AN INTRODUCTION TO RELIABILITY ANALYSIS

Every real-life system has a **capacity** (or resistance) for doing something and is subjected to some sort of **demand** (or load). Both capacity and demand may change depending on various factors and those factors can be viewed as random variables. When demand exceeds the capacity of the system, a **system failure** is reached, given that **the system cannot offer the service that it was designed to provide**. While some system's failure can be permanent, other system's failure can be only temporary. Here are a few examples of real-life systems and some of the factors that could influence their capacities and demands:

System	Description	Some Factors
Highway	A highway's capacity and demand can be measured in terms of vehicles per hour. This system is designed to facilitate transportation. When it fails, people refer to the condition as a traffic jam and it can go back to working condition after the rush hour is over.	-Capacity may depend on the weather. -Demand may depend on the price of the highway's tolls.
Web Site Server	A web site's capacity and demand can be measured in terms of hits per second. This system is designed to provide some sort of digital data. When it fails, the system may require some sort of human intervention in order to go back up.	- Capacity may depend on the type of content being requested from the web site (video, images, etc). - Demand may depend on the popularity of the web site's content.
Building	A building's capacity and demand can be measured in terms of units of force per unit area (N/m^2, lb/ft^2, etc). This system is designed to provide a variety of services (housing, offices, commerce, etc). When it fails, the failure is usually permanent and requires some sort of intervention to be re-established.	- Capacity may depend on the quality of the construction process. - Demand may depend on the distribution of the loads.

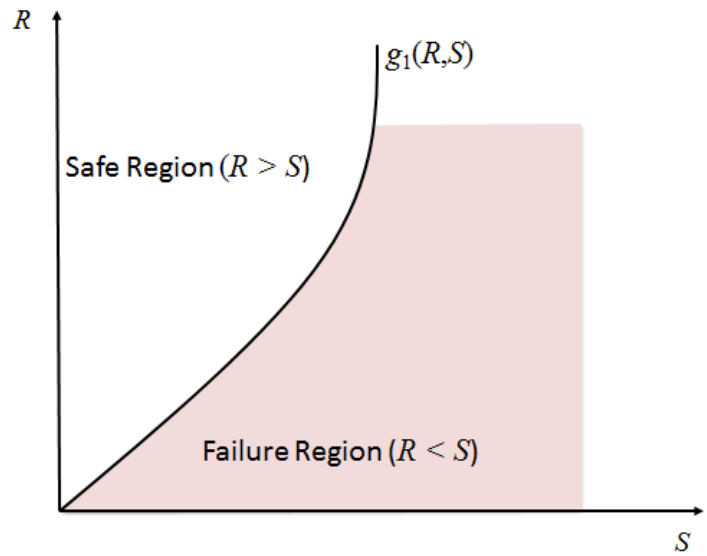
All of the factors mentioned in the table above have one thing in common: they can be estimated, but they are expected to vary within a certain range, with some values being more probable than others. As such, each of them constitutes a random variable.

THE LIMIT STATE FUNCTION

A system's reliability is modeled by what is known as its **limit state function, represented with the letter *g***. The limit state function returns a negative value under system failure conditions and a positive value when the system is stable. Therefore, **it can be viewed as the difference between resistance, R, and load, S**:

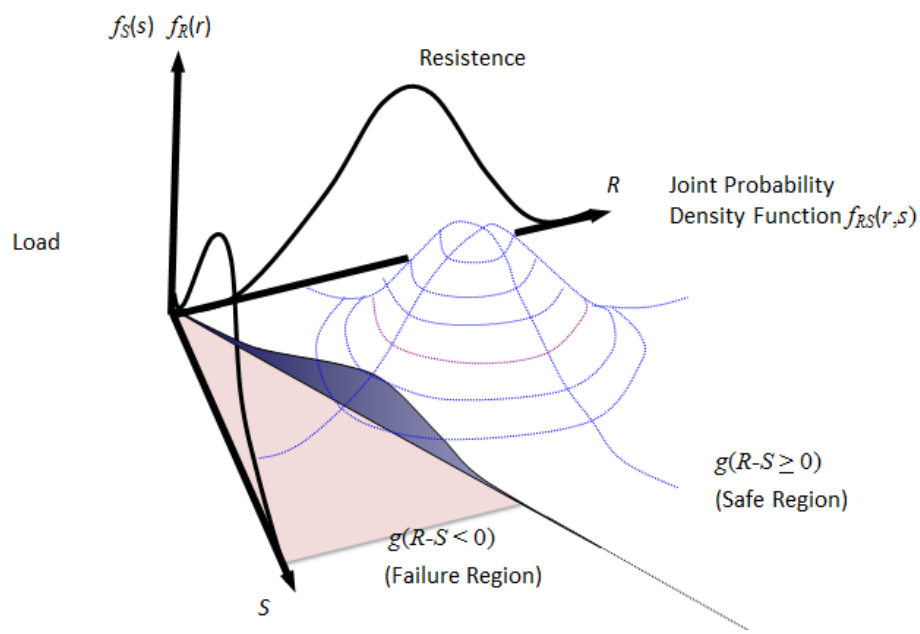
$$g(R,S) = R - S$$

The limit state function is what separates the **safe region** from the **failure region**, as depicted in the graph below:



FAILURE PROBABILITY

As previously mentioned, the **resistance** and **load** of a system both depend on random variables. Consequently, they each have a probability distribution ($f_S(S)$ and $f_R(R)$), which in turn combine to generate a **joint probability density function, $f_{RS}(R,S)$:**



If the joint probability density function is known, the probability of failure (i.e. falling in the failure region) can be calculated directly:

$$P_f = \int_{-\infty}^{\infty} \int_{-\infty}^{r \leq s} f_{RS}(R,S) dr\, ds$$

The equation above can be generalized to N different dimensions, which is important since R and S aren't usually found explicitly in a limit state function and tend to be expressed in terms of their components. Most of the times, the probability distributions of R and S are too complex because of the amount of factors affecting them, so they cannot be determined in a mathematical way and would require some sort of simulation-based analysis to be run beforehand. For simplicity, it is better to work with an N-dimensional limit state function, leading to a generalization:

$$P_f = \left[\int \dots \int f_{\vec{X}}(\vec{x}) d\vec{x} \right]_{g(\vec{x} \leqslant 0)}$$

In words, the generalization says that **the probability of failure of a system associated with an N-dimensional limit state function is equal to the probability of falling in the region where the limit state function is less than 0**.

BASIC FILE OPERATIONS

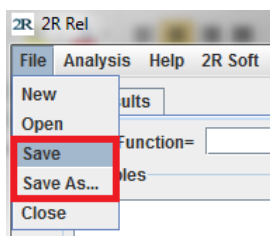
2R Rel is capable of reading and writing **.2rr** files, which contain the complete description of a specific reliability model:

1. The model's equation (limit state function).
2. All the variables and their corresponding probability distribution where appropriate, along with the pertinent parameters.
3. The correlation matrix.

These files are the means for 2R Rel users to save their work, as well as the medium of distribution of models that could be of interest to other 2R Rel users.

SAVING MODELS

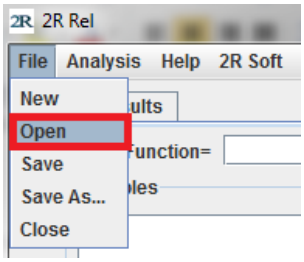
Even if a model isn't yet complete, a user can decide to save its information for later use. In order to do this, the user must navigate through the **File** menu and select the **Save** or **Save As…** option:



The difference between **Save** and **Save As…** is that, while **Save** will only ask for the file's name and destination once and will then overwrite that same file on any subsequent uses, **Save As…** will ask for the file's name and destination every time it is invoked. Thus, **Save As…** is to be used whenever a user wants to save modifications made to a file without modifying the base file.

OPENING MODELS

In order to load the information contained inside a **.2rr** file, the user must navigate through the **File** menu and select the **Open** option:

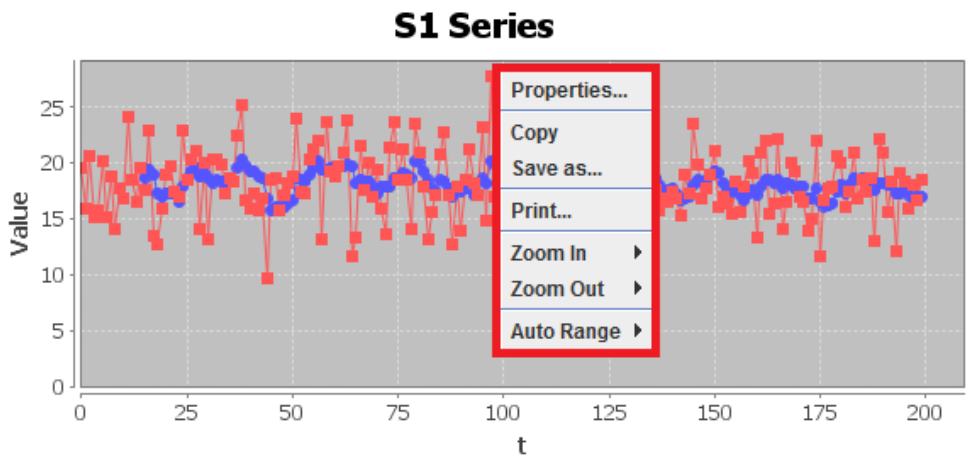


If no errors occur, the loaded model is shown in the **Main** tab (equation, variables, and correlation matrix).

BASICS

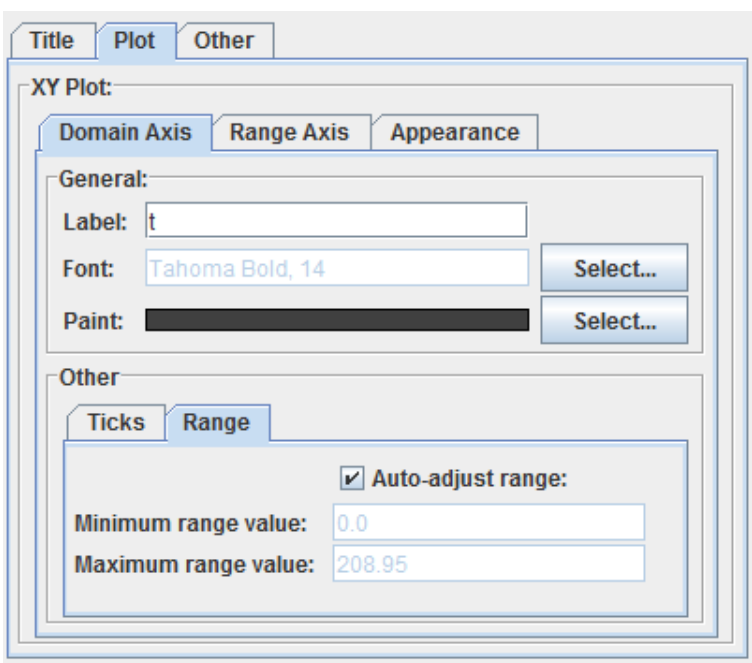
GRAPHS

All of the graphs generated in 2R Soft provide a wide array of options in the form of a context menu. **The context menu appears when you right-click over a graph:**



PROPERTIES PANE

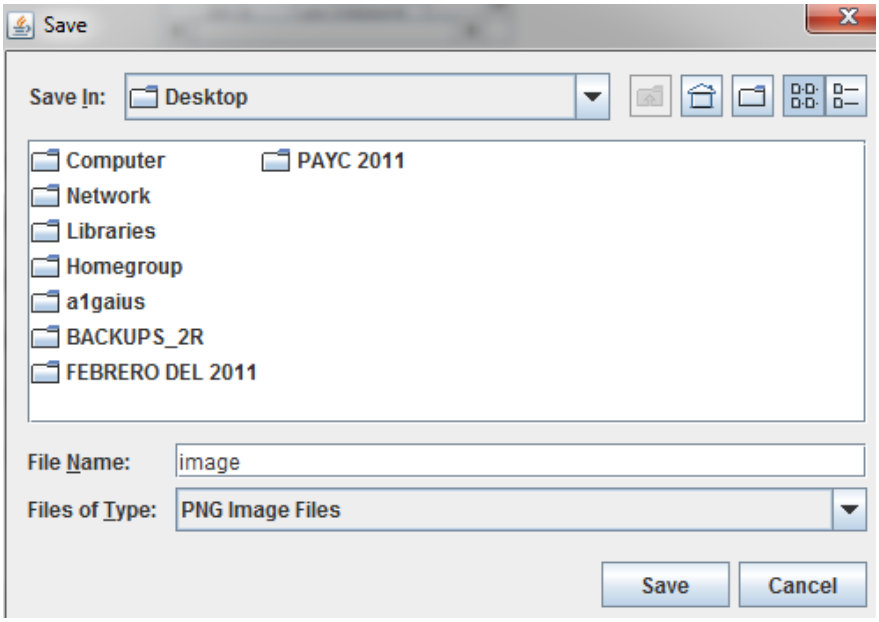
If you select the **Properties…** option, a properties pane appears. The properties pane lets you change the graph title, axis names, axis ranges, and font size.



