# CalREL limit-state function

Here we wish to determine the failure probability accociated with the limit-state function

$$g = 1 - \frac{x2}{1000\,x3} - \left(\frac{x1}{200\,x3}\right)^2;$$

with $x_1$~Lognormal, $x_2$~Lognormal, $x_3$~Uniform having the following second-moment values:

```
μ1 = 500;
σ1 = 100;
μ2 = 2000;
σ2 = 400;
μ3 = 5;
σ3 = 0.5;
```

The correlation matrix is:

```
R = {{1, 0.3, 0.2}, {0.3, 1, 0.2}, {0.2, 0.2, 1}};
MatrixForm[R]
```

which yields: $\begin{pmatrix} 1 & 0.3 & 0.2 \\ 0.3 & 1 & 0.2 \\ 0.2 & 0.2 & 1 \end{pmatrix}$

## Auxiliary quantities

Distribution parameters are:

$$\zeta 1 = \text{Log}[\mu 1] - \frac{1}{2}\text{Log}\left[1 + \left(\frac{\sigma 1}{\mu 1}\right)^2\right];$$

$$\delta 1 = \sqrt{\text{Log}\left[\left(\frac{\sigma 1}{\mu 1}\right)^2 + 1\right]};$$

$$\zeta 2 = \text{Log}[\mu 2] - \frac{1}{2}\text{Log}\left[1 + \left(\frac{\sigma 2}{\mu 2}\right)^2\right];$$

$$\delta 2 = \sqrt{\text{Log}\left[\left(\frac{\sigma 2}{\mu 2}\right)^2 + 1\right]};$$

```
a3 = μ3 - √3 σ3;
b3 = μ3 + √3 σ3;
```

As explained in the 1986 paper by Pei-Ling Liu and Armen Der Kiureghian entitled “Multivariate distribution models with prescribed marginals and covariances” the correlation coefficients for the **z**-

variables, used later, is determined as followed:

$$\text{factor1} = \frac{1}{\sigma 1}\left(\text{InverseCDF}[\text{LogNormalDistribution}[\zeta 1, \delta 1], \text{CDF}[\text{NormalDistribution}[0, 1], \text{zi}]] - \mu 1\right);$$

$$\text{factor2} = \frac{1}{\sigma 2}\left(\text{InverseCDF}[\text{LogNormalDistribution}[\zeta 2, \delta 2], \text{CDF}[\text{NormalDistribution}[0, 1], \text{zj}]] - \mu 2\right);$$

$$\text{phi2} = \frac{1}{2\,\pi\,\sqrt{1-\rho 0\text{ij}^2}}\,\text{Exp}\left[-\frac{\text{zi}^2+\text{zj}^2-2\,\rho 0\text{ij}\,\text{zi}\,\text{zj}}{2\left(1-\rho 0\text{ij}^2\right)}\right];$$

$$\text{Solve}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\text{factor1}\,\text{factor2}\,\text{phi2}\,\text{dzi}\,\text{dzj} == 0.3,\ \rho 0\text{ij}\right]$$

Solving those integrals takes time, so here we simply enter the result:

```
R0 = {{1, 0.30414, 0.206718}, {0.30414, 1, 0.206718},
    {0.206718, 0.206718, 1}};
MatrixForm[R0]
```

which yields:
$$\begin{pmatrix} 1 & 0.30414 & 0.206718 \\ 0.30414 & 1 & 0.206718 \\ 0.206718 & 0.206718 & 1 \end{pmatrix}$$

Cholesky decomposition of the correlation matrix:

```
L = Transpose[CholeskyDecomposition[R0]];
MatrixForm[L]
```

which yields:
$$\begin{pmatrix} 1. & 0. & 0. \\ 0.30414 & 0.952627 & 0. \\ 0.206718 & 0.151 & 0.966678 \end{pmatrix}$$

Inverse of the Cholesky matrix:

```
Linv = Inverse[L];
MatrixForm[Linv]
```

which yields:
$$\begin{pmatrix} 1. & 0. & 0. \\ -0.319264 & 1.04973 & 0. \\ -0.163973 & -0.163973 & 1.03447 \end{pmatrix}$$

