ReliaComp User & Theory Manual

Mayank Chetan

October 23, 2020

Contents

Chapter 1

Introduction

Essentially all models are wrong, but some are useful.
— George Box

ReliaComp is an easy to use tool to evaluate the reliability of a system for risk-based analysis. It uses various statistical models to compute the reliability of a given system based on correlated system uncertainties. The underlying core codes were initially developed as a part of the authors research.

ReliaComp allows the user to define a performance criteria (limit-state function) in terms of a simple MATLAB function which may contain the an analytical input model or be a wrapper for a complex external model. The methods implemented allows for a large number of correlated random variables to be used as inputs. These random variables can be drawn from various standardized distributions such as, normal, log-normal or a uniform distributions. The current implementation of ReliaComp can use Monte Carlo Simulation (MCS), Mean Value First Order Second Moment (MVFOSM) method and Hasofer Lind - Rackwitz Fiessler (HL-RF) [Fiessler and Rackwitz, 1978] methods to computer the reliability of the provided limit-state function

This document is organized as follows, Chapter 2 introduces the user-interface for the tool and details the inputs required to run it. This chapter also goes over validation of the input data that is carried out within ReliaComp and the interpretation of the results. Chapter 3 summarizes the theory and its implementation in ReliaComp. Chapter 4 goes over a few examples that have been included with the tool that can be used as a reference to define custom limit-state functions. Finally, Chapter 5 outlines potential future work.

ReliaComp is currently open sourced through the GPL-3.0 License and can be accessed via https://github.com/mayankchetan/ReliaComp.

Chapter 2

Using ReliaComp

This chapter goes over how to obtain and run ReliaComp in MATLAB. Most of development work of the tool was done on MATLAB 2019a, the ensures compatibility with MATLAB 2019a and above. ReliaComp may work with older version (above 2018a) but is not guaranteed.

2.1 Distributed Files

Users can pull / fork a copy of ReliaComp from its GitHub page, or download it directly at, https://github.com/mayankchetan/ReliaComp/archive/main.zip.

The tool contains the ***utils***, ***examples***, ***docs*** folder. The program can be started through clicking on ***ReliaCompGUI.mlapp*** or running the ***ReliaCompGUI_exported.m*** file from the MATLAB GUI. ***utils*** folder includes the functions used by ReliaComp to carry out the computation. The ***examples*** folder contains three examples that show the various capabilities of ReliaComp, these can be used as templates for the user's own limit-state functions. A detailed description of the examples are provided in Chapter 4. The documentation can be found in the ***docs*** folder.

2.2 Interface Overview

The ReliaComp interface as shown in Figure 2.1 is divided into three main parts: (1) The main control panel, (2) the method specific inputs and (3) the general input-output panel.

Main control Panel The main control panel allows the user to define the main set of parameters to run ReliaComp.Here the user can select the limit-state function in the form of a MATLAB function, define the number of basic variables for the limit-state function as well as select the method of computing the reliability and finally run the computation.

Method Specific Inputs Panel This panel is where the inputs specific to the reliability computation methods are defined. The details related to each method are explained in Section 2.4

Interaction Panel The interaction panel is where the users will define the basic input variables, get feedback from the tool during the computations and be presented with the final results. This panel has three tabs,

1. **Variable Inputs Tab:** This is where the user defines the various basic variables for the limit-state function. An example of this is shown in Figure 2.1.

2. **Output Tab:** This is where the program displays the current process that is being carried out. An example is shown in Figure 2.2.

3. **Results Tab:** The final results of the computation is presented in this tab. An example is shown in Figure 2.3.

Tool Bar The tool bar groups various high level functions together. The "File" menu lets the user, save and load session as well as reset all inputs for the tool. The "Help" menu lets the user bring up this document as well as go the ReliaComp GitHub page.

