QUANTIL

Probabilistic risk and project management

TECHNICAL VALIDATION NOTE

Validation of the calculation engine

Verification of the AACE-oriented Beta-PERT Monte Carlo simulation against an independent reimplementation

Subject	Calculation engine of the probabilistic cost and risk calculation (Monte Carlo engine)
Component	modMonteCarlo_v3 and modQuantilRandom (engine v3.1): Quantil QRCIM (Quantitative Risk-Cost Integration Model)
Method	Beta-PERT Monte Carlo on the MRG32k3a generator, with Cholesky-correlated cost elements (Gaussian copula), a discrete-risk overlay (Bernoulli + independent PERT impact) and a schedule-risk overlay (delay x daily rate)
Verification method	Independent reimplementation in Python (NumPy + SciPy), mirrored line by line
Nature of the verification	Internal validation by the vendor, carried out along an independent implementation route. No external audit party was involved; all tests are documented so that a third party can repeat them (appendices B and C).
Number of tests	14 component and chain tests, plus an end-to-end test on the full real-world register and a control test on the correlation effect
Outcome	All tests passed within predefined tolerances
Reproducibility	Fixed seeds and explicit iteration counts; see appendix C
Status	Fit for use: suitable for substantiating risk reserves
Prepared by	Quantil

Reading guide

This note is structured in layers so that it serves the full audience: executives and clients, risk managers and cost engineers, and quants and programmers who want to be able to assess the calculation engine independently.

Main text (§1 through §6). Accessible, without formulas. Describes the purpose, method, results and conclusion at the level a steering committee or investment committee needs in order to rely on the outcomes.

Appendices (A through E). Technical and complete. The exact algorithms with formulas (A), the full test setup with tolerances and measured values (B), reproducibility and environment (C), the modeling choices and their limits (D), and the literature and standards on which the method rests (E).

Table of contents

Click a line to jump to it. In Word, you refresh the table of contents with F9 (right-click, Update Field).

1. Summary 3
2. Purpose and scope 3
3. Verification method 4
4. Results 4
5. Verification at the scale of large projects 4
5a. Verification on the full real-world register 5
5b. Verification of the random number generator 6
6. Conclusion 6
Appendix A. Algorithms and formulas 7
Appendix B. Test setup and measured values 10
Appendix C. Reproducibility and environment 11
Appendix D. Model choices and their limits 13
Appendix E. Literature and standards 15

1. Summary

The calculation engine runs a Beta-PERT Monte Carlo simulation in accordance with the AACE Total Cost Management framework. Each cost element in the cost breakdown structure (CBS) receives a three-point estimate (minimum, most likely, maximum); the interdependence between those elements is captured in a correlation matrix. On top of that, an overlay of discrete risks (with a probability of occurrence and their own three-point estimate for the impact) and a schedule-risk overlay (delay times daily rate) are applied. From the simulated distribution of total project cost follow the percentiles (P10 through P95), the expected value, the required contingency (probabilistic contingency) and the management reserve.

The verification was carried out by decoupling the complete calculation logic from the tool and rebuilding it independently in Python with NumPy and SciPy. The outcomes of that independent reference were compared with the calculation engine in fourteen component and chain tests against predefined numerical tolerances, supplemented by an end-to-end test on the full real-world register and a control test on the correlation effect. All tests pass. The simulation is mathematically correctly implemented; the outcomes follow solely from the cost and risk register as entered and the chosen parameters.

Key finding for the subject-matter reader: the interdependence between cost elements is modeled through a Gaussian copula based on a Cholesky decomposition of the correlation matrix. The user specifies the full correlation structure, including clusters of related civil and tunnel activities. The individual uncertainty distributions remain unaffected: the correlation only governs the co-occurrence, not the shape of the individual Beta-PERT distributions. The net effect is that related cost overruns occur together more often, which widens the tail of the portfolio distribution and increases the required contingency. The modeling choices and their limits are given in appendix D.

2. Purpose and scope

The purpose is to establish whether the calculation engine works mathematically correctly: whether the sampling, the simulation, the percentile calculation and the correlation structure produce the intended statistical outcomes. The verification focuses on the correctness of the calculation, not on the substantive quality of a specific register.

