Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

457.646 Topics in Structural Reliability

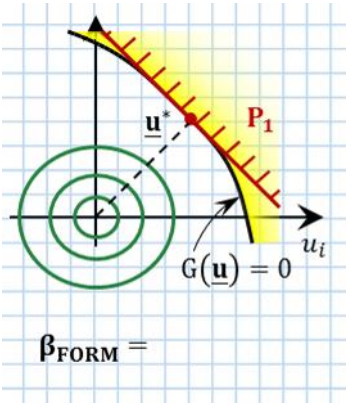
In-Class Material: Class 13

④ First-order reliability method (FORM)

$P_f \cong$ Probability in the linear half space determined by

FO approximation of failure domain at $\mathbf{u} =$

$= p_1$



i)

$G(\mathbf{u})$ +

$=$ ≤ 0

Divide by $\|\nabla G(\mathbf{u}^*)\|$

$(\mathbf{u} - \mathbf{u}^*) \leq 0$

$(\mathbf{u} - \mathbf{u}^*) \leq 0$

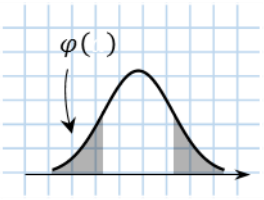
$\therefore p_1 = P($ $\leq 0)$

Consider $Z = \hat{\boldsymbol{\alpha}}\mathbf{u} =$ + +⋯+

i) Type → (_____ function of)

ii) $\mu_Z =$

iii) $\sigma_Z^2 =$



In summary $Z \sim$ ( )

$P_1 = P(\beta_{FORM} -$ $\leq 0)$

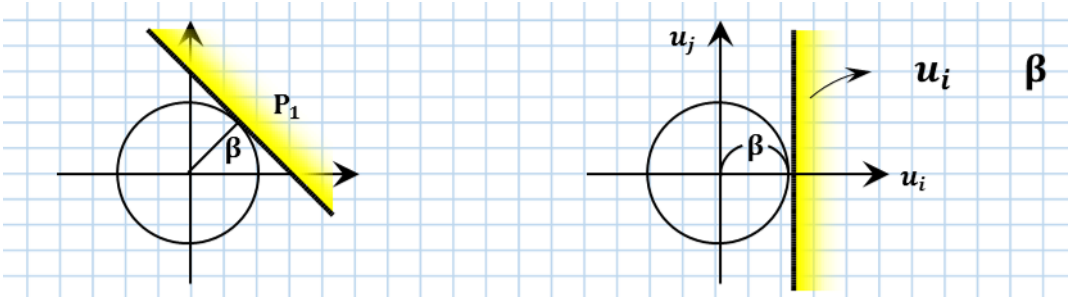
$\therefore$ $= P($ $\geq \beta_{FORM})$

$= \Phi($ $)$

Instructor: Junho Song
junhosong@snu.ac.kr

or

ii) From rotational symmetry



$$p_1 = P(u_i \quad \beta) = \Phi(\quad)$$

※ SORM (p_2): later

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

◎ Probabilistic Transformation & Jacobian (to achieve $\mathbf{U}$ ~)

∴ Transformation (&Jacobian) depends on _____

cf. β_{HL} $\begin{cases} \mathbf{x(u) = DLu + M} \\ \mathbf{J_{x,u} = DL} \end{cases}$

☆ Why do we need $\mathbf{X(u)}$ and $\mathbf{J_{X,u}}$

$\left.\begin{cases} G(\mathbf{u}_i) = g(\ \ ) \\ \nabla_{\mathbf{u}} G(\mathbf{u}_i) = \nabla_{\mathbf{x}} g(\ \ ) \end{cases}\right\} \rightarrow$ need $\begin{cases} \mathbf{x}_i = \mathbf{x}(\mathbf{u}_i) \\ \mathbf{J_{x,u}} = \mathbf{J_{u,x}}^{-1} \text{ at} \end{cases}$

⇒ Four cases

S.I	Dependent
①	② ③ ④

① **X** ~ statistically independent of each other

Each follows general distribution ($F_{X_i}(x_i)$ or $f_{X_i}(x_i)$)

$f_{\mathbf{X}}(\mathbf{x}) =$

⇒ Transformation $\quad u_i = \Phi^{-1}[\qquad]$

Check $f_{\mathbf{U}}(\mathbf{u}) = \varphi_n(\mathbf{u};\ \ ) = \prod_{i=1}^{n}$?

