project.org/web/packages/truncnorm/index.html

Table 19 - Verification of the CDF of the Uniform distribution.

X Values	R 'stats'	Numerics	% Difference
0.10	0.10	0.10	0.0%
0.25	0.25	0.25	0.0%
0.50	0.50	0.50	0.0%
0.75	0.75	0.75	0.0%
0.90	0.90	0.90	0.0%

Table 20 - Verification of the inverse CDF of the Uniform distribution.

P Values	R 'stats'	Numerics	% Difference
0.10	0.10	0.10	0.0%
0.25	0.25	0.25	0.0%
0.50	0.50	0.50	0.0%
0.75	0.75	0.75	0.0%
0.90	0.90	0.90	0.0%

```
library(truncnorm)

# CDF of the Truncated Normal distribution
x = c(0.1, 0.25, 0.5, 0.75, 0.9)
p = ptruncnorm(q = x, a = 0, b = 1, mean = 0.25, sd = 0.3)
# [1] 0.1341936 0.3761035 0.7522070 0.9474634 0.9887291

# Inverse CDF of the Truncated Normal
qtruncnorm(p = p, a = 0, b = 1, mean = 0.25, sd = 0.3)
# [1] 0.10 0.25 0.50 0.75 0.90

library(stats)

# CDF of the Uniform distribution
x = c(0.1, 0.25, 0.5, 0.75, 0.9)
p = punif(q = x, min = 0, max = 1)
# [1] 0.10 0.25 0.50 0.75 0.90

# Inverse CDF of the Uniform
qunif(p = p, min = 0, max = 1)
# [1] 0.10 0.25 0.50 0.75 0.90
```

Figure 8 – Example code for using probability distributions with the R 'truncnorm' and 'stats' packages.

Numerical Integration

In RMC-TotalRisk, within every Monte Carlo realization for every system component, risk is computed using numerical integration. In the technical reference manual [2], risk is formally defined as the expected value of consequences $\mathbb{E}[C]$, which is calculated as:

$$\mathbb{E}[C] = \int_{-\infty}^{\infty} C(x) \cdot f\big(C(x)\big) \cdot dx \qquad \text{Equation 6}$$

where x is the hazard level (e.g., flood discharge or water level); $C(x)$ determines the consequences, such as property damage or life loss, for the hazard level x; and $f\big(C(x)\big)$ is the probability density function (PDF) of the consequences occurring.

Computing risk for multiple system components requires integration over a multidimensional integral. Consider a system with two components, where the consequences of failure from each component are additive. Following the general risk formula provided in Equation 6, the system risk becomes a two-dimensional integral:

$$\mathbb{E}[C]_{\Omega} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \{C_X(x) + C_Y(y)\} \cdot f_{XY}\big(C_X(x), C_Y(y)\big) \cdot dx \cdot dy \qquad \text{Equation 7}$$

where x is the hazard level for system component X; $C_X(x)$ determines the consequences for the hazard level x; y is the hazard level for system component Y; $C_Y(y)$ determines the consequences for the hazard level y; and $f_{XY}\big(C_X(x), C_Y(y)\big)$ is the joint PDF of the combined system consequences occurring.

Single Dimension Integration

In RMC-TotalRisk, single dimension integrals like Equation 6 are solved using an implementation of the Adaptive Simpson's Rule (ASR) method. The ASR algorithm subdivides the interval of integration in a recursive manner until a user-defined tolerance is achieved. The default tolerance level is $1e^{-8}$. More details are provided in [2] and [5].

The numerical integration functionality in RMC-TotalRisk is contained within *Numerics*. Verification of the numerical integration was performed using six analytical example problems of varying complexity with known solutions. The goal of each verification test is to approximate the definite integral:

$$I = \int_{a}^{b} f(x) \cdot dx \qquad \text{Equation 8}$$

The first example is a simple function with a single variable:

$$f(x) = x^3 \qquad \text{Equation 9}$$

Integrating from $a = 0$ to $b = 1$, the exact solution is:

$$\int_0^1 f(x)\,dx = \frac{1}{4}x^4\Big|_0^1 = \frac{1}{4}\cdot 1^4 - 0 = 0.25 \tag{Equation 10}$$

The ASR method will give exact results for 3rd degree (or less) polynomials. The ASR method required only 5 function evaluations to converge with a standard error of 0.

Table 21 - Numerical integral results for example 1.

Exact Solution	Numerics	% Difference
0.25	0.25	0.0%

The next example integrates the following function:

$$f(x) = \cos x \tag{Equation 11}$$

from $a = -1$ to $b = 1$. The exact integral is:

$$\int_{-1}^{1} f(x)\,dx = \sin x\Big|_{-1}^{1} = 2\sin 1 = 1.6829419\ldots \tag{Equation 12}$$

The ASR method required 65 function evaluations to converge with a standard error $< 1e^{-7}$.

Table 22 - Numerical integral results for example 2.

Exact Solution	Numerics	% Difference
1.6829419	1.6829419	0.0%

The next example integrates:

$$f(x) = 0.5 + 24x + 3x^2 \tag{Equation 13}$$

from $a = 0$ to $b = 2$. The exact integral is:

$$\int_0^2 f(x)\,dx = (0.5x + 12x^2 + x^3)\Big|_0^2 = (0.5\cdot 2 + 12\cdot 2^2 + 2^3) - 0 = 57 \tag{Equation 14}$$

The ASR method required 5 function evaluations to converge with a standard error of 0.

Table 23 - Numerical integral results for example 3.

Exact Solution	Numerics	% Difference
57	57	0.0%

The next example integrates:

$$f(x) = 0.5 + 24x + 3x^2 + 8x^3 \quad \text{Equation 15}$$

from $a = 0$ to $b = 2$. The exact integral is:

$$\int_0^2 f(x)\,dx = (0.5x + 12x^2 + x^3 + 2x^4)\Big|_0^2 = (0.5 \cdot 2 + 12 \cdot 2^2 + 2^3 + 2 \cdot 2^4) - 0 = 89 \quad \text{Equation 16}$$

The ASR method required 5 function evaluations to converge with a standard error of 0.

Table 24 - Numerical integral results for example 4.

Exact Solution	Numerics	% Difference
89	89	0.0%

The next two examples are more relevant for computations with RMC-TotalRisk. In the next example, the goal is to compute the mean of a Gamma distribution with a scale of $\theta = 10$ and shape of $\kappa = 5$, which is simply $\theta \cdot \kappa = 50$. The function to integrate is:

$$\mu = \mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot f(x) \cdot dx \quad \text{Equation 17}$$

$$f(x) = \frac{1}{\Gamma(\kappa)\theta^{\kappa}} x^{\kappa-1} e^{-\frac{x}{\theta}} \quad \text{Equation 18}$$

The ASR method required 393 function evaluations to converge with a standard error $< 1e^{-6}$.

Table 25 - Numerical integral results for example 5.

Exact Solution	Numerics	% Difference
50	50	0.0%

The final example for single dimension integration is to compute the conditional expected value of a Ln-Normal distribution with a real-space mean of $\mu_x = 10$ and standard deviation $\sigma_x = 2$. The corresponding log-space mean and standard deviation are $\mu = 2.282975$ and $\sigma = 0.198042$. For more details on the Ln-Normal distribution, please see the technical reference manual [2].

A conditional expectation is defined as the expected value of a random variable given that this value lies within some prescribed probability range [6]:

$$\mathbb{E}[X|X \geq \beta] = \frac{1}{1 - F(\beta)} \int_{\beta}^{\infty} x \cdot f(x) \cdot dx \quad \text{Equation 19}$$

where $f(\cdot)$ is the PDF; and $F(\cdot)$ is the CDF.

where $f(\cdot)$ is the PDF; and $F(\cdot)$ is the CDF. The threshold value β can be defined by a specified nonexceedance probability α, such as $\alpha = 0.99$, rather than a threshold value, β. In this case, $\beta = F^{-1}(\alpha)$. The exact solution for the expectation of a Ln-Normal distribution conditioned on $X \geq \beta$ is provided by:

$$\mathbb{E}[X|X \geq \beta] = \frac{e^{\mu+\frac{\sigma^2}{2}}}{1-\alpha} \cdot \left(1 - \Phi(\Phi^{-1}(\alpha) - \sigma)\right) \quad \text{Equation 20}$$

where $\Phi(\cdot)$ is the CDF of the standard Normal distribution; $\Phi^{-1}(\cdot)$ is the inverse CDF; and μ and σ are the log-space mean and standard deviation, respectively.

The ASR method required 89 function evaluations to converge with a standard error $< 1e^{-6}$.

Table 26 - Numerical integral results for example 6.

Exact Solution	Numerics	% Difference
16.65587	16.65587	0.0%

Additional verification was performed using the *R 'stats'* package. Example code for replicating these three example problems with *R 'stats'* is provided in Figure 9 below. In each case, RMC-TotalRisk produced the same results as the *R 'stats'* package. The package reports the absolute error, which is approximately the same as the square of the standard error reported by the *Numerics* ASR method.

Multidimensional Integration

Solving multidimensional integrals is computationally demanding. If traditional, nonadaptive numerical integration techniques were used, the solution would require K^D iterations, where K is the number of integration steps (or bins) and D is the number of dimensions. If there were 100 integration steps and 5 dimensions, the solution would need 10 billion iterations. To avoid these computational limitations, RMC-TotalRisk uses an adaptive importance sampling algorithm called VEGAS [7] [8]. More details on this method can be found in [1], [2], and [9].

The VEGAS algorithm implemented in RMC-TotalRisk has two steps. First, to establish and refine the importance sampling histogram, by default the routine performs five warmup cycles, each with a maximum of $1{,}000 \cdot D$ integrand evaluations. For instance, if $D = 2$, by default there are $10{,}000$ total warmup evaluations. Next, again by default, $10{,}000$ final integrand evaluations are performed. The solution and resulting standard error are based only on the final evaluations. The user can add more function evaluations to achieve a smaller standard error.

Verification of the multidimensional integration was performed using four analytical example problems of varying complexity with known solutions. The goal of each verification test is to approximate the definite integral:

$$I = \int_{a_1}^{b_1} \cdots \int_{a_D}^{b_D} f(x_1, \cdots, x_D) \cdot dx_1 \cdots dx_D \quad \text{Equation 21}$$

The first example is a simple 2-dimensional problem for computing π:

$$f(x,y) = \begin{cases} 1, & x^2 + y^2 < 1 \\ 0, & x^2 + y^2 \geq 1 \end{cases} \qquad \text{Equation 22}$$

Integrating from $a = \{-1, -1\}$ to $b = \{1, 1\}$, the exact solution is $\pi = 3.141593$ ....

```
library(stats)

# Example 1
fx1 = function(x){return(x^3)}
integrate(f = fx1, lower = 0, upper = 1, rel.tol = 1E-8)
# 0.25 with absolute error < 2.8e-15

# Example 2
fx2 = function(x){return(cos(x))}
integrate(f = fx2, lower = -1, upper = 1, rel.tol = 1E-8)
# 1.682942 with absolute error < 1.9e-14

# Example 3
fx3 = function(x){return(0.5 + 24 * x + 3 * x * x)}
integrate(f = fx3, lower = 0, upper = 2, rel.tol = 1E-8)
# 57 with absolute error < 6.3e-13

# Example 4
fx4 = function(x){return(0.5 + 24 * x + 3 * x * x + 8 * x * x * x)}
integrate(f = fx4, lower = 0, upper = 2, rel.tol = 1E-8)
# 89 with absolute error < 9.9e-13

# Example 5
fx5 = function(x){return(x*dgamma(x=x, shape = 10, scale=5))}
integrate(f = fx5, lower = qgamma(p=1E-16, shape = 10, scale = 5), upper = qgamma(p=1-1E-16, shape
= 10, scale = 5), rel.tol = 1E-8)
# 50 with absolute error < 1e-10

# Example 6
# Get log parameters
mu = 10; sigma = 2; var = sigma^2
lmu = log(mu^2 / sqrt(var + mu^2))
lsigma = sqrt(log(1.0 + var / mu^2))
fx6 = function(x){return(x*dlnorm(x=x, meanlog = lmu, sdlog = lsigma))}

I=integrate(f = fx6, lower = qlnorm(p=0.99, meanlog = lmu, sdlog = lsigma), upper = qlnorm(p=1-1E-
16, meanlog = lmu, sdlog = lsigma), rel.tol = 1E-8)
I$value / (1 - 0.99)
# [1] 16.65587
```

Figure 9 – Example code for performing numerical integration for a single dimension with the R 'stats' package.

Table 27 shows the VEGAS results for this first example. Since VEGAS is an advanced Monte Carlo integration method, the results are approximate in nature. The precision can be assess using the standard error of result (more details are provided in [1], [7] and [8]). The standard error of the VEGAS solution is $\sigma = 0.004019$ and the 90% confidence interval around the result is provided in parentheses. The exact solution is contained within this interval.

Table 27 - VEGAS results for example 1.

Exact Solution	Numerics	% Difference
3.141593	3.141074 (3.134463, 3.147684)	0.0%

The next problem follows the example provided in by the GNU Scientific Library[6]. The example is a 3-dimensional integral from the theory of random walks:

$$f(x, y, z) = \frac{1}{\pi^3[1 - \cos(x) \cdot \cos(y) \cdot \cos(z)]} \qquad \text{Equation 23}$$

Integrating from $a = \{0, 0, 0\}$ to $b = \{\pi, \pi, \pi\}$, the exact solution is given by:

$$\frac{\Gamma\left(\frac{1}{4}\right)^4}{4\pi^3} = 1.393204 \ldots \qquad \text{Equation 24}$$

where $\Gamma(\cdot)$ is the Gamma function. The VEGAS results are provided in Table 28. The standard error is $\sigma = 0.007161$ and the 90% confidence interval around the result is provided in parentheses. The exact solution is contained within this interval.

Table 28 - VEGAS results for example 2.

Exact Solution	Numerics	% Difference
1.393204	1.389101 (1.37732, 1.400881)	0.3%

The remaining two examples are more relevant for system risk computations with RMC-TotalRisk. The goal is to compute the mean of the sum of independent Normal distributions for 5-dimensions and 20-dimensions. A listing of the mean and standard deviations of the Normal distributions are provided in Table 29 below. The multidimensional function to integrate is as follows:

$$f(x_1, \cdots, x_D) = \sum_{k=1}^{D} x_k \cdot \prod_{k=1}^{D} \phi(x_k | \mu_k, \sigma_k) \qquad \text{Equation 25}$$

where $\phi(\cdot)$ is the PDF of the k-th Normal distribution with a mean μ_k and standard deviation σ_k. The integration limits are $a = \{\Phi^{-1}(1e^{-16} | \mu_1, \sigma_1), \cdots, \Phi^{-1}(1e^{-16} | \mu_D, \sigma_D)\}$ and $b = \{\Phi^{-1}(1 -$

[6] https://www.gnu.org/software/gsl/doc/html/montecarlo.html

$1e^{-16}|\mu_1,\sigma_1),\cdots,\Phi^{-1}(1-1e^{-16}|\mu_D,\sigma_D)\}$, where $\Phi^{-1}(\cdot)$ is the inverse CDF of the k-th Normal distribution. In other words, the integration limits cover the full probability domain from ~ 0 to ~ 1. The exact solution to the mean of the sum of Normally distributed random variables is:

$$\mathbb{E}[X] = \sum_{k=1}^{D} \mu_k$$ *Equation 26*

So, the mean of the sum of the first five distributions is:

$$\mathbb{E}[X] = 10 + 30 + 17 + 99 + 68 = 224$$ *Equation 27*

The exact solution to the sum of all twenty distributions is 837.

Table 29 - Normal distribution mean and standard deviations.

Distribution	**Mean, μ**	**Std. Deviation, σ**
1	10	2
2	30	15
3	17	5
4	99	14
5	68	7
6	26	24
7	35	29
8	55	22
9	13	22
10	59	1
11	12	3
12	28	28
13	49	19
14	54	18
15	20	4
16	47	24
17	12	23
18	76	26
19	70	26
20	57	19

The VEGAS results are provided in Table 30 and Table 31, and the standard errors are $\sigma = 0.027803$ and 0.097398, respectively. In both cases, the exact solution is contained within the 90% confidence interval.

Table 30 - VEGAS results for example 3.

Exact Solution	Numerics	% Difference
224	223.984867 (223.939135, 224.030599)	0.0%

Table 31 - VEGAS results for example 4.

Exact Solution	Numerics	% Difference
837	836.965827 (836.805622, 837.126033)	0.0%

RMC-TotalRisk permits an unlimited number of failure modes per system component. However, a single system is limited to 20 components due to virtual memory and computer runtime limitations. Therefore, the 20-dimension verification test shown above provides a stress test to the TotalRisk computation engine. The VEGAS method used in RMC-TotalRisk is capable of accurately estimating high-dimensional integrals.

An additional verification was performed using the *R 'cubature'* package[7]. Example code for replicating the 5-dimension example problem is provided in Figure 10 below. RMC-TotalRisk produced more accurate results than the *R 'cubature'* package while requiring fewer function evaluations, which indicates that the VEGAS implementation in *Numerics* is efficient and robust.

[7] https://cran.r-project.org/web/packages/cubature/index.html

```
library(cubature)

# Array of distribution mean and standard deviations
mu = c(10, 30, 17, 99, 68)
sigma = c(2, 15, 5, 14, 7)

# Computes the mean of the sum of independent Normal distributions
sumNormal = function(x){
  sum = 0
  prod = 1
  for (i in 1:5){
    sum = sum + x[i]
    prod = prod * dnorm(x = x[i], mean = mu[i], sd = sigma[i])
  }
  return(sum*prod)
}

# Get integration limits
lower = numeric(5)
upper = numeric(5)
for (i in 1:5){
  lower[i] = qnorm(p = 1E-16, mean = mu[i], sd = sigma[i])
  upper[i] = qnorm(p = 1 - 1E-16, mean = mu[i], sd = sigma[i])
}

# Perform integration
vegas(sumNormal, lowerLimit = lower, upperLimit = upper, flags=list(verbose=0, final=1))
# $integral
# [1] 223.9502
# $error
# [1] 0.4381816
# $neval
# [1] 1007500
```

Figure 10 – Example code for performing numerical integration for multidimensions with the R 'stats' package.

Numerical Differentiation

In RMC-TotalRisk, a derivative-based sensitivity analysis is provided for the event tree and risk analysis components of the software. The partial derivative measures how sensitive an output component f is with respect to an input parameter θ_i when all other input parameters are held fixed.

$$\frac{\partial f}{\partial \theta_i} \ (i = 1,2,\cdots,n)$$

Equation 28

In RMC-TotalRisk, the partial derivatives are evaluated using numerical differentiation with the two-point formula:

$$\frac{\partial f}{\partial \theta} = \frac{f(\theta + h) - f(\theta - h)}{2h}$$

Equation 29

where h represents a small change in θ. The step size value h is automatically determined according to the magnitude of the function input parameter:

$$h = \begin{cases} |x| \cdot \epsilon^{\frac{1}{2}}, & x \neq 0 \\ \epsilon^{\frac{1}{2}}, & x = 0 \end{cases}$$

Equation 30

Where x is the input parameter; and ϵ is the double precision machine epsilon.

The numerical derivative functionality in RMC-TotalRisk is contained within *Numerics*. Verification of the numerical differentiation was performed using three analytical functions with known solutions. The first example is a simple function with a single variable:

$$f(x) = x^3$$

Equation 31

Differentiating with respect to x gives the following:

$$\frac{\partial f}{\partial x} = 3x^2$$

Equation 32

Evaluating the function at $x = 2$, yields a derivative equal to 12:

$$\frac{\partial f}{\partial x} = 3 \cdot 2^2 = 12$$

Equation 33

Table 32 - Numerical derivative results for example 1.

Derivative	Exact Solution	Numerics	% Difference
$\partial f / \partial x$	12.00	12.00	0.0%

The second example is a function with two variables:

$$f(x, y) = x^2 y^3$$ *Equation 34*

Differentiating with respect to each variable gives:

$$\frac{\partial f}{\partial x} = 2xy^3$$ *Equation 35*

$$\frac{\partial f}{\partial y} = 3x^2 y^2$$ *Equation 36*

Evaluating the function at $x = 2$ and $y = 2$, yields partial derivatives equal to 32 and 48, respectively:

$$\frac{\partial f}{\partial x} = 2 \cdot 2 \cdot 2^3 = 32$$ *Equation 37*

$$\frac{\partial f}{\partial y} = 3 \cdot 2^2 \cdot 2^2 = 48$$ *Equation 38*

Table 33 - Numerical derivative results for example 2.

Derivative	Exact Solution	Numerics	% Difference
$\partial f / \partial x$	32.00	32.00	0.0%
$\partial f / \partial y$	48.00	48.00	0.0%

The third example is a function with three variables:

$$f(x, y, z) = x^3 + y^4 + z^5$$ *Equation 39*

Differentiating with respect to each variable gives:

$$\frac{\partial f}{\partial x} = 3x^2$$ *Equation 40*

$$\frac{\partial f}{\partial y} = 4y^3$$ *Equation 41*

$$\frac{\partial f}{\partial z} = 5z^4$$ *Equation 42*

Evaluating the function at $x = 2$, $y = 2$ and $z = 2$, yields partial derivatives equal to 12, 32, and 80, respectively:

$$\frac{\partial f}{\partial x} = 3 \cdot 2^2 = 12 \quad \text{Equation 43}$$

$$\frac{\partial f}{\partial y} = 4 \cdot 2^3 = 32 \quad \text{Equation 44}$$

$$\frac{\partial f}{\partial z} = 5 \cdot 2^4 = 80 \quad \text{Equation 45}$$

Table 34 - Numerical derivative results for example 3.

