A Review of Recent Features and Improvements Added to FERUM Software

J.-M. Bourinet, C. Mattrand, V. Dubourg
IFMA & Université Blaise Pascal, Laboratoire de Mécanique et Ingénieries
Campus des Cézeaux, BP265, F-63175 Aubière Cedex, France

Keywords: reliability, reliability sensitivities, simulation, reliability-based optimization, FERUM

ABSTRACT: The development of FERUM (**F**inite **E**lement **R**eliability **U**sing **M**atlab) as an open-source Matlab® toolbox was initiated in 1999 under A. Der Kiureghian's leadership at the University of California at Berkeley (UCB). This general purpose structural reliability code was developed and maintained by T. Haukaas, with the contributions of many researchers at UCB. The present paper aims at presenting the main features and capabilities of a new version of this open-source code based on the main contribution of the first author and the help of a few Ph.D. students at the Institut Français de Mécanique Avancée (IFMA) in Clermont-Ferrand, France. The main concepts of FERUM are preserved and this paper lists the major changes operated in this code. Available algorithms are briefly presented and, for some of them, they are applied on reference examples for illustration purposes. This new version offers improved capabilities including simulation-based technique (subset simulation), sensitivity analysis (Sobol's indices) and a Reliability-Based Optimization algorithm. Beyond the new methods implemented in this code, this paper put some emphasis on the new architecture of the code, which now allow distributed computing, either virtually through vectorized calculations within Matlab or for real with multi-processor computers.

1 INTRODUCTION

FERUM (**F**inite **E**lement **R**eliability **U**sing **M**atlab) is a general purpose structural reliability code whose first developments started in 1999 at UC Berkeley (Der Kiureghian *et al.* 2006). This code consists of an open-source Matlab® toolbox, featuring various structural reliability methods. As opposed to commercial structural reliability codes, see *e.g.* reference (Pellissetti & Schuëller 2006) for a review in 2006, the main objective of FERUM is to provide students with a tool immediately comprehensible and easy to use and researchers with a tool very accessible which they may develop for research purposes. The scripting language of Matlab is perfect for such objectives, as it allows users to give commands in a very flexible way, either in an interactive mode or in a batch mode through input files.

FERUM was created under Prof. A. Der Kiureghian's leadership and was managed by T. Haukaas at UCB until 2003. It benefited from a prior experience with CalRel structural reliability code, which features all the methods implemented in the last version of FERUM. It also benefited from the works of many researchers at UCB, who made valuable contributions in the last available version. Version 3.1 is the last release and all necessary Matlab m-files can be downloaded at the following address: http://www.ce.berkeley.edu/FERUM/. Since 2003, this code is no longer officially maintained.

The objective of this paper is to review changes brought to FERUM since 2001 by the first author and other individuals at IFMA. As previously achieved in the past, the main intention is to provide students and researchers with a developer-friendly computational platform which facilitates learning methods and serves as a basis for collaborative research works. FERUM should still be viewed as a development platform for testing new methods and applying them to various challenging engineering problems, either represented by basic analytical models or more elaborated numerical models, through proper user-defined interfaces.

The main architecture of FERUM was preserved in general, see Section 2 for more details. In order to improve its efficiency in terms of computational time, all algorithms have been revisited to extend FERUM capabilities to distributed computing. For example, in its new version, FERUM makes Monte Carlo Simulations (MCS) much faster thanks to limit-state functions defined in a vectorized form or real distributed computing, according that a proper interface is defined for sending multiple jobs to a multi-processor computer platform.

Section 2 clearly states the framework of structural reliability and presents some details on the architecture and main features of this new release of FERUM (version 4.0). Next sections are then dedicated to methods implemented in FERUM 4.0, approximation methods such as FORM and SORM in Section 3, simulation methods in Section 4, Sensitivity Analysis (SA) in Section 5 and Reliability-Based Design Optimization (RBDO) in Section 6. Examples demonstrating advantages and common pitfalls of various methods are given all along the text, to illustrate the potential applications of FERUM.

2 PROBLEM DEFINITION AND STRUCTURE OF FERUM

This section briefly presents the general formulation of time-invariant structural reliability problems. In addition to some brief details about theoretical concepts, this section highlights how these concepts are translated to FERUM structure. This includes for instance the stochastic model, the transformation to standard normal variates, limit-state functions and more generally other aspects regarding computational issues. It is important here to recall that the main structure of input data in FERUM is preserved compared to version 3.1 (same Matlab structure variables: *probdata*, *analysisopt*, *gfundata*, *femodel*, *randomfield* and *system*). Changes brought to FERUM are applied in core m-functions and within the fields of the existing structure variables. Similarly to version 3.1, results are stored in structure variables with the following syntax: *results* keyword appended to the name of the method applied, such as *e.g. formresults*, *sormresults*, etc.