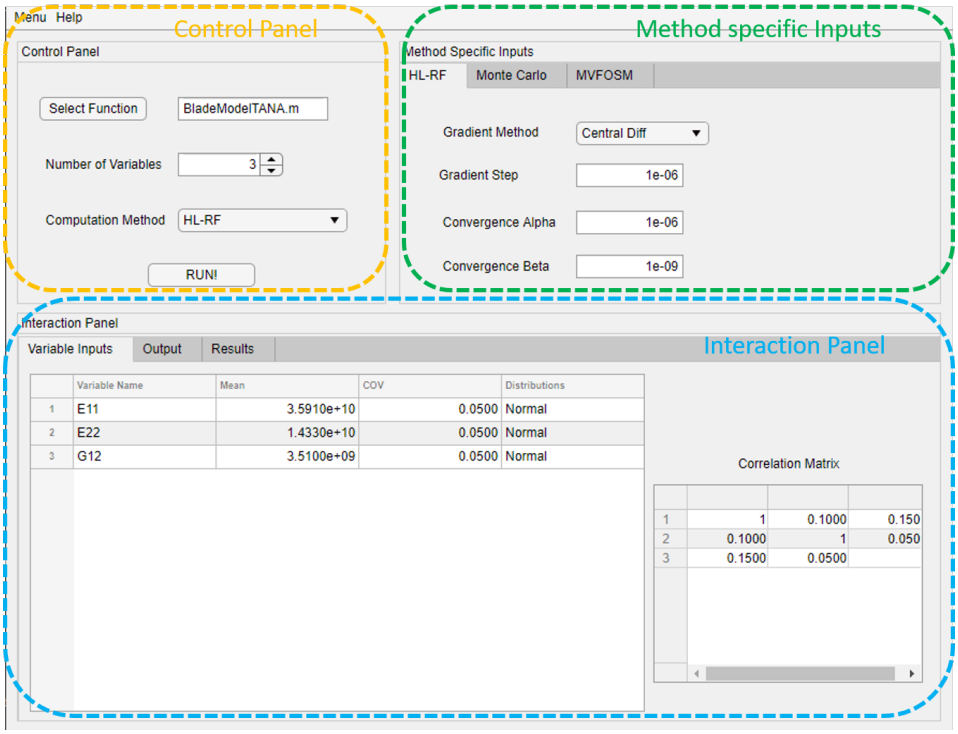

Figure 2.1: The three major parts of the ReliaComp interface.

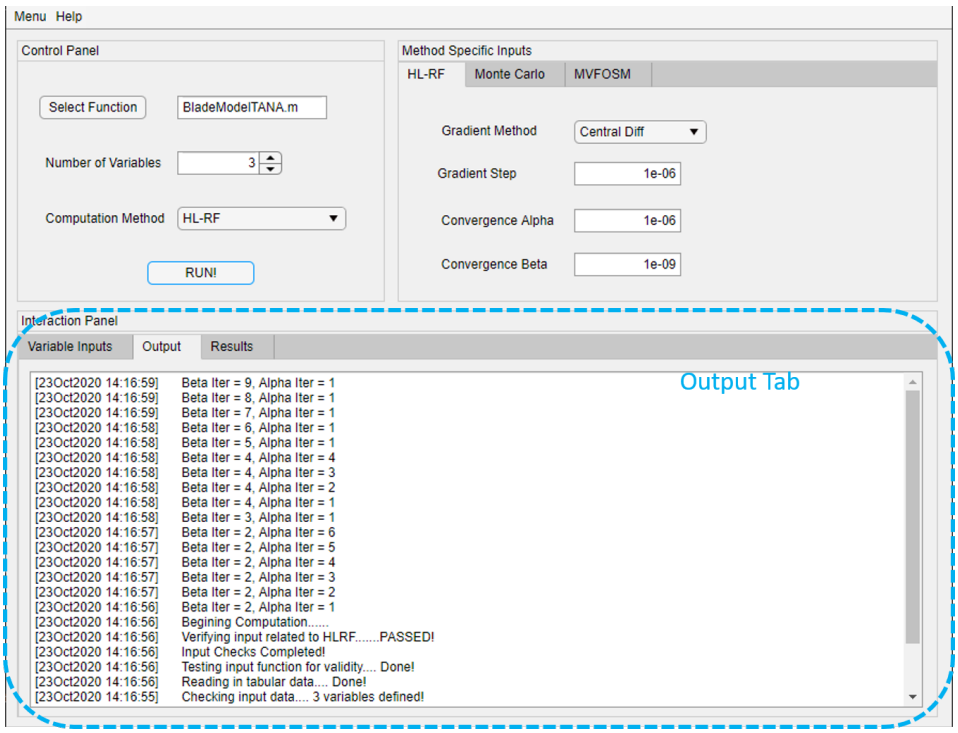

Figure 2.2: The output tab in action.