Derivative	Exact Solution	Numerics	% Difference
$\partial f / \partial x$	12.00	12.00	0.0%
$\partial f / \partial y$	32.00	32.00	0.0%
$\partial f / \partial z$	80.00	80.00	0.0%

In each case, *Numerics* and RMC-TotalRisk produced the correct solution with an absolute error of approximately $\pm 1e^{-8}$. Additional verification was performed using the *R 'numDeriv'* package[8]. Example code for replicating these three example problems with *R 'numDeriv'* is provided in Figure 11 below. In each case, RMC-TotalRisk produced the same results as the *R 'numDeriv'* package.

[8] https://cran.r-project.org/web/packages/numDeriv/index.htm

```
library(numDeriv)

# Example 1 - One variable function
fx = function(x){
  return(x^3)
}

# The grad function returns the partial derivatives with respect to each input
grad(func=fx, x=2)
# [1] 12

# Example 2 - Two variable function
fxy = function(p){
  x = p[1]
  y = p[2]
  return(x^2*y^3)
}

grad(func=fxy, x=c(2,2))
# [1] 32 48

# Example 3 - Three variable function
fxyz = function(p){
  x = p[1]
  y = p[2]
  z = p[3]
  return(x^3+y^4+z^5)
}

grad(func=fxyz, x=c(2,2,2))
# [1] 12 32 80
```

Figure 11 – Example code for performing numerical differentiation with the R 'numDeriv' package.

Linear Regression

In the risk analysis component of RMC-TotalRisk, it is challenging to compute the partial derivatives of the output with respect to each input. Instead, the sensitivity index is derived from a linear regression of inputs and output from a Monte Carlo simulation. For each Monte Carlo realization, the sampled inputs θ and the resulting output y are stored in a matrix. Then, a multiple linear regression is estimated as:

$$y = \sum_{i=1}^{n} \beta_i \theta_i + \varepsilon \qquad \text{Equation 46}$$

where the regression coefficient β_i measures the effect that input θ_i has on the predicted value y; and ε is the model error, or residual.

The linear regression functionality in RMC-TotalRisk is contained within *Numerics*. Verification of the linear regression was performed using the *R 'stats'* package. The regression example uses the *'uschange'* dataset from the *R 'fpp2'* package[9] [10], which provides growth rates of personal consumption and personal income in the USA.

The first verification test is just a simple linear equation that models consumption as a function of income. The next test models consumption as a function of income, production, savings, and the unemployment rate for the US. Results are provided in Table 35 and Table 36, respectively. The regression results from *Numerics* have perfect agreement with the *R 'stats'* package. Example code for performing linear regression in R is provided in Figure 12 below.

Table 35 - Verification results for linear regression for a simple linear model.

Coefficients	R 'stats'	Numerics	% Difference
Intercept	0.54510	0.54510	0.0%
Income	0.28060	0.28060	0.0%
Standard Error	0.60261	0.60261	0.0%

Table 36 - Verification results for linear regression for a linear model with multiple covariates.

Coefficients	R 'stats'	Numerics	% Difference
Intercept	0.26729	0.26729	0.0%
Income	0.71449	0.71449	0.0%
Production	0.04589	0.04589	0.0%
Savings	-0.04527	-0.04527	0.0%
Unemployment	-0.20477	-0.20477	0.0%
Standard Error	0.32860	0.32860	0.0%

[9] https://cran.r-project.org/web/packages/fpp2/index.html

```
library(fpp2)
library(stats)

# The first example is a simple linear regression
simpleLM = lm(formula = Consumption ~ Income, data = uschange)
summary(simpleLM)

# Residuals:
#       Min       1Q    Median       3Q      Max
# -2.40845 -0.31816  0.02558  0.29978  1.45157
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept)  0.54510    0.05569   9.789  < 2e-16 ***
# Income       0.28060    0.04744   5.915 1.58e-08 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.6026 on 185 degrees of freedom
# Multiple R-squared:  0.159,	Adjusted R-squared:  0.1545
# F-statistic: 34.98 on 1 and 185 DF,  p-value: 1.577e-08

# The next example is a multiple linear regression
multipleLM = lm(formula = Consumption ~ Income + Production + Savings + Unemployment, data =
uschange)
summary(multipleLM)

# Residuals:
#       Min       1Q   Median       3Q      Max
# -0.88296 -0.17638 -0.03679  0.15251  1.20553
#
# Coefficients:
#              Estimate Std. Error t value Pr(>|t|)
# (Intercept)   0.26729    0.03721   7.184 1.68e-11 ***
# Income        0.71449    0.04219  16.934  < 2e-16 ***
# Production    0.04589    0.02588   1.773   0.0778 .
# Savings      -0.04527    0.00278 -16.287  < 2e-16 ***
# Unemployment -0.20477    0.10550  -1.941   0.0538 .
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.3286 on 182 degrees of freedom
# Multiple R-squared:  0.754,	Adjusted R-squared:  0.7486
# F-statistic: 139.5 on 4 and 182 DF,  p-value: < 2.2e-16
```

Figure 12 – Example code for performing linear regression with the R 'stats' package.

Verification of Input Functions

RMC-TotalRisk has the following key model inputs: 1) hazard functions; 2) transform function; 3) system response functions; and 4) consequence functions. The input functions can be defined with parametric or nonparametric methods.

Most of the parametric input functions use parametric probability distributions, which were already verified in the previous section. Most of the nonparametric functions rely on linear interpolation, which was also verified in the previous section. The following sections describe the additional input function components that required further verification.

Parametric Bootstrap

In RMC-TotalRisk, the hazard and response functions can be defined as either a parametric or nonparametric distribution. The parametric bootstrap [11] [12] is used to quantify uncertainty in the parametric distributions. The bootstrap procedure involves the following general steps:

1. Randomly sample n values from the user-defined probability distribution, or *parent distribution*, where n is equal to the effective record length (ERL). This is called the *bootstrap sample*.

2. Estimate a new distribution from the bootstrap sample. The distribution can be estimated with product moments, linear moments, or maximum likelihood. See [4], [13], and [14] for more details on these estimation and fitting methods for distributions.

3. Record quantiles for desired nonexceedance probabilities, and any other output of interest.

4. Repeat steps 1 through 3 for a sufficiently large number of realizations, R. Then, derive confidence intervals by computing percentiles from the bootstrapped array for the desired output.

More details on how the parametric bootstrap is used in RMC-TotalRisk can be found in the technical reference manual [2]. The parametric bootstrap analysis and resulting confidence intervals were verified using the *R 'boot'* package[10] for a Log-Normal (base 10) distribution. Figure 13 shows the input options for an example Log-Normal parametric hazard function. The distribution has a mean (of log) of 3.0 and standard deviation (of log) of 0.5, and the ERL is 100. The bootstrap was performed using 10,000 realizations, and the 90% confidence interval was output for each user-defined exceedance probability.

Figure 14 shows a frequency curve plot comparing the bootstrap confidence interval results from RMC-TotalRisk to the *R 'boot'* package. Table 37 lists the confidence interval results and the percent difference between the two software programs. It is clear from these results that RMC-TotalRisk produces effectively identical confidence intervals to those of *R 'boot'*. Any differences in results are due to minor differences that arise from pseudo-random number generators and Monte Carlo sampling errors. Example code for replicating this example with *R 'boot'* is provided in Figure 15 below.

[10] https://cran.r-project.org/web/packages/boot/index.html

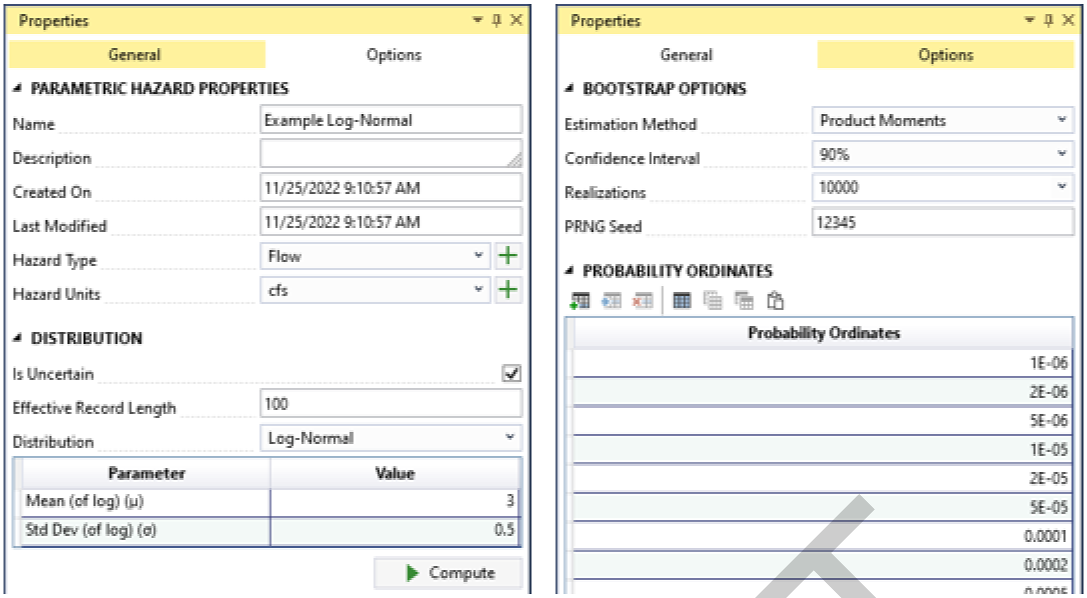

Figure 13 - RMC-TotalRisk inputs for a Log-Normal (base 10) parametric hazard function.

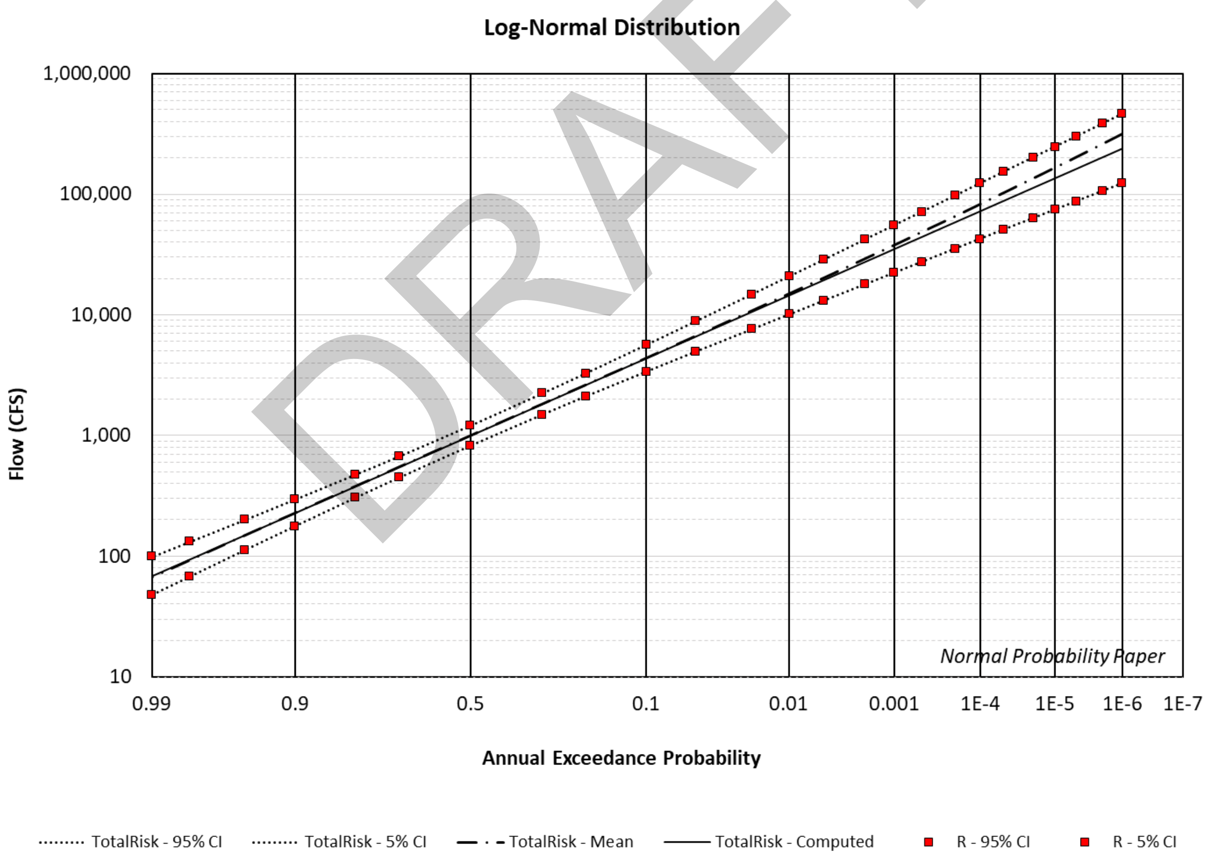

Figure 14 - Comparison of RMC-TotalRisk with R 'boot' confidence intervals for the Log-Normal (base 10) distribution.

Table 37 - Comparison of RMC-TotalRisk with R 'boot' confidence intervals for the Log-Normal (base 10) distribution. Quantile results are shown in log (base 10) space.

AEP	5% - CI			95% - CI		
	***R* 'boot'**	**RMC-TotalRisk**	**% Difference**	***R* 'boot'**	**RMC-TotalRisk**	**% Difference**
1.0E-06	5.664	5.665	0.0%	5.088	5.092	0.1%
2.0E-06	5.585	5.586	0.0%	5.025	5.029	0.1%
5.0E-06	5.477	5.478	0.0%	4.938	4.943	0.1%
1.0E-05	5.392	5.394	0.0%	4.871	4.875	0.1%
2.0E-05	5.304	5.308	0.1%	4.801	4.805	0.1%
5.0E-05	5.184	5.187	0.1%	4.705	4.710	0.1%
1.0E-04	5.088	5.092	0.1%	4.628	4.633	0.1%
2.0E-04	4.989	4.992	0.1%	4.549	4.552	0.1%
5.0E-04	4.851	4.854	0.1%	4.437	4.440	0.1%
1.0E-03	4.741	4.743	0.1%	4.346	4.350	0.1%
2.0E-03	4.624	4.626	0.0%	4.252	4.255	0.1%
5.0E-03	4.458	4.458	0.0%	4.116	4.118	0.1%
1.0E-02	4.320	4.321	0.0%	4.004	4.005	0.0%
2.0E-02	4.170	4.171	0.0%	3.881	3.882	0.0%
5.0E-02	3.946	3.948	0.1%	3.695	3.696	0.0%
1.0E-01	3.749	3.752	0.1%	3.528	3.529	0.0%
2.0E-01	3.515	3.517	0.0%	3.323	3.325	0.1%
3.0E-01	3.349	3.350	0.0%	3.171	3.173	0.1%
5.0E-01	3.082	3.083	0.0%	2.917	2.915	0.0%
7.0E-01	2.826	2.826	0.0%	2.650	2.649	0.0%
8.0E-01	2.676	2.676	0.0%	2.483	2.483	0.0%
9.0E-01	2.472	2.470	0.1%	2.248	2.248	0.0%
9.5E-01	2.306	2.302	0.2%	2.051	2.051	0.0%
9.8E-01	2.120	2.117	0.1%	1.828	1.827	0.1%
9.9E-01	1.998	1.994	0.2%	1.679	1.678	0.1%

```
library(boot)

# This is example code for performing the parametric bootstrap in R. This example estimates 90%
confidence intervals for a Log-Normal distribution (base 10).

# First define the AEP values for computing the curve and confidence intervals.
AEPs = c(0.000001, 0.000002, 0.000005, 0.00001, 0.00002, 0.00005, 0.0001, 0.0002, 0.0005, 0.001,
0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95, 0.98, 0.99)

# Define the parent distribution parameters.
pMu = 3.0
pSigma = 0.5
ERL = 100

# Define a dummy vector of data
pData = rnorm(n=ERL, mean = pMu, sd = pSigma)

# This function returns the bootstrap sample for each bootstrap realization.
bootSample = function(data, parms){
 # The 'data' and 'parms' inputs are required by the package. They are not used for this example.
 return(rnorm(n=ERL, mean = pMu, sd = pSigma))
}

# This function returns the bootstrapped vector of the AEPs given the bootstrapped sample.
bootAEPs = function(data){
 # Estimate new bootstrapped parameters from the bootstrap sample using product moments.
 bMu = mean(data)
 bSigma = sd(data)
 bAEPs = numeric(length(AEPs))
 for (i in 1:length(AEPs)){
  bAEPs[i] = qnorm(p = 1-AEPs[i], mean = bMu, sd = bSigma)
 }
 return(bAEPs)
}

# Perform the parametric bootstrap
bootstrap = boot(data = pData, ran.gen = bootSample, statistic = bootAEPs,   R = 10000, sim =
"parametric")

# Estimate the confidence interval for each AEP using the percentile method
CIs = matrix(nrow = length(AEPs), ncol=2)
for (i in 1:length(AEPs)){
 CIs[i,] = 10^boot.ci(boot.out = bootstrap, conf = 0.9, type = "perc", index = i)$percent[4:5]
}
```

Figure 15 – Example code for performing the parametric bootstrap with the R 'boot' package.

Nonparametric Hazard Function

In RMC-TotalRisk, a nonparametric hazard function can be defined in the same way as the “less simple method” in the flood damage reduction analysis software, *HEC-FDA* [15]. This type of hazard function is intended to provide backwards compatibility with existing *HEC-FDA* models for flood risk management studies.

More details on the nonparametric hazard function can be found in the technical reference manual [2]. The nonparametric confidence intervals were verified using *HEC-FDA version 1.4.3*[11].

Verification was performed using the *‘Beargrass Creek’* example project provided in the *HEC-FDA* user guide [15] and training course[12]. The Beargrass Creek study used for that course consists of two highly urbanized damage reaches on the South Fork of Beargrass Creek.

Figure 16 shows the RMC-TotalRisk input options for the South Fork 8 (SF-8) reach for the *HEC-FDA* model. Figure 17 shows the same inputs with *HEC-FDA*.

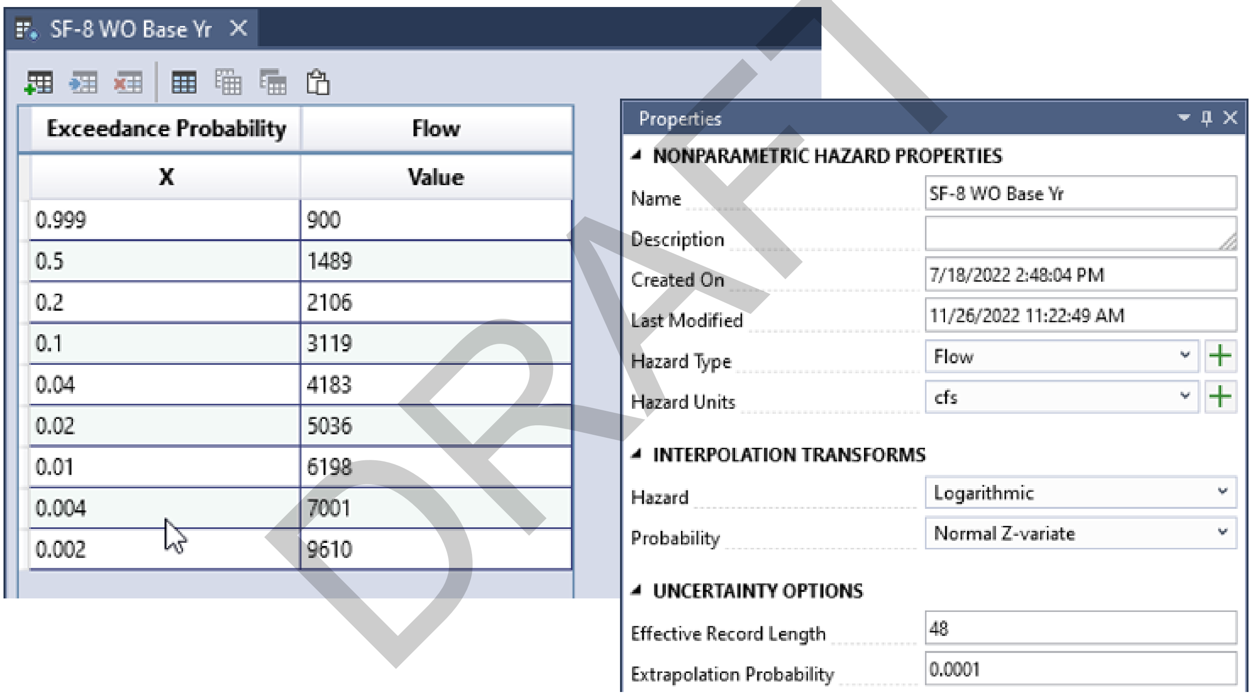

Figure 16 - RMC-TotalRisk inputs for the SF-8 nonparametric hazard function.