2.1 *Time invariant structural reliability*

We consider here only time invariant structural reliability problems, see *e.g.* (Ditlevsen & Madsen 2007). The probability *w.r.t.* an undesired or unsafe state is expressed in terms of a n-dimensional vector $\mathbf{X}$ of random variables with continuous joint density function $f_{\mathbf{X}}(\mathbf{x}, \boldsymbol{\theta}_f)$, where $\boldsymbol{\theta}_f$ stands for a vector of distribution parameters. Failure is defined in terms of a limit-state function $g(\mathbf{x}, \boldsymbol{\theta}_g)$ where $\mathbf{x}$ is a realization of the random vector $\mathbf{X}$ and $\boldsymbol{\theta}_g$ denotes a vector of deterministic limit-state function parameters. We will restrict here the analysis to component reliability with a single g function, but this function may represent multiple failure modes in subset simulation in Section 4.3, without lack of generality. This limit-state function divides the random variable space in a safety domain, $g(\mathbf{x}) > 0$, and a failure domain, $g(\mathbf{x}) \leq 0$. The probability of failure therefore reads:

$$p_f = \int_{g(\mathbf{x}, \boldsymbol{\theta}_g) \leq 0} f_{\mathbf{X}}\left(\mathbf{x}, \boldsymbol{\theta}_f\right) \mathrm{d}\,\mathbf{x} \tag{1}$$

2.2 *Probability distributions and transformation to standard normal space*

The joint density function $f_{\mathbf{X}}(\mathbf{x}, \boldsymbol{\theta}_f)$ is often unknown and replaced by its Nataf counterpart completely defined by specifying marginal distributions and the Gaussian correlation structure between random variables (Liu & Der Kiureghian 1986). This Nataf joint distribution is completely specified by variables *probdata.marg* and *probdata.correlation* in FERUM input files. FERUM has a rich library of probability distribution models, including extreme value distributions and a truncated normal distribution. These distributions can be specified through either their statistical moments or parameters.

The structural reliability problem expressed in the original space of random variables $\mathbf{x}$ in Equation (1) is commonly transformed to a standard normal space $\mathbf{u}$, where $\mathbf{U}$ becomes an independent standard normal vector. For a Nataf joint distribution, physical random variables $\mathbf{X}$ are transformed to correlated standard normal variables $\mathbf{Z}$, whose correlation structure obeys the following integral equation:

$$\rho_{ij} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\left(\frac{x_i - \mu_i}{\sigma_i}\right)\left(\frac{x_j - \mu_j}{\sigma_j}\right)\ldots \\ \ldots\varphi_2\left(z_i, z_j, \rho_{0ij}\right)\mathrm{d}\,z_i\,\mathrm{d}\,z_j \tag{2}$$

where μ_i and σ_i respectively stand for the mean and standard deviation of the i^{th} component of $\mathbf{X}$, and $\varphi_2(\bullet,\bullet,\rho)$ is the 2D standard normal probability density function (pdf) with correlation coefficient ρ.

Independent standard normal variables $\mathbf{U}$ are then obtained from $\mathbf{Z}$ variables such as follows:

$$\mathbf{u} = \mathbf{L}_0^{-1}\mathbf{z} \tag{3}$$

where $\mathbf{L}_0$ is the lower-triangular Cholesky decomposition of $\mathbf{R}_0 = [\rho_{0\,ij}]$ matrix, such that $\mathbf{L}_0\mathbf{L}_0{}^{\mathrm{T}} = \mathbf{R}_0$.

Previous version of FERUM was based on formulae taken from reference (Liu & Der Kiureghian 1986), most of them obtained by least-squared fitting and therefore approximate. FERUM 4.0 is now based on accurate solutions obtained by 2D numerical Gauss integration of Equation (2). A particular attention is paid to strongly correlated random variables, where the number of integration points along each dimension in z_iz_j-space must be selected carefully, for accurate ρ_{0ij} values.

2.3 *Definition of limit-state functions*

As in the previous version, the limit-state function is defined through structure variable *gfundata* of the input file and called through the file named *gfun.m*. Various strategies are now offered in FERUM 4.0. The limit-state function can either be a simple expression directly written in the input file or a Matlab function. For both cases, *gfun.m* calls another func-

tion called *gfunbasic.m*. Another interesting option offered in FERUM 4.0 is that the limit-state function can be defined through a user-provided Matlab function, which calls a third-party software, such as a Finite Element code. Such merging of FERUM with problem-specific external codes was made in various applications, such as probabilistic buckling (Dubourg *et al.* 2009) and crack propagation (Nespurek *et al.* 2006). For controlling such external codes, extra variables are provided to FERUM through the structure variable *femodel* and the user must create an application-specific function. One more option available in FERUM 4.0 is that it takes advantage of gradients *w.r.t.* all or part of basic variables, when available from third-party software. This proves to be very useful when limit-state functions involve very computationally demanding numerical models, as it avoids tedious estimations by finite differences.

2.4 *Vectorized / distributed computing*

A major change brought to FERUM 4.0 is that calls to the limit-state function g can be evaluated in a distributive manner, as opposed to the sequential manner of the previous version. Every algorithm implemented in FERUM was revisited, so as to send multiple calls to g, whenever possible.

If one thinks of FE-based MCS on a multiprocessor computer, the strategy consists in sending calls to the FE code in batches, the number of jobs in each batch being equal to the number of available CPUs. This strategy is known as distributed computing, see *e.g.* examples of applications in (Nespurek *et al.* 2006, Dubourg *et al.* 2009). The number of jobs sent simultaneously is tuned through the variable *analysisopt.block_size*. Such an option is available in FERUM, assuming that the user has a suitable computer platform and all the necessary tools to create, send and post-process multiple jobs (scripting language such as Perl, queuing systems such as OpenPBS on Linux, job schedulers, …). The function in charge of the job allocation is obviously application-specific and is called by *gfun.m*.

Based on the same developments of FERUM algorithms, it is also possible to send multiple calls to a user-defined Matlab limit-state function written in a vectorized manner. Vectorized calculations, in the Matlab sense, eliminate the need to cycle through nested loops and thus run much faster because of the way Matlab handles vectors internally. The principle is similar to distributed computing, the difference being that the multiprocessor computer is virtually replaced by a single computer which can handle a number of runs simultaneously (this maximum number being directly dependent on the memory available on the computer). Here again, the maximum number of runs sent simultaneously is controlled through *analysisopt.block_size* variable.

