

# CalREL limit-state function

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

$$g = 1 - \frac{x_2}{1000 x_3} - \left( \frac{x_1}{200 x_3} \right)^2;$$

with  $x_1 \sim \text{Lognormal}$ ,  $x_2 \sim \text{Lognormal}$ ,  $x_3 \sim \text{Uniform}$  having the following second-moment values:

$$\mu_1 = 500;$$

$$\sigma_1 = 100;$$

$$\delta_1 = \frac{\sigma_1}{\mu_1} // N$$

which yields: 0.2

$$\mu_2 = 2000;$$

$$\sigma_2 = 400;$$

$$\delta_2 = \frac{\sigma_2}{\mu_2} // N$$

which yields: 0.2

$$\mu_3 = 5;$$

$$\sigma_3 = 0.5;$$

$$\delta_3 = \frac{\sigma_3}{\mu_3} // N$$

which yields: 0.1

The correlation matrix is:

$$\rho_{12} = 0.3;$$

$$\rho_{13} = 0.2;$$

$$\rho_{23} = 0.2;$$

$$R = \{\{1, \rho_{12}, \rho_{13}\}, \{\rho_{12}, 1, \rho_{23}\}, \{\rho_{13}, \rho_{23}, 1\}\};$$

$$\text{MatrixForm}[R]$$

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:

$$\xi_1 = \text{Log}[\mu_1] - \frac{1}{2} \text{Log}\left[1 + \left(\frac{\sigma_1}{\mu_1}\right)^2\right];$$

$$\delta_1 = \sqrt{\text{Log}\left[\left(\frac{\sigma_1}{\mu_1}\right)^2 + 1\right]};$$

$$\xi_2 = \text{Log}[\mu_2] - \frac{1}{2} \text{Log}\left[1 + \left(\frac{\sigma_2}{\mu_2}\right)^2\right];$$

$$\delta_2 = \sqrt{\text{Log}\left[\left(\frac{\sigma_2}{\mu_2}\right)^2 + 1\right]};$$

$$a_3 = \mu_3 - \sqrt{3} \sigma_3;$$

$$b_3 = \mu_3 + \sqrt{3} \sigma_3;$$

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  $\mathbf{z}$ -variables, used later, is determined as followed:

$$\text{factor1} = \frac{1}{\sigma_1} (\text{InverseCDF}[\text{LogNormalDistribution}[\xi_1, \delta_1],$$

$$\text{CDF}[\text{NormalDistribution}[0, 1], z_i] - \mu_1);$$

$$\text{factor2} = \frac{1}{\sigma_2} (\text{InverseCDF}[\text{LogNormalDistribution}[\xi_2, \delta_2],$$

$$\text{CDF}[\text{NormalDistribution}[0, 1], z_j] - \mu_2);$$

$$\text{phi2} = \frac{1}{2\pi \sqrt{1 - \rho_{0ij}^2}} \text{Exp}\left[-\frac{z_i^2 + z_j^2 - 2\rho_{0ij} z_i z_j}{2 \times (1 - \rho_{0ij}^2)}\right];$$

$$\text{Solve}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\text{factor1 factor2 phi2 d}z_i\text{ d}z_j = 0.3, \rho_{0ij}\right]$$

Liu and Der Kiureghian developed easy-to-use formulas in order for analysts to avoid that numerical integration in practical applications. For the lognormal and uniform variables in this problem, the result is:

$$\begin{aligned}\rho_{120} &= \rho_{12} \left( \frac{\text{Log}[1 + \rho_{12} \delta_1 \delta_2]}{\rho_{12} \text{Sqrt}[\text{Log}[1 + \delta_1^2] \text{Log}[1 + \delta_2^2]]} \right); \\ \rho_{130} &= \rho_{13} (1.019 + 0.014 \delta_1 + 0.01 \rho_{13}^2 + 0.249 \delta_1^2); \\ \rho_{230} &= \rho_{23} (1.019 + 0.014 \delta_2 + 0.01 \rho_{23}^2 + 0.249 \delta_2^2); \\ R0 &= \{\{1, \rho_{120}, \rho_{130}\}, \{\rho_{120}, 1, \rho_{230}\}, \{\rho_{130}, \rho_{230}, 1\}\}; \\ \text{MatrixForm}[R0]\end{aligned}$$