Gradient of the limit-state function:

```
∆g = {D[g, x1], D[g, x2], D[g, x3]};
MatrixForm[∆g]
```

which yields:
$$\begin{pmatrix} -\frac{x1}{20\,000\,x3^2} \\ -\frac{1}{1000\,x3} \\ \frac{x1^2}{20\,000\,x3^3}+\frac{x2}{1000\,x3^2} \end{pmatrix}$$

Hessian of the limit-state function:

```
H = {D[∆g, x1], D[∆g, x2], D[∆g, x3]};
MatrixForm[H]
```

which yields:
$$\begin{pmatrix} -\frac{1}{20\,000\,x3^2} & 0 & \frac{x1}{10\,000\,x3^3} \\ 0 & 0 & \frac{1}{1000\,x3^2} \\ \frac{x1}{10\,000\,x3^3} & \frac{1}{1000\,x3^2} & -\frac{3\,x1^2}{20\,000\,x3^4}-\frac{x2}{500\,x3^3} \end{pmatrix}$$

# One FORM step

Specify start-point in the **x**-space:

```
xVector = {μ1, μ2, μ3};
MatrixForm[xVector]
```

which yields:
$$\begin{pmatrix} 500 \\ 2000 \\ 5 \end{pmatrix}$$

Store all trial points in a matrix:

```
xMatrix = {xVector};
```

Transform that point into the **z**-space:

```
z1 = InverseCDF[NormalDistribution[0, 1],
    CDF[LogNormalDistribution[ζ1, δ1], xVector[[1]]]];
z2 = InverseCDF[NormalDistribution[0, 1],
    CDF[LogNormalDistribution[ζ2, δ2], xVector[[2]]]];
z3 = InverseCDF[NormalDistribution[0, 1],
    CDF[UniformDistribution[{a3, b3}], xVector[[3]]]];
zVector = {z1, z2, z3};
MatrixForm[zVector] // N
```

which yields: $\begin{pmatrix} 0.0990211 \\ 0.0990211 \\ 0. \end{pmatrix}$

Transform from the **z**-space to the **y**-space:

```
yVector = Linv.zVector;
MatrixForm[yVector]
```

which yields: $\begin{pmatrix} 0.0990211 \\ 0.0723314 \\ -0.0324736 \end{pmatrix}$

Evaluate the limit-state function:

```
gValue = g /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]} // N
```

which yields: `0.35`

Store the first *g*-value for later convergence checks:

```
gFirst = gValue;
```

Evaluate the gradient vector in the **x**-space:

```
∆gValue = ∆g /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]};
MatrixForm[∆gValue] // N
```

which yields: $\begin{pmatrix} -0.001 \\ -0.0002 \\ 0.18 \end{pmatrix}$

Establish the Jacobian for the **x** to **z** transformation:

```
ϕ1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[ζ1, δ1], xVector[[1]]];
ϕ2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[ζ2, δ2], xVector[[2]]];
ϕ3 = PDF[NormalDistribution[0, 1], z3];
f3 = PDF[UniformDistribution[{a3, b3}], xVector[[3]]];
dxdz = {{ϕ1/f1, 0, 0}, {0, ϕ2/f2, 0}, {0, 0, ϕ3/f3}};
MatrixForm[dxdz] // N
```

which yields:
$$\begin{pmatrix} 99.0211 & 0. & 0. \\ 0. & 396.084 & 0. \\ 0. & 0. & 0.690988 \end{pmatrix}$$

Evaluate the gradient in the **y**-space (notice that **L** must be last, or results will be slightly off):

```
△G = dxdz.△gValue.L;
MatrixForm[△G]
```

which yields:
$$\begin{pmatrix} -0.097403 \\ -0.0566831 \\ 0.120233 \end{pmatrix}$$

Calculate the $\alpha$-vector:

```
α = -△G/Norm[△G];
MatrixForm[α]
```

which yields:
$$\begin{pmatrix} 0.591066 \\ 0.343968 \\ -0.729607 \end{pmatrix}$$