For illustration purpose, on an Intel T7800 2.6GHz dual core CPU with 4Gb RAM, a MCS takes 31 min with $1.5 \cdot 10^9$ samples for a basic $g = r - s$ problem, where R and S are normal random variables, in a vectorized manner (FERUM 4.0), as opposed to 6 days 15 hours in the sequential manner (FERUM 3.1).

2.5 *Random fields, system reliability*

Structure variables created for random field problems by B. Sudret and for system reliability analysis by J. Song are still compatible with the new version of FERUM, though not tested extensively. An illustration of the use of random fields can be found in (Dubourg *et al.* 2009).

3 RELIABILITY: APPROXIMATE METHODS

This section briefly presents approximate methods implemented in FERUM 4.0, namely FORM and SORM. It also details some improvements brought to FORM sensitivities.

3.1 *FORM*

First Order Reliability Method (FORM) aims at using a first order approximation of the limit-state function in the standard space at the so-called Most Probable Point (MPP) of failure P^* (or design point), which is the limit-state surface closest point to the origin. Finding the coordinates $\mathbf{u}^*$ of the MPP consists in solving the following constrained optimization problem:

$$\mathbf{u}^* = \arg\min \left\{ \|\mathbf{u}\| \;\middle|\; g\left(x\left(\mathbf{u}\right)\right) = G\left(\mathbf{u}\right) = 0 \right\} \qquad (4)$$

Once the MPP P^* is obtained, the Hasofer and Lind reliability index β is computed as $\beta = \boldsymbol{\alpha}^{\mathrm{T}}\mathbf{u}^*$ where $\boldsymbol{\alpha} = -\nabla_{\mathbf{u}} G(\mathbf{u}^*) / \|\nabla_{\mathbf{u}} G(\mathbf{u}^*)\|$ is the negative normalized gradient vector at the MPP P^*. It represents the distance from the origin to the MPP in the standard space. The first-order approximation of the failure probability is then given by $p_{f1} = \Phi(-\beta)$, where $\Phi(\bullet)$ is the standard normal cdf.

As in FERUM 3.1, the new version is based on the iHLRF algorithm, see (Zhang & Der Kiureghian 1994) for further details. In order to take advantage of distributed computing, g-calls required for gradient evaluations by finite differences at a specific point of the standard space are sent in a single batch. The same technique is applied to step size evaluation with Armijo rule, where all corresponding g-calls are sent simultaneously.

Search for multiple MPPs such as described in (Der Kiureghian & Dakessian 1998) is also implemented in FERUM 4.0. Figure 1 illustrates the use

of this method, applied to a parabolic limit-state function (Der Kiureghian & Dakessian 1998):

$$g(\mathbf{x}) = g(x_1, x_2) = b - x_2 - \kappa (x_1 - e)^2 \tag{5}$$

where $b = 5$, $\kappa = 0.5$ and $e = 0.1$. Both variables x_1 and x_2 are independent and identically distributed (i.i.d.) standard normal random variables.

This problem is characterized by two MPPs at similar distances from the origin and basic FORM algorithm results are therefore not valid. Results in Figure 1 are obtained with parameter values recommended in (Der Kiureghian & Dakessian 1998), *i.e.* $\gamma = 1.1$, $\delta = 0.75$ and $\varepsilon = 0.5$.

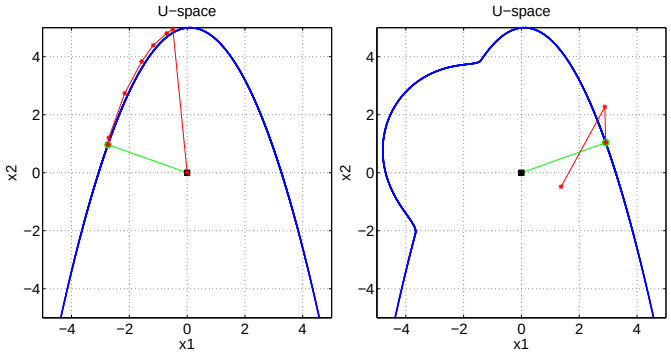

Figure 1 . Parabolic limit-state function: FORM with search for multiple design points.

3.2 *FORM sensitivities and importance measures*

In addition to the reliability index β and the MPP coordinates coming from a FORM analysis, the user may use FERUM 4.0 to calculate the sensitivities of β (or of the failure probability p_f) to distribution parameters $\boldsymbol{\theta}_f$ or to limit-state function parameters $\boldsymbol{\theta}_g$.

For instance, the sensitivity of β *w.r.t.* $\boldsymbol{\theta}_f$ reads:

$$\nabla_{\boldsymbol{\theta}_f} \beta = \mathbf{J}_{\mathbf{u}^*, \boldsymbol{\theta}_f} \left(\mathbf{x}^*, \boldsymbol{\theta}_f \right)^{\mathrm{T}} \boldsymbol{\alpha} \tag{6}$$

where $\mathbf{J}_{\mathbf{u}^*, \boldsymbol{\theta}_f} \left(\mathbf{x}^*, \boldsymbol{\theta}_f \right) = \left[\partial u_i / \partial \theta_{fj} \right]_{\mathbf{x}^*}$

The Jacobian of the transformation is obtained by differentiating Equation (3) *w.r.t.* $\boldsymbol{\theta}_f$ parameters:

$$\frac{\partial \mathbf{u}}{\partial \theta_f} = \mathbf{L}_0^{-1} \frac{\partial \mathbf{z}}{\partial \theta_f} + \frac{\partial \mathbf{L}_0^{-1}}{\partial \theta_f} \mathbf{z} \tag{7}$$

In FERUM 4.0, sensitivities *w.r.t.* distributions parameters $\boldsymbol{\theta}_f$ are evaluated based on both terms of Equation (7), as opposed to FERUM 3.1 which only uses the first term. Sensitivities to correlation are based on the second term of this expression only, as the first one vanishes (Bourinet & Lemaire 2008). Sensitivities are evaluated numerically with the same integration scheme as the one used for obtaining $\mathbf{R}_0$ matrix and it is required to differentiate the Cholesky decomposition algorithm in a step-by-step manner. Examples of application are given in reference (Bourinet & Lemaire 2008).