[11] https://www.hec.usace.army.mil/software/hec-fda/
[12] Flood Damage Assessment Course Content (army.mil)

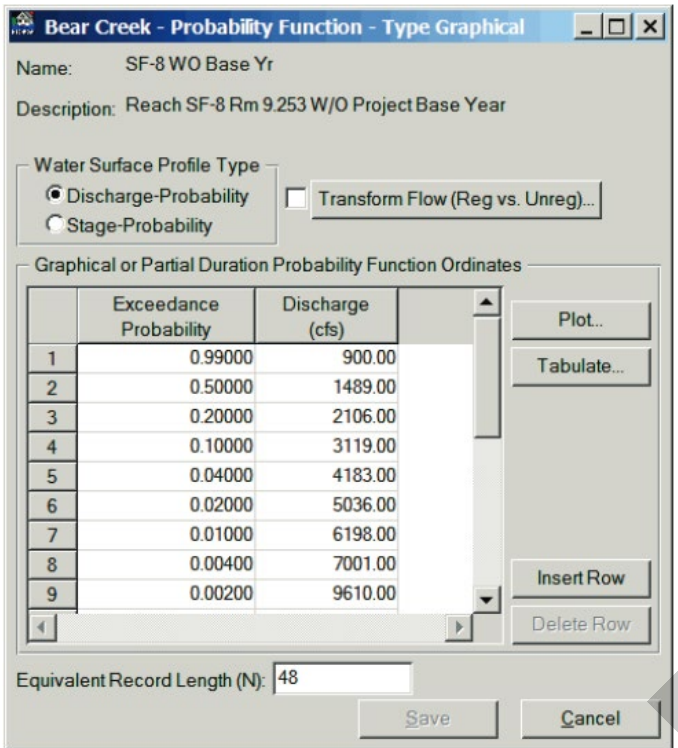

Figure 17 - RMC-TotalRisk inputs for the SF-8 nonparametric hazard function.

The uncertainty in the hazard level for a given exceedance probability is derived using the asymptotic approximation for quantile variance. A detailed proof is provided in [16] and [17]. Additional details related to the *HEC-FDA* implementation are provided in [15]. Details on the RMC-TotalRisk implementation are provided in [2].

Figure 18 shows a frequency curve plot comparing confidence interval results from RMC-TotalRisk to the *HEC-FDA* confidence intervals. Table 38 lists the confidence interval results and the percent difference between the two software programs. RMC-TotalRisk produces confidence intervals that very closely match *HEC-FDA*. The differences in results are primarily due to differences in linear interpolation choices and differences in how RMC-TotalRisk computes the probability density of the nonparametric hazard function. *HEC-FDA* interpolates several more points from the user-defined table before developing confidence intervals. Whereas RMC-TotalRisk only uses the user-defined values. Results are primarily different due to this design choice between software programs.

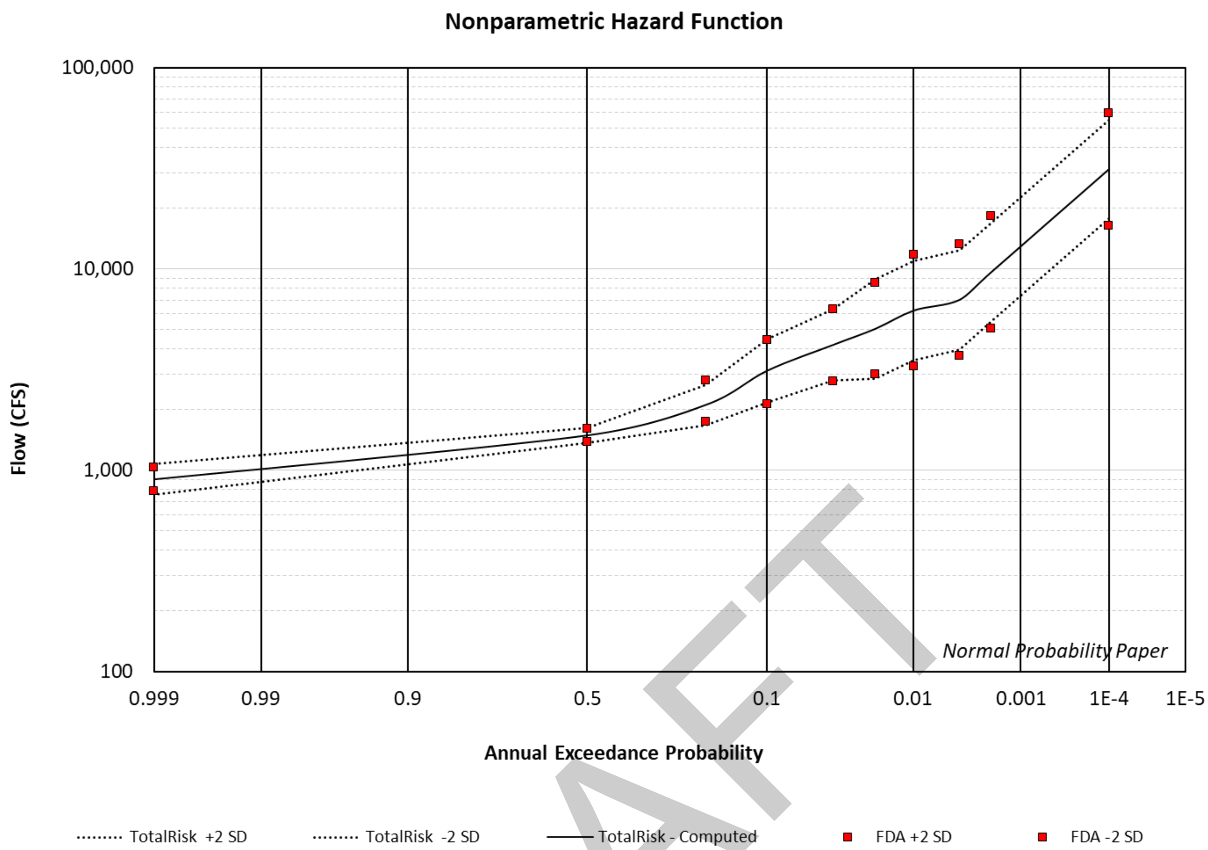

Figure 18 - Comparison of RMC-TotalRisk with HEC-FDA confidence intervals for the nonparametric hazard function.

Table 38 - Comparison of RMC-TotalRisk with HEC-FDA confidence intervals for the nonparametric hazard function. Quantile results are shown in log (base 10) space.

AEP	-2 SD			+2 SD		
	HEC-FDA	RMC-TotalRisk	% Difference	HEC-FDA	RMC-TotalRisk	% Difference
1.00E-04	4.216	4.248	0.8%	4.772	4.740	0.7%
2.00E-03	3.705	3.737	0.9%	4.261	4.229	0.7%
4.00E-03	3.567	3.599	0.9%	4.123	4.091	0.8%
1.00E-02	3.514	3.546	0.9%	4.070	4.038	0.8%
2.00E-02	3.474	3.456	0.5%	3.930	3.948	0.5%
4.00E-02	3.442	3.445	0.1%	3.801	3.798	0.1%
1.00E-01	3.328	3.336	0.2%	3.649	3.652	0.1%
2.00E-01	3.240	3.222	0.5%	3.448	3.424	0.7%
5.00E-01	3.142	3.136	0.2%	3.204	3.210	0.2%
9.99E-01	2.894	2.878	0.6%	3.014	3.030	0.5%

Tabular Hazard Functions

For the tabular hazard function, the user is required to enter a tabular relationship of hazard levels and exceedance probabilities. The user can choose to model the exceedance probabilities as uncertain while holding the hazard levels as fixed, or vice versa. A distribution must be selected to define uncertainty. The parameters for the selected distribution must be entered for every ordinate in the tabular data. The uncertainty at each hazard level must be entered such that the confidence intervals are monotonically increasing with increasing hazard levels.

All tabular (nonparametric) functions in RMC-TotalRisk use the same uncertainty analysis algorithm. In short, for each Monte Carlo realization, a single percentile value (e.g., 0.9) is sampled at random. Every ordinate in the tabular data is then sampled using the same percentile. This ensures each tabular function in the Monte Carlo simulation is generated with monotonically increasing hazard levels. Please see [2] for more details.

This uncertainty analysis approach is the same as the approach taken in HEC-FDA for *graphical* or *non-analytic* relationships [15]. This algorithm is restrictive in terms of the possible shapes of the nonparametric distribution that can be randomly generated, which could lead to a slight overestimation or underestimation in the variance of the risk results. However, as discussed in [15], generalizing the shape of the distribution requires a parametric representation. In the absence of a parametric shaping component, this is currently the best algorithm available for nonparametric uncertainty analysis.

Verification of the tabular function uncertainty analysis was performed using the theoretical confidence intervals for a Normal distribution, which are presented in [18]. The theoretical intervals are as follows:

$$x_p \pm \Phi^{-1}\left(\frac{1+\alpha}{2}\right) \cdot \sigma_x \cdot \sqrt{\frac{1+\frac{1}{2}\cdot\Phi^{-1}(p)^2}{N}}$$

Equation 47

where x_p is the quantile for the desired nonexceedance probability p; σ_x is the standard deviation of the Normal distribution; N is the effective record length; $\Phi^{-1}(\cdot)$ is the inverse CDF of the standard Normal distribution; and α is the confidence interval width (e.g., 0.9).

Following the previous bootstrap example, the tabular hazard function was derived from a Log-Normal distribution with a mean (of log) of 3.0 and standard deviation (of log) of 0.5, and an ERL of 100. Figure 19 shows how the tabular function data is input into RMC-TotalRisk. For this example, the data was entered in log (base 10) space. The mean values (quantiles) were taken from the inverse CDF of the Log-Normal distribution. The standard deviation (in log space) at each quantile is:

$$\sigma_p = \sigma_x \cdot \sqrt{\frac{1+\frac{1}{2}\cdot\Phi^{-1}(p)^2}{N}}$$

Equation 48

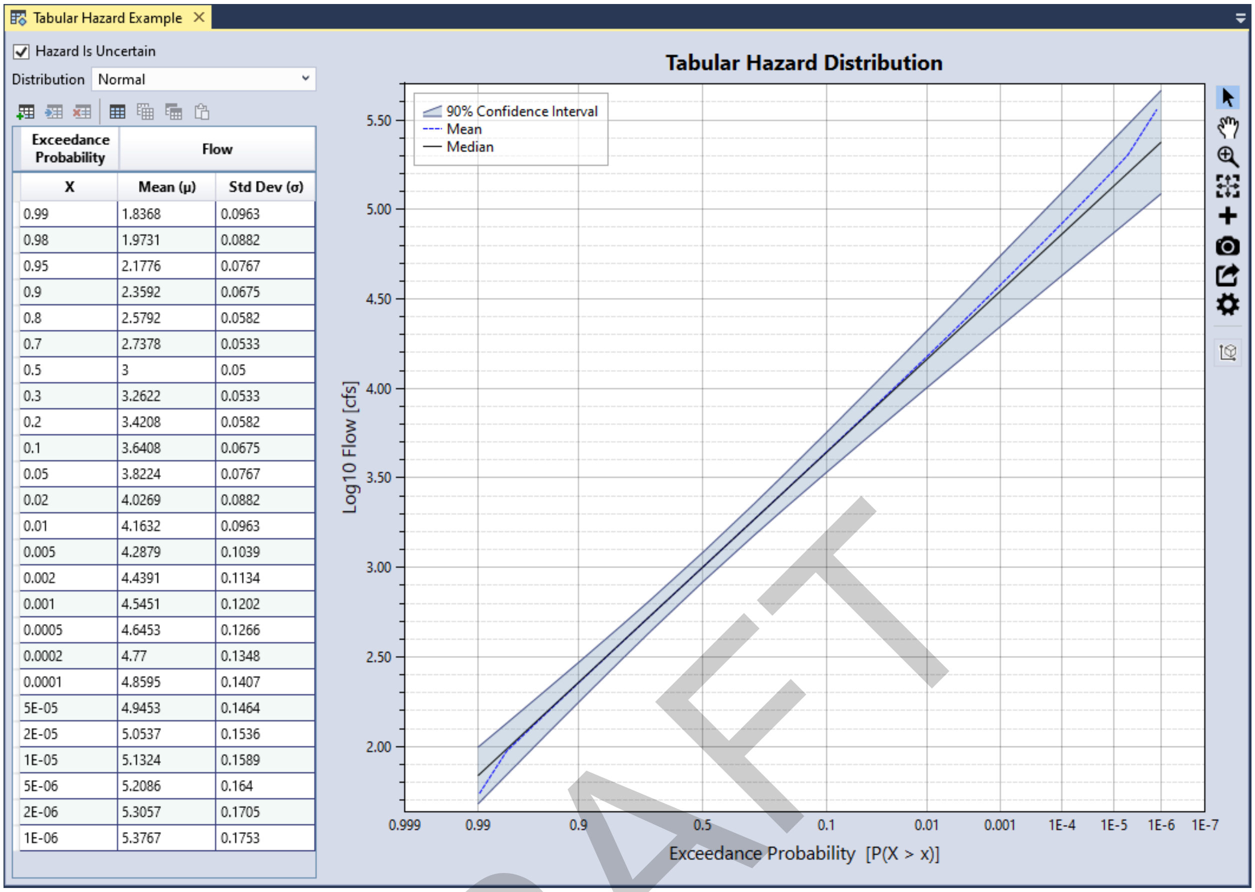

Figure 19 - RMC-TotalRisk inputs for the tabular hazard function.

The uncertainty analysis was performed using 10,000 Monte Carlo realizations, and the 90% confidence interval was output for each user-defined exceedance probability. Figure 20 shows a frequency curve plot comparison, and Table 39 lists the confidence interval results and the percent difference between the theoretical result and the tabular hazard function. RMC-TotalRisk produces effectively identical confidence intervals to the theoretical result from Equation 47. Any differences in results are due to minor Monte Carlo sampling errors.

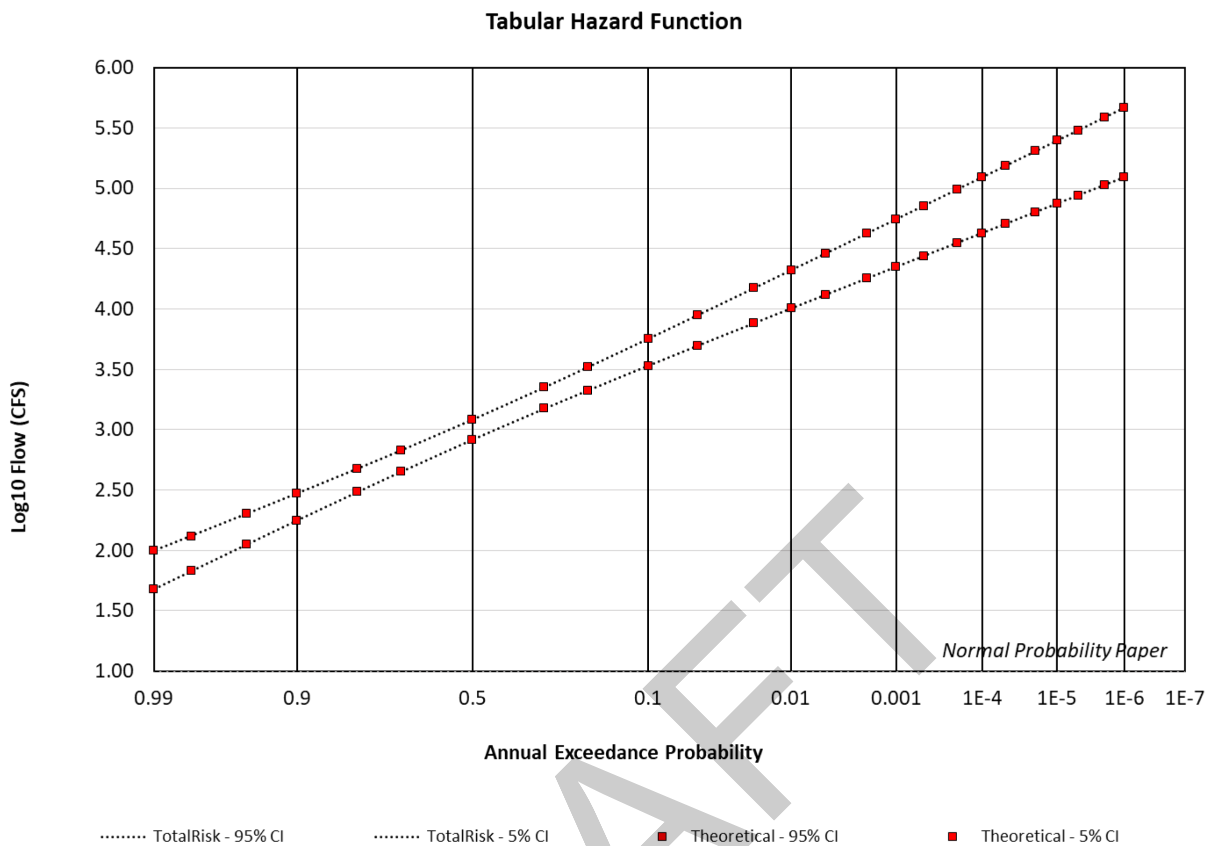

Figure 20 - Comparison of the RMC-TotalRisk tabular hazard function with theoretical confidence intervals for the Log-Normal (base 10) distribution

Table 39 - Comparison of the RMC-TotalRisk tabular hazard function with theoretical confidence intervals for the Log-Normal (base 10) distribution. Quantile results are shown in log (base 10) space.

AEP	5% - CI			95% - CI		
	Theoretical	RMC-TotalRisk	% Difference	Theoretical	RMC-TotalRisk	% Difference
1.0E-06	5.0883	5.0913	0.1%	5.6651	5.6662	0.0%
2.0E-06	5.0252	5.0281	0.1%	5.5862	5.5872	0.0%
5.0E-06	4.9389	4.9416	0.1%	5.4783	5.4793	0.0%
1.0E-05	4.8711	4.8738	0.1%	5.3937	5.3947	0.0%
2.0E-05	4.8011	4.8037	0.1%	5.3064	5.3073	0.0%
5.0E-05	4.7046	4.7070	0.1%	5.1860	5.1869	0.0%
1.0E-04	4.6281	4.6305	0.1%	5.0909	5.0917	0.0%
2.0E-04	4.5484	4.5506	0.1%	4.9917	4.9925	0.0%
5.0E-04	4.4370	4.4391	0.0%	4.8535	4.8543	0.0%
1.0E-03	4.3475	4.3495	0.0%	4.7428	4.7435	0.0%
2.0E-03	4.2526	4.2545	0.0%	4.6256	4.6262	0.0%
5.0E-03	4.1170	4.1188	0.0%	4.4588	4.4594	0.0%
1.0E-02	4.0048	4.0065	0.0%	4.3215	4.3221	0.0%
2.0E-02	3.8819	3.8834	0.0%	4.1719	4.1724	0.0%
5.0E-02	3.6963	3.6976	0.0%	3.9486	3.9490	0.0%
1.0E-01	3.5298	3.5309	0.0%	3.7518	3.7522	0.0%
2.0E-01	3.3251	3.3261	0.0%	3.5165	3.5169	0.0%
3.0E-01	3.1745	3.1754	0.0%	3.3499	3.3502	0.0%
5.0E-01	2.9178	2.9186	0.0%	3.0822	3.0825	0.0%
7.0E-01	2.6501	2.6510	0.0%	2.8255	2.8258	0.0%
8.0E-01	2.4835	2.4845	0.0%	2.6749	2.6752	0.0%
9.0E-01	2.2482	2.2494	0.1%	2.4702	2.4706	0.0%
9.5E-01	2.0514	2.0527	0.1%	2.3037	2.3042	0.0%
9.8E-01	1.8281	1.8296	0.1%	2.1181	2.1187	0.0%
9.9E-01	1.6785	1.6801	0.1%	1.9952	1.9957	0.0%

Event Tree Response Function

RMC-TotalRisk includes the ability to define a system response function using an event tree. Event tree analysis (ETA) represents the logic of how an initiating event, like a flood or earthquake, can lead to various types of damage and failure [19]. It is common practice to develop detailed event trees for individual PFMs to clearly identify the full sequence of steps required to obtain failure or breach. Each identified PFM is decomposed into a sequence of component events and conditions that must occur for there to be a failure. More details on the event tree response function can be found in the technical reference manual [2].

The event tree math is straightforward, requiring simple multiplication of node probabilities. However, the event tree in RMC-TotalRisk provides comprehensive diagnostics and sensitivity analysis results, which are more complex.

The event tree response function and sensitivity analysis results were verified using *Palisade's @Risk* software[13]. Figure 21 provides an example of a simple event tree for a backwards erosion and piping failure mode. Table 40 shows the event tree node probabilities for a single hazard level. Each node has a Triangular distribution.