$f_{\mathbf{u}}(\mathbf{u}) = f_{\mathbf{X}}(\mathbf{x}) \cdot \left|\qquad\right|$

$= \qquad \times$ ———

$=$

$J_{\mathbf{x,u}} = \begin{bmatrix} & \\ & \\ & \end{bmatrix}$

∴ **u** ~

⇒ Jacobian $J_{\mathbf{x,u}}$

Instructor: Junho Song
junhosong@snu.ac.kr

$$J_{ii} = \frac{dx_i}{du_i} = \text{——} \qquad \text{; Ratio of PDFs}$$

Note $F_{X_i}(x_i) = \Phi(u_i)$

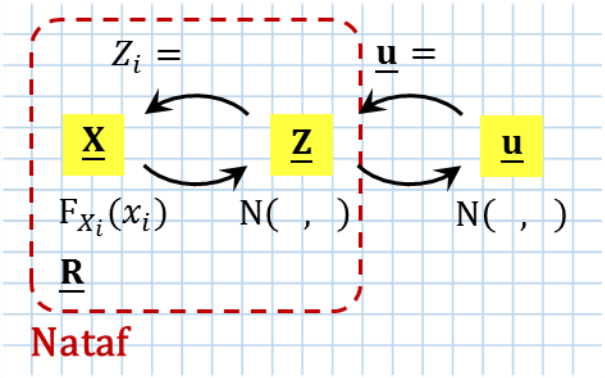
$$f_{X_i}(x_i)dx_i = \varphi(u_i)du_i$$

② **X** ~ Correlated Normal, $\mathbf{N(M, \Sigma)}$

⇒ Transform $\begin{cases} \mathbf{x} = & \mathbf{X} & \square & \mathbf{u} \\ \mathbf{u} = & N(\mathbf{M}, \mathbf{\Sigma}) & & N(\mathbf{0}, \mathbf{I}) \end{cases}$

⇒ Jacobian $J_{\mathbf{x},\mathbf{u}}$ therefore β_{HL} β_{FORM} for $\mathbf{X} \sim N(\quad)$

③ **X** ~ Nataf distribution : & available



note $\begin{cases} \mathbf{X(U)} = \mathbf{DLU} + \mathbf{M} \\ \mathbf{Z} = \end{cases}$

⇒ Transform $\mathbf{u}$ = $\mathbf{z}$ = $\left\{ \quad \right\}$

⇒ Jacobian $\begin{cases} J_{\mathbf{u},\mathbf{x}} = J \quad \cdot J \quad = \\ J_{\mathbf{x},\mathbf{u}} = \end{cases}$

④ Non-normal, non-Nataf, dependent RVs

e.g. Hohenbichler & Rackwitz 1981 (named, Rosenblatt's transformation)

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

Transformation for non-normal, non-Nataf, dependent random variables

>> Rosenblatt's transformation (Rosenblatt 1952; Hohenbichler & Rachwitz 1981)

Given:

$$f_{\mathbf{X}}(\mathbf{x}) = f_{X_n}(x_n \mid x_1,\ldots,x_{n-1})\, f_{X_{n-1}}(x_{n-1} \mid x_1,\ldots,x_{n-2}) \cdots f_{X_2}(x_2 \mid x_1)\, f_{X_1}(x_1)$$

~ conditional PDFs are available.

Transformation: triangular transformation

$$u_1 = \Phi^{-1}\left[F_{X_1}(x_1)\right]$$
$$u_2 = \Phi^{-1}\left[F_{X_2}(x_2 \mid x_1)\right]$$
$$\vdots$$
$$u_n = \Phi^{-1}\left[F_{X_n}(x_n \mid x_1,\ldots,x_{n-1})\right]$$

** **Proof:** $\mathbf{U} \sim N(\mathbf{0},\mathbf{I})$?

$$\begin{aligned} f_{\mathbf{U}}(\mathbf{u}) &= f_{\mathbf{X}}(\mathbf{x})\left|\det \mathbf{J}_{\mathbf{u},\mathbf{x}}\right|^{-1} \\ &= f_{\mathbf{X}}(\mathbf{x})\left[\prod_{i=1}^{n} J_{i,i}\right]^{-1} \quad (\because \mathbf{J}_{\mathbf{u},\mathbf{x}} \text{ lower triangular matrix}) \\ &= f_{\mathbf{X}}(\mathbf{x})\frac{\varphi(u_1)}{f_{X_1}(x_1)}\frac{\varphi(u_2)}{f_{X_2}(x_2 \mid x_1)}\cdots\frac{\varphi(u_n)}{f_{X_n}(x_n \mid x_1,\ldots,x_{n-1})} \\ &= \prod_{i=1}^{n}\varphi(u_i) \quad \text{(uncorrelated standard normal)} \end{aligned}$$