3.3 *SORM*

As in the previous version, FERUM offers two ways for computing a second order approximation of the failure probability. The first method consists in determining the principal curvatures and directions, by solving an eigenproblem involving the Hessian of the limit-state function. The Hessian is computed by finite differences, the perturbations being set in the standard normal space. All calls to the limit-state function corresponding to perturbated points are potentially sent simultaneously, as being all independent from each other. The second method consists in approximating the limit-state function by a piecewise paraboloid surface (Der Kiureghian *et al.* 1987). This approximate surface must be tangent to the limit-state at the design point and coincides with the limit-state at two points on each axis selected on both sides of the origin. It is built iteratively, with a limited number of iterations and all calls to the limit-state function, at each iteration, are potentially sent simultaneously as well. This second approach is advantageous for slightly-noisy limit-state functions (*e.g.* involving a FE code), for problem with a large number of random variables or when the computation of curvatures turns out to be problematic. In both methods, the SORM approximation of the failure probability p_{f2} is computed with Breitung or Tvedt formulae, as in FERUM 3.1.

An example is taken from reference (Der Kiureghian & De Stefano 1990) to illustrate SORM application. A two d.o.f. primary-secondary system with uncertain damped oscillators is considered under a white-noise base excitation. This problem is characterized by a highly nonlinear limit-state around a single design point. The limit-state function is given in Equation (8) and the basic random variables of this problem are gathered in Table 1.

$$\begin{aligned} g(\mathbf{x}) &= F_s - 3k_s \sqrt{\frac{\pi S_0}{4\zeta_s \omega_s^3}} \times \dots \\ &\dots \sqrt{\frac{\zeta_a \zeta_s}{\zeta_p \zeta_s \left(4\zeta_a^2 + \theta^2\right) + \gamma \zeta_a^2} \frac{\left(\zeta_p \omega_p^3 + \zeta_s \omega_s^3\right) \omega_p}{4\zeta_a \omega_a^4}} \end{aligned} \tag{8}$$

where $\begin{cases} \omega_p = \sqrt{k_p / m_p} \quad , \quad \omega_s = \sqrt{k_s / m_s} \quad , \\ \omega_a = \left(\omega_p + \omega_s\right)/2 \quad , \quad \zeta_a = \left(\zeta_p + \zeta_s\right)/2 \\ \gamma = m_s / m_p \quad , \quad \theta = \left(\omega_p - \omega_s\right)/\omega_a \end{cases}$

With iHLRF acceptance tolerances $e_1 = 10^{-3}$ and $e_2 = 5 \cdot 10^{-3}$ (Zhang & Der Kiureghian 1995), FORM takes 294 iterations to converge (2646 calls to the limit-state function). This FORM analysis is based on a step size with a fixed value of 0.025, since convergence cannot be obtained using Armijo rule. FORM result is $p_{f1} = 3.86 \cdot 10^{-5}$. Based on 44 extra calls, SORM results with curvature fitting (first method) are obtained and we find $p_{f2} = 4.15 \cdot 10^{-6}$

(improved Breitung formula). For comparison purpose, the failure probability obtained by averaging the results of 500 subset simulations (see Section 4.3), each of them using 300000 limit-state function evaluations per subset step, gives a reference value $p_f = 4.18 \cdot 10^{-6}$. This proves here that SORM is rather suitable for such a problem with a single MPP.

Table 1. Stochastic model

variable	distribution	mean	c.o.v.
m_p	lognormal	1.5	0.1
m_s		0.01	0.1
k_p	lognormal	1	0.2
k_s		0.01	0.2
ζ_p	lognormal	0.05	0.4
ζ_s		0.02	0.5
F_s	lognormal	24.5	0.1
S_0	lognormal	100	0.1

4 RELIABILITY: SIMULATION METHODS

This section briefly presents simulation methods implemented in FERUM 4.0: well-known simulation methods such as crude Monte Carlo Simulation (MCS) and Importance Sampling (IS), Directional Simulation (DS) and a variance reduction technique known as Subset Simulation (SS).

4.1 *Monte Carlo Simulation / Importance Sampling*

Equation (1) is rewritten as follows:

$$p_f = \int_{D_{f_\mathbf{X}}} I(\mathbf{x}) f_\mathbf{X}(\mathbf{x}) \mathrm{d}\mathbf{x} = E_{f_\mathbf{X}}\left[I(\mathbf{X}) \right] \tag{9}$$

where $D_{f\mathbf{X}}$ represents the integration domain of joint pdf $f_\mathbf{X}(\mathbf{x})$, $I(\bullet)$ is an indicator function which equals 1 if $g(\mathbf{x}) \le 0$, and 0 otherwise, and $E_{f\mathbf{X}}[\bullet]$ denotes the mathematical expectation *w.r.t.* joint pdf $f_\mathbf{X}(\mathbf{x})$.