Determine the HLRF search direction:

```
d = (gValue/Norm[△G] + α.yVector) α - yVector;
MatrixForm[d]
```

which yields:
$$\begin{pmatrix} 1.21964 \\ 0.695057 \\ -1.59527 \end{pmatrix}$$

Set the step size:

```
s = 1;
```

Take a step in the **y**-space:

```
yVector += s d;
MatrixForm[yVector]
```

which yields: $\begin{pmatrix} 1.31866 \\ 0.767389 \\ -1.62775 \end{pmatrix}$

Transform from **y**-space to **z**-space:

```
zVector = L.yVector;
MatrixForm[zVector]
```

which yields: $\begin{pmatrix} 1.31866 \\ 1.13209 \\ -1.18504 \end{pmatrix}$

Transform from **z**-space to **x**-space:

```
xValue1 = InverseCDF[LogNormalDistribution[ζ1, δ1],
   CDF[NormalDistribution[0, 1], zVector[[1]]]];
xValue2 = InverseCDF[LogNormalDistribution[ζ2, δ2],
   CDF[NormalDistribution[0, 1], zVector[[2]]]];
xValue3 = InverseCDF[UniformDistribution[{a3, b3}],
   CDF[NormalDistribution[0, 1], zVector[[3]]]];
xVector = {xValue1, xValue2, xValue3};
MatrixForm[xVector]
```

which yields: $\begin{pmatrix} 636.605 \\ 2454.05 \\ 4.33836 \end{pmatrix}$

Store the trial point for later plotting:

```
AppendTo[xMatrix, xVector];
```

# FORM iterations

The calculations above are here repeated until convergence:

```
counter = 1;
While[counter < 100,
 counter++;

  (* Transform from x-space to y-space *)
```

```
z1 = InverseCDF[NormalDistribution[0, 1],
  CDF[LogNormalDistribution[ζ1, δ1], xVector[[1]]]];
z2 = InverseCDF[NormalDistribution[0, 1],
  CDF[LogNormalDistribution[ζ2, δ2], xVector[[2]]]];
z3 = InverseCDF[NormalDistribution[0, 1],
  CDF[UniformDistribution[{a3, b3}], xVector[[3]]]];
zVector = {z1, z2, z3};
yVector = Linv.zVector;

(* Evaluate the limit-state function *)
gValue = g /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]};

(* Evaluate the gradient in the x-space *)
△gValue =
 △g /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]};

(* Transform the gradient into y-space *)
ϕ1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[ζ1, δ1], xVector[[1]]];
ϕ2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[ζ2, δ2], xVector[[2]]];
ϕ3 = PDF[NormalDistribution[0, 1], z3];
f3 = PDF[UniformDistribution[{a3, b3}], xVector[[3]]];
dxdz = {{ϕ1/f1, 0, 0}, {0, ϕ2/f2, 0}, {0, 0, ϕ3/f3}};
△G = dxdz.△gValue.L;

(* Evaluate the alpha-vector *)
α = -△G/Norm[△G];

(* Check convergence *)
convergenceCheck1 = gValue/gFirst;
convergenceCheck2 = Norm[yVector - (α.yVector) α];

(* Print status of the algorithm, and break if convergence *)
Print["Step ", counter, ": Check1=", convergenceCheck1,
 ", Check2=", convergenceCheck2, ", |y|=", Norm[yVector]];
If[convergenceCheck1 < 0.01 && convergenceCheck2 < 0.01, Break[]];

(* Take a step in the y-space *)
d = (gValue/Norm[△G] + α.yVector) α - yVector;
yVector += s d;
```

```
  (* Transform from y-space to x-space *)
  zVector = L.yVector;
  xValue1 = InverseCDF[LogNormalDistribution[ζ1, δ1],
    CDF[NormalDistribution[0, 1], zVector[[1]]]];
  xValue2 = InverseCDF[LogNormalDistribution[ζ2, δ2],
    CDF[NormalDistribution[0, 1], zVector[[2]]]];
  xValue3 = InverseCDF[UniformDistribution[{a3, b3}],
    CDF[NormalDistribution[0, 1], zVector[[3]]]];
  xVector = {xValue1, xValue2, xValue3};

  (* Store trial points in a matrix for plotting *)
  AppendTo[xMatrix, xVector];
 ]
```

```
Step 2: Check1=-0.297053, Check2=0.76923, |y|=2.23099
Step 3: Check1=0.0903486, Check2=0.504866, |y|=1.70647
Step 4: Check1=0.0265896, Check2=0.271462, |y|=1.74952
Step 5: Check1=0.00910004, Check2=0.162763, |y|=1.76543
Step 6: Check1=0.00330713, Check2=0.0972849, |y|=1.76996
Step 7: Check1=0.0012027, Check2=0.059014, |y|=1.77178
Step 8: Check1=0.000442024, Check2=0.0356695, |y|=1.7724
Step 9: Check1=0.000162124, Check2=0.0216443, |y|=1.77264
Step 10: Check1=0.0000596254
 , Check2=0.0131114, |y|=1.77272
Step 11: Check1=0.000021905
 , Check2=0.00795259, |y|=1.77276
```