Within scope are: the derivation of the Beta-PERT shape parameters; the inverse-beta sampling; the normal distribution function the tool uses without external add-ins; the Cholesky-correlated sampling of the cost elements via a Gaussian copula; the discrete-risk overlay (Bernoulli plus independent impact) and the schedule-risk overlay; the percentile, expected value, standard deviation and skewness calculation; the stability across random sequences; and the behavior in the edge cases.

Out of scope are: the correctness of a register entered by the user, the choice of probabilities and three-point estimates, and organizational aspects of use. The reliability of a forecast remains only as good as the quality of the input.

Nature and independence of the verification. This note describes an internal validation: it was prepared by the vendor of the tool and not by an external audit party. What is independent is the route. The calculation logic was rebuilt line by line in a second environment, with different libraries, a different random number generator and, where possible, an analytical reference instead of a second simulation. An error in the original implementation would almost certainly manifest itself differently along that route and would therefore stand out. That is an established method for validating a numerical engine, but it is not a substitute for an audit by a third party. All tests are therefore documented so that an external party can repeat them: appendix B gives the criteria, the thresholds and the measured values, appendix C the seeds, the iteration counts and the environment. Anyone who wants to see the outcomes confirmed independently can have an external verification carried out on that basis without having to start over.

3. Verification method

The calculation engine was read line by line and the mathematical logic was rebuilt independently in a second environment (Python with NumPy and SciPy). This approach, reimplementation along an independent route and comparison of the outcomes, is the standard method for validating a numerical engine: an error in the original implementation would almost certainly manifest itself differently than in the independent rebuild.

For each component, a testable criterion with a numerical tolerance was defined in advance. Comparison was made against (a) a known analytical formula, (b) an established reference library (SciPy), or (c) a theoretically expected statistical property. For the statistical tests, large samples (up to 200,000 draws) were used in order to establish the properties reliably. The full setup, including seeds and measured values, is given in appendices B and C.

4. Results

The table below summarizes the tests. Each component reproduces the independent reference within the predefined tolerance. The exact measured values and thresholds are given in appendix B.

Tested component	Result	Verdict
Normal distribution function (Abramowitz-Stegun)	max. deviation < 8×10^{-8} vs. SciPy	Pass
Beta-PERT shape parameters	exactly equal to canonical PERT formula	Pass
Beta-PERT mode	mode reproduces the most likely value	Pass
Inverse-beta sampling (bisection)	bounded in [min, max] and monotonic	Pass
Gamma sampling (Marsaglia-Tsang) → Beta-PERT	mean 0.047% vs. analytical	Pass
Cholesky decomposition ($L\times L^T = C$)	max. deviation 2×10^{-16}	Pass
Copula: marginal distribution	mean 0.00 and sd 1.00 preserved	Pass
Copula: correlation matrix reproduction	measured correlation = input (within 0.004)	Pass
Expected value (full chain)	deviation 0.0002% vs. analytical EMV	Pass
Cost percentiles	monotonically increasing and above the base estimate	Pass
Convergence	P85 differs 0.05% between two seeds	Pass
Edge cases (P=0, P=100, no spread)	behavior exactly as analytically expected	Pass
Scale test (100 risks, 250 million, 50k iter)	deviation 0.001% vs. analytical EMV	Pass
Generator MRG32k3a (known-answer and statistics)	reference value reproduced exactly; zero deviations over 1 million draws	Pass

Note on accuracy. Deviations on the order of 10^{-8} and smaller concern the helper function for the normal distribution and the matrix arithmetic and are negligible for any practical application. The statistical deviations (expected value, convergence, correlation) lie well within what is to be expected at the sample size used and decrease further with more iterations.

5. Verification at the scale of large projects

To establish that the calculation engine remains reliable at the size of large construction, infrastructure and capital investment projects, a stress test was carried out: twice the configured limit of cost elements (100), a base estimate of USD 250 million and 50,000 iterations. The arithmetic correctness proved not to depend on the size of the amount.

Number of elements	100 (twice the configured limit)
Base estimate	USD 250,000,000
Number of iterations	50.000
Simulated mean	USD 254.0 million
Analytical expected value	USD 254.0 million
Relative deviation	0,001%
P85 outcome	USD 256.3 million

The simulated mean deviated less than one thousandth of a percent from the analytical expected value and the percentiles were correctly ordered. This substantiates that the outcomes can be used to support a risk reserve at a confidence level such as P85, the type of substantiation a steering committee or investment committee requires. Substantively, the method is in line with what is common practice in the industry: Beta-PERT Monte Carlo within the AACE framework is the established approach to cost estimating and risk reserves for capital-intensive projects (see appendix E).