The expectation in Equation (9) is estimated in a statistical sense for MCS. The *u*-space is randomly sampled with N independent samples u_j, $j = 1, \ldots, N$. These N samples are then transformed to the *x*-space $x_j = x(u_j)$ and an unbiased estimate of p_f is finally obtained from the sample mean of $q_j = I(x_j)$. Note that a standard deviation is also obtained for this sample, providing useful information regarding the accuracy of the estimated value of p_f. It must be stressed out here that MCS requires a high computational effort (large N) for small failure probabilities and a number of variance reduction techniques have been proposed in the past to lower this computational effort.

One of these variance reduction techniques is known as Importance Sampling (IS). Equation (1) is rewritten now in the following form:

$$p_f = \int_{D_h} I(\mathbf{x}) \frac{f_\mathbf{X}(\mathbf{x})}{h(\mathbf{x})} h(\mathbf{x}) \mathrm{d}\mathbf{x} = E_h\left[I(\mathbf{X}) \frac{f_\mathbf{X}(\mathbf{X})}{h(\mathbf{X})} \right] \tag{10}$$

where h is called a sampling density. For IS analysis, it is usual to take $h(\mathbf{x}) = h(x(\mathbf{u})) = \varphi_n(\mathbf{u}-\mathbf{u}^*)$ where φ_n is the n-dimensional standard normal pdf and $\mathbf{u}^*$ is the vector of MPP coordinates coming from a previous FORM analysis. Note that p_f is now obtained from the sample mean of $q_j = I(x_j) f_\mathbf{X}(\mathbf{x}_j) / h(\mathbf{x}_j)$.

FERUM 4.0 features both methods and calls to the limit-state function are sent in a distributed manner, the maximum number of jobs sent being adjusted by the variable *analysisopt.block_size*.

4.2 *Directional Simulation*

The n-dimensional normal vector $\mathbf{U}$ is expressed as $\mathbf{U} = R\mathbf{A}$, $R \ge 0$, where R^2 is a chi-square distributed random variable with n degrees of freedom (d.o.f.), independent of the random unit vector $\mathbf{A}$, which is uniformly distributed on the n-dimensional unit sphere Ω_n. The failure probability p_f can be written as follows, conditioning on $\mathbf{A} = \mathbf{a}$ (Bjerager 1988):

$$p_f = \int_{\mathbf{a} \in \Omega^n} P\left[G(R\mathbf{A}) \le 0 \,|\, \mathbf{A} = \mathbf{a} \right] f_\mathbf{A}(\mathbf{a}) \mathrm{d}\mathbf{a} \tag{11}$$

where $f_\mathbf{A}(\mathbf{a})$ is the uniform density of $\mathbf{A}$ on the unit sphere.

Practically, a sequence of N random direction vectors $\mathbf{a}_j = \mathbf{u}_j / \|\mathbf{u}_j\|$, $j = 1, \ldots, N$, is generated first, then $r_j = \{ r \,|\, G(r\mathbf{a}_j) = 0 \}$ are found iteratively and p_f is finally estimated from the following expression:

$$\hat{p}_f = \frac{1}{N} \sum_{j=1}^{N} \left[1 - \chi_n^2\left(r_j^2 \right) \right] \tag{12}$$

where χ_n^2 is the chi-square cdf with n d.o.f.

In FERUM 4.0, a slightly modified version of this algorithm is proposed. Instead of generating random directions on the unit sphere, it is proposed to divide it into N evenly distributed points, in a deterministic manner, in order to gain an improved accuracy at a given computational cost. Intersections with the limit-state function along each direction are found in a distributive manner, based on a vectorized version of *fzero.m* Matlab function. It is worth noting that DS looses efficiency as the number of random variables n increases.

Table 2 shows the results obtained on the example of reference (Der Kiureghian & Dakessian 1998) presented in Section 3.1. These results must be compared to a reference value of $3.0158 \cdot 10^{-3}$, obtained by averaging the results of 500 subset simulations, each of them using 200000 limit-state function evaluations per subset step. Results appear to be fairly good for small numbers of directions N.

Table 2. Directional Simulation results

N	N_{call}	p_f
10	101	$2.9648 \cdot 10^{-3}$
20	201	$3.0162 \cdot 10^{-3}$
50	551	$3.0163 \cdot 10^{-3}$
100	1101	$3.0163 \cdot 10^{-3}$

4.3 *Subset Simulation*

Starting from the premise that the failure event $F = \{ g(\mathbf{x}) \le 0 \}$ is a rare event, S.-K. Au and J.L. Beck (Au & Beck 2001) proposed to estimate $P(F)$ by means of more frequent intermediate conditional failure events $\{F_i\}_{i=1..m}$ (called subsets) so that $F_1 \supset F_2 \supset ... \supset F_m = F$. The m-sequence of intermediate conditional failure events is selected so that $F_i = \{ g(\mathbf{x}) \le y_i \}$, where y_i's are decreasing values of the limit state function and $y_m = 0$. As a result, the failure probability $p_f = P(F)$ is expressed as a product of the following m conditional probabilities:

$$p_f = P(F) = P(F_m) = P(F_m | F_{m-1}) P(F_{m-1}) = ... = P(F_1) \prod_{i=2}^{m} P(F_i | F_{i-1}) \quad (13)$$

Each subset event F_i (and the related threshold value y_i) is determined so that its corresponding conditional probability equals a sufficiently large value α, in order to be efficiently estimated with a small number of simulations (in practice $\alpha \approx 0.1$-0.2). The first threshold y_1 is obtained by a crude MCS, so that $P(F_1) \approx \alpha$. For further thresholds, new sampling points corresponding to $\{F_i \mid F_{i-1}\}$ conditional events are obtained from Markov Chains Monte Carlo (MCMC), based on a modified Metropolis-Hastings algorithm, see Figure 2 for illustration of main steps.