Jacobian: $\mathbf{J}_{\mathbf{u},\mathbf{x}} = [J_{ij}]$ where

$$J_{ij} = \begin{cases} \dfrac{f_{X_1}(x_1)}{\varphi(u_1)} & i = j = 1 \\ \dfrac{f_{X_i}(x_i \mid x_1,\ldots,x_{i-1})}{\varphi(u_i)} & i = j > 1 \\ \dfrac{1}{\varphi(u_i)}\dfrac{\partial F_{X_i}(x_i \mid x_1,\ldots,x_{i-1})}{\partial x_j} & i > j \\ 0 & i < j \end{cases}$$

** **What does** $F_{X_i}(x_i \mid x_1,\ldots,x_{i-1})$ **mean?**

Instructor: Junho Song
junhosong@snu.ac.kr

First of all, $F_{X_i}(x_i \mid x_1,...,x_{i-1}) \neq \dfrac{F_{X_1...X_i}(x_1,...,x_i)}{F_{X_1...X_{i-1}}(x_1,...,x_{i-1})}$. It is rather the conditional probability that $X_i \le x_i$ given $X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}$ that is,

$$
\begin{aligned}
F_{X_i}(x_i \mid x_1,...,x_{i-1}) &= P(X_i \le x_i \mid X_1 = x_1,...,X_{i-1} = x_{i-1}) \\
&= \int_{-\infty}^{x_i} f_{X_i}(x_i \mid x_1,...,x_{i-1}) dx_i \\
&= \int_{-\infty}^{x_i} \frac{f(x_1,...,x_i)}{f(x_1,...,x_{i-1})} dx_i \\
&= \frac{1}{f(x_1,...,x_{i-1})} \int_{-\infty}^{x_i} \frac{\partial^i F(x_1,...,x_i)}{\partial x_1 \cdots \partial x_i} dx_i \\
&= \frac{1}{f(x_1,...,x_{i-1})} \frac{\partial^{i-1} F(x_1,...,x_i)}{\partial x_1 \cdots \partial x_{i-1}}
\end{aligned}
$$

For example,

$$
F_{X_2}(x_2 \mid x_1) = \frac{1}{f_{X_1}(x_1)} \frac{\partial F(x_1, x_2)}{\partial x_1}, \quad F_{X_3}(x_3 \mid x_1, x_2) = \frac{1}{f(x_1, x_2)} \frac{\partial^2 F(x_1, x_2, x_3)}{\partial x_1 \partial x_2}
$$

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

457.646 Topics in Structural Reliability

In-Class Material: Class 14

FERUM: Finite Element Reliability Using Matlab®

FERUM (URL: http://projects.ce.berkeley.edu/ferum/) is an open source Matlab® toolbox for structural reliability analysis, created by Dr. Terje Haukaas during his Ph.D. study at UC Berkeley (currently at the University of British Columbia).

- **FERUMcore** contains the core algorithms to perform FORM, SORM, Monte Carlo simulations and importance sampling.
- **FERUMlinearfecode** is a simple finite element code provided with FERUM to enable linear finite element reliability analysis with truss, beam or quad4 elements. Limit-state functions can be defined in terms of displacement reponse from this code. Gradients can be computed either by direct differentiation (DDM) or by a forward finite difference scheme.
- **FERUMnonlinearfecode** is an add-on to FERUMlinearfecode to enable nonlinear finite element reliability analysis. The J2 plasticity material is provided, and gradients can be computed by direct differentiation (DDM) or by forward finite difference. Truss and quad4 elements are available.
- **FERUMdynamicfecode** is yet another extension of FERUMlinearfecode to enable limit-state functions being defined in terms of response quantities from a dynamic finite element analysis.
- **FERUMlargedefofecode** is an add-on to enable limit-state functions being defined in terms of response quantities from a finite element code capable of large deformation analysis.
- **FERUMsystems** enables FERUM to perform system reliability analysis using the Matrix-based System Reliability (MSR) method. This part of FERUM was created by Bora Gencturk during his CEE491 term project, and is maintained by Junho Song.
- **FERUMrandomfield** is an add-on to the simple finite element codes provided with FERUM. It addresses the issue of characterizing material properties as random fields. Options for the simple 1D case was provided with the initial versions of FERUM. However, the main contributions to the current version have been made by Bruno Sudret, who has also provided a user's/theory manual for the random field part of FERUM (see the User's Guide section).
- **FERUMfedeasconnection** enables the finite element program FedeasLab developed by Professor Filip Filippou at UC Berkeley to be connected to FERUM. This provides for a quite powerful computational platform for finite element reliability analysis. This part is maintained by Paolo Franchin.
- **FERUMexamples** contains a collection of example input files for FERUM.