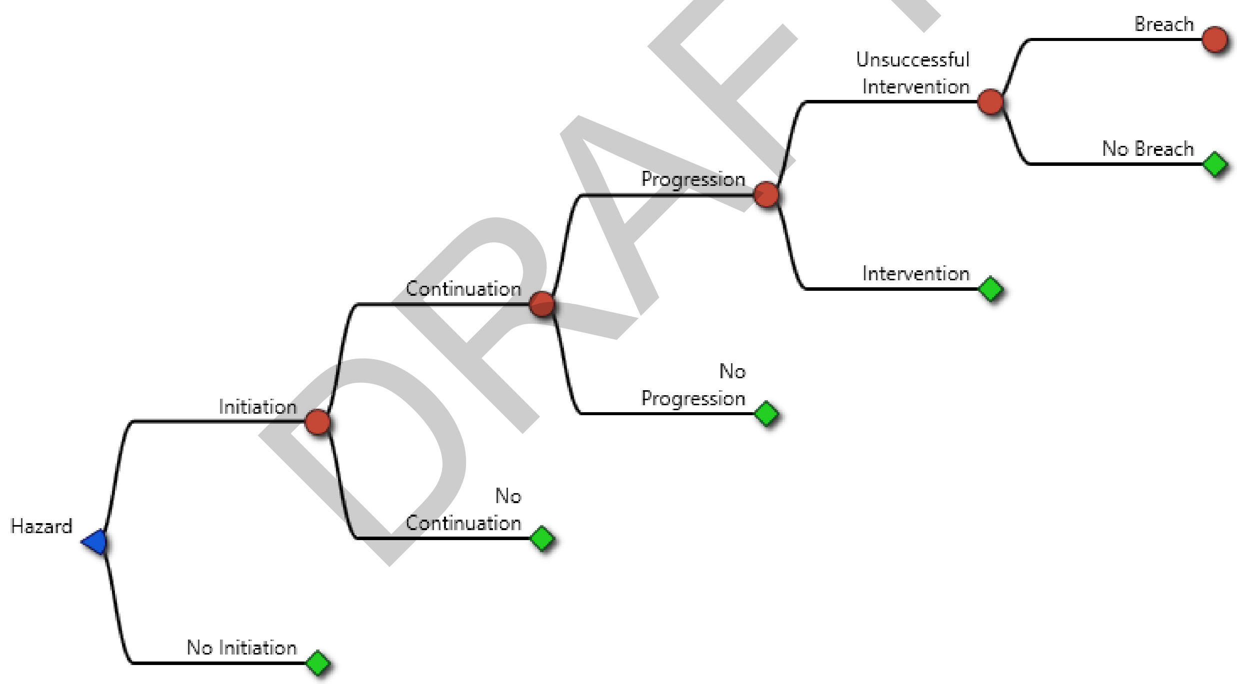

Figure 21 - Example backwards erosion piping event tree.

[13] https://www.palisade.com/risk/default.asp

Table 40 - Event tree node probabilities.

Node	Min	Most Likely	Max
Initiation	0.5	0.8	1.0
Continuation	0.6	0.7	0.9
Progression	0.2	0.3	0.7
Unsuccessful Intervention	0.8	0.9	1.0
Breach	0.4	0.5	0.8

Figure 22 shows an example of the event tree diagnostics tab in RMC-TotalRisk. For this comparison, 10,000 Monte Carlo iterations were performed in both RMC-TotalRisk and *Palisade's @Risk©*.

Table 41 provides a comparison of the summary statistics from the Monte Carlo simulation. Table 42 provides a comparison of the Pearson's correlation coefficients, which describe the strength and direction of an association between the simulated input and output variables in the event tree. Table 43 shows a comparison of the sensitivity indices, which is often referred to as the *contribution to variance* [20]. This provides the fractional contribution of variance from the input to the total output variance.

The event tree analysis results from RMC-TotalRisk very closely match those from *Palisade's @Risk©*. Any differences in results are due to minor differences that arise from pseudo-random number generators and Monte Carlo sampling errors.

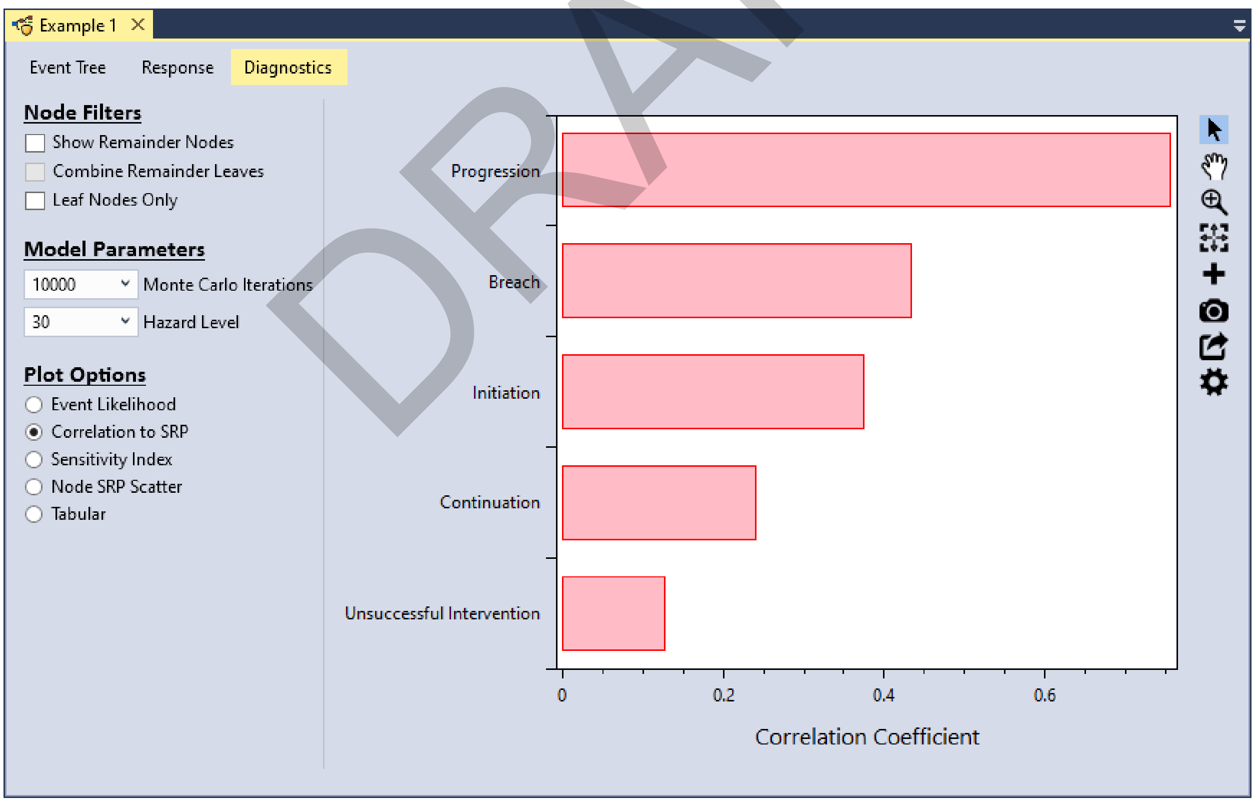

Figure 22 - Example of even tree diagnostics tab in RMC-TotalRisk.

Table 41 - Comparison of event tree response probability summary statistics.

Statistic	Palisade's @RIsk©	RMC-TotalRisk	% Difference
Mean	0.1147	0.1148	0.1%
Std. Deviation	0.0409	0.0412	0.7%
5^{th} %-ile	0.0609	0.0614	0.9%
50^{th} %-ile	0.1074	0.1074	0.1%
95^{th} %-ile	0.1917	0.1920	0.1%

Table 42 - Comparison of event tree node correlation coefficients.

Node	Palisade's @RIsk©	RMC-TotalRisk	% Difference
Initiation	0.37	0.37	0.0%
Continuation	0.24	0.24	0.0%
Progression	0.76	0.76	0.0%
Unsuccessful Intervention	0.13	0.13	0.0%
Breach	0.43	0.43	0.0%

Table 43 - Comparison of event tree node contribution to variance.

Node	Palisade's @RIsk©	RMC-TotalRisk	Difference
Initiation	14.3%	13.7%	0.6%
Continuation	5.7%	5.5%	0.2%
Progression	56.7%	56.9%	0.2%
Unsuccessful Intervention	1.5%	1.6%	0.1%
Breach	18.1%	17.3%	0.8%

Composite Hazard and Response Functions

In RMC-TotalRisk, a composite hazard or response function can be created by assigning weights (or likelihoods) to a list of functions as follows:

$$F(x) = \sum_{i=1}^{n} \omega_i \cdot F_i\,(x) \qquad \text{Equation 49}$$

where $F_i(\cdot)$ is the CDF for function i; and ω_i is the weight or likelihood of function i, with $0 \leq \omega_i \leq 1$ and $\sum_{i=1}^{n} \omega_i = 1$. This type of composite function is traditionally referred to as a *mixture distribution* [21].

In dam safety, it is common practice to evaluate various gate failure or debris blockage scenarios as separate analyses, and then assign a likelihood to each scenario. Similarly, a system response function might be a function of multiple hazard scenarios. The joint probability of the various hazards can be accounted for using weights (or likelihoods) for discrete hazard bins.

For more details on the computation and algorithmic aspects of the composite hazard and response functions, please refer to the technical reference manual [2]. The composite hazard and response functions and resulting confidence intervals were verified using the *R 'mistr'* package[14] for a mixture of three Normal distributions, which are shown in Table 44.

Table 44 – Composite hazard and response distribution parameters and weights.

Distribution	Mean, μ	Std. Deviation, σ	ERL	Weight
1	10	2	100	0.3
2	20	1	100	0.2
3	30	5	100	0.5

Verification of the computed mixture distribution was performed using built-in functions within the *R 'mistr'* package. The confidence intervals were created by performing the bootstrap analysis with 10,000 realizations for each sub-distribution. Then, for each realization, a mixture distribution was created using the weights in Table 44. Finally, confidence intervals can be derived by computing percentiles from the 10,000 bootstrapped mixture distributions (see the **Parametric Bootstrap** section for details on the bootstrap algorithm).

The composite hazard function from RMC-TotalRisk is shown in Figure 23. The composite response function produces the same results, but it is plotted as a CDF with the hazard levels versus the non-exceedance probabilities. Whereas the composite hazard function plots the exceedance probabilities versus the hazard levels.

[14] https://cran.r-project.org/web/packages/mistr/index.html

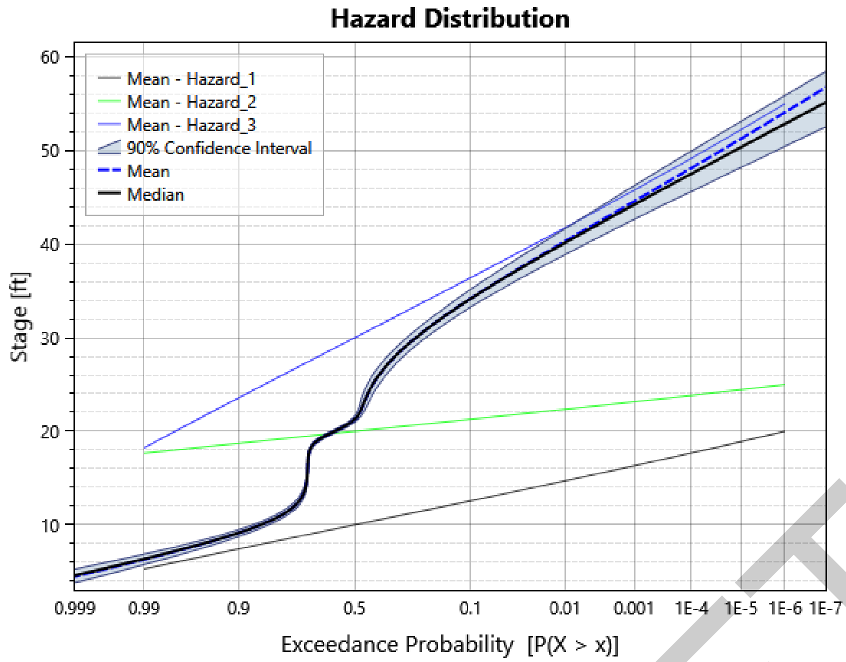

Figure 23 – Composite hazard function for three Normal distributions in RMC-TotalRisk.

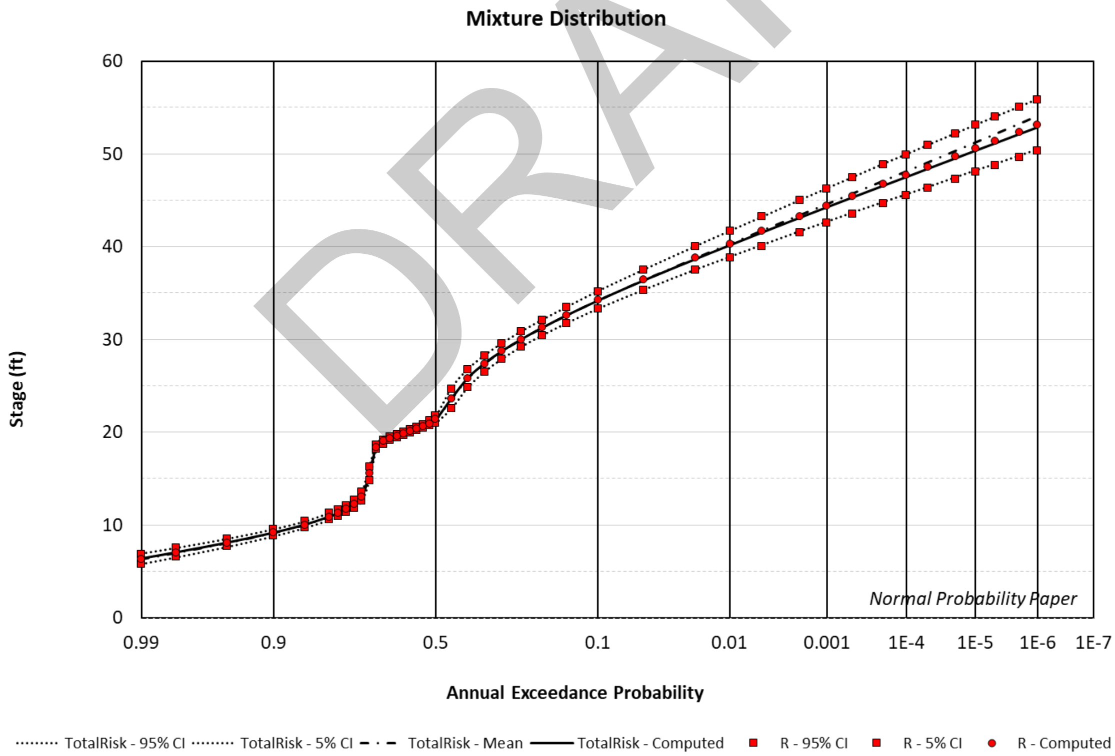

Figure 24 - Comparison of RMC-TotalRisk with R 'mistr' confidence intervals for the mixture distribution.

Figure 24 shows a frequency curve plot comparing the bootstrap confidence interval results from RMC-TotalRisk to the *R 'mistr'* package. Table 45 lists the results for the computed curve and Table 46 lists the confidence interval results and the percent difference between the two software programs. It is clear from these results that RMC-TotalRisk produces effectively identical results to those created with *R*. Any differences in results are due to minor differences that arise from numerical precision differences, differences from pseudo-random number generators, and Monte Carlo sampling errors. Example code for getting the computed curve with *R 'mistr'* is provided in Figure 25 below.

Table 45 - Comparison of RMC-TotalRisk with R 'mistr' computed curve for the mixture distribution.

AEP	*R 'mistr'*	RMC-TotalRisk	% Difference
1.0E-06	53.10	53.06	0.1%
2.0E-06	52.34	52.33	0.0%
5.0E-06	51.33	51.32	0.0%
1.0E-05	50.54	50.54	0.0%
2.0E-05	49.72	49.72	0.0%
5.0E-05	48.60	48.59	0.0%
1.0E-04	47.70	47.70	0.0%
2.0E-04	46.76	46.76	0.0%
5.0E-04	45.45	45.45	0.0%
1.0E-03	44.39	44.39	0.0%
2.0E-03	43.26	43.26	0.0%
5.0E-03	41.63	41.63	0.0%
1.0E-02	40.27	40.27	0.0%
2.0E-02	38.75	38.75	0.0%
5.0E-02	36.40	36.40	0.0%
1.0E-01	34.21	34.21	0.0%
2.0E-01	31.26	31.26	0.0%
3.0E-01	28.72	28.72	0.0%
5.0E-01	21.38	21.38	0.0%
7.0E-01	15.58	15.58	0.0%
8.0E-01	10.88	10.88	0.0%
9.0E-01	9.15	9.15	0.0%
9.5E-01	8.07	8.07	0.0%
9.8E-01	7.00	7.00	0.0%
9.9E-01	6.34	6.34	0.0%

Table 46 - Comparison of RMC-TotalRisk with R 'mistr' confidence intervals for the mixture distribution.

AEP	5% - CI			95% - CI		
	R 'mistr'	**RMC-TotalRisk**	**% Difference**	***R 'mistr'***	**RMC-TotalRisk**	**% Difference**
1.0E-06	50.27	50.46	0.4%	55.80	55.87	0.1%
2.0E-06	49.62	49.78	0.3%	55.00	55.06	0.1%
5.0E-06	48.72	48.88	0.3%	53.94	53.96	0.0%
1.0E-05	48.02	48.19	0.4%	53.07	53.09	0.0%
2.0E-05	47.29	47.45	0.3%	52.16	52.19	0.1%
5.0E-05	46.29	46.42	0.3%	50.90	50.93	0.1%
1.0E-04	45.49	45.62	0.3%	49.91	49.93	0.0%
2.0E-04	44.65	44.80	0.4%	48.87	48.91	0.1%
5.0E-04	43.47	43.63	0.4%	47.43	47.43	0.0%
1.0E-03	42.52	42.66	0.3%	46.26	46.25	0.0%
2.0E-03	41.51	41.64	0.3%	45.00	44.99	0.0%
5.0E-03	40.04	40.14	0.3%	43.21	43.17	0.1%
1.0E-02	38.81	38.89	0.2%	41.71	41.68	0.1%
2.0E-02	37.44	37.50	0.2%	40.05	40.02	0.1%
5.0E-02	35.30	35.33	0.1%	37.50	37.50	0.0%
1.0E-01	33.25	33.28	0.1%	35.15	35.17	0.1%
2.0E-01	30.42	30.42	0.0%	32.09	32.09	0.0%
3.0E-01	27.89	27.89	0.0%	29.57	29.57	0.0%
5.0E-01	20.98	20.98	0.0%	21.73	21.64	0.4%
7.0E-01	14.81	14.78	0.2%	16.29	16.26	0.2%
8.0E-01	10.55	10.50	0.5%	11.24	11.23	0.1%
9.0E-01	8.80	8.76	0.5%	9.51	9.49	0.2%
9.5E-01	7.67	7.62	0.7%	8.49	8.47	0.2%
9.8E-01	6.52	6.48	0.6%	7.49	7.47	0.2%
9.9E-01	5.80	5.77	0.5%	6.87	6.87	0.1%


```
library(mistr)

# define the sub-distributions
n1 = normdist(mean = 10, sd = 2)
n2 = normdist(mean = 20, sd = 1)
n3 = normdist(mean = 30, sd = 5)

# create the mixture
mix = mixdist(n1, n2, n3, weights = c(0.3, 0.2, 0.5))

# get a list of non-exceedance probabilities given the x-values
pVals = p(mix, q= seq(0, 60, 1))
```
Figure 25 – Example code for creating a mixture distribution with the R 'mistr' package.

Composite Consequence Function

In dam and levee safety, there is often a need to combine multiple consequence functions into a single composite function. RMC-TotalRisk provides three methods for creating a composite consequence function: 1) consequence functions are summed; 2) consequence functions are averaged based on user-defined weights; and 3) the uncertainty in consequence functions is treated as a mixture distribution based on user-defined weights.

Additive

A composite consequence function can be created by summing across a list of consequences functions. This capability is useful when flood damages are estimated separately by types or economic sectors. For example, there could be damages to private properties, industrial buildings, agriculture, etc. The damage to each sector can be estimated separately and then aggregated to a total damage using the composite consequence function.

The uncertainty analysis procedures for composite consequence functions are described in the technical reference manual [2]. A verification of the uncertainty routine for additive consequences was performed using the theoretical solution for the sum of three independent Normal distributions.

Three hypothetical consequence functions representing life loss for different river stages (in feet) were created for the purposed of this verification. At a stage of zero, the life loss was zero for all three functions. The distribution of life loss at a stage of 10 ft was given by a Normal distribution for each function as shown in Table 47. The resulting additive composite consequence function is shown in Figure 26 below.

Table 47 - Listing of consequence uncertainty for a single hazard level at three consequence functions.

Function	Mean Life Loss, μ	Std. Deviation, σ
1	10	2
2	20	1
3	100	5

The mean (μ_{XY}) and standard deviation (σ_{XY}) of the sum of two independent random variables is as follows:

$$\mu_{XY} = \mu_X + \mu_Y \qquad \text{Equation 50}$$

$$\sigma_{XY} = \sqrt{\sigma_X^2 + \sigma_Y^2} \qquad \text{Equation 51}$$

These formulas are easily extended to multiple random variables; for example the mean of the sum of three random variables is $\mu_{XYZ} = \mu_X + \mu_Y + \mu_Z$ and standard deviation is $\sigma_{XYZ} = \sqrt{\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2}$.