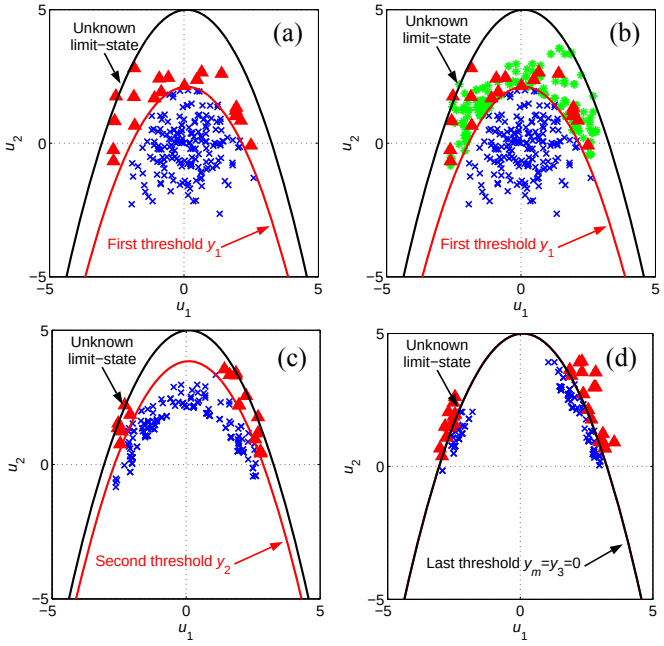

Figure 2 . Main steps of Subset Simulation algorithm.

Figure 3 shows how the coefficient of variation (c.o.v.) of the failure probability of subset simulations varies in terms of the number of simulations per subset step N_s, on the two d.o.f. primary-secondary system presented at Section 3.3. For each N_s value, the c.o.v. is estimated empirically by replicate applications of subset simulations (here, 500 times). For comparison purpose, Figure 3 also gives a lower (respectively upper) bound estimate, which assumes uncorrelated (respectively fully correlated) conditional probability estimates (Au & Beck 2001).

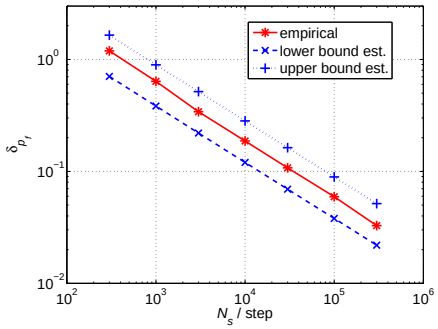

Figure 3 . Two d.o.f. primary-secondary system: Subset results.

5 GLOBAL SENSITIVITY ANALYSIS

Global sensitivity analysis aims at quantifying the impact of the variability in each (or group of) input variates on the variability of the output of a model in apportioning the output model variance to the variance in the input variates. Sobol' indices (Sobol 1993) are the most usual global sensitivity measures. They can be evaluated in FERUM 4.0.

We consider here a model given by:

$$Y = g(\mathbf{X}) = g(X_1, X_2, ..., X_n) \quad (14)$$

where $\mathbf{X} = (X_1, X_2, ..., X_n)$ is a vector of n independent random input variates, g is a deterministic model and Y is a scalar random output.

In order to determine the importance of each input variate, we consider how the variance of the output Y decreases when variate X_i is fixed to a given x_i^* value:

$$V(Y | X_i = x_i^*) \quad (15)$$

where $V(\bullet)$ denotes the variance function.

Since x_i^* value is unknown, we take the expectation of Equation (15) and, by virtue of the law of total variance, we can write:

$$V(E[Y | X_i]) = V(Y) - E[V(Y | X_i)] \quad (16)$$

The global sensitivity index of the first order is defined as follows, for $i = 1, \ldots, n$:

$$S_i = \frac{V(E[Y | X_i])}{V(Y)} = \frac{V_i}{V} \quad (17)$$

Indices of higher orders are defined in a similar manner, *e.g.* for the second order:

$$S_{ij} = \frac{V(E[Y | X_i, X_j]) - V_i - V_j}{V(Y)} = \frac{V_{ij}}{V} \quad (18)$$

First order indices inform about the influence of each variate taken alone whereas higher order indices account for possible influences between various

parameters. Total sensitivity indices are also usually introduced. They express the total sensitivity of Y variance to X_i input, including all interactions that involve X_i:

$$S_{T_i} = \sum_{i \subset k} S_k \tag{19}$$

where $i \subset k$ denotes the set of indices containing i.

From a computational viewpoint, Sobol' indices can be assessed using Crude Monte Carlo (CMC) or Quasi-Monte Carlo (QMC) simulations. This latter technique is based on low-discrepancy sequences, which usually outperform CMC simulations in terms of accuracy, at a given computational cost. Both CMC and QMC methods are implemented in FERUM 4.0. Another available option consists in building a Support Vector surrogate function, by regression on a sample set of well-chosen points. This option based on statistical learning theory proves to be a rather cost efficient technique for evaluating sensitivities of models of moderate complexity.