COPY AND SAVE AS…

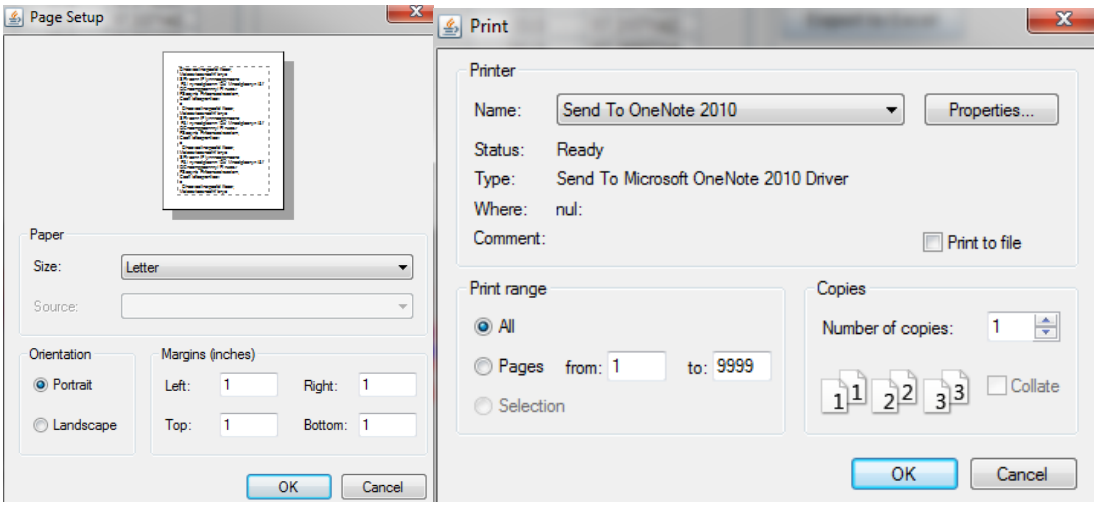
If you select **COPY**, the graph is copied to the system clipboard, so you can **PASTE** it anywhere else (Microsoft Word, Microsoft PowerPoint, etc).

Meanwhile, if **Save As…** is selected, 2R Soft will save the graph as a **PNG** image file in your hard disk after selecting the desired output folder and file name:



PRINT

The **Print** option does just that: it sends the graph to the printer of your choice (local or networked):

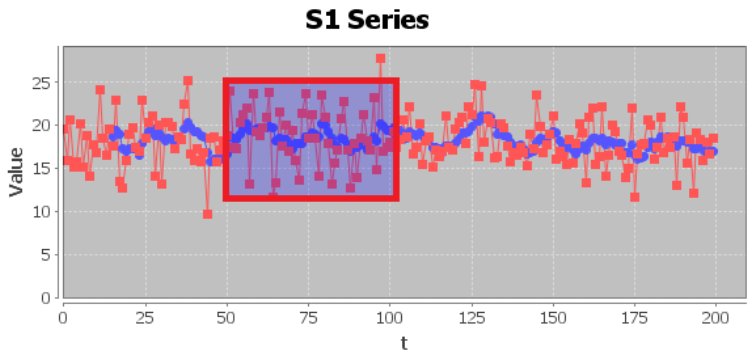


SCALE OPTIONS

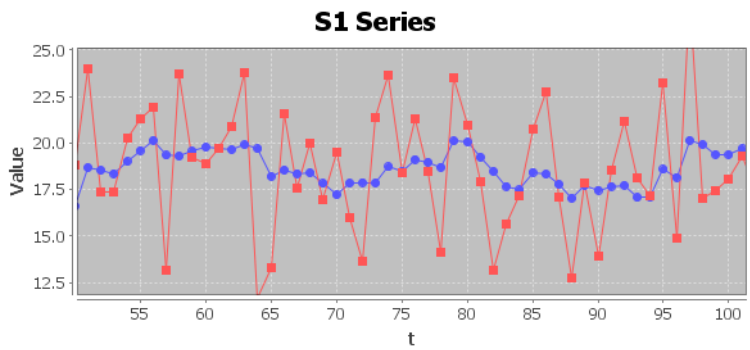
The **Auto Range**, **Zoom In**, and **Zoom Out** options are a quick way to inspect the graph. If a very specific range is needed for an axis, we highly recommend the Properties Pane.

MANUAL ZOOM IN

For user convenience, all **2R Soft graphs support manual zoom in by regions.** If you're interested in a specific region**, hold your left-click and drag the mouse** to generate a highlighted box around that region:



The end result:



EQUATION EDITOR

User-entered equations are common in 2R Soft. This section of the document explains the use of the equation editor.

FUNCTIONS AND OPERATIONS

Equations can contain the following functions and operations:

Function/Op.	Description	Usage (A and B are declared variables or numeric values)
+	Addition.	A+B Example: 4+7=11
-	Subtraction.	A-B Example: 4-7=-3
*	Multiplication.	A*B Example: 4*7=28
/	Division.	A/B Example: 6/4=1.5
^	Returns the value of the first operand raised to the power of the second operand. (Microsystems)	A^B Example: 5^3=125 Example: 5^(-3)=1/125
%	Modulo operation. Divides the value of one expression by the value of another, and returns the remainder. (MSDN)	A%B Example: 7%3=1 Example: 67%10=7
cos	Returns the trigonometric cosine of an angle. (Microsystems)	cos(A), where A is in radians Example: cos(3.14) ≈ 1.0

sin	Returns the trigonometric sine of an angle. (Microsystems)	sin(A), where A is in radians Example: sin(3.14) ≈ 0.0
tan	Returns the trigonometric tangent of an angle. (Microsystems)	tan(A), where A is in radians Example: tan(3.14) ≈ 0.0
acos	Returns the arc cosine of an angle, in the range of 0.0 through pi. (Microsystems)	acos(A), where A is the value whose arc cosine is to be returned. Example: acos(1)=0.0
asin	Returns the arc sine of an angle, in the range of -pi/2 through pi/2 (Microsystems)	asin (A), where A is the value whose arc sine is to be returned. Example: asin(0)=0.0
atan	Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. (Microsystems)	atan(A), where A is the value whose arc tangent is to be returned. Example: atan(0)=0.0
sqrt	Returns the correctly rounded positive square root of a positive real number. (Microsystems)	sqrt(A), where A is a real positive number. Example: sqrt(9)=3
sqr	Returns the value of the argument squared (to the power of 2).	sqr(A) Example: sqr(-4)=16 Example: sqr(2)=4
ln	Returns the natural logarithm (base e) of a real value. (Microsystems)	ln(A), where A is a positive real number greater than zero. Example: ln(1)=0 Example: ln(e^2)=2
min	Returns the smaller of two real values. That is, the result is the value closer to negative infinity. (Microsystems)	min(A,B) Example: min(4,9)=4 Example: min(-4,-11)=-11
max	Returns the greater of two real values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. (Microsystems)	max(A,B) Example: max(4,9)=9 Example: max(-4,-11)=-4
ceil	Returns the smallest (closest to negative infinity) real value that is not less than the argument and is equal to a mathematical integer. (Microsystems)	ceil(A) Example: ceil(9.23)=10 Example: ceil(-1.25)=-1
floor	Returns the largest (closest to positive infinity) value that is not greater than the argument and is equal to a mathematical integer (Microsystems)	floor(A) Example: floor(9.23)=9 Example: floor(-1.25)=-2
abs	Returns the absolute value of a real value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. (Microsystems)	abs(A) Example: abs(4.15)=4.15 Example: abs(-4.3)=4.3 Example: abs(0)=0
neg	Changes the sign of the value received as an argument (negative to positive or positive to negative).	neg(A) Example: neg(-1)=1 Example: neg(1)=-1 Example: neg(0)=0
rnd	Returns a pseudo-random real value between 0.0 (included) and the value received as an argument (excluded).	rnd(A) Example: rnd(50)=0,1.45,2.78,49.9 Example: rnd(-20)=0,-1.45,-18.392,-19.61
exp	Returns Euler's number e raised to the power of a real value. (Microsystems)	exp(A) Example: exp(0)=1 Example: exp(1)=e Example: exp(-2)=1/(e^2)
log	Returns the logarithm (base 10) of a real value. (Microsystems)	log(A), where A is a positive real number greater than zero. Example: log(1)=0 Example: log(10^2)=2

PROBABILITY DISTRIBUTION TYPES

When declaring a variable with a known distribution, the user can select one of many types of probability distributions.

Distribution	Description	Parameters
Beta	The *beta* distribution has shape parameters $\alpha > 0$ and $\beta > 0$ over the interval (a, b), where $a < b$. It has density: $f(x) = (x - a)^{\alpha-1}(b - x)^{\beta-1}/[B(\alpha, \beta)(b - a)^{\alpha+\beta-1}]$ for $a < x < b$, and 0 elsewhere. It has the following distribution function: $F(x) = I_{\alpha, \beta}(x) = \int_a^x(\xi - a)^{\alpha-1}(b - \xi)^{\beta-1}/[B(\alpha, \beta)(b - a)^{\alpha+\beta-1}]d\xi$, for $a < x < b$ (Simard)	Alpha – shape parameter, alpha > 0 Beta – shape parameter, beta > 0 a – lower bound of the interval b – upper bound of the interval, b > a
Binomial	The binomial distribution with parameters n and p, where n is a positive integer and $0 <= p <= 1$. Its mass function is given by: $p(x) = \mathrm{nCr}(n, x)p^x(1 - p)^{n-x} = n!/[x!(n - x)!]\ p^x(1 - p)^{n-x}$ for $x = 0, 1, 2,\ldots n$, and its distribution function is: $F(x) = \sum_{j=0}^{x}\mathrm{nCr}(n, j)\ p^j(1 - p)^{n-j}$ for $x = 0, 1, 2,\ldots n$, where $\mathrm{nCr}(n, x)$ is the number of possible combinations of x elements chosen among a set of n elements. (Simard)	p – probability of success on each trial ($0 \leq p \leq 1$) n – number of trials (integer), n > 0
Chi Square	The *chi-square* distribution with n degrees of freedom, where n is a positive integer. Its density is: $f(x) = x^{(n/2)-1}e^{-x/2}/(2^{n/2}\Gamma(n/2))$, for $x > 0$ where $\Gamma(x)$ is the gamma function. The *chi-square* distribution is a special case of the *gamma* distribution with shape parameter $n/2$ and scale parameter1/2. (Simard)	n – degrees of freedom (integer), n>0
Deterministic	Distribution that represents a constant value, *val*. Consequently: $f(x) = \begin{cases} 1\ if\ x = val \\ 0\ if\ x \neq val \end{cases}, \quad F(x) = \begin{cases} 1\ if\ x \geq val \\ 0\ if\ x < val \end{cases}$	Value – any real number
Discrete Uniform	The *discrete uniform* distribution over the integers in the range $[i, j]$. Its mass function is given by: $p(x) = 1/(j - i + 1)$ for $x = i, i + 1,\ldots, j$ and 0 elsewhere. The distribution function is: $F(x) = (\mathrm{floor}(x) - i + 1)/(j - i + 1)$ for $i <= x <= j$ and its inverse is: $F^{-1}(u) = i + (j - i + 1)u$ for $0 <= u <= 1$. (Simard)	Min. – lower bound (integer) Max. – upper bound (integer) (Max. > Min.)
Exponential	The *exponential* distribution with mean $1/\lambda$ where $\lambda > 0$. Its density is: $f(x) = \lambda e^{-\lambda x}$ for $x >= 0$, its distribution function is: $F(x) = 1 - e^{-\lambda x}$, for $x >= 0$, and its inverse distribution function is: $F^{-1}(u) = - \ln(1 - u)/\lambda$, for $0 < u < 1$ (Simard)	Lambda – rate parameter, lambda > 0
F-Distribution	The Fisher F distribution with n and m degrees of freedom, where n and m are positive integers. Its density is: $f(x) = \Gamma((n + m)/2)n^{n/2}m^{m/2}/[\Gamma(n/2)\Gamma(m/2)]x^{(n-2)/2}/(m + nx)^{(n+m)/2}$, for $x > 0$. where $\Gamma(x)$ is the gamma function (Simard)	D.O.F. 1 – the n degrees of freedom (integer), D.O.F. 1 > 0 D.O.F. 2 – the m degrees of freedom (integer), D.O.F. 2 > 0