Since the three input distributions are Normally distributed, the resulting distribution of the sum is also Normally distributed. Therefore, the exact theoretical solution is easily obtained. Table 48 shows the verification results for the additive composite consequence functions. RMC-TotalRisk produces a near perfect match with the theoretical solution. The minor differences are due to Monte Carlo sampling errors in the RMC-TotalRisk uncertainty routine.

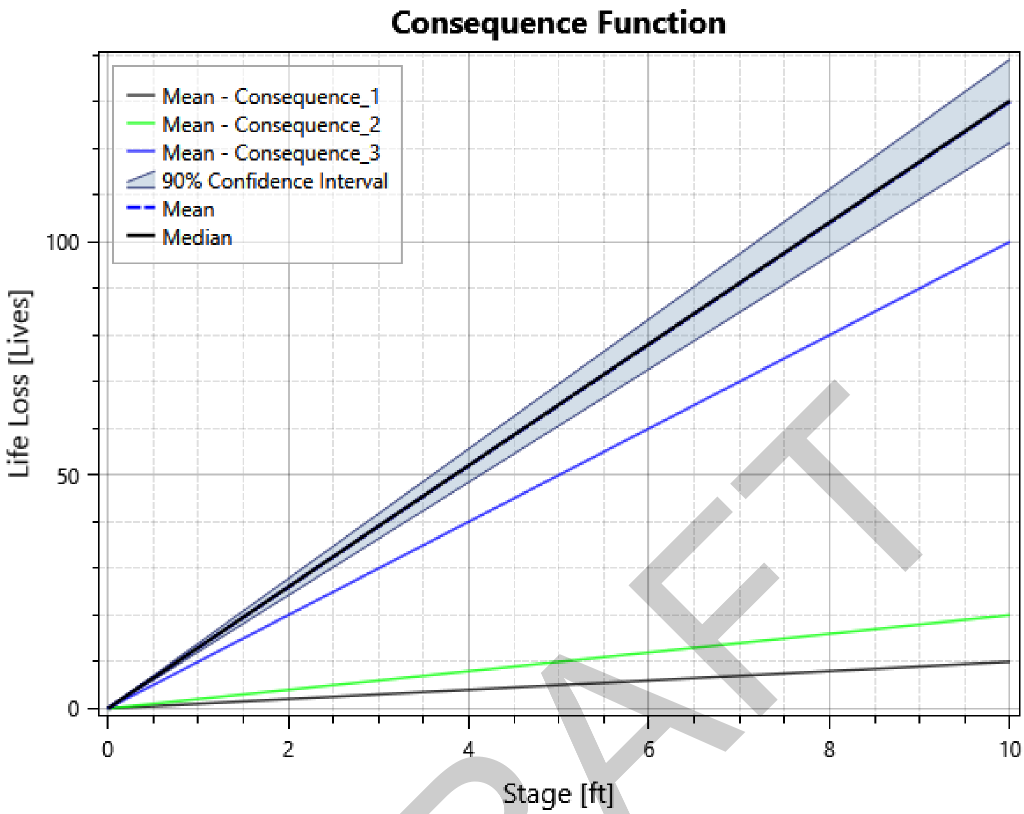

Figure 26 - Composite consequence function with additive consequences.

Table 48 - Verification results for the additive composite consequence function.

Statistic	Exact Solution	RMC-TotalRisk	% Difference
Mean	130.00	130.01	0.0%
Std. Deviation	5.48	5.49	0.2%
5^{th} %-ile	120.99	120.94	0.0%
95^{th} %-ile	139.01	139.03	0.0%

Average

In Dam Safety risk analysis, it has been common practice to evaluate daytime and nighttime consequences separately, and then assigning a weight (or likelihood) to each scenario. For example, daytime consequences are typically given a weight of 0.42 and nighttime consequences are given a weight of 0.58. Then, the composite consequences are derived by treating day and night consequences as a weighted average.

A verification of the uncertainty routine for averaged consequences was performed using the theoretical solution for the average of three independent Normal distributions. Verification of the averaged composite consequence was performed using the same functions as shown in Table 47. Each function was given a weight of 0.3, 0.2, and 0.5, respectively. The resulting average composite consequence function is shown in Figure 27 below.

The mean and standard deviation of the average of three independent random variables is:

$$\mu_{XYZ} = \omega_X \cdot \mu_X + \omega_Y \cdot \mu_Y + \omega_Z \cdot \mu_Z \qquad \text{Equation 52}$$

$$\sigma_{XYZ} = \sqrt{\omega_X^2 \cdot \sigma_X^2 + \omega_Y^2 \cdot \sigma_Y^2 + \omega_Z^2 \cdot \sigma_Z^2} \qquad \text{Equation 53}$$

where ω_X is the weight given to variable X, ω_Y is the weight given to variable Y, and ω_Z is the weight given to variable Z, with $\omega_X + \omega_Y + \omega_Z = 1$.

Since the three input distributions are Normally distributed, the resulting distribution of the average is also Normally distributed. Therefore, the exact theoretical solution is easily obtained. Table 49 shows the verification results for the average composite consequence functions. RMC-TotalRisk produces a near perfect match with the theoretical solution. The minor differences are due to Monte Carlo sampling errors in the RMC-TotalRisk uncertainty routine.

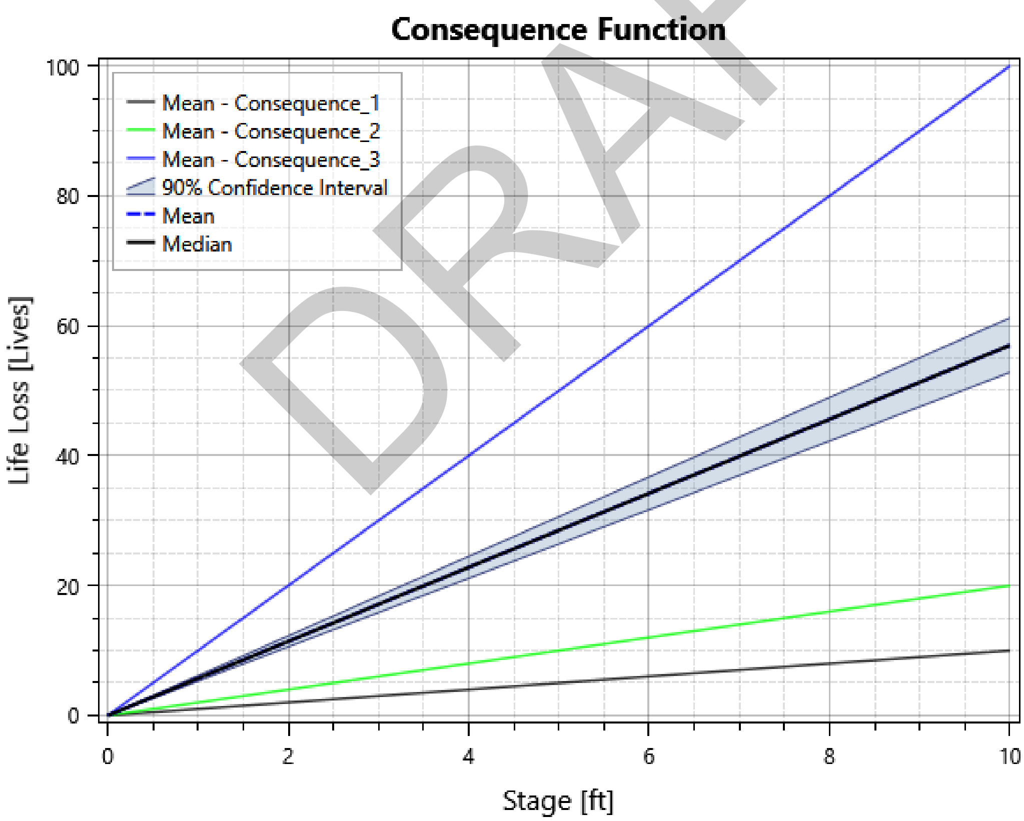

Figure 27 - Composite consequence function with averaged consequences.

Table 49 - Verification results for the averaged composite consequence function.

Statistic	Exact Solution	RMC-TotalRisk	% Difference
Mean	57.00	57.00	0.0%
Std. Deviation	2.58	2.58	0.2%
5th %-ile	52.76	52.75	0.0%
95th %-ile	61.24	61.27	0.0%

Mixture

Rather than treating consequences as a weighted average, uncertainty from different consequence scenarios can be treated as a mixture distribution. In the past, consequences from day and nighttime exposure scenarios were combined as a weighted average. However, these scenarios are more appropriately combined as a mixture, which will fully capture the uncertainty from all scenarios.

A verification of the uncertainty routine for mixture consequences was performed using the theoretical solution for the mixture of three independent Normal distributions. An additional Monte Carlo simulation was performed to verify the percentiles. Verification of the mixture composite consequence was performed using the same functions as shown previously in Table 47. Each function was given a weight of 0.3, 0.2, and 0.5, respectively. The resulting mixture composite consequence function is shown in Figure 28 below.

The mean and standard deviation of a mixture of three distributions is:

$$\mu_{XYZ} = \omega_X \cdot \mu_X + \omega_Y \cdot \mu_Y + \omega_Z \cdot \mu_Z \qquad \textit{Equation 54}$$

$$\sigma_{XYZ} = \sqrt{\omega_X \cdot (\mu_X^2 + \sigma_X^2) + \omega_Y \cdot (\mu_Y^2 + \sigma_Y^2) + \omega_Z \cdot (\mu_Z^2 + \sigma_Z^2) - \mu_{XYZ}^2} \qquad \textit{Equation 55}$$

where ω_X is the weight given to variable X, ω_Y is the weight given to variable Y, and ω_Z is the weight given to variable Z, with $\omega_X + \omega_Y + \omega_Z = 1$.

The mean of the average of random variables (Equation 52) is equivalent to the mean of the mixture distribution (Equation 54). However, the standard deviation of the average (Equation 53) can be much smaller than the mixture (Equation 55).

Even though the three input distributions are Normally distributed, the resulting mixture distribution is not Normally distributed. Therefore, the exact theoretical solution for the mean and standard deviation can be obtained, but the percentiles of the mixture does not have an exact solution. As such, and additional Monte Carlo simulation with 10 million samples was performed to verify the RMC-TotalRisk results. Table 50 shows the verification results for the mixture composite consequence functions as compared to the exact theoretical solution. Table 51 shows the comparison of RMC-TotalRisk to the Monte Carlo simulation results. RMC-TotalRisk produces a near perfect match with the theoretical and Monte Carlo solutions. The minor differences are due to Monte Carlo sampling errors in the RMC-TotalRisk uncertainty routine.

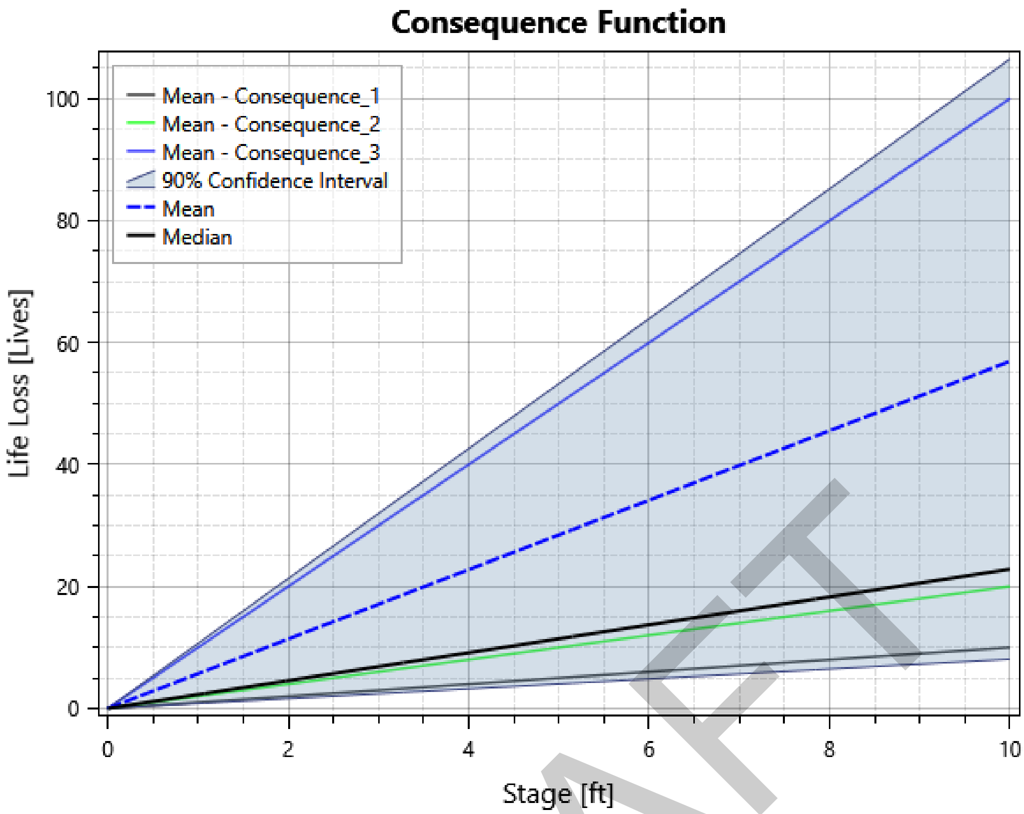

Figure 28 - Composite consequence function with uncertainty in consequences treated as a mixture.

Table 50 - Verification results for the mixture composite consequence function.

Statistic	Exact Solution	RMC-TotalRisk	% Difference
Mean	57.00	57.01	0.0%
Std. Deviation	43.30	43.28	0.0%

Table 51 – Comparison of RMC-TotalRisk to a Monte Carlo simulation for the mixture composite consequence function.

Statistic	Monte Carlo	RMC-TotalRisk	% Difference
Mean	56.86	57.01	0.3%
Std. Deviation	43.29	43.28	0.0%
5^{th} %-ile	8.04	8.04	0.0%
95^{th} %-ile	106.38	106.39	0.0%

Verification of Risk Analysis

In RMC-TotalRisk, within every Monte Carlo realization for every system component, risk is computed using numerical integration. The **Numerical Integration** section provided verification of the Adaptive Simpson's Rule (ASR) and VEGAS methods. ASR is used for calculating risk for a system with a single component (or single dimension), whereas VEGAS is used for calculating risk for systems with two or more components (multidimensions).

The ASR and VEGAS methods were verified against analytical integrand functions with known solutions. The integrands for real-world risk analyses are often very complex and they do not have a closed form analytical solution. Consequently, verification of the risk analysis was performed using Monte Carlo simulation. The Monte Carlo simulation approach randomly samples millions of events, and the expected value is obtained from a simple arithmetic average from all the samples. Monte Carlo simulation is especially useful for higher-dimensional problems where analytical solutions are not available.

RMC- TotalRisk can perform multi-failure mode risk analysis for a single system component or for a complex system with multiple components. In addition, the user has control over the dependencies between failure modes and system components, as well as how consequences of joint failures should be handled. With this level of complexity, there are several aspects of the risk analysis that must be verified.

To capture a reasonable range of complexity, nearly 50 verification tests were performed using Monte Carlo simulation. Each system component has the same hazard function, which was a Ln-Normal distribution with a real-space mean $\mu = 85$ and standard deviation $\sigma = 20$. There are up to five potential failure modes, each entered as a Normal distribution (Table 52). Each failure mode has associated failure consequences which were entered as tabular functions (Table 53). Each system component has the same non-failure consequences. These inputs were used to develop the various system configurations listed in Table 54.

Table 52 - Potential failure mode inputs.

Failure Modes	**Mean, μ**	**Std. Deviation, σ**
PFM-1	140	30
PFM-2	160	10
PFM-3	150	20
PFM-4	130	35
PFM-5	160	15

Table 53 - Consequence function inputs.

Stage (ft)	**Non-Fail**	**PFM-1**	**PFM-2**	**PFM-3**	**PFM-4**	**PFM-5**
60	0	0	0	0	0	0
100	1	5	3	10	2	8
140	10	50	30	100	20	80
200	100	500	300	1,000	200	800
250	150	750	450	1,500	300	1,200

Table 54 - Listing of all risk analysis verification tests.

Test	System Options			Failure Mode Options			
	Components	**Dependency**	**Joint Consequences**	**Modes**	**Method**	**Dependency**	**Joint Consequences**
1	1	N/A	N/A	2	CCA	Independent	N/A
2	1	N/A	N/A	2	CCA	Negative	N/A
3	1	N/A	N/A	5	CCA	Independent	N/A
4	1	N/A	N/A	5	CCA	Negative	N/A
5	1	N/A	N/A	2	Competing	Independent	N/A
6	1	N/A	N/A	5	Competing	Independent	N/A
7	1	N/A	N/A	2	Joint	Independent	Additive
8	1	N/A	N/A	2	Joint	Independent	Average
9	1	N/A	N/A	2	Joint	Independent	Maximum
10	1	N/A	N/A	2	Joint	Independent	Minimum
11	1	N/A	N/A	2	Joint	Negative	Additive
12	1	N/A	N/A	2	Joint	Negative	Average
13	1	N/A	N/A	2	Joint	Negative	Maximum
14	1	N/A	N/A	2	Joint	Negative	Minimum
15	1	N/A	N/A	5	Joint	Independent	Additive
16	1	N/A	N/A	5	Joint	Independent	Average
17	1	N/A	N/A	5	Joint	Independent	Maximum
18	1	N/A	N/A	5	Joint	Independent	Minimum
19	1	N/A	N/A	5	Joint	Negative	Additive
20	1	N/A	N/A	5	Joint	Negative	Average
21	1	N/A	N/A	5	Joint	Negative	Maximum
22	1	N/A	N/A	5	Joint	Negative	Minimum
23	2	Independent	Additive	1	N/A	N/A	N/A
24	2	Independent	Average	1	N/A	N/A	N/A
25	2	Independent	Maximum	1	N/A	N/A	N/A
26	2	Independent	Minimum	1	N/A	N/A	N/A
27	2	Positive	Additive	1	N/A	N/A	N/A
28	2	Positive	Average	1	N/A	N/A	N/A
29	2	Positive	Maximum	1	N/A	N/A	N/A
30	2	Positive	Minimum	1	N/A	N/A	N/A
31	2	Negative	Additive	1	N/A	N/A	N/A
32	2	Negative	Average	1	N/A	N/A	N/A
33	2	Negative	Maximum	1	N/A	N/A	N/A
34	2	Negative	Minimum	1	N/A	N/A	N/A
35	2	Independent	Additive	2	Joint	Independent	Additive
36	2	Positive	Additive	2	Joint	Independent	Additive
37	2	Negative	Additive	2	Joint	Independent	Additive
38	5	Independent	Additive	1	N/A	N/A	N/A
39	5	Independent	Average	1	N/A	N/A	N/A
40	5	Independent	Maximum	1	N/A	N/A	N/A
41	5	Independent	Minimum	1	N/A	N/A	N/A
42	5	Positive	Additive	1	N/A	N/A	N/A
43	5	Positive	Average	1	N/A	N/A	N/A
44	5	Positive	Maximum	1	N/A	N/A	N/A
45	5	Positive	Minimum	1	N/A	N/A	N/A
46	5	Negative	Additive	1	N/A	N/A	N/A
47	5	Negative	Average	1	N/A	N/A	N/A
48	5	Negative	Maximum	1	N/A	N/A	N/A
49	5	Negative	Minimum	1	N/A	N/A	N/A

Every risk analysis listed above in Table 54 was estimated using the "Simulate Mean Risk Only" option [2] in RMC-TotalRisk and compared against a Monte Carlo simulation with 10 million samples. For every verification test, all five risk types were computed and reported to 6 decimal places. As discussed in [2], the five risk types are: 1) incremental $\mathbb{E}[C_\Delta]$; 2) background $\mathbb{E}[C_B]$; 3) total $\mathbb{E}[C_T]$; 4) failure $\mathbb{E}[C_F]$; and 5) non-failure $\mathbb{E}[C_{NF}]$.

For a single system component, the total risk is the sum of incremental plus background risk, and the sum of failure plus non-failure risk:

$$\mathbb{E}[C_T] = \mathbb{E}[C_\Delta] + \mathbb{E}[C_B] \quad \text{Equation 56}$$

$$\mathbb{E}[C_T] = \mathbb{E}[C_F] + \mathbb{E}[C_{NF}] \quad \text{Equation 57}$$

However, in RMC-TotalRisk, a system can have multiple components, each with multiple failure modes where each component also has a separate non-failure mode. As such, there is a potential for some embedded correlation between incremental consequences and non-failure consequences across system components. Therefore, for a system with multiple components, depending on the joint consequence rule, total risk will be greater than or equal to incremental plus background risk:

$$\mathbb{E}[C_T]_\Omega \geq \mathbb{E}[C_\Delta]_\Omega + \mathbb{E}[C_B]_\Omega \quad \text{Equation 58}$$

$$\mathbb{E}[C_T]_\Omega = \mathbb{E}[C_F]_\Omega + \mathbb{E}[C_{NF}]_\Omega \quad \text{Equation 59}$$

The complete mathematic details behind these risk types are provided in the technical reference manual [2]. The following subsections provide a full listing of the verification test results and a description of the Monte Carlo algorithms used for various system configurations.

