The Second-order Reliability Method (SORM)

This material was first described to me in a course taught by my PhD thesis supervisor Professor Armen Der Kiureghian at the University of California at Berkeley. In 2005 he made an excellent description available in Chapter 14 "First- and second-order reliability methods" of the CRC Engineering Design Reliability Handbook edited by Nikolaidis, Ghiocel and Singhal, published by the CRC Press in Boca Raton, Florida.

Essentially, the second-order reliability method (SORM) is an extension of FORM. Specifically, a hyper-paraboloid is employed to approximate the limit-state surface instead of a hyper-plane. The exact expression for the probability mass outside a hyper-paraboloid with distance β from the origin in the uncorrelated standard normal space is

$$p_f = \varphi(\beta) \cdot \mathrm{Re}\left(i \cdot \sqrt{\frac{2}{\pi}} \cdot \int_0^{i\omega} \frac{1}{s} \cdot e^{\left(\frac{(s+\beta)^2}{2}\right)} \cdot \prod_{i=1}^{n-1} \frac{1}{\sqrt{1+\kappa_i s}} ds \right) \tag{1}$$

where n is the number of random variables. An alternative expression is the asymptotic approximation

$$p_f \approx \Phi(-\beta) \cdot \prod_{i=1}^{n-1} \frac{1}{\sqrt{1+\psi(\beta)\cdot\kappa_i}} \tag{2}$$

in which the FORM probability is multiplied by a correction factor that depends upon the principal curvatures of the limit-state surface at the design point, κ_i and

$$\psi(\beta) = \frac{\varphi(\beta)}{\Phi(-\beta)} \tag{3}$$

The key question in SORM is how to determine the curvatures, κ_i. There are several options, some of which are outlined in the following sections.

Curvature Fitted SORM

Suppose the second-derivatives of the limit-state function are not too difficult to compute analytically. Then consider the second-order Taylor series expansion of the limit-state function in the standard normal space:

$$G(\mathbf{y}) \approx \underbrace{G(\mathbf{y}^*)}_{=0} + \nabla G(\mathbf{y}^*)^T (\mathbf{y}-\mathbf{y}^*) + \frac{1}{2}\cdot(\mathbf{y}-\mathbf{y}^*)^T \underbrace{\left[\frac{\partial^2 G}{\partial y_i \partial y_j}\right]}_{=\mathbf{H}} (\mathbf{y}-\mathbf{y}^*) \tag{4}$$

where the Hessian matrix, $\mathbf{H}(\mathbf{y})$, of second-derivatives is identified. Take notice that $\mathbf{H}$ is indeed here defined in the standard normal $\mathbf{y}$-space of random variables. Transformation of the Hessian in the original $\mathbf{x}$-space to the $\mathbf{y}$-space is addressed later. Substitution of the

FORM-linearization of the limit-state function into Eq. (4), i.e., writing the linear Taylor series expansion as $G(\mathbf{y}) \approx \|\nabla G\|(\beta - \boldsymbol{\alpha}^T \mathbf{y})$ yields

$$G(\mathbf{y}) \approx \left\|\nabla G(\mathbf{y}^*)\right\|(\beta - \boldsymbol{\alpha}^T \mathbf{y}) + \frac{1}{2} \cdot (\mathbf{y} - \mathbf{y}^*)^T \mathbf{H}(\mathbf{y} - \mathbf{y}^*) \tag{5}$$

To construct a hyper-paraboloid at the design point in the standard normal space the coordinate system is first rotated so that one of the random variable axes align with the alpha-vector. Usually, the last random variable is arbitrarily selected for this purpose. This is shown for the simple example of two random variables in Figure 1. The tilde is used to identify the rotated coordinate system.

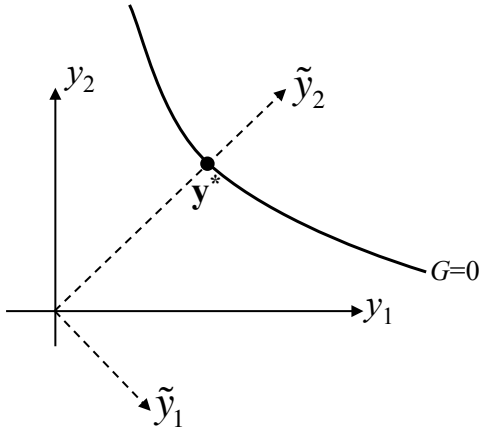

Figure 1: Rotation of the standard normal space to construct hyper-paraboloid.