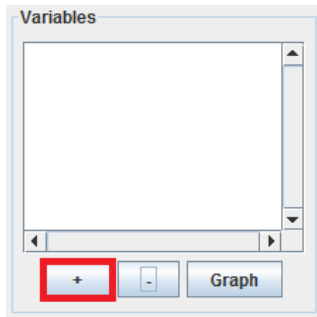
Gamma	**The *gamma* distribution with shape parameter $\alpha > 0$ and scale parameter $\lambda > 0$. The density is:** $f(x) = \lambda^{\alpha} x^{\alpha-1} e^{-\lambda x} / \Gamma(\alpha)$, **for $x > 0$,** **where Γ is the gamma function, defined by:** $\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx$. **In particular, $\Gamma(n) = (n - 1)!$ when n is a positive integer.** **(Simard)**	**Alpha – shape parameter, alpha > 0** **Lambda – scale parameter, lambda>0**
Geometric	The *geometric* distribution with parameter p, where $0 < p < 1$. Its mass function is: $p(x) = p(1 - p)^x$, for $x = 0, 1, 2,\ldots$ The distribution function is given by: $F(x) = 1 - (1 - p)^{x+1}$, for $x = 0, 1, 2,\ldots$ and its inverse is: $F^{-1}(u) = \text{floor}(\ln(1 - u)/\ln(1 - p))$, for $0 <= u < 1$ (Simard)	p – probability of success on each trial (0 < p < 1)
Gumbel	The Gumbel distribution, with location parameter δ and scale parameter $\beta \neq 0$. Using the notation $z = (x - \delta)/\beta$, it has density: $f(x) = e^{-z} e^{-e^{-z}} / \lvert \beta \rvert$, for $-\infty < x < \infty$. and distribution function: $F(x) = e^{-e^{-z}}$, for $\beta > 0$ $F(x) = 1 - e^{-e^{-z}}$, for $\beta < 0$ (Simard)	Beta – scale parameter, beta ≠ 0 Delta – location parameter, any real number
Hypergeometric	The *hypergeometric* distribution with k elements chosen among l, m being of one type, and $l - m$ of the other. The parameters m, k and l are positive integers where $1 <= m <= l$ and $1 <= k <= l$. Its mass function is given by: $p(x) = \text{nCr}(m, x)\text{nCr}(l - m, k - x)/\text{nCr}(l, k)$ for $\max(0, k - l + m) <= x <= \min(k, m)$ where $\text{nCr}(n, x)$ is the number of possible combinations of x elements chosen among a set of n elements. (Simard)	m – number of elements of one type (integer), m > 0 l – total elements (integer), l > 0 k – number of elements chosen among l (integer), k > 0
Logistic	The *logistic* distribution. It has location parameter α and scale parameter $\lambda > 0$. The density is: $f(x) = (\lambda e^{-\lambda(x-\alpha)})/((1 + e^{-\lambda(x-\alpha)})^2)$ for $-\infty < x < \infty$. and the distribution function is: $F(x) = 1/[1 + e^{-\lambda(x-\alpha)}]$ for $-\infty < x < \infty$. For $\lambda = 1$ and $\alpha = 0$, one can write: $F(x) = (1+\tanh(x/2))/2$. The inverse distribution function is given by: $F^{-1}(u) = \ln(u/(1 - u))/\lambda + \alpha$ for $0 <= u < 1$ (Simard)	Alpha – location parameter, any real number Lambda – scale parameter, lambda>0
Lognormal	The *lognormal* distribution. It has scale parameter μ and shape parameter $\sigma > 0$. The density is: $f(x) = ((2\pi)^{1/2}\sigma x)^{-1} e^{-(\ln(x)-\mu)2/(2\sigma 2)}$ for $x > 0$, and 0 elsewhere. The distribution function is: $F(x) = \Phi((\ln(x)-\mu)/\sigma)$ for $x > 0$, where Φ is the standard normal distribution function. Its inverse is given by: $F^{-1}(u) = e^{\mu+\sigma\Phi-1(u)}$ for $0 <= u < 1$ If $\ln(Y)$ has a *normal* distribution, then Y has a *lognormal* distribution with the same parameters. (Simard)	log mu – scale parameter, any real number log sigma – shape parameter, log sigma > 0

Negative Binomial	**The negative binomial distribution with real parameters γ and p, where $\gamma > 0$ and $0 <= p <= 1$. Its mass function is:** **$p(x) = \Gamma(\gamma + x)/(x!\ \Gamma(\gamma))p^{\gamma}(1 - p)^{x}$, for $x = 0, 1, 2,\ldots$** **where Γ is the gamma function.** **If γ is an integer, $p(x)$ can be interpreted as the probability of having x failures before the γ-th success in a sequence of independent Bernoulli trials with probability of success p.** **(Simard)**	**Gamma – number of failures until the experiment is stopped, Gamma > 0** **p – success probability in each experiment, $0 \leq p \leq 1$**
Normal	The *normal* distribution. It has mean μ and variance σ^2. Its density function is: $f(x) = e^{-(x-\mu)2/(2\sigma2)}/((2\pi)^{1/2}\sigma)$ for $-\infty < x < \infty$, where $\sigma > 0$. When $\mu = 0$ and $\sigma = 1$, we have the *standard normal* distribution, with corresponding distribution function: $F(x) = \Phi(x) = \int_{-\infty}^{x} e^{-t2/2}\, dt/(2\pi)^{1/2}$ for $-\infty < x < \infty$. (Simard)	Mean – self-explanatory, any real number Standard Deviation – self-explanatory, Std. Dev. > 0
Pareto	The *Pareto* family, with shape parameter $\alpha > 0$ and location parameter $\beta > 0$. The density for this type of Pareto distribution is: $f(x) = \alpha\beta^{\alpha}/x^{\alpha+1}$ for $x >= \beta$, and 0 otherwise. The distribution function is: $F(x) = 1 - (\beta/x)^{\alpha}$ for $x >= \beta$, and the inverse distribution function is: $F^{-1}(u) = \beta(1 - u)^{-1/\alpha}$ for $0 <= u < 1$ (Simard)	Alpha – shape parameter, alpha > 0 Beta – location parameter, beta > 0
Poisson	The *Poisson* distribution with mean $\lambda >= 0$. The mass function is: $p(x) = e^{-\lambda}\lambda^{x}/(x!)$, for $x = 0, 1,\ldots$ and the distribution function is: $F(x) = e^{-\lambda}\sum_{j=0}^{x} \lambda^{j}/(j!)$, for $x = 0, 1,\ldots.$ (Simard)	Lambda – mean, lambda ≥ 0
Student's T	The *Student-t* distribution with n degrees of freedom, where n is a positive integer. Its density is: $f(x) = [\Gamma((n + 1)/2)/(\Gamma(n/2)(\pi n)^{1/2})][1 + x^2/n]^{-(n+1)/2}$ for $-\infty < x < \infty$, where $\Gamma(x)$ is the gamma function (Simard)	D.O.F – degrees of freedom (integer), D.O.F > 0
Triangular	The *triangular* distribution with domain $[a, b]$ and mode (or shape parameter) m, where $a <= m <= b$. The density function is: $f(x) = 2(x - a)/[(b - a)(m - a)]$ for $a <= x <= m$, $f(x) = 2(b - x)/[(b - a)(b - m)]$ for $m <= x <= b$, $f(x) = 0$ elsewhere, the distribution function is: $F(x) = 0$ for $x < a$, $F(x) = (x - a)^2/[(b - a)(m - a)]$ if $a <= x <= m$, $F(x) = 1 - (b - x)^2/[(b - a)(b - m)]$ if $m <= x <= b$, $F(x) = 1$ for $x > b$, and the inverse distribution function is given by: $F^{-1}(u) = a + ((b - a)(m - a)u)^{1/2}$ if $0 <= u <= (m - a)/(b - a)$, $F^{-1}(u) = b - ((b - a)(b - m)(1 - u))^{1/2}$ if $(m - a)/(b - a) <= u <= 1$ (Simard)	a – lower bound of the domain, any real number b – upper bound of the domain, any real number mode – shape parameter, any real number "$a \leq mode \leq b$" must be satisfied
Uniform	The *uniform* distribution over the interval $[a, b]$. Its density is: $f(x) = 1/(b - a)$ for $a <= x <= b$, and 0 elsewhere. The distribution function is: $F(x) = (x - a)/(b - a)$ for $a <= x <= b$ and its inverse is: $F^{-1}(u) = a + (b - a)u$ for $0 <= u <= 1$ (Simard)	Min. – lower bound Max. – upper bound (Max. > Min.)

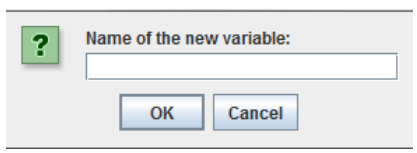
Weibull	The *Weibull* distribution with shape parameter $\alpha > 0$, location parameter δ, and scale parameter $\lambda > 0$. The density function is: $f(x) = \alpha\lambda^{\alpha}(x - \delta)^{\alpha-1}e^{-(\lambda(x-\delta))\alpha}$ for $x > \delta$. the distribution function is: $F(x) = 1 - e^{-(\lambda(x-\delta))\alpha}$ for $x > \delta$, and the inverse distribution function is: $F^{-1}(u) = (- \ln(1 - u))^{1/\alpha}/\lambda + \delta$ for $0 <= u < 1$ (Simard)	Alpha – shape parameter, alpha > 0 Lambda – scale parameter, lambda>0 Delta – location parameter, any real number

ADDING VARIABLES

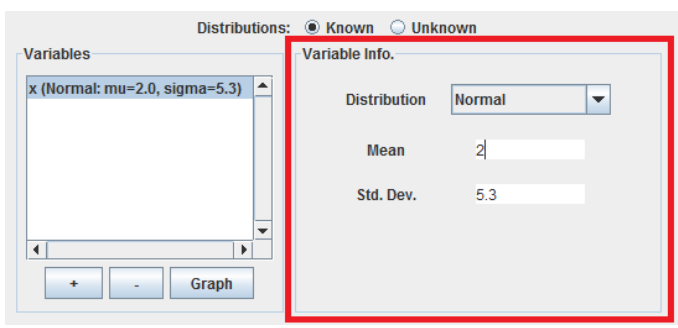
In order to add a variable to a model, the user must left-click over the **+** button that can be found in the **Main** tab:



The user will be asked to enter the new variable's name:

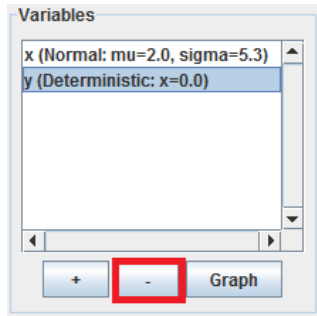


As long as there is no other previously declared variable with the same name as the one entered, the new variable will show up in the list of variables. Afterwards, the user must change the variable's information in the **Variable Info** pane, depending on the type of distribution that is to be assigned to the variable:



REMOVING VARIABLES

Removing variables is straight-forward. The user just needs to select the variable that is to be removed from the simulation model in the **Variables** pane and then left-click over the **–** button:

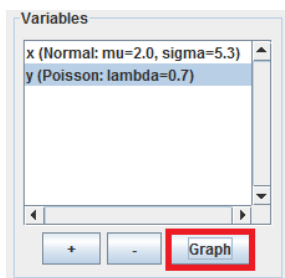


In the screenshot above, the variable **y** would disappear from the model after clicking the **–** button.