For illustration, Sobol' indices are evaluated on the following example taken from reference (Nie & Ellingwood 2004). The model is given by:

$$\begin{aligned} g(\mathbf{x}) = g(x_1, x_2, x_3) = \\ -\left(x_1^2 + x_2^2 + x_3^2 - 2x_1x_2 - 2x_2x_3 - 2x_3x_1\right)/2 \\ -\left(x_1 + x_2 + x_3\right)/\sqrt{3} + 3 \end{aligned} \tag{20}$$

where x_i, $i = 1, 2, 3$, are i.i.d. standard normal random variables.

Sobol's indices obtained by QMC simulations are given in Table 3. This model is sensitive to interactions between variables, since second order indices are not equal to zero. It is worth noting that, in this specific example, indices obtained from a SVR surrogate based on a set of 100 points are as accurate as those estimated by QMC with 20000 points.

Table 3. Sobol' indices computed with 20000 simulation points using Quasi-Monte Carlo (QMC) simulation method

1st order	mean	stdv	2nd order	mean	stdv
S_1	0.1499	0.0147	S_{12}	0.1826	0.0192
S_2	0.1453	0.0124	S_{13}	0.1866	0.0179
S_3	0.1494	0.0130	S_{23}	0.1866	0.0143
3rd order	**mean**	**stdv**	**Total**	**mean**	**stdv**
S_{123}	≈0	0.0146	S_{T1}	0.5186	0.0132
			S_{T2}	0.5141	0.0139
			S_{T3}	0.5222	0.0121

6 RELIBIALITY-BASED OPTIMIZATION

FERUM 4.0 now offers Reliability-Based Optimization (RBO) capabilities. The problem of interest reads, in its most basic and general formulation:

$$\min_{\boldsymbol{\theta}} \quad c(\boldsymbol{\theta}) \quad \text{s.t.} \begin{cases} f_1(\boldsymbol{\theta}), \ldots, f_{q-1}(\boldsymbol{\theta}) \le 0 \\ f_q(\mathbf{x}, \boldsymbol{\theta}) = \beta_t - \beta(\mathbf{x}, \boldsymbol{\theta}) \le 0 \end{cases} \tag{21}$$

where:

- $\boldsymbol{\theta}$ stands for the design variables of the problem, either purely deterministic variables $\boldsymbol{\theta}_g$ or distribution parameters $\boldsymbol{\theta}_f$,
- $c(\boldsymbol{\theta})$ is the cost function to be minimized,
- $f_1(\boldsymbol{\theta}) \ldots f_{q-1}(\boldsymbol{\theta})$ is a vector of deterministic constraints over the design variables $\boldsymbol{\theta}$,
- $f_q(\mathbf{x}, \boldsymbol{\theta})$ is the reliability constraint enforcing the respect of the design rule referred to as the limit-state function and considering the uncertainty to which some of the model parameters $\mathbf{x}$ are subjected to. β_t is the targeted safety index.

One way to answer the problem in Equation (21) consists in a brute-force outer optimization loop over the reliability evaluation, here termed "nested bi-level approach". This might be computational expensive in the case of simulation-based methods such as MCS and DS, as addressed in (Royset & Polak 2004) and (Royset & Polak 2007) respectively. However, if based on FORM, this brute-force method gives a solution within a reasonable amount of calls to the limit-state function.

The outer optimization loop makes use of the Polak-He optimization algorithm (Polak 1997) and requires the gradients of both cost and constraints functions, which themselves require the gradient of the reliability index *w.r.t.* design variables $\boldsymbol{\theta}$.

Previous RBO applications of the Polak-He algorithm showed that its rate of convergence highly depends on the order of magnitude of design parameters, cost and constraints functions. In FERUM 4.0, all these values are normalized at each Polak-He iteration, thus improving and ensuring convergence, whatever the initial scaling of the problem in Equation (21). Convergence to an optimum is assumed to be obtained when the cost function has reached a stable value and all the constraints are satisfied, *i.e.* $f_1(\boldsymbol{\theta}) \ldots f_q(\boldsymbol{\theta}) \le 0$.

For illustration purpose, RBO is applied to a spherical tank under internal pressure p. Failure is defined when the von Mises stress exceeds the yield strength σ_y of the elastic constitutive material. Variables r_0 and r_1 denoting the internal and external radii respectively, the limit-state function thus reads:

$$g(\mathbf{x}) = g(p, \sigma_y, r_0, r_1) = \sigma_y - \frac{3p}{2} \frac{r_1^3}{r_1^3 - r_0^3} \tag{22}$$

The proposed optimization problem enunciates:

$$\begin{aligned} &\min_{(r_0, r_1)} c(r_0, r_1) = V_{\text{sphere}}(r_0, r_1) = \frac{4\pi}{3}\left(r_1^3 - r_0^3\right) \\ &\text{s.t.} \begin{cases} 40 - r_0 \le 0 \quad (i) & r_0 - r_1 \le 0 \quad (iii) \\ r_1 - 150 \le 0 \quad (ii) & \beta_t - \beta(r_0, r_1) \le 0 \quad (iv) \end{cases} \end{aligned} \tag{23}$$

The internal pressure p and the yield strength σ_y are considered as statistically independent lognormal random variables with means and standard deviations equal to (130 MPa, 8 MPa) and (300 MPa,

20 MPa) respectively, whereas the design variables r_0 and r_1 are supposed to be deterministic. The initial design set as well as the RBO results for various reliability targets β_t are provided in Table 4. The last column example ($\beta_t = 5$) exhibits an exponential increase of the cost function *w.r.t.* iterations (it is worth noting that the initial cost is amplified by a factor greater than 700 after 7 iterations). This example demonstrates the ability of the RBO algorithm to adapt itself very rapidly to fulfill the reliability constraint, which is not reachable here. This example is in fact characterized by an upper bound reliability index $\beta_{max} \approx 4.7494$, which corresponds to the limit-state function in Equation (22) when r_1 tends toward infinity. From results in Table 4, it is observed that the deterministic constraints are not always accurately satisfied. This is due to the rescaling procedure used at each Polak-He iteration.