which yields: 
$$\begin{pmatrix} 1 & 0.30406 & 0.206388 \\ 0.30406 & 1 & 0.206388 \\ 0.206388 & 0.206388 & 1 \end{pmatrix}$$

Cholesky decomposition of the correlation matrix:

$$\begin{aligned}L &= \text{Transpose}[\text{CholeskyDecomposition}[R0]]; \\ \text{MatrixForm}[L]\end{aligned}$$

which yields: 
$$\begin{pmatrix} 1. & 0. & 0. \\ 0.30406 & 0.952653 & 0. \\ 0.206388 & 0.150772 & 0.966784 \end{pmatrix}$$

Inverse of the Cholesky matrix:

$$\begin{aligned}L_{\text{inv}} &= \text{Inverse}[L]; \\ \text{MatrixForm}[L_{\text{inv}}]\end{aligned}$$

which yields: 
$$\begin{pmatrix} 1. & 0. & 0. \\ -0.319172 & 1.0497 & 0. \\ -0.163703 & -0.163703 & 1.03436 \end{pmatrix}$$

Gradient of the limit-state function:

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

which yields:

$$\begin{pmatrix} -\frac{x_1}{20\,000\,x_3^2} \\ -\frac{1}{1000\,x_3} \\ \frac{x_1^2}{20\,000\,x_3^3} + \frac{x_2}{1000\,x_3^2} \end{pmatrix}$$

Hessian of the limit-state function:

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

which yields:

$$\begin{pmatrix} -\frac{1}{20\,000\,x_3^2} & 0 & \frac{x_1}{10\,000\,x_3^3} \\ 0 & 0 & \frac{1}{1000\,x_3^2} \\ \frac{x_1}{10\,000\,x_3^3} & \frac{1}{1000\,x_3^2} & -\frac{3\,x_1^2}{20\,000\,x_3^4} - \frac{x_2}{500\,x_3^3} \end{pmatrix}$$

## 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[ξ1, δ1], xVector[[1]]]];
z2 = InverseCDF[NormalDistribution[0, 1],
  CDF[LogNormalDistribution[ξ2, 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: 
$$\begin{pmatrix} 0.0990211 \\ 0.0723377 \\ -0.0324201 \end{pmatrix}$$

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:

```

ΔgValue = Δg /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]};
MatrixForm[ΔgValue] // N

```

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

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

```

phi1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[xi1, delta1], xVector[[1]]];
phi2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[xi2, delta2], xVector[[2]]];
phi3 = PDF[NormalDistribution[0, 1], z3];
f3 = PDF[UniformDistribution[{a3, b3}], xVector[[3]]];
dxdz = {{phi1/f1, 0, 0}, {0, phi2/f2, 0}, {0, 0, phi3/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):

```

DeltaG = dxdz.DeltaGValue.L;
MatrixForm[DeltaG]

```

which yields: 
$$\begin{pmatrix} -0.0974377 \\ -0.0567135 \\ 0.120247 \end{pmatrix}$$

Calculate the  $\alpha$ -vector:

```

alpha = - DeltaG / Norm[DeltaG];
MatrixForm[alpha]

```

which yields: 
$$\begin{pmatrix} 0.591132 \\ 0.344067 \\ -0.729508 \end{pmatrix}$$

Determine the HLRF search direction:

$$\mathbf{d} = \left( \frac{\text{gValue}}{\text{Norm}[\Delta \mathbf{G}]} + \alpha \cdot \mathbf{yVector} \right) \alpha - \mathbf{yVector};$$

MatrixForm[d]

which yields: 
$$\begin{pmatrix} 1.21946 \\ 0.695084 \\ -1.5947 \end{pmatrix}$$

Set the step size:

$$s = 1;$$

Take a step in the y-space:

$$\mathbf{yVector} += s \mathbf{d};$$

MatrixForm[yVector]

which yields: 
$$\begin{pmatrix} 1.31848 \\ 0.767422 \\ -1.62712 \end{pmatrix}$$

Transform from y-space to z-space:

$$\mathbf{zVector} = \mathbf{L} \cdot \mathbf{yVector};$$

MatrixForm[zVector]

which yields: 
$$\begin{pmatrix} 1.31848 \\ 1.13198 \\ -1.18525 \end{pmatrix}$$

Transform from z-space to x-space:

```

xValue1 = InverseCDF[LogNormalDistribution[ $\xi_1$ ,  $\delta_1$ ],
  CDF[NormalDistribution[0, 1], zVector[[1]]]];
xValue2 = InverseCDF[LogNormalDistribution[ $\xi_2$ ,  $\delta_2$ ],
  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:  $\begin{pmatrix} 636.582 \\ 2454. \\ 4.33829 \end{pmatrix}$

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[ $\xi_1$ ,  $\delta_1$ ], xVector[[1]]]];
  z2 = InverseCDF[NormalDistribution[0, 1],
    CDF[LogNormalDistribution[ $\xi_2$ ,  $\delta_2$ ], 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 *)
ΔgValue =
  Δg /. {x1 → xVector[[1]], x2 → xVector[[2]], x3 → xVector[[3]]};

(* Transform the gradient into y-space *)
φ1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[ξ1, δ1], xVector[[1]]];
φ2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[ξ2, 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}};
ΔG = dxdz.ΔgValue.L;

(* Evaluate the alpha-vector *)
α = -  $\frac{\Delta G}{\text{Norm}[\Delta G]}$ ;

(* Check convergence *)
convergenceCheck1 =  $\frac{\text{gValue}}{\text{gFirst}}$ ;
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 =  $\left( \frac{\text{gValue}}{\text{Norm}[\Delta G]} + \alpha.yVector \right) \alpha - yVector$ ;
yVector += s d;

(* Transform from y-space to x-space *)
zVector = L.yVector;

```

```

xValue1 = InverseCDF[LogNormalDistribution[ $\xi_1$ ,  $\delta_1$ ],
  CDF[NormalDistribution[0, 1], zVector[[1]]]];
xValue2 = InverseCDF[LogNormalDistribution[ $\xi_2$ ,  $\delta_2$ ],
  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=-0.296988, Check2=0.769028,  $|y|=2.23044$

Step 3: Check1=0.0903339, Check2=0.504676,  $|y|=1.70609$

Step 4: Check1=0.0265809, Check2=0.271301,  $|y|=1.74914$

Step 5: Check1=0.00908765, Check2=0.162594,  $|y|=1.76504$

Step 6: Check1=0.00330063, Check2=0.0971494,  $|y|=1.76958$

Step 7: Check1=0.00119928, Check2=0.0589068,  $|y|=1.77139$

Step 8: Check1=0.000440435, Check2=0.0355908,  $|y|=1.77201$

Step 9: Check1=0.000161405, Check2=0.0215876,  $|y|=1.77225$

Step 10: Check1=0.0000593138

, Check2=0.0130718,  $|y|=1.77233$

Step 11: Check1=0.0000217725

, Check2=0.00792531,  $|y|=1.77236$

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

```
yStar = yVector;
```

```
xStar = xVector;
```

```
zStar = zVector;
```

```
 $\Delta G$ star =  $\Delta G$ ;
```

```
dxdzStar = dxdz;
```

```
 $\Delta g$ Star =  $\Delta g$ Value;
```

The reliability index is:

$$\beta_{\text{FORM}} = \text{Norm}[y_{\text{Star}}]$$

which yields: 1.77236

The associated failure probability from FORM is:

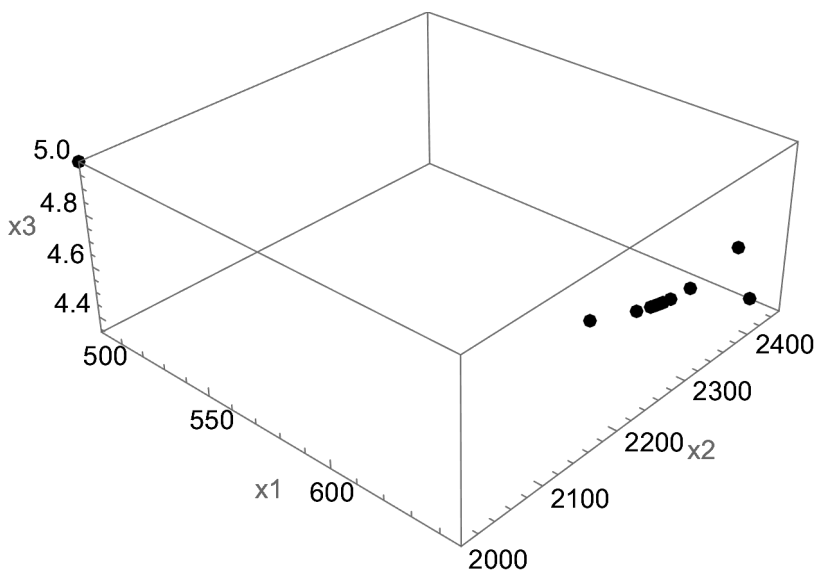
$$\text{pf}_{\text{FORM}} = \text{CDF}[\text{NormalDistribution}[0, 1], -\beta_{\text{FORM}}];$$

$$\text{ScientificForm}[\text{pf}_{\text{FORM}}]$$

which yields:  $3.81673 \times 10^{-2}$

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:

$$\alpha // \text{MatrixForm}$$

which yields:

$$\begin{pmatrix} 0.722146 \\ 0.271312 \\ -0.636314 \end{pmatrix}$$

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: 
$$\begin{pmatrix} 0.722146 & 0.271312 & -0.636314 \\ 0.691741 & -0.283237 & 0.664282 \\ -1.28624 \times 10^{-16} & 0.919873 & 0.392216 \end{pmatrix}$$

Make the  $\alpha$ -vector the last row:

```
P = {P[[2]], P[[3]], P[[1]]};
MatrixForm[P]
```

which yields: 
$$\begin{pmatrix} 0.691741 & -0.283237 & 0.664282 \\ -1.28624 \times 10^{-16} & 0.919873 & 0.392216 \\ 0.722146 & 0.271312 & -0.636314 \end{pmatrix}$$

Check that the rows are unit vectors:

```
Norm[P[[1]]]
```

which yields: 1.

```
Norm[P[[2]]]
```

which yields: 1.

```
Norm[P[[3]]]
```

which yields: 1.

Check that the rows are orthogonal:

```
Dot[P[[1]], P[[2]]]
```

which yields:  $-5.55112 \times 10^{-17}$

```
Dot[P[[1]], P[[3]]]
```

which yields:  $5.55112 \times 10^{-17}$

```
Dot[P[[2]], P[[3]]]
```

which yields:  $-1.66533 \times 10^{-16}$

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

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

which yields: 
$$\begin{pmatrix} -2.43833 \times 10^{-6} & 0 & 0.000680923 \\ 0 & 0 & 0.0000487665 \\ 0.000680923 & 0.0000487665 & -0.192609 \end{pmatrix}$$