Mathematically, the rotation of the coordinate system is expressed by means of a rotation matrix, **P**:

$$\tilde{\mathbf{y}} = \mathbf{P}\mathbf{y} \tag{6}$$

where **P** is an *n*-dimensional square matrix with the alpha-vector as its last row, where *n* is the number of random variables. P satisfies the property $\mathbf{P}^T\mathbf{P}=\mathbf{I}$ so that

$$\mathbf{y} = \mathbf{P}^T \tilde{\mathbf{y}} \tag{7}$$

The other rows of **P** are orthogonal to the alpha-vector and orthogonal to each other. They serve the purpose of ensuring a proper Cartesian coordinate system of the rotated system. Algorithms like the Gram-Schmidt procedure are employed to establish the set of orthogonal vectors. Substitution of Eq. (6) into the Taylor approximation in Eq. (4) yields

$$\begin{aligned} G(\tilde{\mathbf{y}}) &\approx \left\|\nabla G(\mathbf{y}^*)\right\|\left(\beta - \boldsymbol{\alpha}^T \left(\mathbf{P}^T \tilde{\mathbf{y}}\right)\right) + \frac{1}{2} \cdot \left(\mathbf{P}^T \tilde{\mathbf{y}} - \mathbf{P}^T \tilde{\mathbf{y}}^*\right)^T \mathbf{H}\left(\mathbf{P}^T \tilde{\mathbf{y}} - \mathbf{P}^T \tilde{\mathbf{y}}^*\right) \\ &= \left\|\nabla G(\mathbf{y}^*)\right\|\left(\beta - \tilde{y}_n\right) + \frac{1}{2} \cdot \left(\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^*\right)^T \mathbf{P}\mathbf{H}\mathbf{P}^T \left(\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^*\right) \end{aligned} \tag{8}$$

where $\tilde{y}_n \equiv \boldsymbol{\alpha}^T \left(\mathbf{P}^T \tilde{\mathbf{y}}\right)$. To further simplify, the limit-state function is scaled by the norm of the gradient at the design point, so that

$$\tilde{G}(\tilde{\mathbf{y}}) \approx \beta - \tilde{y}_n + \frac{1}{2} \cdot \left(\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^*\right)^T \mathbf{A} \left(\tilde{\mathbf{y}} - \tilde{\mathbf{y}}^*\right) \tag{9}$$

where $\tilde{G}(\tilde{\mathbf{y}}) \equiv G(\tilde{\mathbf{y}}) \big/ \left\| \nabla G(\mathbf{y}^*) \right\|$ and $\mathbf{A} \equiv \mathbf{P}\mathbf{H}\mathbf{P}^T \big/ \left\| \nabla G(\mathbf{y}^*) \right\|$. Two facts help simplify Eq. (9): First, because of the rotation the last row and column of $\mathbf{A}$ contain only zeros. Second, it is noted that the design point coordinates in the rotated coordinate system is

$$\tilde{\mathbf{y}}^* = \left\{ \begin{array}{ccccc} 0 & 0 & \cdots & \beta \end{array} \right\}^T \tag{10}$$

These two facts allows the matrix multiplication in Eq. (9) to be reduced to

$$\tilde{G}(\tilde{\mathbf{y}}) \approx \beta - \tilde{y}_n + \frac{1}{2} \cdot \tilde{\mathbf{y}}_{\text{cut}}^T \mathbf{A}_{\text{cut}} \tilde{\mathbf{y}}_{\text{cut}} \tag{11}$$

where the subscript "cut" indicates removal of the last vector element and the last matrix row and column. Finally, $\mathbf{A}_{\text{cut}}$ is diagonalized by computing its eigenvalues and Eq. (11) is rewritten as

$$\tilde{G}(\tilde{\mathbf{y}}) \approx \beta - \tilde{y}_n + \frac{1}{2} \cdot \sum_{i=1}^{n-1} \kappa_i \cdot \tilde{y}_i^2 \tag{12}$$

where κ_i are the n-1 eigenvalues of $\mathbf{A}_{\text{cut}}$, which are the principal curvatures of the hyper-paraboloid. These are the curvatures that enter into Eq. (2) to compute the failure probability from SORM. A positive curvature, κ_i, implies an outwards-curving limit-state surface in the $\tilde{y}_i, \tilde{y}_n$ coordinate system. If the curvature is negative it cannot be so large that points on the limit-state surface are closer to the origin than the design point. That would invalidate the design point from FORM. In other words, the absolute value of negative curvatures cannot exceed the curvature of a circle around the origin with curvature $1/\beta$:

$$\kappa_i > -\frac{1}{\beta} \tag{13}$$

An important task above is to calculate the Hessian. The limit-state function is often defined in terms of numerical responses collected in $\mathbf{u}$, which depends upon the input variables $\mathbf{x}$. In that case the Hessian is obtained by invoking the product rule and chain rule of differentiation:

$$\begin{aligned} \frac{\partial^2 G}{\partial y_i \partial y_j} &= \frac{\partial}{\partial y_i}\left(\frac{\partial G}{\partial y_j}\right) = \frac{\partial}{\partial y_i}\left(\frac{\partial g}{\partial u_k}\frac{\partial u_k}{\partial x_m}\frac{\partial x_m}{\partial y_j}\right) \\ &= \left(\frac{\partial u_p}{\partial x_q}\frac{\partial x_q}{\partial y_i}\underbrace{\frac{\partial^2 g}{\partial u_p \partial u_k}}_{A}\right)\frac{\partial u_k}{\partial x_m}\frac{\partial x_m}{\partial y_j} + \frac{\partial g}{\partial u_k}\left(\frac{\partial x_q}{\partial y_i}\underbrace{\frac{\partial^2 u_k}{\partial x_q \partial x_m}}_{B}\right)\frac{\partial x_m}{\partial y_j} + \frac{\partial g}{\partial u_k}\frac{\partial u_k}{\partial x_m}\left(\underbrace{\frac{\partial^2 x_m}{\partial y_i \partial y_j}}_{C}\right) \end{aligned} \tag{14}$$

where summation over repeated indices is implied. The matrix identified by A, not to be confused with the matrix $\mathbf{A}$ above, is readily obtained because the limit-state function is

usually an algebraic function of the responses **u**. The matrix identified by *B* contains the response sensitivities from the numerical response algorithm. The matrix identified by *C* is obtained by differentiating the probability transformation twice. Suppose the Nataf transformation is used, then the **x**-variables are linked with **z**-variables:

$$F(x_i) = \Phi(z_i) \tag{15}$$

The first differentiation yields:

$$\begin{aligned}
&\frac{\partial}{\partial z_j}\Big(F(x_i) = \Phi(z_i)\Big) \\
&\Rightarrow \frac{\partial}{\partial x_k}\frac{\partial x_k}{\partial z_j}F(x_i) = \frac{\partial}{\partial z_j}\Phi(z_i) \\
&\Rightarrow \frac{\partial F(x_i)}{\partial x_k}\frac{\partial x_k}{\partial z_j} = \frac{\partial \Phi(z_i)}{\partial z_j} \\
&\Rightarrow f(x_i)\frac{\partial x_i}{\partial z_i} = \varphi(z_i) \\
&\Rightarrow \frac{\partial x_i}{\partial z_i} = \frac{\varphi(z_i)}{f(x_i)}
\end{aligned} \tag{16}$$

where the result is a diagonal matrix because each marginal probability distribution varies only with its own random variable. The second differentiation yields:

$$\begin{aligned}
&\frac{\partial}{\partial z_j}\left(\frac{\partial x_i}{\partial z_i} = \frac{\varphi(z_i)}{f(x_i)}\right) \\
&\Rightarrow \frac{\partial^2 x_i}{\partial z_i \partial z_j} = \frac{\partial \varphi(z_i)}{\partial z_j}\cdot\frac{1}{f(x_i)} + \frac{\partial}{\partial z_j}\left(\frac{1}{f(x_i)}\right)\cdot\varphi(z_i) \\
&\Rightarrow \frac{\partial^2 x_i}{\partial z_i \partial z_j} = \frac{\partial \varphi(z_i)}{\partial z_j}\cdot\frac{1}{f(x_i)} + \frac{\partial}{\partial x_j}\left(\frac{1}{f(x_i)}\right)\cdot\frac{\partial x_j}{\partial z_j}\cdot\varphi(z_i) \\
&\Rightarrow \frac{\partial^2 x_i}{\partial z_i \partial z_j} = \frac{\partial \varphi(z_i)}{\partial z_j}\cdot\frac{1}{f(x_i)} - \frac{1}{f(x_i)^2}\cdot\frac{\partial f(x_i)}{\partial x_j}\cdot\frac{\partial x_j}{\partial z_j}\cdot\varphi(z_i)
\end{aligned} \tag{17}$$

The result is zero for $i \neq j$ because each marginal probability distribution varies only with its own random variable. This means that Eq. (17) is a square diagonal tensor, with dimension equal to the number of random variables, just like Eq. (16) was a square diagonal matrix. In turn, the **z**-variables are linked with the **y**-variables via the Cholesky decomposition of the modified correlation matrix:

$$z_i = L_{ij} y_j \tag{18}$$

Differentiation yields

$$\frac{\partial}{\partial y_k}\left(z_i = L_{ij} y_j\right) \Rightarrow \frac{\partial z_i}{\partial y_k} = L_{ij}\frac{\partial y_j}{\partial y_k} \Rightarrow \frac{\partial z_i}{\partial y_j} = L_{ij} \tag{19}$$