Table 4. RBO results

β_t	Initial	3.28*	2.0	5.0	5.0**
N_{iter}	–	7	7	7	7
β	3.2761	3.2804	2.0003	4.5878	4.7484
c / c_0	1	0.5154	0.2588	4.4117	729.33
r_0 (mm)	50	40.05	40.01	38.47	39.35
r_1 (mm)	100	80.17	66.23	157.64	860.97
N_{FORM}	1	29	29	101	42
N_{call}	30	907	898	3949	1648

*: iso-reliability optimization **: constraint (*ii*) released

The proposed RBO algorithm is part of FERUM 4.0 and it uses all distributed features of FORM. It also allows optimization *w.r.t.* distribution parameters $\boldsymbol{\theta}_f$, *e.g.* mean values of r_0 and r_1, if the two radii are considered as random variables too.

7 CONCLUSION

This paper briefly presents new features available in FERUM 4.0. For more details on each method, the reader may refer to more comprehensive papers referenced in the text. The objectives of this new version are similar to those of the previous one: providing students and researchers with a very flexible tool, which facilitates learning and developing new techniques for research purposes. The Matlab source files of this new version are downloadable both at IFMA (www.ifma.fr/FERUM/) and UC Berkeley (www.ce.berkeley.edu/FERUM/).

8 ACKNOWLEDGEMENTS

The development of this code has benefited from the work of many individuals. The first author wants to gratefully acknowledge Prof. A. Der Kiureghian for inspiring him such developments which started in 2001, all the researchers at UCB who made contributions to previous versions of FERUM and former/current IFMA/LaMI Ph.D. students who contributed to this new version.

REFERENCES

Au, S.-K. & Beck, J.L. 2001. Estimation of small failure probabilities in high dimension by subsets simulations. *Probabilistic Engineering Mechanics* 16(4): 263-277.

Bjerager, P. 1988. Probability integration by directional simulation. *Journal of Engineering Mechanics* 114(8): 1285-1302.

Bourinet, J.-M. & Lemaire, M. 2008. FORM sensitivities to correlation: Application to fatigue crack propagation based on Virkler data. In P.K Das (ed), *Proc. of the 4th International ASRANet Colloquium, Athens, Greece, June 25-27, 2008*.

Der Kiureghian, A., Lin, H.-Z. & Hwang, S.-J. 1987. Second-order reliability approximations. *Journal of Engineering Mechanics* 113(8): 1208-1225.

Der Kiureghian, A. & De Stefano, M. 1990. An efficient algorithm for second-order reliability analysis, report no. UCB/SEMM-90/20. *Report No. UCB/SEMM-90/20, Department of Civil and Environmental Engineering, University of California, Berkeley.*

Der Kiureghian, A. & Dakessian, T. 1998. Multiple design points in first and second-order reliability, *Structural Safety* 20(1): 37-49.

Der Kiureghian, A. & Haukaas, T. & Fujimura, K. 2006. Structural reliability software at the University of California, Berkeley. *Structural Safety* 28(1-2): 44-67.

Ditlevsen, O. & Madsen, H.O. 2007. *Structural reliability methods*. Internet edition 2.3.7.

Dubourg, V., Noirfalise C., Bourinet, J.-M. & Fogli, M. 2009. FE-based reliability analysis of the buckling of shells with random shape, material and thickness imperfections. *Proc. of ICOSSAR 2009, Osaka, Japan, September 13-17, 2009*.

Liu, P.-L. & Der Kiureghian, A. 1986. Multi-variate distribution models with prescribed marginals and covariance, *Probabilistic Engineering Mechanics* 1(2): 105-112.

Nespurek, L., Bourinet, J.-M., Gravouil, A. & Lemaire, M. 2006. Some approaches to improve the computational efficiency of the reliability analysis of complex crack propagation problems. In P.K Das (ed), *Proc. of the 3rd International ASRANet Colloquium, Glasgow, U.K., July 10-12, 2006*.

Nie, J. & Ellingwood, B.R. 2004. A new directional simulation method for system reliability, part II: application of neural networks. *Probabilistic Engineering Mechanics* 19: 437-447.

Pellissetti, M.F. & Schuëller, G.I. 2006. On general purpose software in structural reliability - An overview. *Structural Safety* 28: 3-16.

Polak, E. 1997. *Optimization: Algorithms and Consistent Approximations*. Springer-Verlag New York Inc.

Royset, J.O. & Polak, E. 2004. Reliability-based optimal design using sample average approximations. *Probabilistic Engineering Mechanics* 19(4): 331-343.

Royset, J.O. & Polak, E. 2007. Extensions of stochastic optimization results to problems with system failure probability functions, *Journal of Optimization Theory and its Application* 132(2): 1-18.

Sobol', I.M. 1993. Sensitivity estimates for nonlinear mathematical models. Mathematical *Modelling and Computational Experiments* 1: 407- 414.

Zhang, Y. & Der Kiureghian, A. 1994. Two improved algorithms for reliability analysis. In R. Rackwitz, G. Augusti & A. Borri (eds), *Reliability and Optimization of Structural Systems; Proc. of the 6th IFIP WG 7.5 Working Conf. on Reliability and Optimisation of Structural Systems, 1994*: 297-304.