For future use we store the design point coordinates in the standard normal space:

```
yStar = yVector;
xStar = xVector;
zStar = zVector;
△Gstar = △G;
dxdzStar = dxdz;
△gStar = △gValue;
```

The reliability index is:

```
βFORM = Norm[yStar]
```

which yields: `1.77276`

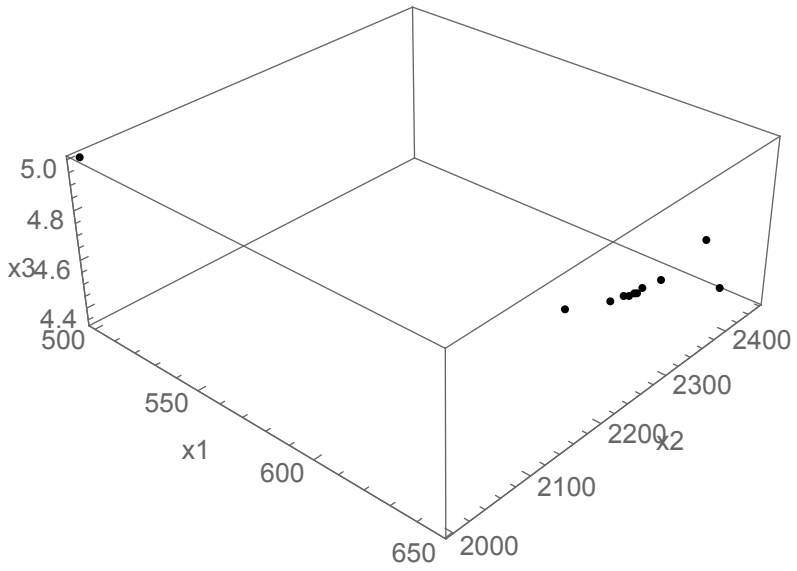
The associated failure probability from FORM is:

```
pfFORM = CDF[NormalDistribution[0, 1], -βFORM];
ScientificForm[pfFORM]
```

which yields: $3.81346 \times 10^{-2}$

Here is a plot that shows the first trial point in the upper left corner, and an increasing density of trial points as the design point is approached:

```
ListPointPlot3D[{xMatrix}, AxesLabel → {"x1", "x2", "x3"},
 PlotStyle → {Black}, PlotRange → All]
```




# SORM

The $\alpha$-vector at the design point is:

```
α // MatrixForm
```

which yields:
$$\begin{pmatrix} 0.722078 \\ 0.271177 \\ -0.636448 \end{pmatrix}$$

That vector is used together with the Gram Schmidt algorithm to establish the rotation matrix:

```
P = Orthogonalize[{α, {1, 0, 0}, {0, 0, 1}}, Method → "GramSchmidt"];
MatrixForm[P]
```

which yields:
$$\begin{pmatrix} 0.722078 & 0.271177 & -0.636448 \\ 0.691811 & -0.283041 & 0.664293 \\ -4.07099\times10^{-16} & 0.919973 & 0.391981 \end{pmatrix}$$

Evaluate the Hessian in the **x**-space:

```
Hvalue = H /. {x1 → xStar[[1]], x2 → xStar[[2]], x3 → xStar[[3]]};
MatrixForm[Hvalue]
```

which yields:
$$\begin{pmatrix} -2.4382\times10^{-6} & 0 & 0.000680896 \\ 0 & 0 & 0.000048764 \\ 0.000680896 & 0.000048764 & -0.192601 \end{pmatrix}$$