Multiple Failure Modes

The first 22 verification tests evaluate a single system component with multiple failure modes. RMC-TotalRisk provides three computational methods for assessing multiple failure modes: 1) the common cause adjustment, 2) competing failure modes, and 3) joint failure modes. The mathematic details behind these computational methods are provided in the technical reference manual [2].

Common Cause Failure Modes

The Common Cause Adjustment (CCA) is a method that was originally intended for failure modes that are not mutually exclusive and that can occur simultaneously at multiple sections of a dam due to a single or common cause initiating event [22]. The CCA was originally intended for positively correlated or independent failure modes. However, there are situations where failure modes can be negatively dependent, which will lead to a higher combined probability of failure for the system. RMC-TotalRisk employs a generalized version of the CCA that can also work with negative dependency.

The description of the Monte Carlo routine for estimating the risk of failure for independent CCA failure modes is provided in Algorithm 1. This routine can be expanded to include all risk types [2].

Algorithm 1 – Simulate Risk of Failure with Independent CCA Failure Modes

$R \leftarrow$ number of Monte Carlo realizations
$M \leftarrow$ number of failure modes
for $i \leftarrow 1$ to R **do**
    $h \leftarrow F_H^{-1}(r_i)$ where $r_i \sim \mathrm{U}(0,1)$ ▻Randomly sample a hazard level
    for $j \leftarrow 1$ to M **do**
        $p_{f_j} \leftarrow F_{FM_j}(h)$ ▻ Get the probability of failure of each failure mode given the hazard level
    end for
    for $j \leftarrow 1$ to M **do**
        $p_{f_j} \leftarrow c \cdot p_{f_j}$ ▻ Perform common cause adjustment of each failure probability
        $F_{p_f} \leftarrow F_{p_f} + p_{f_j}$ ▻ Create a cumulative distribution across all modes
        If $r_j \leq F_{p_f}$ where $r_j \sim \mathrm{U}(0,1)$ **then** ▻ Randomly sample to determine failure
            $N_f \leftarrow N_f + C_{f_j}(h)$ ▻ The system failed, so get the consequences of failure given the hazard level
            break
        end if
    end for
end for
Estimate the mean risk of failure $\mathbb{E}[N_f] \leftarrow N_f / R$

The verification results for the CCA method are provided in Table 55 through Table 58. RMC-TotalRisk has near perfect agreement with the Monte Carlo results. The expected values $\mathbb{E}[\mathrm{N}]$ of all five risk types are provided in the tables. As shown in Table 55, total risk is the sum of incremental plus background risk:

$$\mathbb{E}[C_T] = \mathbb{E}[C_\Delta] + \mathbb{E}[C_B]$$ *Equation 60*

$$3.080821 = 1.654059 + 1.426762$$ *Equation 61*

Likewise, total risk is also the sum of failure plus non-failure risk:

$$\mathbb{E}[C_T] = \mathbb{E}[C_F] + \mathbb{E}[C_{NF}]$$ *Equation 62*

$$3.080821 = 2.113075 + 0.967746$$ *Equation 63*

Table 55 - 1 system component with 2 independent common cause failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental, $\mathbb{E}[C_{\Delta}]$	1.655845	1.654059	0.1%
Background, $\mathbb{E}[C_B]$	1.427955	1.426762	0.1%
Total, $\mathbb{E}[C_T]$	3.083800	3.080821	0.1%
Failure, $\mathbb{E}[C_F]$	2.115712	2.113075	0.1%
Non-Failure, $\mathbb{E}[C_{NF}]$	0.968088	0.967746	0.0%

Table 56 - 1 system component with 2 negatively dependent common cause failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.736219	1.738393	0.1%
Background	1.427955	1.426760	0.1%
Total	3.164173	3.167373	0.1%
Failure	2.218916	2.222265	0.2%
Non-Failure	0.945257	0.945108	0.0%

Table 57 - 1 system component with 5 independent common cause failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.715177	2.711843	0.1%
Background	1.427955	1.426754	0.1%
Total	4.143132	4.138598	0.1%
Failure	3.494614	3.490207	0.1%
Non-Failure	0.648518	0.648391	0.0%

Table 58 - 1 system component with 5 negatively dependent common cause failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.847377	2.841821	0.2%
Background	1.427955	1.426752	0.1%
Total	4.275332	4.268584	0.2%
Failure	3.667178	3.660357	0.2%
Non-Failure	0.608154	0.608227	0.0%

Competing Failure Modes

A competing failure analysis represents a combination of two or more failure modes that are "competing" to the end of life of a series system. The competing failure mode approach can be thought of as a race to see which failure mode will fail first. The key assumption is that each failure mode proceeds independently of every other one until failure occurs. At the point of first failure, each failure mode is mutually exclusive from one another; i.e., there cannot be joint failures.

The description of the Monte Carlo routine for estimating the risk of failure with independent competing failure modes is provided in Algorithm 2 below. This routine can be expanded to include all risk types. The verification results for competing failures are provided in Table 59 and Table 60. RMC-TotalRisk has near perfect agreement with the Monte Carlo results.

Table 59 - 1 system component with 2 independent competing failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.745274	1.744245	0.1%
Background	1.427955	1.426705	0.1%
Total	3.173229	3.170950	0.1%
Failure	2.204366	2.203265	0.0%
Non-Failure	0.968863	0.967685	0.1%

Table 60 - 1 system component with 5 independent competing failure modes

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.204566	2.197086	0.3%
Background	1.427955	1.426719	0.1%
Total	3.632521	3.623804	0.2%
Failure	2.984006	2.975383	0.3%
Non-Failure	0.648515	0.648421	0.0%

Algorithm 2 – Simulate Risk of Failure with Independent Competing Failure Modes

$R \leftarrow$ number of Monte Carlo realizations
$M \leftarrow$ number of failure modes
for $i \leftarrow 1$ to R **do**
  $h \leftarrow F_H^{-1}(r_i)$ where $r_i \sim \mathrm{U}(0,1)$ ⊳Randomly sample a hazard level
  for $j \leftarrow 1$ to M **do**
    If $r_j \leq F_{FM_j}(h)$ where $r_j \sim \mathrm{U}(0,1)$ **and** $\min_j F_{FM_j}^{-1}(r_j)$ **then** ⊳ Randomly sample to determine failure
    ⊳ The failure mode that fails first is the one with the minimum hazard level given the random sample
      $N_f \leftarrow N_f + C_{f_j}(h)$ ⊳ The system failed, so get the consequences of failure given the hazard level
      break
    end if
  end for
end for
Estimate the mean risk of failure $\mathbb{E}[N_f] \leftarrow N_f / R$

Joint Failure Modes

A joint failure modes analysis directly allows for dependency between failure modes and allows for simultaneous (joint) failures. The only assumption required for this analysis is that a rule must be assumed to account for the joint consequences of failure.

When there are multiple failure modes, the number of possible ways the system can fail is $2^n - 1$. The joint failure mode approach enumerates all possible combinations of failure and non-failure. As the number of failure modes and system components increase, the number of ways the system can fail increases exponentially [2]. Considering this, in RMC-TotalRisk, if the joint failure mode option is selected, the maximum number of failure modes allowable for a single system component is 20.

The description of the Monte Carlo routine for estimating the risk of failure with independent joint failure modes and additive joint consequences is provided in Algorithm 3. This routine can be expanded to include all risk types. Example code for performing a risk analysis for a single system component with joint failure modes and additive consequences is provide in Figure 29 below.

Algorithm 3 – Simulate Risk of Failure with Independent Joint Failure Modes and Additive Joint Consequences

$R \leftarrow$ number of Monte Carlo realizations
$M \leftarrow$ number of failure modes
for $i \leftarrow 1$ to R **do**
    $h \leftarrow F_H^{-1}(r_i)$ where $r_i \sim \mathrm{U}(0,1)$ ▻Randomly sample a hazard level
    for $j \leftarrow 1$ to M **do**
        If $r_j \leq F_{FM_j}(h)$ where $r_j \sim \mathrm{U}(0,1)$ **then** ▻ Randomly sample to determine failure
            $N_f \leftarrow N_f + C_{f_j}(h)$ ▻ The system failed, so get the consequences of failure given the hazard level
        end if
    end for
end for
Estimate the mean risk of failure $\mathbb{E}[N_f] \leftarrow N_f / R$

The verification results for joint failures are provided in Table 61 through Table 76. RMC-TotalRisk has near perfect agreement with the Monte Carlo results.

For a graphical comparison of risk analysis results, Figure 30 shows the FN curves for a single system component with two joint failure modes and additive consequences. The TotalRisk results are shown as thicker transparent lines, and the Monte Carlo results are plotted as dashed lines. The ASR integration approach used by RMC-TotalRisk arrives at nearly the same results as the Monte Carlo simulation; however, the ASR method can do so with only a couple hundred function evaluations rather than the 10 million required by the simulation.

```
library(stats)

Realz = 10000000
incremental = numeric(Realz); background = numeric(Realz); total = numeric(Realz); fail = numeric(Realz); nonfail =
numeric(Realz)
set.seed(12345)

for (i in 1:Realz){

 # Get hazard level
 # Hazard distribution is a Ln-Normal
 mu = 85; sigma = 20; var = sigma^2
 lmu = log(mu^2 / sqrt(var + mu^2)); lsigma = sqrt(log(1.0 + var / mu^2))
 h = qlnorm(p = runif(1), meanlog = lmu, sdlog = lsigma)

 # Get non-fail damages
 nfc = approx(x = c(60, 100, 140, 200, 250), y = c(0, 1, 10, 100, 150), xout = h, yleft = 0, yright = 150)$y

 # Get failure damages
 failed = FALSE; fc = 0; ic = 0
 # PFM 1
 if (runif(1) <= pnorm(q = h, mean = 140, sd = 30)){
  failed = TRUE
  f1 = approx(x = c(60, 100, 140, 200, 250), y = c(0, 5, 50, 500, 750), xout = h, yleft = 0, yright = 750)$y
  fc = fc + f1
}
 # PFM 2
 if (runif(1) <= pnorm(q = h, mean = 160, sd = 10)){
  failed = TRUE
  f2 = approx(x = c(60, 100, 140, 200, 250), y = c(0, 3, 30, 300, 450), xout = h, yleft = 0, yright = 450)$y
  fc = fc + f2
 }
if (failed == TRUE) ic = fc – nfc

 # Record results
 incremental[i] = ic
 background[i] = nfc
 if (failed == TRUE){ nfc = 0 }
 total[i] = fc + nfc
 fail[i] = fc
 nonfail[i] = nfc

}

# Expected consequences
mean(incremental); mean(background); mean(total); mean(fail); mean(nonfail)
# [1] 2.137346
# [1] 1.427623
# [1] 3.56497
# [1] 2.597125
# [1] 0.9678445
```
Figure 29 - Example code for estimation risk for a single system component with joint failure modes with the R ‘stats’ package.

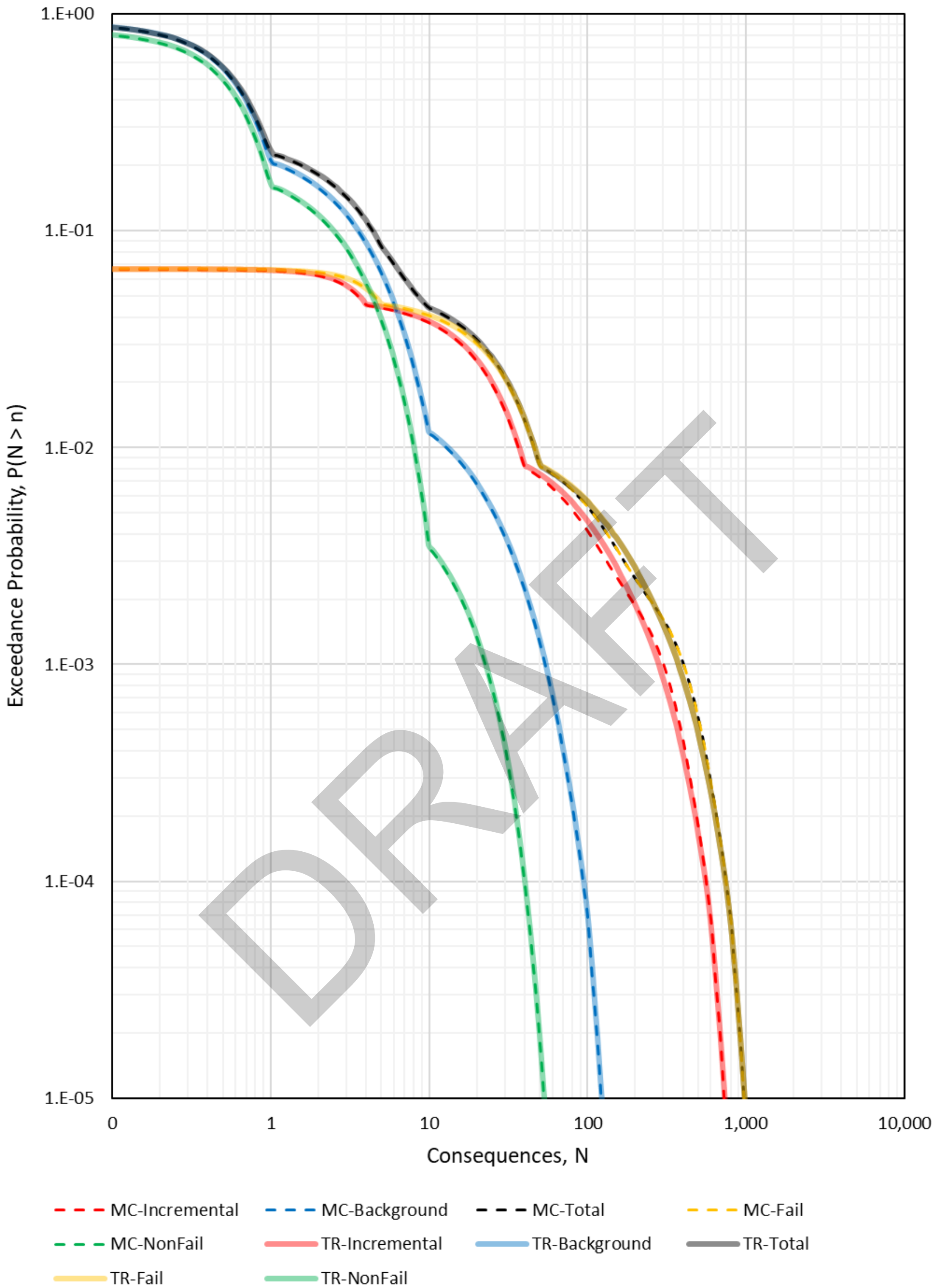

Figure 30 – The F-N plot for a single system component and two independent joint failure modes with additive consequences.

Table 61 - 1 system component with 2 independent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.138221	2.136250	0.1%
Background	1.427955	1.426757	0.1%
Total	3.566176	3.563007	0.1%
Failure	2.597312	2.595259	0.1%
Non-Failure	0.968863	0.967748	0.1%

Table 62 - 1 system component with 2 independent failure modes and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.665706	1.666106	0.0%
Background	1.427955	1.426764	0.1%
Total	3.093661	3.092869	0.0%
Failure	2.124798	2.125125	0.0%
Non-Failure	0.968863	0.967745	0.1%

Table 63 - 1 system component with 2 independent failure modes and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.783835	1.783639	0.0%
Background	1.427955	1.426760	0.1%
Total	3.211790	3.210399	0.0%
Failure	2.242926	2.242653	0.0%
Non-Failure	0.968863	0.967746	0.1%

Table 64 - 1 system component with 2 independent failure modes and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.547577	1.548567	0.1%
Background	1.427955	1.426773	0.1%
Total	2.975532	2.975340	0.0%
Failure	2.006669	2.007600	0.0%
Non-Failure	0.968863	0.967740	0.1%

Table 65 - 1 system component with 2 negatively dependent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.114628	2.111377	0.2%
Background	1.427955	1.426755	0.1%
Total	3.542583	3.540351	0.1%
Failure	2.598992	2.595238	0.1%
Non-Failure	0.943591	0.945113	0.2%

Table 66 - 1 system component with 2 negatively dependent failure modes and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.740963	1.740667	0.0%
Background	1.427955	1.426760	0.1%
Total	3.168918	3.169647	0.0%
Failure	2.225327	2.224539	0.0%
Non-Failure	0.943591	0.945108	0.2%

Table 67 - 1 system component with 2 negatively dependent failure modes and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.834379	1.833352	0.1%
Background	1.427955	1.426760	0.1%
Total	3.262334	3.262331	0.0%
Failure	2.318743	2.317222	0.1%
Non-Failure	0.943591	0.945109	0.2%

Table 68 - 1 system component with 2 negatively dependent failure modes and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.647547	1.647979	0.0%
Background	1.427955	1.426760	0.1%
Total	3.075501	3.076959	0.0%
Failure	2.131910	2.131851	0.0%
Non-Failure	0.943591	0.945108	0.2%

Table 69 - 1 system component with 5 independent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.936764	6.920990	0.2%
Background	1.427955	1.426735	0.1%
Total	8.364718	8.347725	0.2%
Failure	7.716204	7.699330	0.2%
Non-Failure	0.648515	0.648395	0.0%

Table 70 - 1 system component with 5 independent failure modes and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.630434	2.626588	0.1%
Background	1.427955	1.426754	0.1%
Total	4.058388	4.053342	0.1%
Failure	3.409874	3.404951	0.1%
Non-Failure	0.648515	0.648391	0.0%

Table 71 - 1 system component with 5 independent failure modes and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	3.874881	3.868535	0.2%
Background	1.427955	1.426748	0.1%
Total	5.302836	5.295283	0.1%
Failure	4.654321	4.646891	0.2%
Non-Failure	0.648515	0.648392	0.0%

Table 72 - 1 system component with 5 independent failure modes and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.517102	1.515594	0.1%
Background	1.427955	1.426785	0.1%
Total	2.945057	2.942378	0.1%
Failure	2.296542	2.293994	0.1%
Non-Failure	0.648515	0.648385	0.0%

Table 73 - 1 system component with 5 negatively dependent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.890224	6.880921	0.1%
Background	1.427955	1.426735	0.1%
Total	8.318179	8.307686	0.1%
Failure	7.710042	7.699460	0.1%
Non-Failure	0.608137	0.608226	0.0%

Table 74 - 1 system component with 5 negatively dependent failure modes and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.797486	2.794816	0.1%
Background	1.427955	1.426753	0.1%
Total	4.225441	4.221599	0.1%
Failure	3.617304	3.613377	0.1%
Non-Failure	0.608137	0.608222	0.0%

Table 75 - 1 system component with 5 negatively dependent failure modes and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	4.004253	3.999984	0.1%
Background	1.427955	1.426744	0.1%
Total	5.432208	5.426759	0.1%
Failure	4.824072	4.818534	0.1%
Non-Failure	0.608137	0.608224	0.0%

Table 76 - 1 system component with 5 negatively dependent failure modes and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.713543	1.712076	0.1%
Background	1.427955	1.426765	0.1%
Total	3.141498	3.138871	0.1%
Failure	2.533361	2.530657	0.1%
Non-Failure	0.608137	0.608215	0.0%

Multiple System Components

Computing risk for multiple system components requires integration over a multidimensional integral. RMC-TotalRisk uses an adaptive importance sampling algorithm called VEGAS [7] [8]. More details on this method can be found in [1], [2], and [9].

The dependency between system components is defined based on the dependency between hazard functions. The dependency between system components can be set as perfectly independent, positive, or negatively dependent. There is also an option to set the dependency between system components with a user-defined correlation matrix [2]. The user must also select the joint consequence rule. The joint failure mode approach in the previous section is used to estimate the combined consequences of failure and non-failure of the system. In the current version of RMC-TotalRisk, the failure modes within a system component are statistically independent from failure modes within all other system components.

The description of the Monte Carlo routine for estimating the risk of failure with independent system components, each with independent joint failure modes and additive joint consequences, is provided in Algorithm 4. This routine can be expanded to include all risk types.

Algorithm 4 – Simulate Risk of Failure with Independent System Components Each with Independent Joint Failure Modes and Additive Joint Consequences

$R \leftarrow$ number of Monte Carlo realizations
$D \leftarrow$ number of system components
for $i \leftarrow 1$ to R **do**
    for $j \leftarrow 1$ to D **do**
        $M_j \leftarrow$ number of failure modes in component j
        $h \leftarrow F_H^{-1}(r_{i,j})$ where $r_{i,j} \sim \mathrm{U}(0,1)$ ⊳Randomly sample a hazard level for component
        for $k \leftarrow 1$ to M_j **do**
            If $r_{j,k} \leq F_{FM_{j,k}}(h)$ where $r_{j,k} \sim \mathrm{U}(0,1)$ **then**   ⊳ Randomly sample to determine failure
                $N_f \leftarrow N_f + C_{f_{j,k}}(h)$ ⊳ The system component failed so, get the consequences of failure given the hazard level
            end if
        end for
    end for
end for
Estimate the mean risk of failure $\mathbb{E}[N_f] \leftarrow N_f / R$

Figure 31 shows the FN curves for a system with two components, each with one failure mode and additive consequences. The TotalRisk results are shown as thicker transparent lines, and the Monte Carlo results are plotted as dashed lines. The VEGAS multidimensional integration approach used by RMC-TotalRisk produces nearly the same results as the Monte Carlo simulation; however, the VEGAS method can typically do so with less than 20 thousand function evaluations, rather than the 10 million required by the Monte Carlo simulation.