5a. Verification on the full real-world register

In addition to the component and scale tests, the complete chain was tested on the full register as entered in the tool: twelve interrelated cost elements with a 12x12 correlation matrix, eight discrete risks and eight schedule activities. This tests the calculation engine not on a synthetic setup, but on the type of register that is actually run in practice, from input to percentile outcome in a single pass. The calculation engine was run with a fixed seed (42) and 10,000 iterations, and compared with an independent reimplementation along a different route: vectorized in NumPy, with scipy.stats.beta for the inverse and a different random number generator (PCG64), over 200,000 iterations.

The independent reference reproduced the outcomes of the calculation engine across the reported percentiles (P50 through P90) and the expected value within 0.06%, well within the Monte Carlo noise at the iteration count used. The core check, whether the simulated mean coincides with the analytical expected value, passes at portfolio level within 0.0002%.

Metric	Calculation engine, seed 42	Independent reference	Deviation
Base estimate	**USD 416.00 million**	**USD 416.00 million**	**0.00%**
P50 (median)	**USD 476.11 million**	**USD 476.05 million**	**0.01%**
P80 (budget level)	**USD 507.67 million**	**USD 507.50 million**	**0.03%**
P85 (management reserve)	**USD 515.26 million**	**USD 514.93 million**	**0.06%**
P90 (contingency cap)	**USD 524.58 million**	**USD 524.43 million**	**0.03%**
Expected value	**USD 478.16 million**	**USD 477.97 million**	**0.04%**

The percentiles are correctly ordered and lie above the base estimate. This confirms at the level of a full real-world register what the component and scale tests demonstrate at the component level: the calculation engine adds no bias of its own and the outcome follows solely from the register and the chosen parameters. The chain from input to risk reserve at P85 has thus been tested as a whole, not just the individual components.

5b. Verification of the random number generator

All draws in the calculation engine come from MRG32k3a (L'Ecuyer, 1999), the generator also used in @RISK, Crystal Ball, SAS and MATLAB. The period is approximately 2^191 and the generator passes the full TestU01 battery. Because the generator is fully specified, an outcome with a fixed seed can be recomputed outside Excel; that is not possible with VBA's built-in Rnd, which has a period of 16.8 million draws and is undocumented.

The implementation was tested on three points. First, with a known-answer test: from L'Ecuyer's default state, the generator yields the published value 0.127011122046577, and the first six output values agree with the published sequence to 10^-12. Second, on arithmetic exactness: over one million draws, the output of the Double implementation in VBA was compared with exact integer arithmetic, with zero deviations; the largest intermediate product remains 6.03 x 10^15 and thus below the exactness limit of 9.007 x 10^15 of a Double. Third, on the statistical properties: mean 0.5008 against 0.5 analytical, standard deviation 0.288156 against 0.288675 analytical, autocorrelation at lag 1 equal to +0.0032, and a chi-square of 16.61 over twenty classes against a critical value of 30.14. Together these three points form test 14 in appendix B. The known-answer test is also included as check V-5 in the verification suite in the tool itself, so that the user can repeat it at any time.

6. Conclusion

On all tested points (the fourteen component and chain tests, the end-to-end test on the full real-world register and the control test on the correlation effect) the calculation engine reproduces the independent reference within the predefined tolerances, including at the scale of large projects and in the edge cases. The simulation is mathematically correctly implemented and the outcomes follow solely from the register as entered and the chosen parameters. Subject to the modeling choices and boundary conditions in appendix D, the calculation engine is suitable for substantiating risk reserves for construction, infrastructure and capital investment projects.

The tests are reproducible with the seeds and parameters listed in appendix C; running them yields statistically equivalent outcomes.

Appendix A. Algorithms and formulas

This appendix gives the exact mathematical content of the calculation engine, as implemented in the module modMonteCarlo_v3 (engine v3.1). The notation follows the implementation.