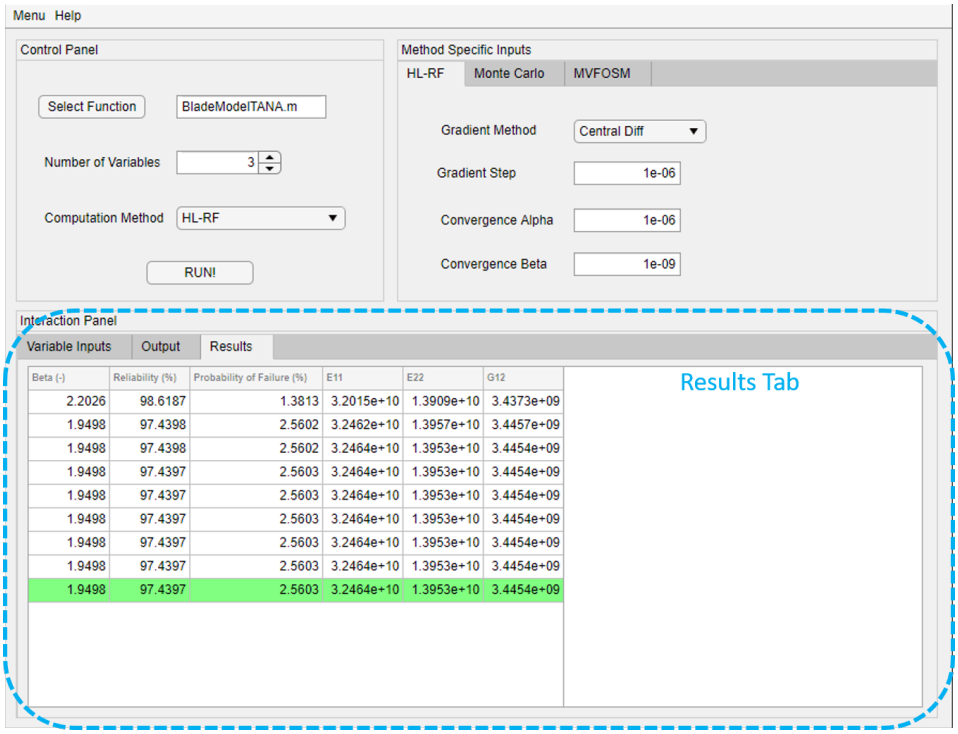

Figure 2.3: An example of the final results tabulated in the results tab.

2.3 Program Architecture

Figure 2.4 shows the program architecture implemented in ReliaComp. The different aspects of the program are explained in further detail in the following sections and chapters.

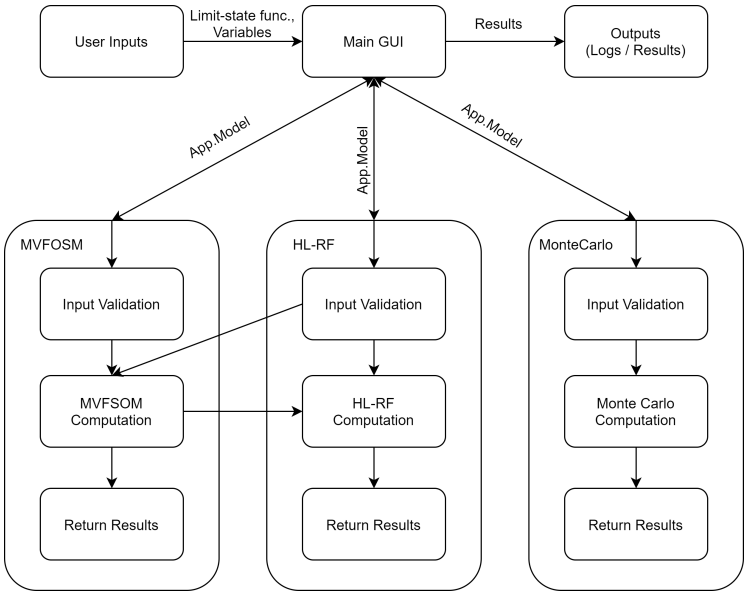

Figure 2.4: ReliaComp Architecture.