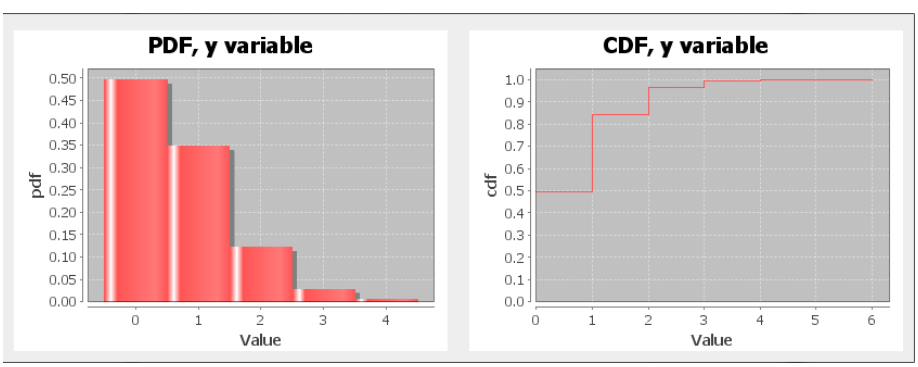
GRAPHING A SINGLE VARIABLE

In order to verify that the parameters of the variables coincide with what the user expects, 2R Soft provides the option to visualize the PDF (probability density function) and CDF (cumulative density function) curves associated with each variable before running a model.

To view the behavior of a particular variable in graph form, the user must select such variable in the **Variables** pane and then left-click over the **Graph** button:



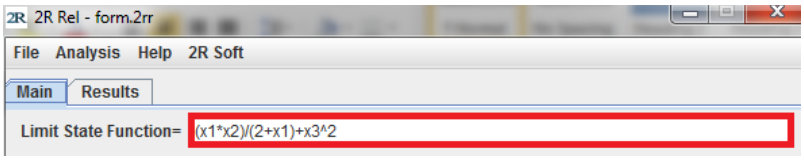
A new window will pop-up with the PDF and CDF curves:



MAIN TAB

EQUATION

2R Rel only supports explicit equations (limit state functions), and only one equation can be associated with a specific model. The equation is to be written in the **Main** tab:



Refer to the Equation Editor section for information on supported functions and equation syntax in general.

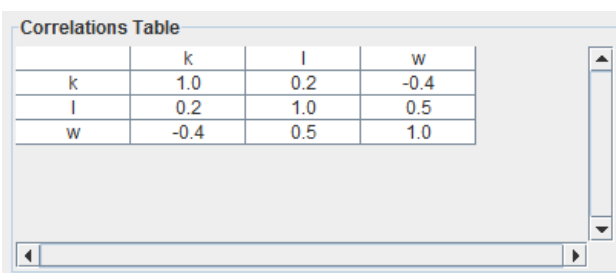
CORRELATIONS MATRIX (CORRELATIONS TABLE)

The correlation coefficient is a single number between -1 and 1 that describes the dependence between two variables. The formula used to compute the correlation coefficient of two variables from experimental data is (Lane):

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

When two variables are independent from each other, their correlation coefficient is equal to 0. On the other hand, if two variables are perfectly dependent, their correlation coefficient is equal to 1 (as one increases the other one also increases) or -1 (as one increases the other one decreases, and vice-versa).

During a simulation run, random values of the different variables must be generated. Hence, 2R Rel must take correlations into account to make simulations as accurate as possible, since the value generated for a variable might influence the probability distribution of another variable in a simulation's iteration. This information is entered by the user in the form of a **Correlation Matrix** that is shown in the **Main** tab:

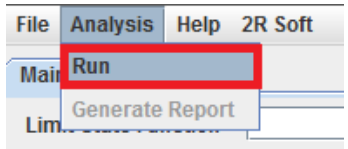


	k	l	w
k	1.0	0.2	-0.4
l	0.2	1.0	0.5
w	-0.4	0.5	1.0

The resulting matrix is always symmetric and shows the multiple dependencies that exist between all the variables contained in the current model. To edit a correlation coefficient, the user must double-click over the corresponding cell and enter the new value.

RUNNING A RELIABILITY ANALYSIS

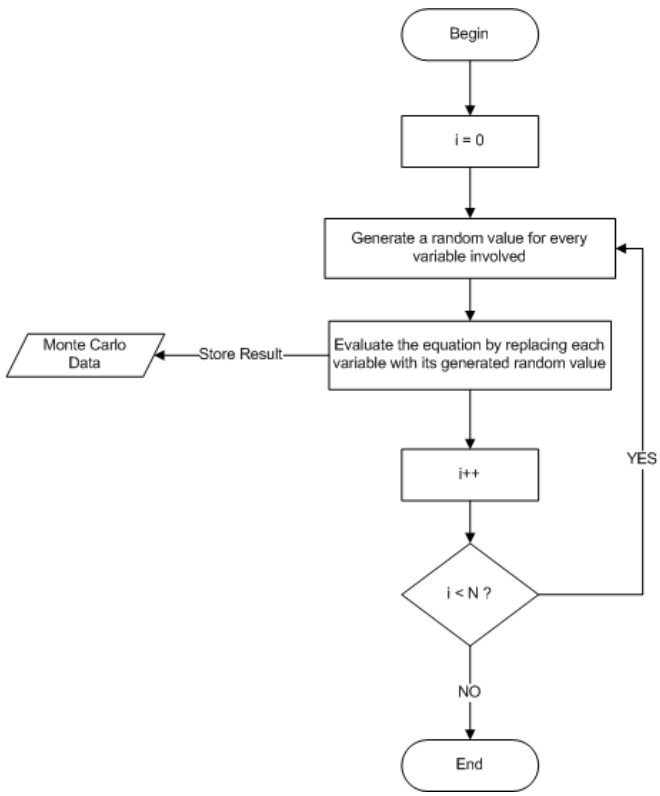
When the reliability model is complete (limit state function, variables, correlation matrix), the analysis process can be started by navigating through the **Analysis** menu and selecting the **Run** option:



MONTE CARLO

THEORY

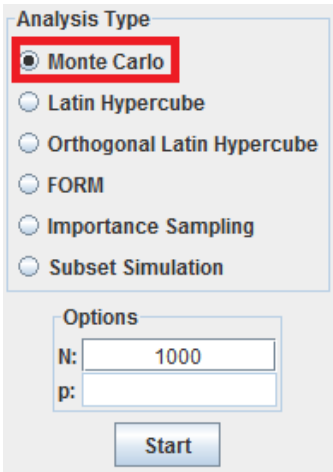
The Monte Carlo algorithm follows the flow chart shown below: (N is the number of simulations to be carried out)



Although the Monte Carlo algorithm is simple, one run can be time-consuming because the generation of random numbers and the evaluation of an equation are processor-intensive procedures. Nonetheless, each iteration is independent from the rest, which enables the use of a thread pool to run various iterations at the same time depending on the amount of processing cores available in the computer running the program. 2R Rel uses the thread pool strategy to speed up Monte Carlo runs.

INPUT

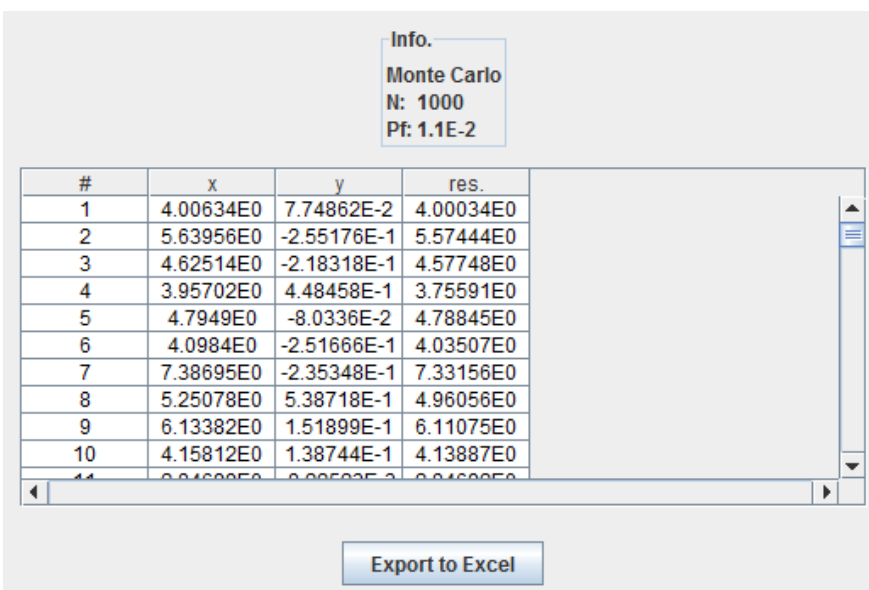
To begin a Monte Carlo simulation, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu:



- **N:** number of reliability simulations to carry out

OUTPUT

The user is presented with the probability of failure (**Pf**) drawn from the simulated sample, along with data for every simulation performed. Such data can be exported to Microsoft Excel with no hassle by clicking on the **Export to Excel** button below the table:



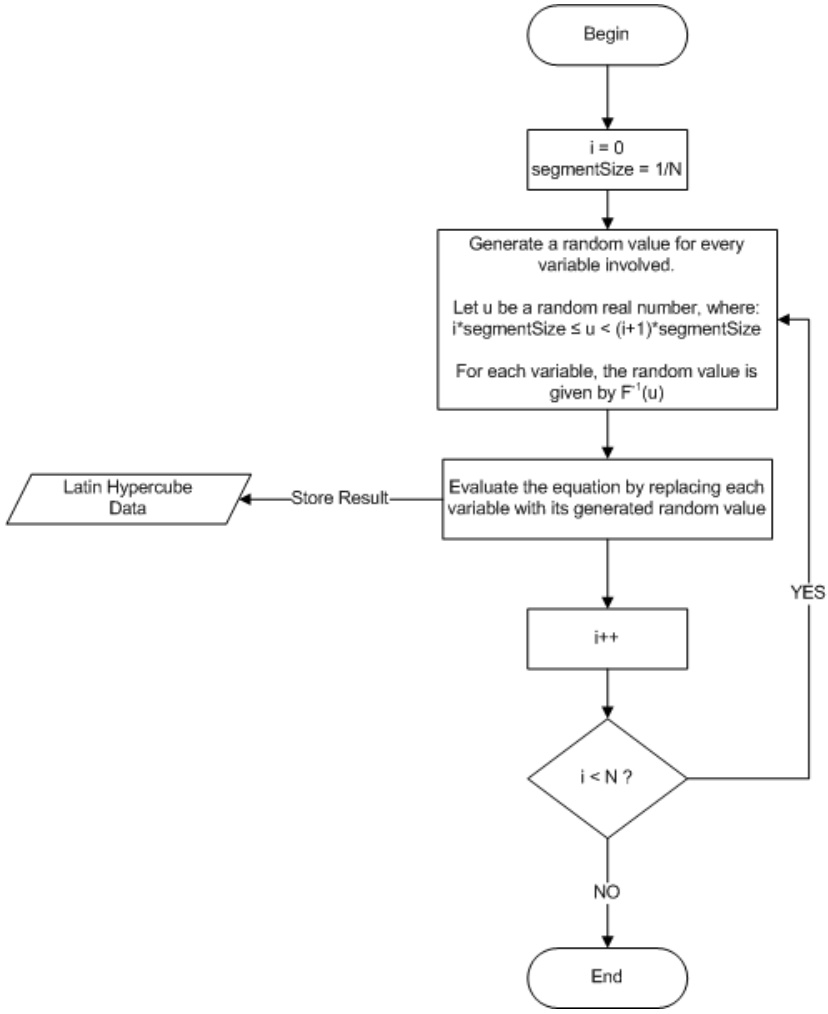
#	x	y	res.
1	4.00634E0	7.74862E-2	4.00034E0
2	5.63956E0	-2.55176E-1	5.57444E0
3	4.62514E0	-2.18318E-1	4.57748E0
4	3.95702E0	4.48458E-1	3.75591E0
5	4.7949E0	-8.0336E-2	4.78845E0
6	4.0984E0	-2.51666E-1	4.03507E0
7	7.38695E0	-2.35348E-1	7.33156E0
8	5.25078E0	5.38718E-1	4.96056E0
9	6.13382E0	1.51899E-1	6.11075E0
10	4.15812E0	1.38744E-1	4.13887E0

In the screenshot above, the system failed on 11 of the 1000 times it was tested, leading to a probability of failure of 1.1E-2. As the amount of simulations, N, increases, the probability of failure is expected to be more consistent and closer to the real value.

LATIN HYPERCUBE

THEORY

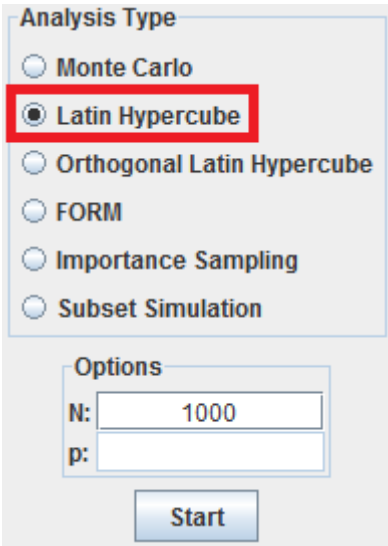
The Latin Hypercube algorithm uses a sampling process different from the one in the basic Monte Carlo algorithm, leading to a better coverage of the sampling space with a smaller number of iterations. As expected, the resulting flow chart is more elaborate than the one shown in the previous section: (N is the number of simulations to be carried out)



As the flow chart shows, the Latin Hypercube algorithm divides the range [0,1] in N partitions of the same size (segmentSize), and then takes one value, u, from each of those partitions to generate the random values of the variables. The thread pool strategy mentioned in the previous section is also employed by 2R Rel with this algorithm, taking into account that the iterations are still independent from each other.

INPUT

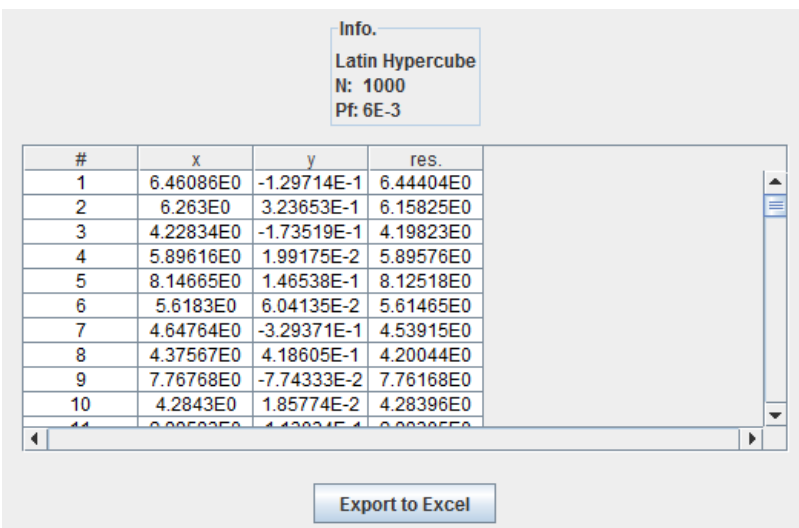
To begin a Latin Hypercube simulation, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu:



- **N:** number of reliability simulations to carry out

OUTPUT

The user is presented with the probability of failure (**Pf**) drawn from the simulated sample, along with data for every simulation performed. Such data can be exported to Microsoft Excel with no hassle by clicking on the **Export to Excel** button below the table:



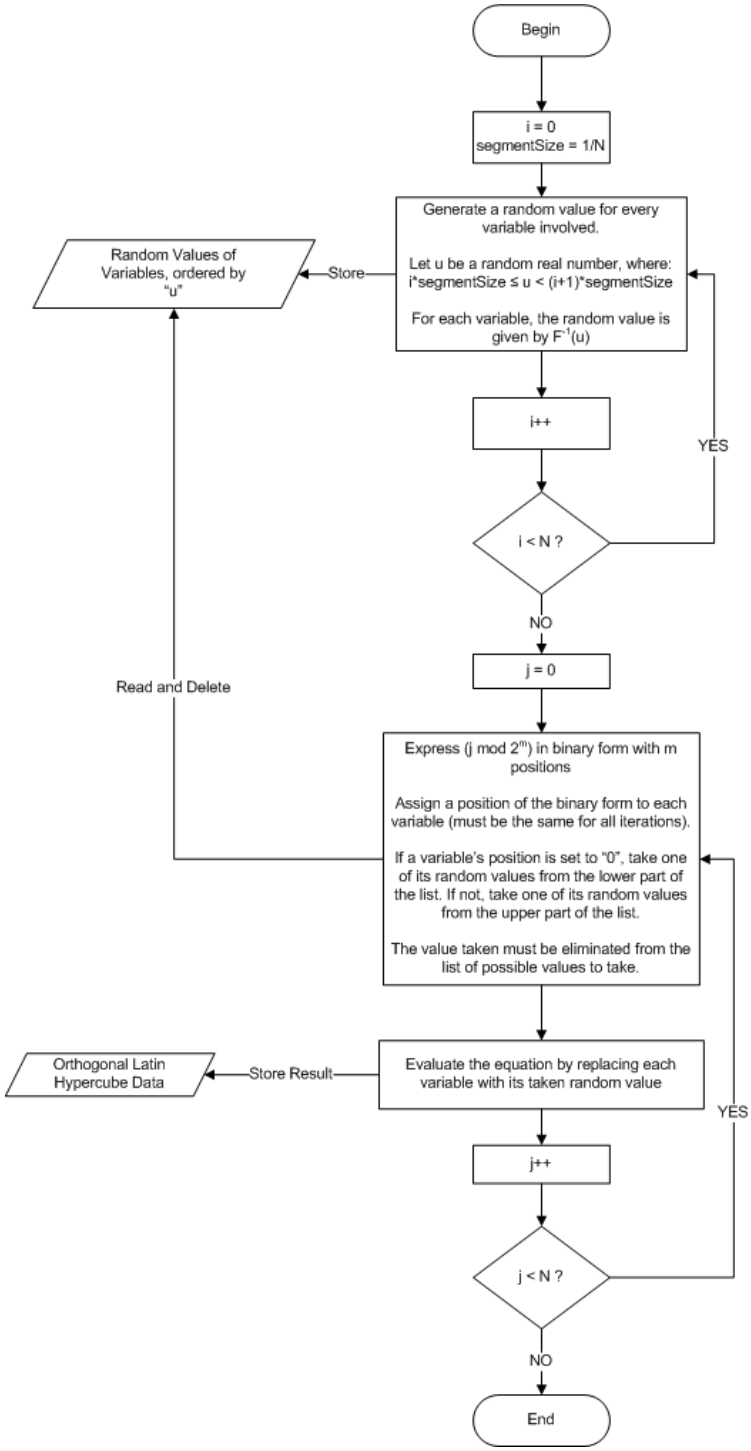
#	x	y	res.
1	6.46086E0	-1.29714E-1	6.44404E0
2	6.263E0	3.23653E-1	6.15825E0
3	4.22834E0	-1.73519E-1	4.19823E0
4	5.89616E0	1.99175E-2	5.89576E0
5	8.14665E0	1.46538E-1	8.12518E0
6	5.6183E0	6.04135E-2	5.61465E0
7	4.64764E0	-3.29371E-1	4.53915E0
8	4.37567E0	4.18605E-1	4.20044E0
9	7.76768E0	-7.74333E-2	7.76168E0
10	4.2843E0	1.85774E-2	4.28396E0

In the screenshot above, the system failed on 6 of the 1000 times it was tested, leading to a probability of failure of 6E-3. As the amount of simulations, N, increases, the probability of failure is expected to be more consistent and closer to the real value.

ORTHOGONAL LATIN HYPERCUBE

THEORY

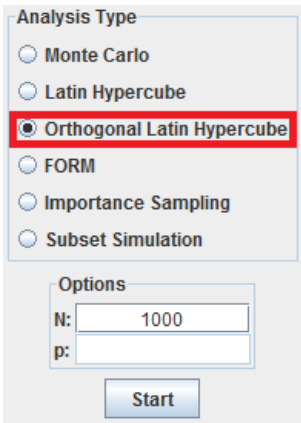
This algorithm refines the sampling process of the normal Latin Hypercube by dividing the sampling space into subspaces with the same probability, where the amount of samples taken from each of those subspaces is the same. If each variable's sampling space is divided into two sections (upper and lower), there will be 2^m equally probable subspaces to be monitored, being *m* the number of variables being used. The resulting flow chart is: (N is the number of simulations to be carried out, and is a multiple of 2^m)



As shown in the flow diagram, the Orthogonal Latin Hypercube algorithm is comprised of two cycles: one that calculates random values for the variables and another one that picks generated random values according to the appropriate subspaces and evaluates the equation with the values picked. The binary form is used as a way to assure that each subspace is sampled the same number of times as the rest. As with the previous algorithms, the Orthogonal Latin Hypercube is optimized in 2R Rel by using a thread pool due to the independence of the iterations.