The second differentiation yields zero. We are now ready to substitute the derivatives of the probability transformation into the expressions for the gradient and Hessian. Using the above equations we obtain the gradient vector in the **y**-space as follows:

$$\frac{\partial G}{\partial y_i} = \frac{\partial g}{\partial u_j}\cdot\frac{\partial u_j}{\partial x_k}\cdot\frac{\partial x_k}{\partial z_l}\cdot\frac{\partial z_l}{\partial y_i} = \frac{\partial g}{\partial u_j}\cdot\frac{\partial u_j}{\partial x_k}\cdot\left[\frac{\varphi(z)}{f(x)}\right]_{kl}\cdot L_{li} \tag{20}$$

where square brackets are used to denote a diagonal matrix. Next we reconsider Eq. (14), now including the intermediate **z**-variables:

$$\begin{aligned}
\frac{\partial G}{\partial y_i \partial y_j} &= \frac{\partial}{\partial y_j}\left(\frac{\partial g}{\partial u_k}\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i}\right) \\
&= \frac{\partial}{\partial y_j}\left(\frac{\partial g}{\partial u_k}\right)\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i} + \frac{\partial g}{\partial u_k}\cdot\frac{\partial}{\partial y_j}\left(\frac{\partial u_k}{\partial x_l}\right)\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i} \\
&+ \frac{\partial g}{\partial u_k}\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial}{\partial y_j}\left(\frac{\partial x_l}{\partial z_m}\right)\cdot\frac{\partial z_m}{\partial y_i} + \frac{\partial g}{\partial u_k}\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial}{\partial y_j}\left(\frac{\partial z_m}{\partial y_i}\right) \\
&= \left(\frac{\partial^2 g}{\partial u_n \partial u_k}\cdot\frac{\partial u_n}{\partial x_o}\cdot\frac{\partial x_o}{\partial z_p}\cdot\frac{\partial z_p}{\partial y_j}\right)\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i} + \frac{\partial g}{\partial u_k}\cdot\left(\frac{\partial^2 u_k}{\partial x_l \partial x_n}\cdot\frac{\partial x_n}{\partial z_o}\cdot\frac{\partial z_o}{\partial y_j}\right)\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i} \\
&+ \frac{\partial g}{\partial u_k}\cdot\frac{\partial u_k}{\partial x_l}\cdot\left(\frac{\partial^2 x_l}{\partial z_m \partial z_n}\cdot\frac{\partial z_n}{\partial y_j}\right)\cdot\frac{\partial z_m}{\partial y_i} + \frac{\partial g}{\partial u_k}\cdot\frac{\partial u_k}{\partial x_l}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial^2 z_m}{\partial y_i \partial y_j}
\end{aligned} \tag{21}$$

The last term is zero because the second differentiation of **z**=**Ly** is zero. To further simplify, consider explicit limit-state functions where g depends directly on **x** without any **u**-variables. Eq. (21) then reads:

$$\begin{aligned}
\frac{\partial G}{\partial y_i \partial y_j} &= \frac{\partial^2 g}{\partial x_l \partial x_n}\cdot\frac{\partial x_n}{\partial z_o}\cdot\frac{\partial z_o}{\partial y_j}\cdot\frac{\partial x_l}{\partial z_m}\cdot\frac{\partial z_m}{\partial y_i} + \frac{\partial g}{\partial x_l}\cdot\frac{\partial^2 x_l}{\partial z_m \partial z_n}\cdot\frac{\partial z_n}{\partial y_j}\cdot\frac{\partial z_m}{\partial y_i} \\
&= \frac{\partial^2 g}{\partial x_l \partial x_n}\cdot\left[\frac{\varphi(z)}{f(x)}\right]_{no}\cdot L_{oj}\cdot\left[\frac{\varphi(z)}{f(x)}\right]_{lm}\cdot L_{mi} \\
&+ \frac{\partial g}{\partial x_l}\cdot\left[\frac{\partial\varphi(z)}{\partial z}\cdot\frac{1}{f(x)} - \frac{1}{f(x)^2}\cdot\frac{\partial f(x)}{\partial x}\cdot\frac{\partial x}{\partial z}\cdot\varphi(z)\right]_{lmn}\cdot L_{nj}\cdot L_{mi}
\end{aligned} \tag{22}$$