2.4 Program Inputs

In this section we go over the inputs that the user has to provide to run ReliaComp. Most of the inputs from the user are validated before the computation is carried out, if any errors are found the user is notified.

2.4.1 Setting up Inputs

limit-state Function The user can select a limit-state function that they have defined using the "Select Function" button and selecting the MATLAB function. A simplest form of the limit-state function takes in a vector of the basic variables and provides a scalar value of the limit-state. An example of a limit-state function that can be used as a template is provided in Appendix A.

Number of Variables The number of variables that is taken in by the limit-state function is defined here. Based on the number of variables selected, the tables to populate the values are updated.

Computation Method The user can select the method for carrying out the reliability analysis of the limit-state funciton. Currently, there are two methods implemented, (1) Hasofer Lind - Rackwitz Fiessler method (HL-RF) which is an advanced first-order secomd-moment (AFSOM) reliability technique, (2) Monte Carlo Simulations (MCS) method which carries out a large number of samples to compute the reliability.

HL-RF When HL-RF method is selected a few more inputs are necessary to define the parameters of method.

1. **Gradient Method:** The user can select the method to calculate the gradients while running the HL-RF method. Currently only the central difference method is implemented. A future version will allow the user to provide analytical gradients via the limit-state function.
2. **Gradient Perturbation:** This is the perturbation carried out about a point while computing the gradients. A typical value based on the authors experience is 1e-6.
3. **Convergence Alpha:** This is the convergence value for the variable α in the HL-RF method. the details regarding this are provided in Chapter 3. A typical value based on the authors experience is 1e-6.
4. **Convergence Beta:** This is the convergence value for the variable β in the HL-RF method. the details regarding this are provided in Chapter 3. A typical value based on the authors experience is 1e-9.

MCS - Number of Samples When MCS method is selected the user must provide the number of samples that are used to compute the reliability of the limit-state function.

Variable Inputs This is where the basic variables are defined. Depending on the number of input variables defined by the user, the rows are made editable. The first column takes in the name of the variable, this is just for the users reference and is not used for computation. Next column is the mean value of the variable, followed by its coefficient of variation (COV). The final column lets the user select the distribution of the the input variable. Currently there are three distributions implemented, (1) normal distribution, (2) log-normal distribution and (3) uniform distribution.

Correlation Matrix Here the user can provide the correlation between the variables. This matrix is used while computing the random variables. The use is only required to fill in the upper or lower triangle of the matrix.

2.4.2 Input verification

Validating the inputs is an important step to ensure the robustness of the computation. For this reason when the user run the computation in ReliaComp, it carries out a wide range of verification's on the inputs provided by the user. Few of the major verification's are detailed below.

1. **Limit-state function:** The program verifies if the user has selected limit-state function MATLAB function.
2. **Number of variables:** The program verifies the number of variables that the user has provided with the given limit-state function. If the number of input variables are lower than that accepted by the limit-state function the user is alerted. If there are a larger number of variables than what is accepted by the limit-state functions, then the first N variables are used, where N is the number of accepted available. This limitation can be overcome by the having check on the number of inputs in the limit-state function itself.
3. **Method Specific Inputs:** The program verifies if the user provided values are within the recommended values based on the authors experience. If not the user is notified to modify them.

4. **Basic Variable Inputs:** The program verifies if all the data for the basic variables are populated by the user.

5. **Correlation Matrix:** The program verifies the correlation matrix is symmetric, positive-semi definite and that all elements are no larger than 1.

2.5 Interpretation of Outputs

When running computations, ReliaComp outputs the progress to the output tab as shown in Figure 2.2. Once computation is completed, ReliaComp presents the results in the results tab. The way the results are presented depend on the methods used for the computation as they yield. This section discusses how to interpreter them based for the different methods. ReliaComp also stored the all the results from the computation on the MATLAB workshops, that can be accessed via the traditional MATLAB GUI.