INPUT

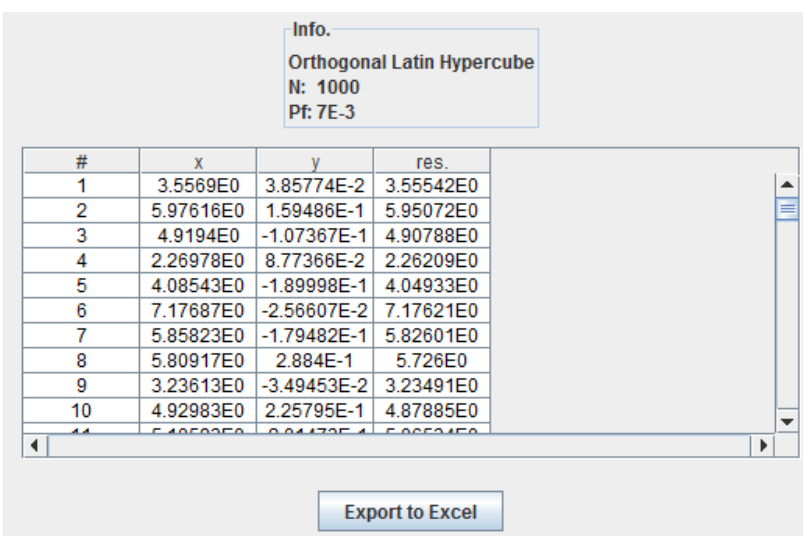
To begin a Latin Hypercube simulation, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu:



- **N:** number of reliability simulations to carry out

OUTPUT

The user is presented with the probability of failure (**Pf**) drawn from the simulated sample, along with data for every simulation performed. Such data can be exported to Microsoft Excel with no hassle by clicking on the **Export to Excel** button below the table:



#	x	y	res.
1	3.5569E0	3.85774E-2	3.55542E0
2	5.97616E0	1.59486E-1	5.95072E0
3	4.9194E0	-1.07367E-1	4.90788E0
4	2.26978E0	8.77366E-2	2.26209E0
5	4.08543E0	-1.89998E-1	4.04933E0
6	7.17687E0	-2.56607E-2	7.17621E0
7	5.85823E0	-1.79482E-1	5.82601E0
8	5.80917E0	2.884E-1	5.726E0
9	3.23613E0	-3.49453E-2	3.23491E0
10	4.92983E0	2.25795E-1	4.87885E0

In the screenshot above, the system failed on 7 of the 1000 times it was tested, leading to a probability of failure of 7E-3. As the amount of simulations, N, increases, the probability of failure is expected to be more consistent and closer to the real value.

FORM (FIRST ORDER RELIABILITY METHOD)

THEORY

Note: this section is based on (Lee 2008). Refer to that text to gain a deeper understanding of the topic.

INTRODUCTION

A reliability analysis entails the calculation of the probability of failure, denoted by P_F, which is defined using a multi-dimensional integral (Madsen et al., 1986):

$$P_F \equiv P[G(\mathbf{X}) > 0] = \int_{G(\mathbf{X})<0} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

Where $\mathbf{X}=\{X_1, X_2, \ldots, X_N\}^T$ is an N-dimensional random vector, G(**X**) is the limit state function such that G(**X**) < 0 is defined as system failure, and $f_x(\mathbf{x})$ is a joint probability density function (PDF) of the random variable **X**. In most engineering applications, the exact evaluation of the equation shown above is very difficult or often impossible to obtain since $f_x(\mathbf{x})$ is generally non-Gaussian and G(**X**) is highly non-linear. To handle the non-Gaussian $f_x(\mathbf{x})$, a transformation from the original X-space into the independent standard normal U-space is introduced. In addition, FORM approximates G(**X**) using a First Order Taylor Series Expansion to attenuate its non-linearity.

TRANSFORMATION TO U-SPACE

Consider an N-dimensional random vector **X** with a joint cumulative distribution function (CDF) $F_x(\mathbf{x})$. Let T: **X → U** denote a transformation from X-space to U-space that is defined by Rosenblatt (Rosenblatt, 1952):

$$T: \begin{cases} u_1 = \Phi^{-1}\left[F_{X_1}(x_1)\right] \\ u_2 = \Phi^{-1}\left[F_{X_2}(x_2|x_1)\right] \\ \quad\vdots \\ u_N = \Phi^{-1}\left[F_{X_N}(x_N|x_1, x_2, \cdots, x_{N-1})\right] \end{cases}$$

where $F_{X_i}(x_i|x_1, x_2, \cdots, x_{i-1})$ is the conditional CDF given by

$$F_{X_i}(x_i|x_1, x_2, \cdots, x_{i-1}) = \frac{\int_{-\infty}^{x_i} f_{X_1X_2\cdots X_i}(x_1, x_2, \cdots, x_{i-1}, \xi) d\xi}{f_{X_1X_2\cdots X_{i-1}}(x_1, x_2, \cdots, x_{i-1})}$$

and $\Phi(\bullet)$ is the standard normal CDF given by

$$\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp\left(-\frac{1}{2}\xi^2\right) d\xi$$

The inverse transformation can be obtained from the first equation:

$$T^{-1}: \begin{cases} x_1 = F_{X_1}^{-1}\left[\Phi(u_1)\right] \\ x_2 = F_{X_2}^{-1}\left[\Phi(u_2 \;\; x_1)\right] \\ \\ x_N = F_{X_N}^{-1}\left[\Phi(u_N \;\; x_1, x_2, \quad , x_{N-1})\right] \end{cases}$$

If the N-dimensional random vector **X** is independent, that is, the joint PDF is given by:

$$f_{\mathbf{X}}(\mathbf{x}) = f_{X_1}(x_1) \times f_{X_2}(x_2) \times \quad \times f_{X_N}(x_N)$$

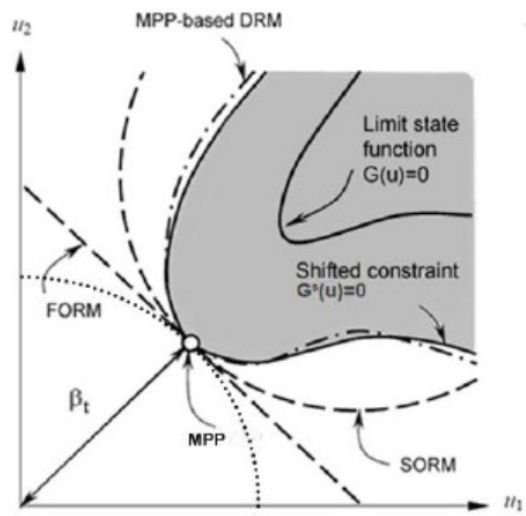
Where $f_{xi}(x_i)$ are the marginal PDFs, then the Rosenblatt transformation and the inverse transformation are simplified as:

$$u_i = \Phi^{-1}\left[F_{X_i}(x_i)\right] \text{ and } x_i = F_{X_i}^{-1}\left[\Phi(u_i)\right]$$

Where $F_{xi}(x_i)$ are the marginal CDFs.

FORM

To calculate the probability of failure of the system with limit state function G(**x**) using FORM, it is necessary to find the most probable point (MPP), which is defined as the point **u*** on the limit state function (g(**u**)=0) closest to the origin in the standard normal U-space, as shown below:



The limit state function in U-space is defined as g(**u**)=G(**x**(**u**))=G(**x**) using the Rosenblatt transformation. Hence, the MPP can be found by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} && \|\mathbf{u}\| \\ &\text{subject to} && g(\mathbf{u}) = 0 \end{aligned}$$

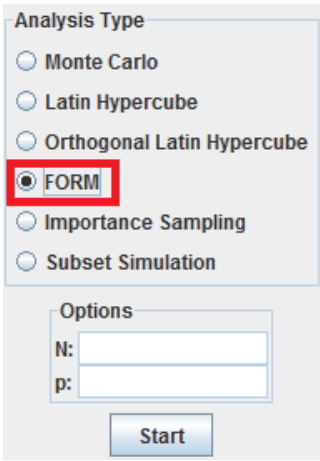
After finding the MPP, the distance from the MPP to the origin is commonly called the Hasofer-Lind reliability index (Hasofer and Lind, 1974) and denoted by β_{HL}, that is, $\beta_{HL} = \|\boldsymbol{u}^*\|$. Using the reliability index β_{HL}, FORM can approximate the probability of failure using linear approximation of the limit state function as:

$$P_F^{\text{FORM}} \cong \Phi(-\beta_{\text{HL}})$$

The FORM algorithm used by 2R Rel is fully described in (SÁNCHEZ-SILVA 2010). 2R Rel supposes that the random variables are independent and calculates the joint PDF as the multiplication of the independent PDFs.

INPUT

To begin a FORM analysis, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu:

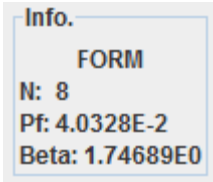


No additional user input is required.

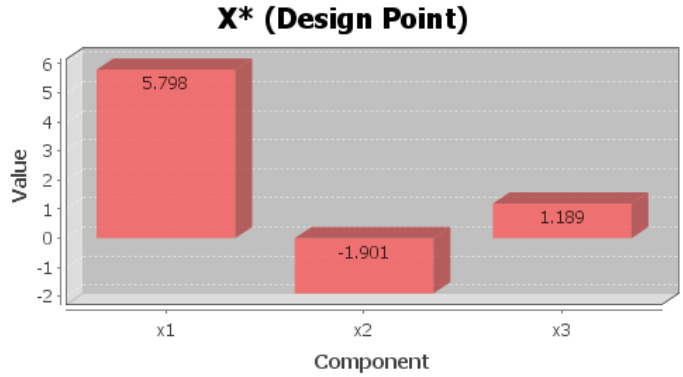
OUTPUT

The user is presented with the probability of failure drawn from the FORM analysis, along with 3 complimentary graphs:

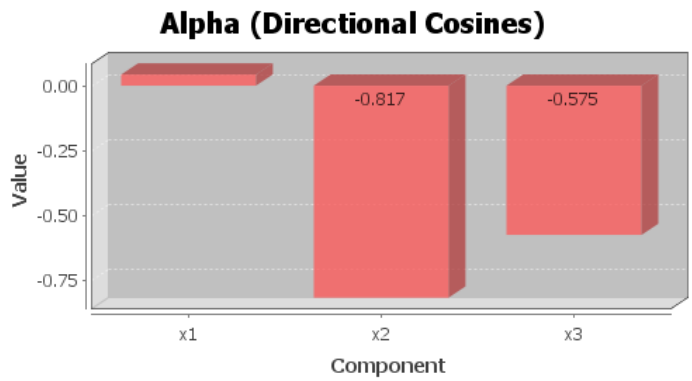
- The **info.** section indicates the amount of FORM iterations that were run, **N**, as well as the probability of failure that was obtained, **Pf**, and its associated **Beta**. Refer to the Theory section for information on the meaning of Beta.



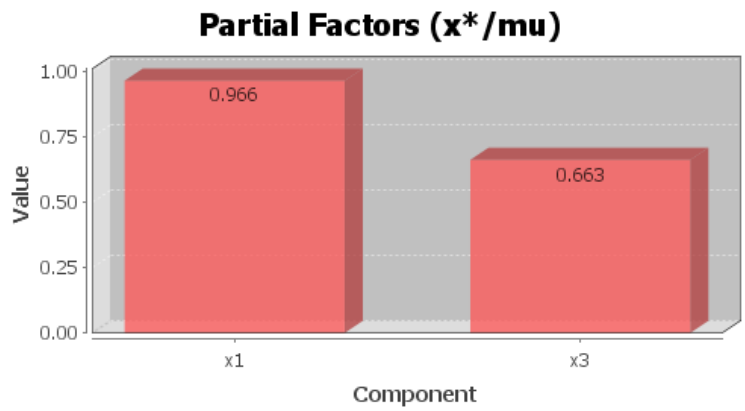
- A graph then shows the components of the **Design Point**, also known as the **Most Probable Point (MPP)**. Refer to the Theory section for information on the meaning of the MPP.



- Another graph is used to show the directional cosines, which are the components of $\boldsymbol{\alpha}$ in the equation $\overrightarrow{\boldsymbol{u}^*} = \vec{\boldsymbol{\alpha}}\beta$. As mentioned in the Theory section, $\overrightarrow{\boldsymbol{u}^*}$ is the **Most Probable Point (MPP)** and β is the Hasofer-Lind reliability index.



- Last but not least, the partial factors are shown to give an idea of the MPP's components relative to the mean of each variable.



IMPORTANCE SAMPLING

THEORY

Note: this section is based on (SÁNCHEZ-SILVA 2010). Refer to that book for a deeper understanding of the topic.

Normal Monte Carlo analysis failure probabilities are calculated as follows:

$$P_f \approx \frac{1}{N}\sum_{i=1}^{N} I\left[g(\overrightarrow{\hat{x}_i})\right] = \frac{N_F}{N}$$

Where N_F is the number of system failures, N is the number of simulations, $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ is the value of the random variables in the i-th simulation, and $g(\vec{x})$ is the limit state function.