Prepare the second-order derivative of the probability transformation, in order to transform the Hessian into the standard normal space:

```
ϕ1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[ζ1, δ1], xVector[[1]]];
ϕ2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[ζ2, δ2], xVector[[2]]];
ϕ3 = PDF[NormalDistribution[0, 1], z3];
f3 = PDF[UniformDistribution[{a3, b3}], xVector[[3]]];
dϕdz = D[PDF[NormalDistribution[0, 1], z], z];
df1dx = D[PDF[LogNormalDistribution[ζ1, δ1], x], x];
df2dx = D[PDF[LogNormalDistribution[ζ2, δ2], x], x];
df3dx = D[PDF[UniformDistribution[{a3, b3}], x], x];
dϕdz1 = dϕdz /. z → zStar[[1]];
df1dx1 = df1dx /. x → xStar[[1]];
dϕdz2 = dϕdz /. z → zStar[[2]];
df2dx2 = df2dx /. x → xStar[[2]];
dϕdz3 = dϕdz /. z → zStar[[3]];
df3dx3 = df3dx /. x → xStar[[3]];
seconddxdz1 = (dϕdz1/f1 - 1/f1^2 df1dx1 ϕ1/f1 ϕ1);
seconddxdz2 = (dϕdz2/f2 - 1/f2^2 df2dx2 ϕ2/f2 ϕ2);
seconddxdz3 = (dϕdz3/f3 - 1/f3^2 df3dx3 ϕ3/f3 ϕ3);
```

Transform Hessian into the **y**-space:

```
factor1 = Hvalue.dxdzStar.L;
factor2 = dxdzStar.L;
firstTerm = Transpose[factor1].factor2;
factor3 = {{△gStar[[1]] seconddxdz1, 0, 0},
   {0, △gStar[[2]] seconddxdz2, 0}, {0, 0, △gStar[[3]] seconddxdz3}};
secondTerm = Transpose[L].factor3.L;
Htrans = firstTerm + secondTerm;
MatrixForm[Htrans]
```

which yields:
$$\begin{pmatrix} -0.0551878 & 0.00611412 & 0.0616488 \\ 0.00611412 & -0.0131489 & 0.0218201 \\ 0.0616488 & 0.0218201 & 0.0705671 \end{pmatrix}$$

The **A**-matrix is the rotated and scaled Hessian matrix:

$$\mathtt{A} = \frac{\mathtt{P.Htrans.P^T}}{\mathtt{Norm[\triangle Gstar]}};$$

```
MatrixForm[A]
```

which yields:
$$\begin{pmatrix} -0.241275 & -0.177085 & -0.0376754 \\ -0.177085 & 0.190608 & 0.204346 \\ -0.0376754 & 0.204346 & 0.0592136 \end{pmatrix}$$

Reduced **A**-matrix:

```
Acut = {{A[[2, 2]], A[[2, 3]]}, {A[[3, 2]], A[[3, 3]]}};
MatrixForm[Acut]
```

which yields:
$$\begin{pmatrix} 0.190608 & 0.204346 \\ 0.204346 & 0.0592136 \end{pmatrix}$$

Curvatures calculated by eigenvalue analysis:

```
κ = Eigenvalues[Acut]
```

which yields: {0.339558, -0.0897364}

The asymptotic SORM correction factors:

$$\mathtt{correction1} = \frac{1}{\sqrt{1 + \frac{\mathtt{PDF[NormalDistribution[0,1],\beta FORM]}}{\mathtt{pfFORM}}\,\kappa[[1]]}}$$

which yields: 0.758524

$$\text{correction2} = \frac{1}{\sqrt{1 + \frac{\text{PDF}[\text{NormalDistribution}[0,1],\beta\text{FORM}]}{\text{pfFORM}}\,\kappa[[2]]}}$$

which yields: 1.11459

Failure probability according to SORM:

$$\text{pfSORM} = \text{pfFORM}\ \text{correction1}\ \text{correction2}$$

which yields: 0.0322406

Corresponding reliability index:

$$\beta\text{SORM} = -\text{InverseCDF}[\text{NormalDistribution}[0, 1], \text{pfSORM}]$$