A.1 Beta-PERT distribution and shape parameters

For each cost element or risk with a three-point estimate (a = minimum, m = most likely, b = maximum), a Beta distribution on the interval [a, b] is used. The tool derives the shape parameters directly from the three points and the shape factor lambda (default lambda = 4):

```
alpha = 1 + lambda * (m – a) / (b – a)
beta  = 1 + lambda * (b – m) / (b – a)
mu    = (a + lambda*m + b) / (lambda + 2)          (PERT mean)

with lambda = 4 (default); a = min, m = most likely, b = max
lower bound alpha, beta >= 1 ; if b = a, m is used directly
```

This standard form is algebraically equal to the classical derivation via the PERT mean mu, but contains no division that can tend to zero for a symmetric three-point estimate; that is why it is used in the calculation engine. With lambda = 4, the expected value of the distribution coincides with the well-known PERT mean (a + 4m + b) / 6 and the mode coincides with m. Both properties have been tested (appendix B, tests 2 and 3). If b = a (no spread), m is used directly; this degenerate case has been tested separately.

A.2 Sampling from the Beta-PERT distribution

The calculation engine has two sampling routes, both validated:

Inverse transform (correlated cost elements). For a uniform value u from the copula (A.3), the inverse of the regularized incomplete Beta function is determined via at most 30 bisection steps, and rescaled to [a, b]. The loop stops earlier as soon as the interval is narrower than 1e-8, which at the amounts used is well below one cent:

```
lo = 0 ; hi = 1
repeat max. 30x:  mid = (lo+hi)/2
                  if I(mid; alpha, beta) < u  then lo = mid  else hi = mid
                  stop as soon as hi – lo < 1e-8
x = a + (b – a) * mid

I = regularized incomplete beta (modified Lentz continued fraction in the tool, at most 60 terms)
  = scipy.special.betainc in the reference
```

Gamma quotient (discrete risks and schedule). For the overlays, the tool draws two Gamma variates via the Marsaglia-Tsang algorithm and forms a Beta draw from them:

```
Ga ~ Gamma(alpha) ; Gb ~ Gamma(beta)   (Marsaglia-Tsang)
Beta = Ga / (Ga + Gb)
x    = a + (b – a) * Beta
```

Because the inverse is monotonically increasing in u and its image lies in [0,1], x is guaranteed to be bounded in [a, b] and monotonic in u. Both routes have been confirmed numerically (appendix B, tests 4 and 5).

A.3 Correlation: Cholesky-correlated Gaussian copula

The interdependence between the cost elements is captured in a correlation matrix C. The tool decomposes C into a lower triangular matrix L (Cholesky-Banachiewicz) such that $C = L \times L^{T}$. In each iteration, independent standard normal draws are transformed into correlated normals, which are converted to uniforms via the normal distribution function and drive the inverse-Beta sampling:

```
C       = L * L^T                       (Cholesky, lower triangular)
z_ind   = [Box-Muller(), ...]           (n independent normals)
z_corr  = L * z_ind                     (correlated normals)
u_j     = Phi(z_corr[j])                (Phi = normal CDF, A.4)
x_j     = InvBetaPERT(u_j; a_j,m_j,b_j) (per element, A.2)
```

Properties relevant to interpretation. First, the marginal distribution of each z_corr remains standard normal (mean 0, sd 1), so that the individual cost uncertainty remains unbiased. Second, the empirical correlation of the transformed draws reproduces the specified matrix C. The correlation therefore governs only the co-occurrence; the magnitude per element is subsequently drawn from its own Beta-PERT distribution. Both points have been tested explicitly (appendix B, tests 7 and 8) and are interpreted substantively in appendix D.

Check on inconsistent input. A matrix entered by the user is not always mathematically valid (positive definite). The calculation engine does not work around this: if the diagonal term drops below zero, the simulation stops and the tool reports at which cost item the matrix becomes impossible. Two smaller checks precede this: values outside the range [-1, +1] are clipped to the bound, and if the lower triangle contradicts the upper triangle, the tool reports for which pair of items this occurs before the mirroring is applied. In this way an invalid correlation structure never silently produces an outcome that does not correspond to the specified correlation. The real-world matrix tested is positive definite (smallest eigenvalue 0.35) and requires no repair.

A.4 Helper function for the normal distribution

The tool uses no external add-ins. The normal distribution function (CDF) is implemented via the Abramowitz-Stegun approximation 26.2.17 (five-term polynomial in t = 1/(1 + 0.2316419×|z|)). The standard normal draws come from the Box-Muller transformation, fed by the MRG32k3a generator (L'Ecuyer, 1999). Box-Muller yields two normals per pair of draws; the calculation engine stores the second and uses it on the next call, so that no draws are discarded. The CDF has been tested against SciPy as reference (appendix B, test 1).