Caution must be exercised when conducting matrix multiplications to evaluate Eq. (22). The first term has factors with indices *ln*, *no*, *oj*, *lm*, *mi*. Here one can first multiply the three first matrices *ln*, *no*, *oj* giving *lj*. The last to matrices *lm*, *mi* are multiplied to give *li*. Hence, the final value of the first term is the *lj* matrix transposed times the *li* matrix. The second term has factors with indices *l*, *lmn*, *nj*, *mi*. The product of the first two is an *mn* matrix with values $\left(\partial g/\partial x\right)\cdot\left(\partial^2 x/\partial z^2\right)$ along the diagonal. Then what remains in the second term is *mn*, *nj*, *mi*. To multiply this out with matrix products the last of them, *mi*, is transposed and placed first to give the matrix product *im*, *mn*, *nj*. These calculations are

demonstrated for the CalREL limit-state function on the example-webpage. In summary, the procedure for curvature-fitted SORM is:

1. Have an analytical expression for the limit-state function
2. Differentiate it twice to obtain gradient and Hessian
3. Establish the **A**-matrix use an orthogonalization algorithm
4. Perform eigenvalue analysis to obtain principal curvatures
5. Update the failure probability using Eq. (2)

This approach is straightforward but it is limited to explicit limit-state functions, or at least limit-state functions for which second-derivative computations are possible. Obtaining those derivatives can be computationally expensive, and the results may be prone to numerical noise in the calculation of the limit-state function.

Gradient-based SORM

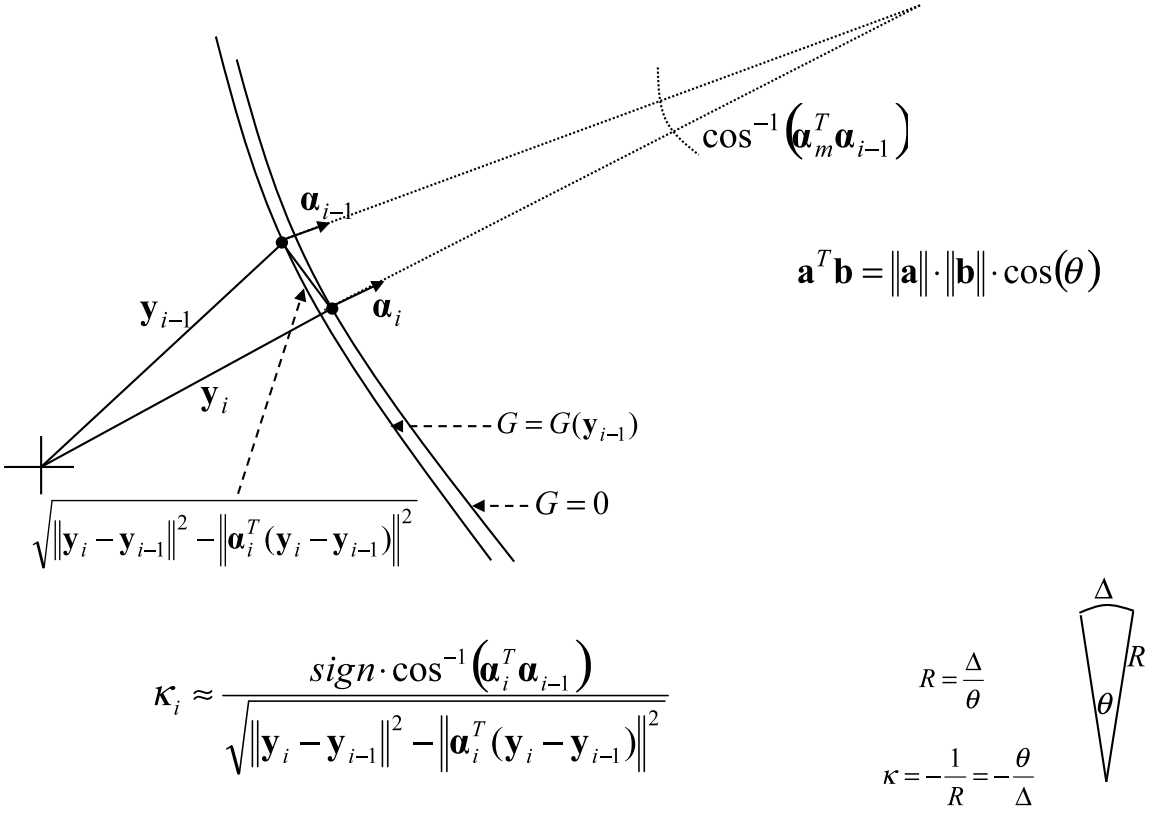

Figure 2: Geometrical considerations for gradient-based SORM.