The verification results for joint failures are provided in Table 77 through Table 103. RMC-TotalRisk has near perfect agreement with the Monte Carlo results.

Table 77 – 2 independent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.017288	2.018285	0.0%
Background	2.855437	2.852825	0.1%
Total	4.872725	4.871109	0.0%
Failure	2.593366	2.594673	0.1%
Non-Failure	2.279360	2.276436	0.1%

Table 78 - 2 independent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.004752	2.006091	0.1%
Background	1.427719	1.426486	0.1%
Total	3.758951	3.759305	0.0%
Failure	2.575244	2.577056	0.1%
Non-Failure	1.183707	1.182249	0.1%

Table 79 - 2 independent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.013030	2.013996	0.0%
Background	2.416342	2.413949	0.1%
Total	4.487518	4.486161	0.0%
Failure	2.587785	2.589050	0.0%
Non-Failure	1.899734	1.897111	0.1%

Table 80 - 2 independent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.996474	1.998192	0.1%
Background	0.439095	0.438931	0.0%
Total	3.030384	3.032464	0.1%
Failure	2.562703	2.565090	0.1%
Non-Failure	0.467681	0.467374	0.1%

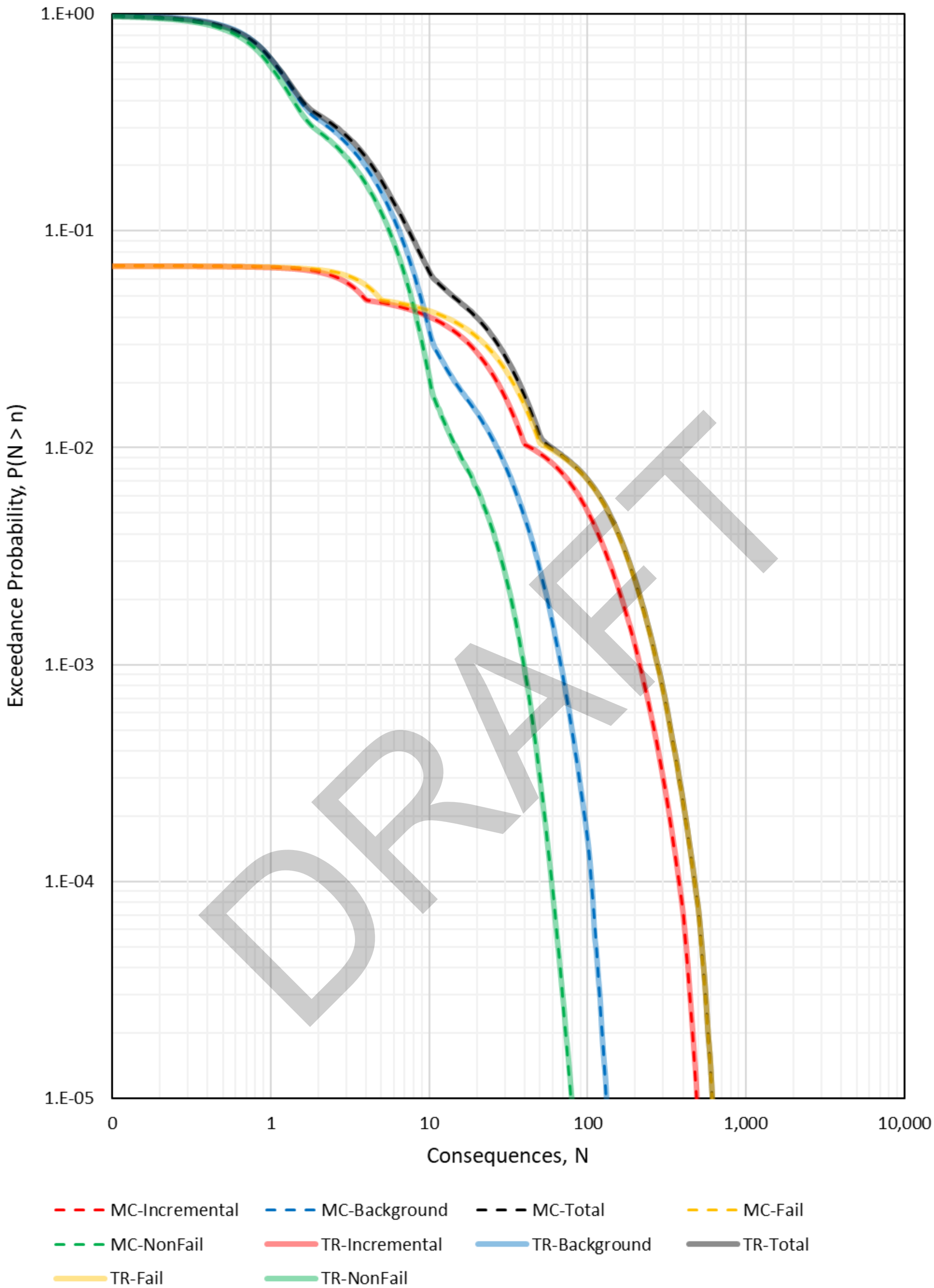

Figure 31 – The F-N plot for a system with two components, each with one failure mode and additive consequences.

Table 81 – 2 positively dependent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.017594	2.018545	0.0%
Background	2.856296	2.853464	0.1%
Total	4.873890	4.872009	0.0%
Failure	2.593824	2.595047	0.0%
Non-Failure	2.280066	2.276962	0.1%

Table 82 - 2 positively dependent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.664981	1.665942	0.1%
Background	1.428148	1.426719	0.1%
Total	3.434284	3.434103	0.0%
Failure	2.123673	2.124913	0.1%
Non-Failure	1.310610	1.309190	0.1%

Table 83 - 2 positively dependent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.782519	1.783491	0.1%
Background	1.428148	1.426752	0.1%
Total	3.551821	3.551691	0.0%
Failure	2.241211	2.242467	0.1%
Non-Failure	1.310610	1.309223	0.1%

Table 84 - 2 positively dependent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.547443	1.548412	0.1%
Background	1.428148	1.426726	0.1%
Total	3.316746	3.316582	0.0%
Failure	2.006136	2.007381	0.1%
Non-Failure	1.310610	1.309201	0.1%

Table 85 – 2 negatively dependent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.014946	2.018793	0.2%
Background	2.852113	2.853782	0.1%
Total	4.867058	4.872576	0.1%
Failure	2.589851	2.595380	0.2%
Non-Failure	2.277207	2.277196	0.0%

Table 86 - 2 negatively dependent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.014888	2.018602	0.2%
Background	1.426056	1.426874	0.1%
Total	3.735204	3.740538	0.1%
Failure	2.589764	2.595126	0.2%
Non-Failure	1.145439	1.145412	0.0%

Table 87 - 2 negatively dependent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.014946	2.018674	0.2%
Background	2.593128	2.594702	0.1%
Total	4.617460	4.622746	0.1%
Failure	2.589851	2.595231	0.2%
Non-Failure	2.027609	2.027516	0.0%

Table 88 - 2 negatively dependent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.014830	2.018619	0.2%
Background	0.258984	0.259115	0.1%
Total	2.852948	2.858527	0.2%
Failure	2.589678	2.595149	0.2%
Non-Failure	0.263270	0.263378	0.0%

Table 89 – 2 independent system components each with 2 independent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	4.265464	4.271295	0.1%
Background	2.852090	2.853490	0.0%
Total	7.117553	7.124785	0.1%
Failure	5.182284	5.189128	0.1%
Non-Failure	1.935270	1.935658	0.0%

Table 90 – 2 positively dependent system components each with 2 independent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	4.269872	4.272228	0.1%
Background	2.853251	2.853518	0.0%
Total	7.123123	7.125745	0.0%
Failure	5.187768	5.190203	0.0%
Non-Failure	1.935355	1.935543	0.0%

Table 91 – 2 negatively dependent system components each with 2 independent failure modes and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	4.278800	4.272820	0.1%
Background	2.854866	2.853630	0.0%
Total	7.133666	7.126450	0.1%
Failure	5.197708	5.190912	0.1%
Non-Failure	1.935957	1.935539	0.0%

Table 92 – 5 independent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.108527	6.122738	0.2%
Background	7.127700	7.119127	0.1%
Total	13.236227	13.241865	0.0%
Failure	7.681828	7.708889	0.4%
Non-Failure	5.554399	5.532976	0.4%

Table 93 - 5 independent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	5.568252	5.556979	0.2%
Background	1.425540	1.424691	0.1%
Total	8.189262	8.184263	0.1%
Failure	7.025408	7.023737	0.0%
Non-Failure	1.163855	1.160526	0.3%

Table 94 - 5 independent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.025967	6.029169	0.1%
Background	4.598846	4.599493	0.0%
Total	10.968579	10.970967	0.0%
Failure	7.558594	7.573999	0.2%
Non-Failure	3.409985	3.396968	0.4%

Table 95 - 5 independent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	5.098583	5.096918	0.0%
Background	0.153408	0.154650	0.8%
Total	6.633722	6.643208	0.1%
Failure	6.478531	6.486784	0.1%
Non-Failure	0.155191	0.156424	0.8%

Table 96 – 5 positively dependent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.143270	6.123966	0.3%
Background	7.140574	7.138028	0.0%
Total	13.283844	13.261993	0.2%
Failure	7.723975	7.700912	0.3%
Non-Failure	5.559869	5.561081	0.0%

Table 97 - 5 positively dependent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.633103	2.627145	0.2%
Background	1.428115	1.427658	0.0%
Total	4.774072	4.767531	0.1%
Failure	3.412262	3.405693	0.2%
Non-Failure	1.361811	1.361838	0.0%

Table 98 - 5 positively dependent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	3.878425	3.868349	0.3%
Background	1.428115	1.428004	0.0%
Total	6.019395	6.008984	0.2%
Failure	4.657584	4.646713	0.2%
Non-Failure	1.361811	1.362271	0.0%

Table 99 - 5 positively dependent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.516603	1.515976	0.0%
Background	1.426499	1.427498	0.1%
Total	3.656122	3.656317	0.0%
Failure	2.294723	2.294713	0.0%
Non-Failure	1.361399	1.361604	0.0%

Table 100 – 5 negatively dependent system components each with 1 failure mode and additive joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.095003	6.059984	0.6%
Background	7.125940	7.107293	0.3%
Total	13.220943	13.167277	0.4%
Failure	7.666340	7.611653	0.7%
Non-Failure	5.554603	5.555624	0.0%

Table 101 - 5 negatively dependent system components each with 1 failure mode and average joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	5.829741	5.793946	0.6%
Background	1.425188	1.419711	0.4%
Total	8.492369	8.435112	0.7%
Failure	7.344564	7.288417	0.8%
Non-Failure	1.147805	1.146695	0.1%

Table 102 - 5 negatively dependent system components each with 1 failure mode and maximum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	6.070426	6.033674	0.6%
Background	4.965946	4.955526	0.2%
Total	11.280812	11.232915	0.4%
Failure	7.627586	7.570491	0.7%
Non-Failure	3.653226	3.662424	0.3%

Table 103 - 5 negatively dependent system components each with 1 failure mode and minimum joint consequences

Risk Type	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	5.590722	5.568786	0.4%
Background	0.100801	0.100250	0.5%
Total	7.165477	7.126707	0.5%
Failure	7.063415	7.025254	0.5%
Non-Failure	0.102062	0.101453	0.6%

Annual Probability of Inundation

RMC-TotalRisk provides capabilities to support the National Flood Insurance Program (NFIP). The USACE uses a risk-informed approach to perform NFIP Levee System Evaluations (LSEs) to make a recommendation about whether to certify and accredit a levee system. USACE guidance for conducting LSEs to assess levee accreditation is currently outlined in Engineering and Construction Bulletin (ECB) 2019-11 [23]. All LSEs must include a computation of assurance of the 0.01 AEP which is the probability that the 0.01 AEP event will not be exceeded [23].

Computation of assurance of the 0.01 AEP for the NFIP requires an estimate of the annual probability of inundation (API), which is the probability that the leveed area will be inundated due to levee overtopping or breach in any given year. In RMC-TotalRisk, the API is computed as shown below in Equation 65. More details on API and assurance can be found in [2].

$$API = \sum_{i=1}^{n} P(x_i) \cdot P(F|x_i) + \sum_{i=x_T}^{n} P(x_i) \cdot \{1 - P(F|x_i)\} \qquad \text{Equation 64}$$

where $P(x_i)$ is the probability of the hazard level x_i; $P(F|x_i)$ is the conditional probability of failure given the hazard level x_i; x_T is the top of levee height; and $\{1 - P(F|x_i)\}$ is the probability of non-failure given the hazard level x_i, which is simply the complement of the probability of failure at a given hazard level.

Verification of the API calculation was performed using the hypothetical levee risk analysis provided by Smith [24]. An illustration of the idealized cross section of a river channel with a single levee is shown in Figure 32. As discussed in [24], the levee has a backwards erosion piping (BEP) failure mode, which is a function of the height, base, and crest width of the levee. In addition, there is an overtopping failure mode, which is only a function of the height. For simplicity, the economic consequences do not vary based on the failure mechanism and are instead a function of the water level. Figure 33 shows the failure and non-failure consequence functions for a levee height of 70 ft.

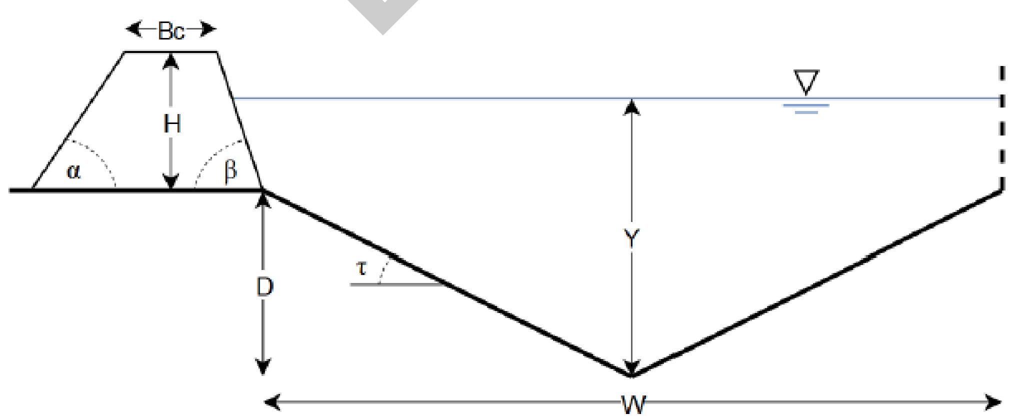

Figure 32 - Idealized cross section of a river channel with a single levee (taken from Smith [24]).

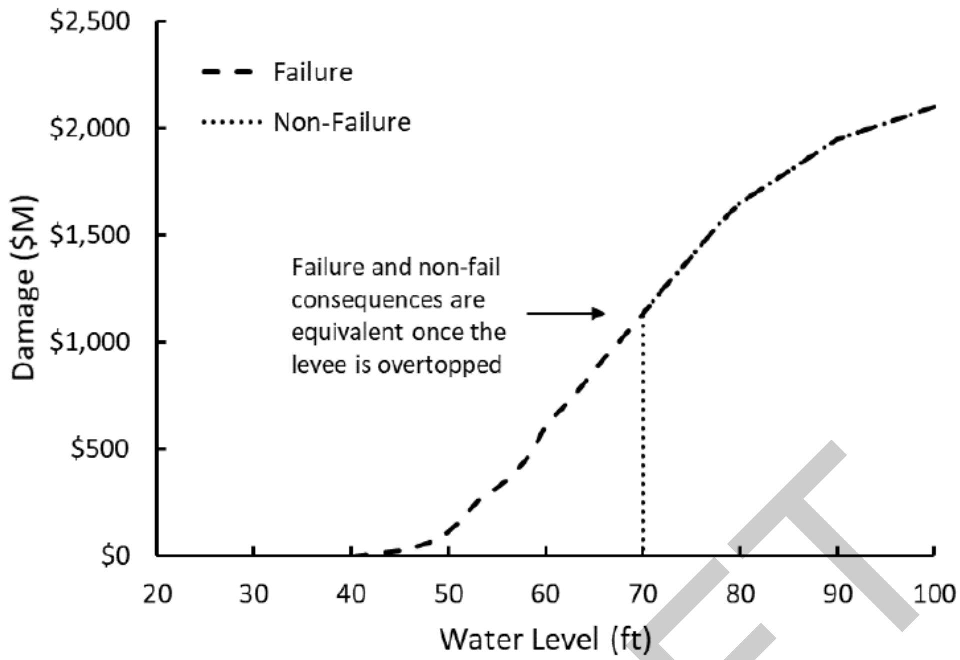

Figure 33 - Property damage in $millions as a function of water level (taken from Smith [24]).

For verification, five different levee configurations were modeled, each with a different levee height. A Monte Carlo simulation was performed using 10 million samples following Algorithm 5 below. Results are shown in Table 104. RMC-TotalRisk has very close agreement with the Monte Carlo results. The minor differences are due to Monte Carlo sampling errors.

Table 104 - Comparison of RMC-TotalRisk to Monte Carlo simulation for the Annual Probability of Inundation.

Levee Height	Monte Carlo	RMC-TotalRisk	% Difference
50	0.038197	0.038212	0.0%
55	0.018829	0.018807	0.1%
60	0.010254	0.010237	0.2%
65	0.006042	0.006043	0.0%
70	0.003820	0.003809	0.3%

Algorithm 5 – Simulate the Probability of Inundation

$R \leftarrow$ number of Monte Carlo realizations
$M \leftarrow$ number of failure modes
$ToL \leftarrow$ top of levee height
for $i \leftarrow 1$ to R **do**
    $h \leftarrow F_H^{-1}(r_i)$ where $r_i \sim \mathrm{U}(0,1)$ ⊳Randomly sample a hazard level
    if $h > ToL$ **then**
        $N_I \leftarrow N_I + 1$ ⊳ The levee was overtopped, so there is inundation
    else
        for $j \leftarrow 1$ to M **do**
            if $r_j \leq F_{FM_j}(h)$ where $r_j \sim \mathrm{U}(0,1)$ **then** ⊳ Randomly sample to determine failure
                $N_I \leftarrow N_I + 1$ ⊳ The system failed, so there is inundation
            end if
        end for
    end if
end for
Estimate the probability of inundation $API \leftarrow N_I/R$

Comparison with DAMRAE

The Dam Safety Risk Analysis Engine, *DAMRAE©*, is a software tool for performing event tree calculations and risk analysis for dam safety risk assessment studies [25]. A comparison was made between RMC-TotalRisk and *DAMRAE©* following the same hypothetical examples provided in [26]. The "Simulate Mean Risk Only" option [2] was used to estimate risk with TotalRisk.

There were four example risk analyses: 1) a single dam with a single failure mode; 2) a dam with two failure modes; 3) a dam with three failure modes; and 4) a dam with ten failure modes. The multiple failure modes were combined using the common cause adjustment (CCA) method. The incremental risk results for these comparisons are provided in Table 105 through Table 108. RMC-TotalRisk very closely matches *DAMRAE©*.

Differences in results are most likely due to differences in numerical integration techniques. Each of the *DAMRAE©* risk analyses were estimated using the Trapezoidal Rule integration with 50 bins. Whereas RMC-TotalRisk uses an Adaptive Simpson's Rule (ASR) approach, which required approximately 300 function evaluations to reach a tolerance of $1e^{-8}$ for each of the risk analyses. The differences between the software are minor and would not change any real-world investment decisions.

Table 105 – Comparison of RMC-TotalRisk to DAMRAE© for incremental risk for 1 system component with 1 failure mode.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
DAMRAE©	1.620E-07	1047	1.690E-04
RMC-TotalRisk	1.618E-07	1047	1.694E-04
% Difference	0.1%	0.0%	0.2%

Table 106 – Comparison of RMC-TotalRisk to DAMRAE© for incremental risk for 1 system component with 2 failure modes.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
DAMRAE©	3.150E-07	1010	3.180E-04
RMC-TotalRisk	3.146E-07	1010	3.177E-04
% Difference	0.1%	0.0%	0.1%

Table 107 – Comparison of RMC-TotalRisk to DAMRAE© for incremental risk for 1 system component with 3 failure modes.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
DAMRAE©	7.990E-07	977	7.810E-04
RMC-TotalRisk	7.854E-07	977	7.676E-04
% Difference	1.7%	0.0%	1.7%

Table 108 – Comparison of RMC-TotalRisk to DAMRAE© for incremental risk for 1 system component with 10 failure modes.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
DAMRAE©	1.700E-05	1180	2.000E-02
RMC-TotalRisk	1.694E-05	1180	1.998E-02
% Difference	0.4%	0.0%	0.1%

Comparison with DamonRAE

Since 2019, the USACE Risk Management Center has primarily performed dam and levee safety risk analyses with a spreadsheet tool colloquially referred to as *DamonRAE*, which is named after the developer of the tool. This spreadsheet tool uses *Palisade's @Risk©* for performing Monte Carlo simulation of event trees and other risk inputs. Within each Monte Carlo realization, incremental, background, and total risk are estimated using the Trapezoidal Rule method with 50 bins (please see the technical reference manual [2] for details on these risk types). *DamonRAE* was previously validated against *DAMRAE©* and shown to produce the same results [26].