A.5 Discrete-risk and schedule overlay

In addition to the cost elements, the calculation engine processes two overlays, independently in each iteration:

```
DISCRETE RISK (per risk with probability p):
  occurs  <=>  rnd() < p
  if so: impact x ~ BetaPERT(a,m,b)  ->  totalCost += x

SCHEDULE (per activity, planned duration D, daily rate r):
  duration t ~ BetaPERT(a,m,b)
  if t > D:  totalCost += (t – D) * r
```

The probability of occurrence and the impact magnitude are drawn independently, so that a risk retains its full impact spread. The schedule contribution only counts when the planned duration is exceeded. Both overlays are part of the end-to-end test (appendix B).

A.6 Aggregation, sorting and percentiles

After N iterations, the total costs are sorted (quicksort in the tool). Percentiles are read off via linear interpolation between the two nearest ranks:

```
sort totalCost ascending
k    = p*(N-1) + 1 ;  lo = Int(k) ;  hi = lo+1
P(p) = totalCost[lo] + (k – Int(k)) * (totalCost[hi] – totalCost[lo])

mean = (1/N) * sum(totalCost)
sd   = sqrt( (1/N) * sum(totalCost^2) – mean^2 )   (population sd)
```

```
skew = ( (1/N)*sum(c^3) – 3*mean*sd^2 – mean^3 ) / sd^3
```

The linearly interpolating percentile method is a common choice; at the iteration counts used (10,000 to 50,000) the difference from a nearest-rank method is negligible. The standard deviation uses the population form (divisor N); at these iteration counts the difference from the sample form (N-1) is negligible.

Appendix B. Test setup and measured values

For each test, a criterion and a numerical tolerance were defined in advance. The table below gives, per test, the measured value, the threshold and the sample/iteration size used. All values are the actual outcomes of the independent reimplementation.

Test	Measured value	Threshold	Sample / setup
1. Abramowitz-Stegun CDF	7.45×10^{-8}	1.00×10^{-7}	801 points, z∈[-4;4]
2. Beta-PERT shape parameters	0 (exact)	1.00×10^{-9}	analytical
3. Beta-PERT mode	0 (exact)	0,5	analytical
4. Inverse-beta bisection	3.71×10^{-8}	7.45×10^{-8}	200 points + 100k draws
5. Gamma -> Beta-PERT mean	4.70×10^{-4}	0,0100	100,000 draws
6. Cholesky decomposition ($L\times L^{T}=C$)	2.22×10^{-16}	1.00×10^{-9}	12×12 matrix
7. Copula: marginal (mean/sd)	5.35×10^{-3}	0,0200	200,000 draws
8. Copula: reproduction of matrix	4.33×10^{-3}	0,0200	200,000 draws, 12×12
9. Expected value (chain)	2.14×10^{-6}	0,0050	200,000 iterations
10. Cost percentiles	0 (exact)	logical	200,000 iterations
11. Convergence (2 seeds)	5.15×10^{-4}	0,0050	200,000 iter, seeds 11/99
12. Edge cases	0 (exact)	logical	10,000 draws
13. Scale test	1.08×10^{-5}	0,0010	100 elements, 50,000 iter
14. Generator MRG32k3a	0 (exact)	1.00×10^{-12}	known-answer + 1 million draws

Selected measured values in detail. Abramowitz-Stegun: max. |deviation| = 7.45×10^{-8} over 801 points. Beta-PERT: shape parameters reproduce the canonical PERT mean (23.333) exactly and the mode falls on m. Inverse-beta: bounded in [min, max] and monotonic, max. deviation 3.71×10^{-8} vs. SciPy. Expected value (full chain): simulated USD 478,063,439 against analytical USD 478,064,460 (deviation 0.0002%). Convergence: P85 = USD 514,972,789 versus USD 514,707,613 over two seeds (0.05%). Scale test: simulated USD 254.0 million against analytical USD 254.0 million (0.001%), P85 = USD 256.3 million.

B.1 Reproduction of the correlation matrix

To confirm that the copula actually reproduces the specified correlation, the empirical correlation of the transformed draws was compared with the entered 12x12 matrix of the real-world register (cluster structure 0.65 within the civil/tunnel group, 0.35 as baseline). The largest deviation over all 66 pairs was 0.0043, well within the threshold of 0.01 and entirely explained by sampling noise. The marginal distributions remained standard normal (largest deviation of the mean 0.005, of the standard deviation 0.002).