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

```
phi1 = PDF[NormalDistribution[0, 1], z1];
f1 = PDF[LogNormalDistribution[xi1, delta1], xVector[[1]]];
phi2 = PDF[NormalDistribution[0, 1], z2];
f2 = PDF[LogNormalDistribution[xi2, delta2], xVector[[2]]];
phi3 = PDF[NormalDistribution[0, 1], z3];
f3 = PDF[UniformDistribution[{a3, b3}], xVector[[3]]];
dphiDz = D[PDF[NormalDistribution[0, 1], z], z];
df1dx = D[PDF[LogNormalDistribution[xi1, delta1], x], x];
df2dx = D[PDF[LogNormalDistribution[xi2, delta2], x], x];
df3dx = D[PDF[UniformDistribution[{a3, b3}], x], x];
dphiDz1 = dphiDz /. z → zStar[[1]];
df1dx1 = df1dx /. x → xStar[[1]];
dphiDz2 = dphiDz /. z → zStar[[2]];
df2dx2 = df2dx /. x → xStar[[2]];
dphiDz3 = dphiDz /. z → zStar[[3]];
df3dx3 = df3dx /. x → xStar[[3]];
secondDxdz1 =  $\left( \frac{d\phi Dz1}{f1} - \frac{1}{f1^2} df1dx1 \frac{\phi1}{f1} \phi1 \right)$ ;
secondDxdz2 =  $\left( \frac{d\phi Dz2}{f2} - \frac{1}{f2^2} df2dx2 \frac{\phi2}{f2} \phi2 \right)$ ;
secondDxdz3 =  $\left( \frac{d\phi Dz3}{f3} - \frac{1}{f3^2} df3dx3 \frac{\phi3}{f3} \phi3 \right)$ ;
```

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

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

which yields: 
$$\begin{pmatrix} -0.0552294 & 0.00609296 & 0.0616301 \\ 0.00609296 & -0.0131601 & 0.0218098 \\ 0.0616301 & 0.0218098 & 0.0706147 \end{pmatrix}$$

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

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

MatrixForm[A]

which yields: 
$$\begin{pmatrix} 0.190516 & 0.204338 & -0.177138 \\ 0.204338 & 0.0592553 & -0.0377401 \\ -0.177138 & -0.0377401 & -0.241245 \end{pmatrix}$$

Reduced **A**-matrix:

```
Acut = {{A[[1, 1]], A[[1, 2]]}, {A[[2, 1]], A[[2, 2]]}};
MatrixForm[Acut]
```

which yields: 
$$\begin{pmatrix} 0.190516 & 0.204338 \\ 0.204338 & 0.0592553 \end{pmatrix}$$

Curvatures calculated by eigenvalue analysis:

```
κ = Eigenvalues[Acut]
```

which yields: {0.339504, -0.0897332}

The asymptotic SORM correction factors:

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

which yields: 0.758575

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

which yields: 1.11456

Failure probability according to SORM:

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

which yields: 0.0322696

Corresponding reliability index:

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

which yields: 1.84844

## 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[ $\xi_1$ ,  $\delta_1$ ],
    CDF[NormalDistribution[0, 1], zVector[[1]]]];
  xValue2 = InverseCDF[LogNormalDistribution[ $\xi_2$ ,  $\delta_2$ ],
    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 indicator function if failure occurred *)
  If[gValue < 0, qSum++];
]

```



The estimate of the failure probability is:

$$p_{fMCS} = \frac{qSum}{numSamples} // N$$

which yields: 0.0319

Corresponding reliability index:

$$\beta_{MCS} = -\text{InverseCDF}[\text{NormalDistribution}[0, 1], p_{fMCS}]$$

which yields: 1.85357

## 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[ξ1, δ1],
  CDF[NormalDistribution[0, 1], zVector[[1]]]];
xValue2 = InverseCDF[LogNormalDistribution[ξ2, 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[ $\frac{1}{2} (\text{Norm}[y\text{Vector} - y\text{Star}]^2 - \text{Norm}[y\text{Vector}]^2)$ ];

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

```

The new estimat of the failure probability is:

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

which yields: 0.0342119

Corresponding reliability index:

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

which yields: 1.82221