A comparison was made between RMC-TotalRisk and *DamonRAE* using three example dams within the USACE Dam Safety portfolio[15]. The "Simulate Mean Risk Only" option [2] was used to estimate risk with TotalRisk. For each example, multiple failure modes were combined using the CCA method.

The first example dam has two failure modes: 1) spillway erosion; and 2) concentrated leak erosion along the embankment at foundation contact. Results for this example are provided in Table 109 and Table 110. RMC-TotalRisk very closely matches *DamonRAE* for this example.

The primary differences in results are most likely due to differences in numerical integration techniques. RMC-TotalRisk uses an ASR approach, which required 117 function evaluations to reach a tolerance of $1e^{-8}$ for this example risk analysis.

Table 109 – Comparison of RMC-TotalRisk to DamonRAE for incremental risk at example 'Dam 1' with 2 Failure Modes.

	Ex. Probability, α	Conditional Mean, η	Mean, $\mathbb{E}[\mathbf{N}]$
DamonRAE	8.970E-06	127.00	1.140E-03
RMC-TotalRisk	8.883E-06	127.83	1.136E-03
% Difference	1.0%	0.7%	0.4%

Table 110 – Comparison RMC-TotalRisk to DamonRAE for example 'Dam 1' with 2 Failure Modes.

Risk Type	DamonRAE, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	1.140E-03	1.136E-03	0.4%
Background	3.720E-05	3.656E-05	1.7%
Total	1.180E-03	1.172E-03	0.7%

The next example dam also has two failure modes: 1) overtopping of dam crest leading to breach; and 2) backwards erosion piping on the left abutment. Results for this example are provided in Table 111 and Table 112. RMC-TotalRisk closely matches *DamonRAE* for this example. While the percent differences are noticeable, the absolute differences are very small.

The primary differences in results are most likely due to differences in numerical integration techniques. RMC-TotalRisk required 413 function evaluations to reach a tolerance of $1e^{-8}$ for this example risk analysis. Therefore, the 50 bins used in *DamonRAE* are likely insufficient to reach an equivalent level of

[15] The names of these dams and the inputs are not provided since the risk results are for internal use only.

precision. In addition, due to the challenge of converting the spreadsheet model to TotalRisk, there could be minor differences in how uncertainty is sampled between inputs.

Table 111 – Comparison of RMC-TotalRisk to DamonRAE for incremental risk at example 'Dam 2' with 2 Failure Modes.

	Ex. Probability, α	Conditional Mean, η	Mean, $\mathbb{E}[\mathbf{N}]$
DamonRAE	2.370E-07	9,454.00	2.240E-03
RMC-TotalRisk	2.402E-07	9,452.28	2.271E-03
% Difference	1.4%	0.0%	1.4%

Table 112 – Comparison RMC-TotalRisk to DamonRAE for example 'Dam 2' with 2 Failure Modes.

Risk Type	DamonRAE, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.240E-03	2.271E-03	1.4%
Background	6.960E-01	6.772E-01	2.7%
Total	6.980E-01	6.794E-01	2.7%

The final example dam has three failure modes: 1) overtopping of dam crest leading to breach; 2) backwards erosion piping of foundation soils in the terrace section; and 3) backwards erosion piping of foundation soils in the transition section. Results for this example are provided in Table 113 and Table 114. RMC-TotalRisk does not agree well with *DamonRAE* for this example. The percent differences are greater than five percent, but the absolute differences are relatively small; certainly not large enough to change any investment decisions. Nevertheless, as mentioned previously, differences greater than five percent required additional analysis and justification.

Table 113 – Comparison of RMC-TotalRisk to DamonRAE for incremental risk at example 'Dam 3' with 3 Failure Modes.

	Ex. Probability, α	Conditional Mean, η	Mean, $\mathbb{E}[\mathbf{N}]$
DamonRAE	3.600E-04	8.00	2.840E-03
RMC-TotalRisk	3.308E-04	7.86	2.600E-03
% Difference	8.1%	1.8%	8.4%

Table 114 – Comparison RMC-TotalRisk to DamonRAE for example 'Dam 3' with 3 Failure Modes.

Risk Type	DamonRAE, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Incremental	2.840E-03	2.600E-03	8.4%
Background	4.550E-02	3.746E-02	17.7%
Total	4.840E-02	4.006E-02	17.2%

As an additional verification test, a Monte Carlo simulation with 10 million samples (see Algorithm 1) was used to calculate risk for example Dam #3. The results from this comparison are provided in Table 115 and Table 116. RMC-TotalRisk very closely matches the Monte Carlo results. Therefore, it can be concluded that the primary differences between TotalRisk and *DamonRAE* for example Dam #3 are most likely due to differences in numerical integration techniques. RMC-TotalRisk required 209 function evaluations to reach a tolerance of $1e^{-8}$ for this example risk analysis. Therefore, the 50 bins used in *DamonRAE* are likely insufficient to reach an equivalent level of precision. In addition, due to the challenge of converting the spreadsheet model to TotalRisk, there could be minor differences in how uncertainty is sampled between inputs.

Table 115 – Comparison of RMC-TotalRisk to Monte Carlo for incremental risk at example 'Dam 3' with 3 Failure Modes.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
Monte Carlo	3.314E-04	7.86	2.604E-03
RMC-TotalRisk	3.308E-04	7.86	2.600E-03
% Difference	0.2%	0.0%	0.1%

Table 116 – Comparison RMC-TotalRisk to Monte Carlo for example 'Dam 3' with 3 Failure Modes.

Risk Type	**Monte Carlo, $\mathbb{E}[\mathbf{N}]$**	**RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$**	**% Difference**
Incremental	2.60E-03	2.600E-03	0.1%
Background	3.74E-02	3.746E-02	0.2%
Total	4.00E-02	4.006E-02	0.1%

Comparison with LST 2.0

The USACE Levee Screening Tool (LST) 2.0 is used to perform screening-level quantitative risk analysis for the Levee Safety program [27]. The LST has been used to calculate screening-level risk for several thousand levee segments and is considered a state-of-the-art risk analysis and portfolio management tool. The LST 2.0 uses the computing failure mode approach for multiple failure modes [2].

A comparison between RMC-TotalRisk and LST 2.0 was made using a real levee segment from the USACE Levee Safety portfolio[16]. The levee has the following failure modes:

- Backwards erosion piping in the foundation
- Backwards erosion piping through the embankment
- Backwards erosion piping of the floodwall foundation
- Embankment erosion
- Embankment stability
- Floodwall instability
- Inoperability of closures
- Overtopping

The LST 2.0 combines the multiple failure modes using the competing failure modes method. Results for this comparison are provided in Table 117 and Table 118. The “Simulate Mean Risk Only” option [2] was used to estimate risk with TotalRisk. RMC-TotalRisk very closely matches the results from the LST. The minor differences are primarily due to differences in numerical integration approaches.

The LST computes risk using the Trapezoidal Rule with approximately 100 integration bins. Whereas, TotalRisk uses the ASR method, which required 253 function evaluations to reach a tolerance of $1e^{-8}$ for this example. In addition, RMC-TotalRisk pre-processes the cumulative incident functions (CIFs) needed for competing failure modes using 200 uniformly spaced hazard levels. The differences between results are very small and would not lead to different investment decisions.

Table 117 – Comparison of RMC-TotalRisk to LST for incremental risk for levee segment with 8 Failure Modes.

	Ex. Probability, α	**Conditional Mean, η**	**Mean, $\mathbb{E}[\mathbf{N}]$**
LST	1.378E-03	68.39	9.426E-02
RMC-TotalRisk	1.398E-03	66.84	9.345E-02
% Difference	1.4%	2.3%	0.9%

Table 118 – Comparison RMC-TotalRisk to LST for levee segment with 8 Failure Modes.

Risk Type	**LST, $\mathbb{E}[\mathbf{N}]$**	**RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$**	**% Difference**
Incremental	9.426E-02	9.345E-02	0.9%
Background	3.042E-02	3.026E-02	0.5%
Total	1.247E-01	1.237E-01	0.8%

[16] The names of the levee segment and the inputs are not provided since the risk results are for internal use only.

Comparison with HEC-FDA

The flood damage reduction analysis software, *HEC-FDA* [15], has been the primary software used for assessing expected annual flood damages in USACE since 1994. The underlying quantitative risk analysis framework employed by *HEC-FDA* is documented in [28], and it is like RMC-TotalRisk. Both tools divide the risk analysis inputs into four primary components: the hazard, transform, response, and consequence functions.

However, there are some significant differences between the tools. RMC-TotalRisk is part of a larger risk analysis software suite [29] which can import results from state-of-the-art flood hazard and consequence tools, such as *HEC-SSP*[17], *RMC-BestFit* [30], and *LifeSim* [31]. System response functions can be derived from event tree analysis, or from one of the RMC toolboxes[18]. Most significantly, RMC-TotalRisk can perform risk analysis for complex systems that have multiple dependent system components, each with multiple failure modes, and different joint consequence rules.

HEC-FDA does perform system risk, but the calculations assume that each system component is independent, and that inundation areas do not overlap, and thus joint consequences are additive.

Consider a system with two levee segments, where the consequences of failure from each segment are additive. Computing risk for multiple system components requires integration over a multidimensional integral. Following the general risk formula provided in [2], the system risk becomes a two-dimensional integral:

$$\mathbb{E}[C]_{\Omega} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \{C_X(x) + C_Y(y)\} \cdot f_{XY}\big(C_X(x), C_Y(y)\big) \cdot dx \cdot dy \qquad \textit{Equation 65}$$

where x is the hazard level for system component X; $C_X(x)$ determines the consequences for the hazard level x; y is the hazard level for system component Y; $C_Y(y)$ determines the consequences for the hazard level y; and $f_{XY}\big(C_X(x), C_Y(y)\big)$ is the joint PDF of the combined system consequences occurring.

The complex integral in Equation 65 can be simplified. For any two random variables X and Y, the expected value of the sum of those variables will be equal to the sum of their expected values:

$$\mathbb{E}[C]_{\Omega} = \mathbb{E}[C_X + C_Y] = \mathbb{E}[C_X] + \mathbb{E}[C_Y] \qquad \textit{Equation 66}$$

Considering this, *HEC-FDA* computes system risk by estimating the expected damages at each individual levee segment, then simply summing the expected damages at each segment to get the overall system risk.

[17] https://www.hec.usace.army.mil/software/hec-ssp/
[18] https://www.rmc.usace.army.mil/Software/

The expected value at each levee segment is computed using simple Monte Carlo integration with a convergence rule based on the following criteria:

$$\frac{z_{1-\alpha/2} \cdot \sigma_C}{\mathbb{E}[C] \cdot \sqrt{N}} \leq \varepsilon \quad \textit{Equation 67}$$

where z is the standard Normal deviate for a confidence level $\alpha = 0.05$; $\mathbb{E}[C]$ is the expected consequence from N samples; $\varepsilon = 0.01$ is the relative tolerance; and $N \leq 500{,}000$.

RMC-TotalRisk allows for dependency between system components, and the user must also select the joint consequence rule, which can be additive, average, maximum, or minimum. This means that RMC-TotalRisk must use a more robust and efficient multidimensional integration approach. To avoid computational limitations, RMC-TotalRisk uses an adaptive importance sampling algorithm called VEGAS [7] [8]. More details on this method can be found in [1], [2], and [9]. This approach was verified in the previous **Multidimension Integration** and **Multiple System Components** sections of this report.

A comparison between RMC-TotalRisk and *HEC-FDA* was performed using the *'Beargrass Creek'* example project provided in the *HEC-FDA* user guide [15] and training course[19]. The Beargrass Creek study used for that course consists of two highly urbanized damage reaches on the South Fork of the Beargrass Creek. In this example project, there are two levee reaches per system, with four alternatives for reducing flood risk:

- Without project condition
- Plan 1: Detention and channel improvement
- Plan 2: Floodwall only
- Plan 3: Detention, channel improvement, and floodwall.

For this comparison, results were evaluated at the current year (2021) and a future year (2030). Each analysis in TotalRisk was performed using the "Simulate Mean Risk Only" option [2]. Results of the comparison are provided in Table 119 and Table 120. For the most part the differences are relatively minor. However, there are two alternatives that have differences greater than five percent, which required additional analysis and justification.

Table 119 - Comparison of RMC-TotalRisk to HEC-FDA for Bear Creek 2021 EAD.

Alternative	HEC-FDA, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Without	952.48	916.41	3.8%
Plan 1	588.53	571.04	3.0%
Plan 2	551.85	512.91	7.1%
Plan 3	173.37	156.59	9.7%

[19] https://www.hec.usace.army.mil/confluence/fdadocs/fdatutorials/flood-damage-assessment-course-content

Table 120 - Comparison of RMC-TotalRisk to HEC-FDA for Bear Creek 2030 EAD.

Alternative	HEC-FDA, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Without	1,097.25	1,111.27	1.3%
Plan 1	610.62	613.10	0.4%
Plan 2	723.74	730.40	0.9%
Plan 3	181.27	183.84	1.4%

An additional comparison was performed using Monte Carlo simulation with 10 million samples, and results are provided in Table 121 and Table 122. RMC-TotalRisk has near perfect agreement with these Monte Carlo results.

Table 121 - Comparison of RMC-TotalRisk to Monte Carlo simulation for Bear Creek 2021 EAD.

Alternative	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Without	916.75	916.41	0.0%
Plan 1	571.15	571.04	0.0%
Plan 2	513.35	512.91	0.1%
Plan 3	156.71	156.59	0.1%

Table 122 - Comparison of RMC-TotalRisk to Monte Carlo simulation for Bear Creek 2030 EAD.

Alternative	Monte Carlo, $\mathbb{E}[\mathbf{N}]$	RMC-TotalRisk, $\mathbb{E}[\mathbf{N}]$	% Difference
Without	1,111.66	1,111.27	0.0%
Plan 1	613.16	613.10	0.0%
Plan 2	730.83	730.40	0.1%
Plan 3	183.98	183.84	0.1%

The Monte Carlo simulation results would tend to indicate that the primary differences between RMC-TotalRisk and *HEC-FDA* are most likely due to differences in numerical integration techniques. Additional sources of differences between software programs could be some combination of the following:

- Minor differences in how uncertainty is quantified in the nonparametric hazard functions. In the Nonparametric Hazard Function section of this report, the differences between RMC-TotalRisk were shown to be minor.
- The damage functions are automatically computed and aggregated in *HEC-FDA*. For this example, those curves were extracted and combined as a composite consequence function in RMC-TotalRisk. There could be minor differences in how this input is treated between the software programs.
- There could be differences in linear interpolation transforms used between software. The nonparametric hazard functions in RMC-TotalRisk were set to use a logarithm transform for flows and a Normal z-variate transform for probabilities. It is unclear what is used in *HEC-FDA*. Likewise, it is unclear if any transforms are used to improve interpolation on any of the other input functions in *HEC-FDA*.

Conclusions

As demonstrated in this report, the computational methods used in RMC-TotalRisk have been verified. The general numerical methods have been verified against known theoretical and analytical solutions, and widely used, documented, and verified R packages. The input functions were verified with *R* and *Palisade's @Risk*. The risk analysis was verified using Monte Carlo simulation for a variety of complex systems. In all cases, RMC-TotalRisk produced valid results.

Risk analysis results from RMC-TotalRisk were also compared with other risk analysis software and tools, such as *DAMRAE©*, *DamonRAE*, *LST*, and *HEC-FDA*. In most cases, TotalRisk produced similar results that differed by less than five percent difference. Most of the differences in precision between these various tools were inconsequential and would not lead to a different risk-informed investment decision.

References

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes: The Art of Scientific Computing., 3rd ed., Cambridge, UK: Cambridge University Press, 2017.

[2] C. H. Smith, W. L. Fields and D. A. Margo, "(DRAFT) RMC-TR-2022-XX Quantitative Risk Analysis with the RMC-TotalRisk Software," U.S. Army Corps of Engineers, Lakewood, CO, 2022.

[3] C. H. Smith, "RMC-TR-2020-02 Verification of the Bayesian Estimation and Fitting Software (RMC-BestFit)," U.S. Army Corps of Engineers, Lakewood, CO, 2020.

[4] J. R. M. Hosking and J. R. Wallis, Regional Frequency Analysis: An Approach Based on L-Moments, Cambridge, UK: Cambridge University Press, 1997.

[5] P. J. Davis and P. Rabinowitz, Methods of Numerical Integration, 2nd ed., Mineola, New York: Dover Publications, Inc., 2007.

[6] Y. Y. Haimes, Risk Modeling, Assessment, and Management, Hoboken, NJ: John Wiley & Sons, Inc., 2004.

[7] G. Lepage, "A New Algorithm for Adaptive Multidimensional Integration," *Journal of Computational Physics,* vol. 27, no. 1, pp. 192-203, 1978.

[8] G. Lepage, "VEGAS: Ana Adaptive Multidimensional Integration Program," Cornell University, 1980.

[9] A. Ciric, A Guide to Monte Carlo & Quantum Monte Carlo Methods, Createspace Independent Publishing Platform, 2016.

[10] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 2nd ed., Melbourne: OTexts.com/fpp2, 2018.

[11] B. Efron and R. J. Tibshirani, An Introduction to the Bootstrap, Boco Raton: CRC Press LLC, 1998.

[12] B. Efron and T. Hastie, Computer Age Statistical Inference: Algorithms, Evidence and Data Science, New York, NY: Cmbridge University Press, 2016.

[13] A. R. Rao and K. H. Hamed, Flood Frequency Analysis, Boca Raton, FL: CRC Press LLC, 2000.

[14] K. Krishnamoorthy, Handbook of Statistical Distributions with Applications, Boca Raton, FL: CRC Press, 2016.

[15] U.S. Army Corps of Engineers, "HEC-FDA Flood Damage Reduction Analysis User's Manual," U.S. Army Corps of Engineers, Davis, CA, 2016.

[16] U.S. Army Corps of Engineers, "ETL 1110-2-537 Uncertainty Estimates for Nonanalytical Frequency Curves," U.S. Army Corps of Engineers, Washington, DC, 1997.

[17] A. Stuart and K. Ord, Kendall's Advanced Theory of Statistics: Volume 1 Distribution Theory, 6th ed., vol. 1, New Delhi: Wiley, 1994.

[18] J. R. Stedinger, "Confidence Intervals For Design Events," *Journal of Hydraulic Engineering,* pp. 13-27, 1982.

[19] D. N. D. Hartford and G. B. Baecher, Risk and Uncertainty in Dam Safety, London, England: Thomas Telford, 2004.

[20] I. M. Sobol', "Sensitivity analysis for non-linear mathematical models," *Mathematical Modeling & Computational Experiment,* vol. 1, no. 4, pp. 407-414, 1993.

[21] S. Fruhwirth-Schnatter, G. Celeux and C. P. Robert, Handbook of Mixture Analysis, Boca Raton, FL: CRC Press, 2020.

[22] P. I. Hill, D. S. Bowles, R. J. Nathan and R. Herveynen, "On the art of event tree modeling for portfolio risk analyses," in *Australian National Committee on Large Dams (ANCOLD)*, Auckland, NZ, 2001.

[23] U.S. Army Corps of Engineers, "ECB 2019-11 Transition Guidance for Levee System Evaluations for the National Flood Insurance Program (NFIP)," U.S. Army Corps of Engineers, Washington, DC, 2019.

[24] C. H. Smith, "Improving the Economic Evaluation of Flood Risk Management Studies," Colorado School of Mines, Golden, CO, 2022.

[25] RAC Engineers and Economists, LLC, "Technical Guide for DAMRAE version 4.0.0.0," U.S. Army Corps of Engineers, Lakewood, CO, 2017.

[26] D. P. Amlung, "Spreadsheet Validation: DamonRAE," U.S. Army Corps of Engineers, Lakewood, CO, 2019.

[27] U.S. Army Corps of Engineers, "USACE Levee Screening Tool application guide and technical reference manual," U.S. Army Corps of Engineers, Lakewood, CO, 2015.

[28] U.S. Army Corps of Engineers, "EM 1110-2-1619 Risk-Based Analysis for Flood Damage Reduction Studies," Washington, D.C., 1996.

[29] C. H. Smith, W. L. Fields and N. J. Snorteland, "A New Suite of Risk Analysis Software for Dam and Levee Safety," *The Journal of Dam Safety,* pp. 36-46, 2021.

[30] C. H. Smith and M. Doughty, "RMC-TR-2020-03 RMC-BestFit Quick Start Guide," U.S. Army Corps of Engineers, Lakewood, CO, 2020.

[31] U.S. Army Corps of Engineers, "HEC-LifeSim Life Loss Estimation User's Manual," Davis, CA, 2018.