B.2 Effect of the correlation on the outcome

A control test compares the full real-world register with correlation against the same input without correlation (all elements independent). It confirms the expected behavior: the mean remains virtually the same, while the spread and the upper-tail percentiles increase clearly.

Metric	With correlation	Without correlation	Effect
Mean	USD 478.14 million	USD 478.11 million	virtually the same
Standard deviation	USD 34.81 million	USD 25.91 million	+34% spread
P90	USD 524.47 million	USD 512.50 million	+USD 12.0 million

Positive correlation thus makes the tail of the portfolio distribution fatter and increases the required contingency, precisely the effect for which a correlation matrix is used. Without correlation, independent deviations partly average each other out and the spread (and with it the reserve at P85/P90) is underestimated.

Appendix C. Reproducibility and environment

The verification was designed to be repeatable. A third party wishing to reassess the calculation engine can reproduce the same outcomes with the data below.

C.1 Environment

Reference implementation: Python with NumPy (fixed generator numpy.random.default_rng) and SciPy (scipy.special.betainc / betaincinv, scipy.stats.norm).

Test source: the calculation engine modMonteCarlo_v3 and the generator modQuantilRandom (engine v3.1) from the tool, mirrored line by line, including the MRG32k3a recursions, the Abramowitz-Stegun coefficients, the bisection on the regularized incomplete beta and the Cholesky transformation.

Basis of comparison: known analytical formulas, SciPy as an independent library, and theoretically expected statistical properties. The end-to-end test compares the calculation engine at seed 42 and 10,000 iterations with an independent rebuild along a different route; the supplied workbook carries the reference run of the demo case (seed 42, 10,000 iterations), so that the recipient can repeat the test directly; the comparison itself does not rely on those stored outcomes but on a fresh run.

C.2 Seeds and iteration counts

Each statistical test uses a fixed seed, so that the outcome is deterministically reproducible:

Test	Seed / iterations
Gamma -> Beta-PERT	seed 42: 100,000 draws
Copula (marginal and reproduction)	seed 3: 200,000 draws
Expected value (chain)	seed 2025: 200,000 iterations
Convergence	seeds 11 and 99: 200,000 iterations each
Correlation effect	seed 55: 200,000 iterations
End-to-end vs. tool	seed 2026: 200,000 iterations
Edge cases	seed 1: 10,000 draws
Scale test	seed 777 (setup), seed 123 (draw): 50,000 iter
Generator (known-answer and statistics)	L'Ecuyer's default state: 1,000,000 draws

Small differences in the generator or the seed lead to statistically equivalent, not bit-identical outcomes; the tolerances in appendix B are set accordingly.

C.3 Binding to the delivered file

This note validates the calculation engine as included in engine v3.1. To establish that a specific file actually contains that version, the SHA-256 checksum of the workbook is recorded with every delivery:

File name: 01. Quantil_QRCIM_Tool.xlsm

SHA-256: ca62a802deec34aa30ac38c69c0b563480359061925ed714dca2fc596ae22818

Established on: 2026-11-17

The recipient can recompute that checksum independently without installing anything. On Windows via PowerShell with Get-FileHash -Algorithm SHA256 '01. Quantil_QRCIM_Tool.xlsm' or via the command prompt with certutil -hashfile "01. Quantil_QRCIM_Tool.xlsm" SHA256 ; on macOS and Linux with shasum -a 256 . If the result matches the value above, the file is bit-for-bit the copy to which this note refers; if it differs, the file was modified in transit and the results recorded here do not apply to it.

Once the VBA project is digitally signed, the signature provides the same binding at code level: Excel then shows the publisher and warns as soon as the code has been modified after signing. The two complement each other. The checksum covers the entire workbook, including the sheets and the formulas; the signature covers only the VBA project, but is checked automatically by Excel without the user having to do anything.

Appendix D. Model choices and their limits

An honest validation states not only that the implementation is correct, but also which model choices were made and where their limits lie. The following choices are deliberate and defensible, but they do determine how the results are to be interpreted.