Other notable reliability software packages:

- **FERUM4.0** by Jean-Marc Bourinet (SIGMA Clermont): https://www.sigma-clermont.fr/en/ferum
- **UQLab** by B. Sudret and S. Marelli (ETH Zürich): https://www.uqlab.com/

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

◎ FERUM Example (Example 14.3.1.1 in Chapter 14 of CRC Handbook (ADK 2005))

Limit-state function for a short column (elastic-perfect-plastic) under axial force and axial bending:

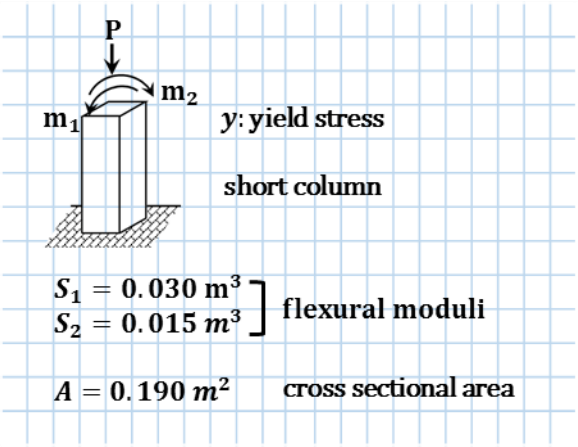
$$g(\mathbf{x}) = 1 - \frac{m_1}{s_1 y} - \frac{m_2}{s_2 y} - \left(\frac{P}{Ay}\right)^2$$

m_1 : Normal

m_2 : Normal

P : Gumbel

y : Weibull



⇒ FERUM results

$\beta_{FORM} = 2.47$

$\mathbf{u}^* = \{1.21 \quad 0.699 \quad \theta.941\}^T$

$\mathbf{x}^* = \{341 \quad 170 \quad 3223\}^T \quad 31.$

$\hat{\boldsymbol{\alpha}} = \{0.491 \quad 0.283 \quad \theta.381$

$P_f \simeq \Phi(-\beta_{FORM}) = 0.00682$

Instructor: Junho Song
junhosong@snu.ac.kr

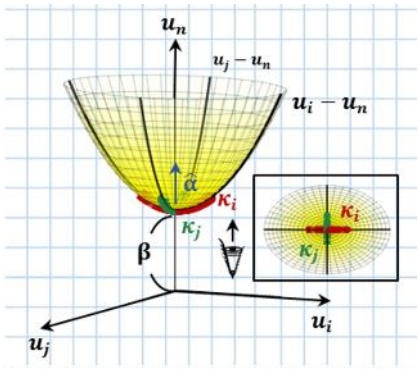
% FERUM INPUTFILE

```
clear probdata femodel analysisopt gfundata randomfield systems results
output_filename

output_filename = 'output_Ch14_Example.txt';


probdata.marg(1,:) =  [ 1   2.5e5   2.5e5*0.3   2.5e5  0 0 0 0 0];
probdata.marg(2,:) =  [ 1  1.25e5  1.25e5*0.3  1.25e5  0 0 0 0 0];
probdata.marg(3,:) =  [15   2.5e6   2.5e6*0.2   2.5e6  0 0 0 0 0];
probdata.marg(4,:) =  [16   4.0e7   4.0e7*0.1   4.0e7  0 0 0 0 0];


probdata.correlation = [1.0 0.5 0.3 0.0;
                        0.5 1.0 0.3 0.0;
                        0.3 0.3 1.0 0.0;
                        0.0 0.0 0.0 1.0];


probdata.parameter = distribution_parameter(probdata.marg);


analysisopt.ig_max    = 100;
analysisopt.il_max    = 5;
analysisopt.e1       = 0.001;
analysisopt.e2       = 0.001;
analysisopt.step_code = 1;
analysisopt.grad_flag = 'DDM';
analysisopt.sim_point = 'dspt';
analysisopt.stdv_sim  = 1;
analysisopt.num_sim   = 100000;
analysisopt.target_cov = 0.0125;


gfundata(1).evaluator = 'basic';
gfundata(1).type = 'expression';
gfundata(1).parameter = 'no';
gfundata(1).expression = '1-x(1)/0.030/x(4)-x(2)/0.015/x(4)-
(x(3)/0.190/x(4))^2';
gfundata(1).dgdq = { '-1/0.030/x(4)' ;
                     '-1/0.015/x(4)';
                     '-2*x(3)/0.190^2/x(4)^2';
'x(1)/0.030/x(4)^2+x(2)/0.015/x(4)^2+2*x(3)^2/0.190^2/x(4)^3'};

femodel = 0;
randomfield.mesh = 0;
```