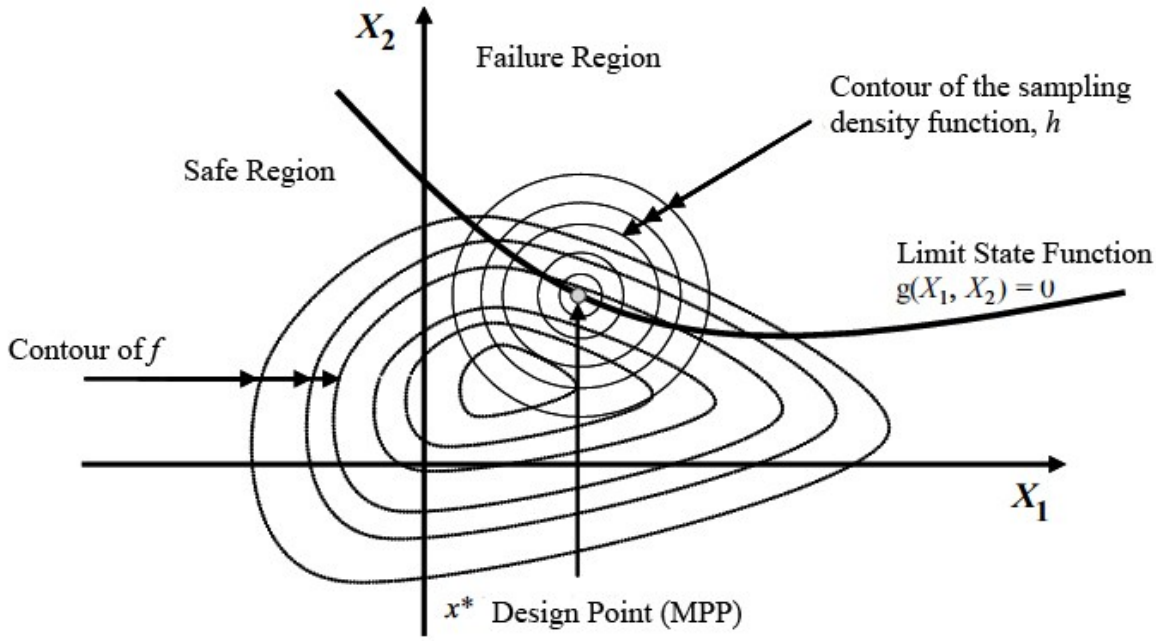
Importance sampling focuses on the region of space that contributes the most to the failure probability of a system. It uses a new function, $h(\vec{x})$, known as the *sampling density function*, to such end. Once this function is defined, the probability of failure can be calculated as: (Let $f_{\vec{X}}(\vec{x})$ be the joint PDF of the variables involved)

$$P_f = \int \left[I\left[\overrightarrow{x}\right] \frac{f_{\overrightarrow{X}}(\overrightarrow{x})}{h(\overrightarrow{x})} \right] h(\overrightarrow{x}) d\overrightarrow{x}$$

Consequently, the failure probability can be statistically approximated by the equation shown below:

$$P_f \approx \frac{1}{N}\sum_{i=1}^{N}\left\{ I\left(\overrightarrow{\widehat{x}_i}\right)\frac{f_{\vec{X}}(\overrightarrow{\widehat{x}_i})}{h(\overrightarrow{\widehat{x}_i})}\right\}$$

A graphical representation of the Importance Sampling approach in a 2D sampling space would be the following:



SAMPLING FUNCTION

Selecting $h(\vec{x})$ is no simple task. **In 2R Rel, the sampling density function for an N-dimensional sampling space is taken to be the multiplication of N normally distributed variables such that:**

$$h(\vec{x}) = \varphi(x_1, x_1^*, \sigma_{x1})\varphi(x_2, x_2^*, \sigma_{x2}) \dots \varphi(x_N, x_N^*, \sigma_{xN})$$

Where $\varphi(x_i, x_i^*, \sigma_{xi})$ is a normally distributed random variable, x_i, with the i-th design point (MPP) component, x_i^*, as its mean and standard deviation, σ_{xi}, equal to the original random variable's, X_i, standard deviation.

This function selection enables a higher sampling density in the region of interest. Nonetheless, **variable independence is assumed, which leads to biased results if the correlation between any pair of variables is different to zero.**

DESIGN POINT (MPP)

The design point or MPP (Most Probable Point), x^*, is defined as the point on the limit state function ($g(\vec{x})$=0) where the maximum value of the joint PDF, $f_{\vec{x}}(\vec{x})$, is obtained. **2R Rel supposes that the random variables are independent and calculates the joint PDF as the multiplication of the independent PDFs. The MPP is estimated by using numerical methods.**

ALGORITHM

The Importance Sampling algorithm is:

1. Define the limit state function, $g(\vec{x})$.
2. Define each random variable's, X_i, probability distribution, as well as the resulting joint PDF, $f_{\vec{X}}(\vec{x})$.
3. Find the design point or MPP (Most Probable Point), x^*.
4. Generate a vector of normally distributed random numbers, $\hat{\boldsymbol{x}}_i$, with x^* as mean. Then, calculate $h(\hat{\boldsymbol{x}}_i)$ as $\varphi(x_1, x_1^*, \sigma_{x1})\varphi(x_2, x_2^*, \sigma_{x2}) \dots \varphi(x_N, x_N^*, \sigma_{xN})$, where $\varphi(x_j, x_j^*, \sigma_{xj})$ is the j-th component of $\hat{\boldsymbol{x}}_i$.
5. Calculate the failure probability of the iteration:

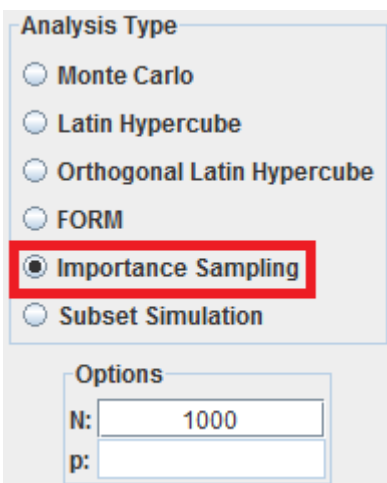
$$P_{f_i} = \frac{I\left[g(\vec{\hat{x}}_i) \leq 0\right] f_{\vec{X}}(\vec{\hat{x}}_i)}{h(\vec{\hat{x}}_i)}$$

6. Repeat steps 4 and 5 until N simulations are completed.
7. The final failure probability will be given by:

$$P_f = \frac{1}{N}\sum_{i=1}^{N} P_{f_i}$$

INPUT

To begin an Importance Sampling analysis, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu:

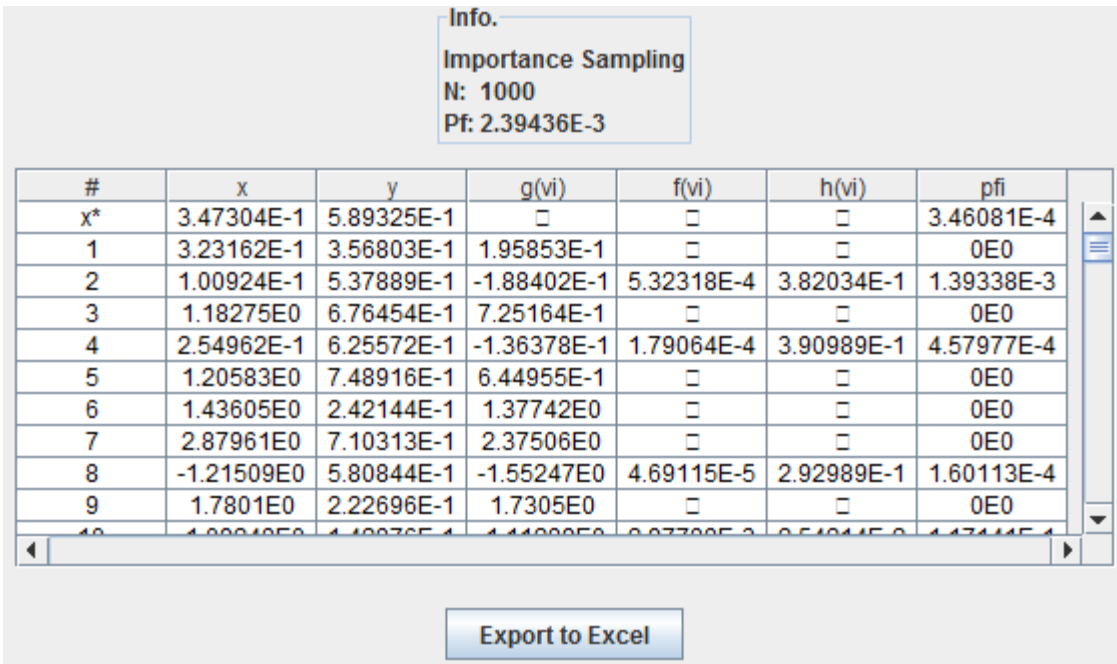


- **N**: the number of simulations to carry out for the analysis. Refer to the Theory section to gain some understanding on the logic behind the algorithm.

OUTPUT

The analysis output comes in the form of a basic summary (**Info.** section) and a step-by-step description (data table).

- The **Info.** section summarizes the number of iterations that were run, **N**, as well as the probability of failure that was obtained, **Pf**.
- The data table contains the information of each iteration per row. **The first row shows the coordinates of the estimated design point (MPP) and the joint probability density function's value on that point.** Some rows are left with blank values for $h(\hat{\boldsymbol{x}}_i)$ and $f(\hat{\boldsymbol{x}}_i)$, since a positive limit state function, $g(\hat{\boldsymbol{x}}_i) > 0$, leads to an associated failure probability, P_{fi}, equal to zero regardless of those two values. Refer to the Theory section for more information on the meaning of the columns presented in the data table.



#	x	y	g(vi)	f(vi)	h(vi)	pfi
x*	3.47304E-1	5.89325E-1				3.46081E-4
1	3.23162E-1	3.56803E-1	1.95853E-1			0E0
2	1.00924E-1	5.37889E-1	-1.88402E-1	5.32318E-4	3.82034E-1	1.39338E-3
3	1.18275E0	6.76454E-1	7.25164E-1			0E0
4	2.54962E-1	6.25572E-1	-1.36378E-1	1.79064E-4	3.90989E-1	4.57977E-4
5	1.20583E0	7.48916E-1	6.44955E-1			0E0
6	1.43605E0	2.42144E-1	1.37742E0			0E0
7	2.87961E0	7.10313E-1	2.37506E0			0E0
8	-1.21509E0	5.80844E-1	-1.55247E0	4.69115E-5	2.92989E-1	1.60113E-4
9	1.7801E0	2.22696E-1	1.7305E0			0E0

SUBSET SIMULATION (SUBSET SAMPLING)

THEORY

Note: this section is a word-by-word copy of the content found in ((Dresden)). We invite you to visit that web site to learn more about uncertainty in engineering.