which yields: 1.84884

# Monte Carlo sampling

```
numSamples = 10 000;
counter = 0;
qSum = 0;
For[i = 1, i < numSamples + 1, i++,
 counter++;

 (* Generate outcomes of random variables between 0 and 1 *)
 u1 = RandomReal[];
 u2 = RandomReal[];
 u3 = RandomReal[];

 (* Transform to standard normal variables *)
 y1Value = InverseCDF[NormalDistribution[0, 1],
   CDF[UniformDistribution[{0, 1}], u1]];
 y2Value = InverseCDF[NormalDistribution[0, 1],
   CDF[UniformDistribution[{0, 1}], u2]];
 y3Value = InverseCDF[NormalDistribution[0, 1],
   CDF[UniformDistribution[{0, 1}], u3]];
 yVector = {y1Value, y2Value, y3Value};

 (* Transform from y-space to x-space *)
 zVector = L.yVector;
 xValue1 = InverseCDF[LogNormalDistribution[ζ1, δ1],
   CDF[NormalDistribution[0, 1], zVector[[1]]]];
 xValue2 = InverseCDF[LogNormalDistribution[ζ2, δ2],
   CDF[NormalDistribution[0, 1], zVector[[2]]]];
 xValue3 = InverseCDF[UniformDistribution[{a3, b3}],
   CDF[NormalDistribution[0, 1], zVector[[3]]]];

 (* Evaluate the limit-state function *)
 gValue = g /. {x1 → xValue1, x2 → xValue2, x3 → xValue3};

 (* Add to sum of indictor function if failure occurred *)
 If[gValue < 0, qSum++];
]
```

The estimate of the failure probability is:

$$\texttt{pfMCS} = \frac{\texttt{qSum}}{\texttt{numSamples}} \texttt{//} \texttt{N}$$

which yields: `0.0353`

Corresponding reliability index:

```
βMCS = -InverseCDF[NormalDistribution[0, 1], pfMCS]
```

which yields: `1.80804`

# Importance sampling

```
numSamples = 10 000;
counter = 0;
qSum = 0;
For[i = 1, i < numSamples + 1, i++,
 counter++;

 (* Generate outcomes of random variables between 0 and 1 *)
 u1 = RandomReal[];
 u2 = RandomReal[];
 u3 = RandomReal[];

 (* Transform to y-space with a shift toward the design point *)
 y1Value =
  InverseCDF[NormalDistribution[0, 1],
    CDF[UniformDistribution[{0, 1}], u1]] + yStar[[1]];
 y2Value =
  InverseCDF[NormalDistribution[0, 1],
    CDF[UniformDistribution[{0, 1}], u2]] + yStar[[2]];
 y3Value =
  InverseCDF[NormalDistribution[0, 1],
    CDF[UniformDistribution[{0, 1}], u3]] + yStar[[3]];
 yVector = {y1Value, y2Value, y3Value};

 (* Transform from y-space to x-space *)
 zVector = L.yVector;
 xValue1 = InverseCDF[LogNormalDistribution[ζ1, δ1],
   CDF[NormalDistribution[0, 1], zVector[[1]]]];
 xValue2 = InverseCDF[LogNormalDistribution[ζ2, δ2],
   CDF[NormalDistribution[0, 1], zVector[[2]]]];
 xValue3 = InverseCDF[UniformDistribution[{a3, b3}],
   CDF[NormalDistribution[0, 1], zVector[[3]]]];

 (* Evaluate the limit-state function *)
 gValue = g /. {x1 → xValue1, x2 → xValue2, x3 → xValue3};

 (* Calculate the correction factor phi/h *)
 phiOverh = Exp[1/2 (Norm[yVector - yStar]^2 - Norm[yVector]^2)];

 (* Add to sum of the corrected I-function if failure occurred *)
 If[gValue < 0, qSum += phiOverh];
]
```

The new estimat of the failure probability is:

$$\text{pfIS} = \frac{\text{qSum}}{\text{numSamples}} \text{ // N}$$

which yields: 0.0347589

Corresponding reliability index:

$$\beta\text{IS} = -\text{InverseCDF[NormalDistribution[0, 1], pfIS]}$$

which yields: 1.81504