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

◎ Second- Order Reliability Method (read CRC Ch.14)



$$P_f \simeq \text{Prob in paraboloid}(\beta, \kappa_i)$$
$$= p_2$$
$$= P(\beta - u_n + \frac{1}{2}\sum_{i=1}^{n-1}\kappa_i u_i^2 \le 0)$$

(κ : principal curvature in $u_i - u_n$ plane)

※ Formulas for p_2

① Tvedt (exact; under the condition $\beta\kappa_i > -1$)

$$p_2 = \varphi(\beta)\,\mathrm{Re}\left\{ i\sqrt{\frac{2}{\pi}}\int_0^{i\infty}\frac{1}{s}\exp\left[\frac{(s+\beta)^2}{2}\right]\prod_{i=1}^{n-1}\frac{1}{\sqrt{1+\kappa_i s}}\,ds\right\}$$

② (Karl) Breitung (simpler; derived earlier; approximate)

$$p_2 \cong \Phi(-\beta)\prod_{i=1}^{n-1}\frac{1}{\sqrt{1+\beta\kappa_i}}$$

$\kappa > 0$ $\quad p_2 \quad p_1$

$\kappa = 0$ $\quad p_2 \quad p_1$

$\kappa < 0$ $\quad p_2 \quad p_1$

③ Improved Breitung

$$p_2 \cong \Phi(-\beta)\prod_{i=1}^{n-1}\frac{1}{\sqrt{1+\psi(\beta)\kappa_i}} \quad \text{where} \quad \psi(\beta) = \frac{\varphi(\beta)}{\Phi(-\beta)} \quad (\leftarrow \text{erratum in Ch.14})$$

※ How to get κ_i's, $i = 1, \cdots, n-1$? (κ : principal curvature)

Seoul National University
Dept. of Civil and Environmental Engineering

Instructor: Junho Song
junhosong@snu.ac.kr

① Curvature-fitting SORM (see in-class material)

⇒ Find ( ) matrix $\mathbf{H}=\left[\dfrac{\partial^2 G}{\partial u_i \partial u_j}\right]$ at $\mathbf{u}=$

⇒ Two rotations & eigenvalue analysis to obtain $\beta - u_n + \frac{1}{2}\sum \kappa_i u_i^2 \le 0$

⇒ Getting Hessian → Costly & Inaccurate

② Gradient-based SORM (ADK & De Stefano 1991)

⇒ Find the largest principal curvature from the trajectory of $\mathbf{u}$'s during HL-RF search to get $\mathbf{u}^*$

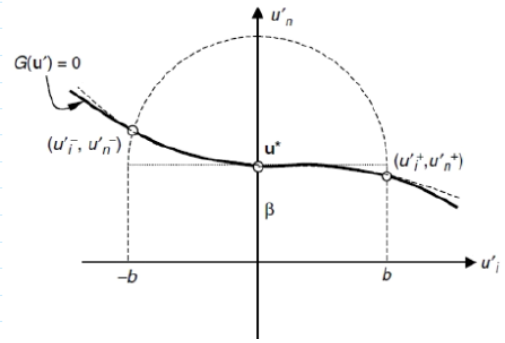
⇒ For the 2nd largest, perform HL-RF in the subspace orthogonal to u_n and u_i (that has the largest κ_i)

⇒ stop searching when $|\kappa_i| < \varepsilon$

⇒ does not need $\mathbf{H}$; can stop when $\|\kappa_i\|$ small

⇒ implementation issue?

③ Point-fitting SORM (ADK, Liu and Hwang 1987)



Fit by piecewise paraboloid surface

$$G(\mathbf{u}) \simeq \beta - u_n + \frac{1}{2}\sum_{i=1}^{n-1} a_i^{\mathrm{sgn}(u_i)} u_i^2$$

where $a_i^{\mathrm{sgn}(u_i)} = \dfrac{2(u_n^{\mathrm{sgn}(u_i)} - \beta)}{2(u_i^{\mathrm{sgn}(u_i)})^2}$

$$b = \begin{cases} 1 & \text{if } |\beta| \le 1 \\ |\beta| & \text{if } 1 < |\beta| \le 3 \\ 3 & \text{if } |\beta| > 3 \end{cases}$$

Merit: Insensitive to the noise in calculating $g(\mathbf{x})$

Does not require derivative calculations ($\mathbf{H}$)

Drawback: $2\times(n-1)$ fitting points ⇒ solve numerically

Not invariant (rotation not unique)