D.1 Correlation via a Gaussian copula

The correlation between cost items is modeled via a Gaussian copula based on the Cholesky decomposition of the correlation matrix. This is the established approach and allows any valid correlation structure, including clusters of correlated activities. The copula links the co-occurrence, while the magnitude of each item is drawn from its own Beta-PERT distribution; each item thus retains its full range without the individual distributions being distorted. Point of attention: a Gaussian copula captures the correlation in the center of the distribution and has no additional tail dependence; for portfolios where the joint extremes are dominant, a copula with tail dependence (for example a t-copula) may be more appropriate.

D.2 Shape factor lambda = 4

The Beta-PERT shape factor is set to lambda = 4 by default, the classic PERT setting that weights the most likely value more heavily than a triangular distribution. A higher lambda makes the distribution more peaked around m (less spread), a lower lambda flatter. The value is adjustable; lambda = 4 is the customary choice and the one tested here.

D.3 Discrete risks and schedule as overlay

The discrete risks and the schedule delays act as an overlay on top of the base estimate: probability of occurrence and impact magnitude are drawn independently, and the schedule contribution only counts when the planned duration is exceeded. This keeps the base uncertainty (cost items) and the event uncertainty (risks) cleanly separated. The limit: a risk that has already been incorporated in the uncertainty range of a cost item must not also be entered as a separate risk, otherwise double counting occurs. This is an input convention, not a computational limitation.

D.4 Capacity and percentile method

Capacity. The calculation engine handles up to approximately 50 cost items, 30 discrete risks and 20 schedule activities per analysis. This is a design limit, not a computational one; the scaling test confirms correctness up to 100 items, twice that limit.

Percentiles via linear interpolation. At 10,000 to 50,000 iterations, the difference from a nearest-rank method is smaller than the remaining uncertainty.

Input dependency. The calculation engine adds no assumptions of its own to the register. The reliability of a forecast is only as good as the quality of the estimates, probabilities and three-point estimates entered, hence the importance of a careful, facilitated build-up of the register.

D.5 Positioning and scope

The calculation engine is built and validated for a specific purpose: quantifying cost and schedule uncertainty within the AACE Total Cost Management framework and substantiating a contingency at a chosen confidence level. Within that purpose it does what the established method prescribes, and the correctness of that has been demonstrated in this note. For a fair expectation, it is worth stating what it is not intended for.

Schedule risk. The schedule aggregation works at activity level via delay times daily rate, not as a full schedule network with logical dependencies. For a schedule risk analysis that computes the logic of a Primavera or MS

Project schedule at network level, a network-based schedule risk package is the right choice; this calculation engine is a complement to that, not a replacement.

Scale and environment. The engine runs in Excel/VBA. That is a deliberate choice for transparency and accessibility, since the user can open and follow the calculation and there is no black box, with the downside that execution is slower than a compiled engine. For portfolios with thousands of risks or for very high iteration counts under time pressure, a compiled solution is more efficient.

In short: for cost estimating and substantiating contingencies on construction, infrastructure and capital investment projects, the calculation engine is a fully fledged and validated instrument. For network-based schedule risk, tail-dependent correlation or very large portfolios, specialized packages (such as @RISK or Primavera Risk Analysis) are the appropriate tools. The two can complement each other.

Appendix E. Literature and standards

The method applied and the validation approach rest on established sources:

AACE International, Recommended Practice 40R-08, Contingency Estimating: General Principles; and 41R-08 (Risk Analysis and Contingency Determination Using Range Estimating) and 57R-09 / 65R-11 for Monte Carlo-based contingency determination.

Vose, D. (2008), Risk Analysis: A Quantitative Guide (3rd ed.), Beta-PERT distribution, inverse transform sampling and copula correlation.

Abramowitz, M. & Stegun, I. (1964), Handbook of Mathematical Functions, formula 26.2.17, the normal CDF approximation used by the tool.

Marsaglia, G. & Tsang, W. (2000), A simple method for generating gamma variables, ACM TOMS: the gamma sampling used by the tool. | L'Ecuyer, P. (1999), Good parameters and implementations for combined multiple recursive random number generators, Operations Research 47(1), 159-164: the MRG32k3a generator that supplies all draws.

Malcolm, Roseboom, Clark & Fazar (1959): original PERT publication; basis of the (a + 4m + b)/6 derivation.

The combination of Beta-PERT sampling with a Cholesky-correlated Gaussian copula is the customary approach for cost estimating and contingency determination on capital-intensive construction, infrastructure and investment projects.

Prepared by

Quantil is a trading name of Bouman Management B.V.

Website: quantiltools.com · Support: support@quantiltools.com