The basic idea of subset sampling is the subdivision of the failure event into a sequence of m partial failure events (subsets) $F_1, F_2, \cdots, F_m = F$. Generally, the determination of small failure probabilities P_F with the aid of Monte Carlo simulation requires the expensive simulation of rare events. The division into subsets (subproblems) offers the possibility to transfer the simulation of rare events into a set of simulations of more frequent events. The determination of the failure regions F_i can be effected by presetting a series $g_i | i=1...m$ of limit values, whereas m denotes the (unknown) total number of subsets.

$$F_i = \{x : g(\underline{x}) \leq g_i\}$$

This enables the computation of the failure probability as a product of conditional probabilities $P(F_{i+1}|F_i)$ and $P(F_1)$.

$$P_F = P(F_m) = P(F_m|F_{m-1})P(F_{m-1}|F_{m-2}) \cdots P(F_2|F_1)P(F_1)$$
$$= P(F_1) \prod_{i=1}^{m-1} P(F_{i+1}|F_i)$$

The determination of the failure regions F_i and subsequently the partial failure probabilities $P_i = P(F_{i+1} \mid F_i)$ influences the accuracy of the simulation. It is convenient to specify the limit values $g_i | i=1...m$ so that nearly equal partial failure probabilities $P_i | i=2...m$ are obtained for each subset. Unfortunately, it is difficult to specify the limit values g_i in advance according to the prescribed probability P_i. Therefore the limit values have to be determined adaptively within the simulation.

ALGORITHM

In the first step the probability $P_1=P(F_1)$ is determined by application of the direct Monte Carlo simulation.

$$P(F_1) \approx \hat{P}_1 = \frac{1}{N_1} \sum_{k=1}^{N_1} I_{F_1}(\underline{x}_k^{(1)})$$

To obtain conditional probabilities $P(F_{i+1}|F_i)$ the evaluation of the respective probability functions

$$f(\underline{x} \mid F_i) = \frac{I_{F_i}(\underline{x}) f(\underline{x})}{P(F_i)}$$

is necessary. **With the application of the Markov Chain Monte Carlo (MCMC) simulation in conjunction with the Metropolis-Hastings algorithm samples may be generated in a numerically efficient way according to $f(\underline{x}|F_i)$.**

$$P(F_{i+1}|F_i) \approx \hat{P}_{i+1} = \frac{1}{N_{i+1}} \sum_{k=1}^{N_{i+1}} I_{F_{i+1}}(\underline{x}_k^{(i+1)})$$

The starting sample of the subset i+1 is selected randomly from the samples $\underline{x}^{(i)} | g_i(\underline{x}^{(i)}) \leq g_i$, i=1,...,m−1 of subset i that are located in the failure region F_i. The limit value g_i of the i-th partial subset is determined adaptively during the simulation. Therefore, g_i is determined from a list of ascending sorted tuple $(\underline{x}_k^{(i)}, g(\underline{x}_k^{(i)})) | k=1,...,N_i$ according to the values $g(\underline{x}_k^{(i)})$. The limit value g_i is given by

$$g_i = g\left(x_j^{(i)}\right) \mid j = int(P_i \cdot N_i)$$

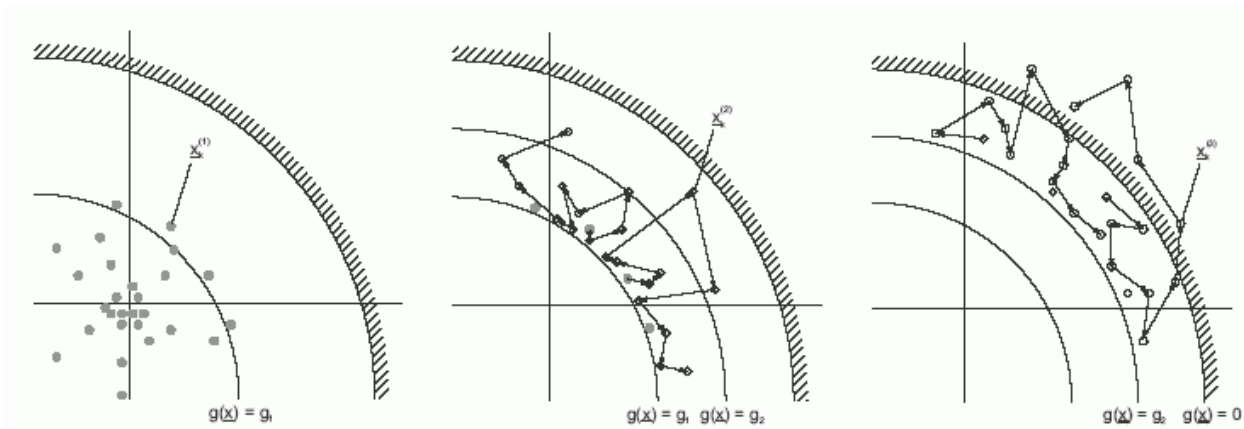
Under the condition $g_i \leq 0$ the last subset m of the simulation has been reached. The last failure probability $P(F_m | F_{m-1})$ can be estimated with

$$P(F_m|F_{m-1}) \approx \hat{P}_m = \frac{1}{N_m} \sum_{k=1}^{N_m} I_{F_m}(\underline{x}_k^{(m)})$$

The failure probability P_F may now be computed as

$$P_F = P_1 \cdot \prod_{i=2}^{m} P_i$$

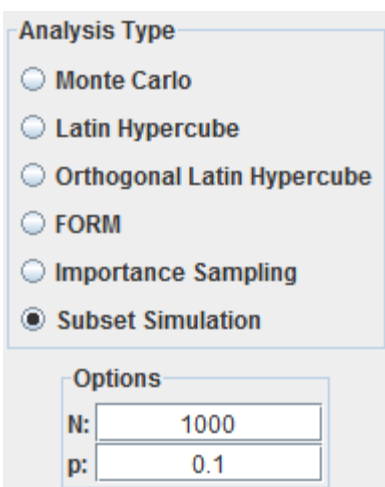
The subset sampling algorithm is exemplified for three subsets in the following picture:



The Subset Simulation algorithm used by 2R Rel is fully described in (SÁNCHEZ-SILVA 2010).

INPUT

To begin a Subset Simulation analysis, the user must select the appropriate radio button after going into the **Run** option found in the **Analysis** menu

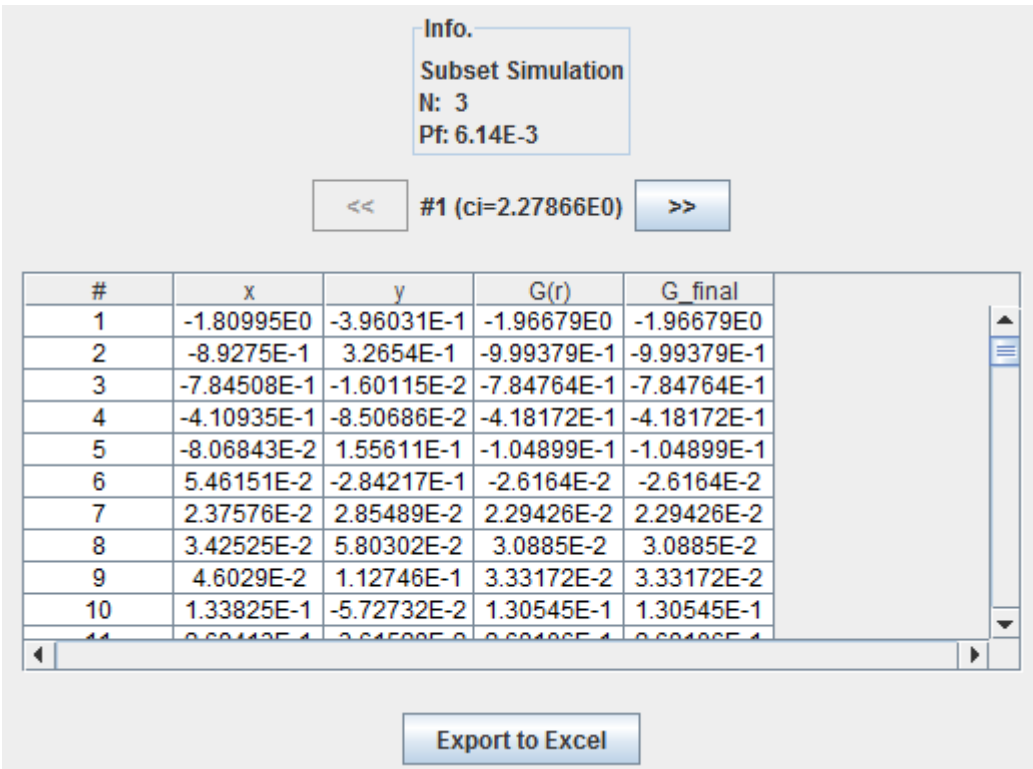


- **N:** the amount of samples to be generated per subset.
- **p:** the cutoff probability of the subsets. For example, if p=0.1 and N=1000, only the 100 (that is, N*p) generated values with the lowest values of g(**x**) in a subset are retained to generate the next subset.

OUTPUT

The analysis output comes in the form of a basic summary (**Info.** section) and a step-by-step description (data table).

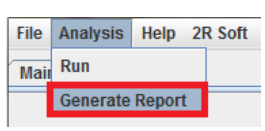
- The **Info.** section summarizes the number of subsets that were generated, **N**, as well as the probability of failure that was obtained, **Pf**.
- The data table is generated for each subset. To view a different subset, press the **[<<]** and **[>>]** buttons. **The table's contents are sorted by ascending value of G_final.**
- For each subset, the cutoff value of g(**x**) (or **G_final**) is represented by **ci**. For example, in the results shown below, only the points with **G_final** $\leq$ 2.27866 were retained from the first subset.
- Each row contains the coordinates for each generated point and its corresponding g(**x**) in the **G(r)** column. The **G_final** column shows the final value of g(**x**) for a point after being processed by the Markov Chain Monte Carlo (MCMC) and Metropolis-Hastings algorithms. **The G_final column contains the only g(x) values that matter.**



#	x	y	G(r)	G_final
1	-1.80995E0	-3.96031E-1	-1.96679E0	-1.96679E0
2	-8.9275E-1	3.2654E-1	-9.99379E-1	-9.99379E-1
3	-7.84508E-1	-1.60115E-2	-7.84764E-1	-7.84764E-1
4	-4.10935E-1	-8.50686E-2	-4.18172E-1	-4.18172E-1
5	-8.06843E-2	1.55611E-1	-1.04899E-1	-1.04899E-1
6	5.46151E-2	-2.84217E-1	-2.6164E-2	-2.6164E-2
7	2.37576E-2	2.85489E-2	2.29426E-2	2.29426E-2
8	3.42525E-2	5.80302E-2	3.0885E-2	3.0885E-2
9	4.6029E-2	1.12746E-1	3.33172E-2	3.33172E-2
10	1.33825E-1	-5.72732E-2	1.30545E-1	1.30545E-1

REPORT GENERATION

2R Rel is capable of exporting all the useful information that results from an analysis process to PDF format. To do this, a user must navigate through the **Analysis** menu and select the **Generate Report** option. This option will only be enabled if a successful run has been completed.



The user is then prompted for the new report's name and location before 2R Rel creates the PDF file.

BIBLIOGRAPHY

(Dresden), I. o. S. a. D. o. S. T. "Subset Sampling." from http://www.uncertainty-in-engineering.net/applications/efficiency/subset_sampling.

Lane, D. "Computing Pearson's Correlation Coefficient." Retrieved 05/17/2010, from http://davidmlane.com/hyperstat/A51911.html.

Lee, I. (2008) RELIABILITY-BASED DESIGN OPTIMIZATION AND ROBUST DESIGNOPTIMIZATION USING UNIVARIATE DIMENSION REDUCTION METHOD.

Microsystems, S. "Math JAVA J2SE." Retrieved 05/17/2010, from http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Math.html.

MSDN, M. "Modulus Operator (%)." Retrieved 05/17/2010, from http://msdn.microsoft.com/en-us/library/h6zfzfy7(VS.80).aspx.

SÁNCHEZ-SILVA, M. (2010). INTRODUCCIÓN A LA CONFIABILIDAD Y EVALUACIÓN DE RIESGOS: teoría y aplicaciones en ingeniería, Universidad de los Andes.

Simard, R. "Package umontreal.iro.lecuyer.probdist." Retrieved 05/17/2010, from http://www.iro.umontreal.ca/~simardr/ssj/doc/html/umontreal/iro/lecuyer/probdist/package-summary.html.