2.5.1 HL-RF

Once the computations are completed using the HL-RF method, the results from the iterations are tabulated with the final results on the last row highlighted in green. The "Reliability Index" β is shown in the first column. This is a common measure of reliability of a limit-state function, further details are provided in Chapter 3. The second column provide the reliability of the limit-state function as a percentage followed by the probability of failure in the next column. The next N columns represent the set of N input variables that results in the point on the failure surface that has the highest probability of failure for the limit-state function. A set of larger results are saved to the MATLAB base work-space.

2.5.2 Monte Carlo

Once the computations are completed using the Monte Carlo method, the final results are tabulated in the results tab. The reliability of the limit-state function is shown in the first column. The next two columns show the upper and lower 95% confidence intervals for the computation. The final column shows the number of simulations carried out to compute the reliability.

2.5.3 MVFOSM

Once the computations are completed using the MVFSOM method, the final results are tabulated in the results tab. The columns of the results are the same as the at of HR-RF.

Chapter 3

Theory and implementation

3.1 General Introduction

Reliability based design is a method of design that involves the consideration of the probability of occurrence of the input variables and system parameters. Consider a simple example of a load S on a structure and its capacity to resist the load R. When the load exceeds the capacity to resist the load the structure fails. Mathematically we can write this as,

$$Z = G(R, S) = R - S \tag{3.1}$$

Here, Z is called the limit-state function who's value when goes less than zero indicates that the system has failed. The load S and resistance to load R are both random in nature. Figure 3.1 shows a graphical representation of Equation 3.1 in the physical domain.

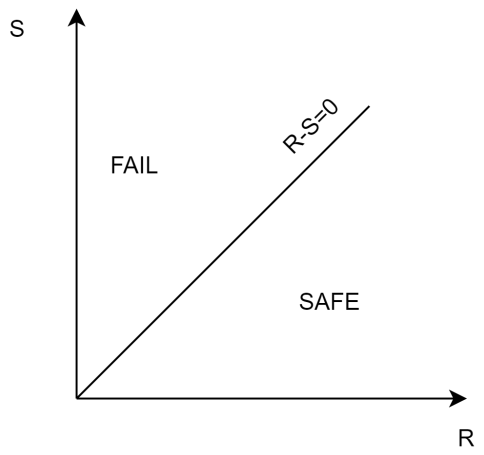

Figure 3.1: Limit-state Concept in the physical domain.

The probability of failure of the limit-state function Z is given by,

$$p_f = \int ... \int_{g()<0} f_X(x_1, x_2, ...x_n) dx_1 dx_2 .... dx_n \tag{3.2}$$

where $f_X(x_1, x_2, ...x_n)$ is the joint probability distribution for the basic random variables $X_1, X_2, ...X_n$, for which the integration is carried out over the failure region $g() < 0$.

If we assume that R and S are statistically independent and normally distributed random variables, the Z is also a normal random variable. We can evaluate the failure of Z as,

$$p_f = P(Z < 0) = \Phi\left(\frac{0 - (\mu_R - \mu_S)}{\sqrt{\sigma_R^2 + \sigma_R^2}}\right) = 1 - \Phi\left(\frac{(\mu_R - \mu_S)}{\sqrt{\sigma_R^2 + \sigma_R^2}}\right) \tag{3.3}$$

Where Φ is the CDF of the standard normal variate. Since the physical variables are now transformed into the standard normal domain, the transformed limit-state function is shown in Figure 3.2.

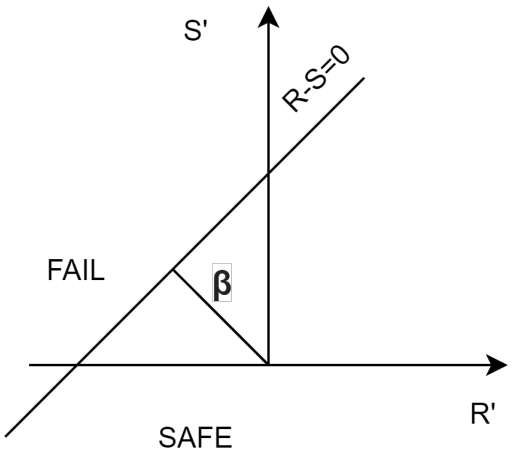

Figure 3.2: Limit-state Concept in the standard normal domain.

from Equation 3.3, we can observe that the probability of failure of the limit-state function depends on the ratio of mean value of Z to its standard deviation. This ratio is called the reliability index or Cornell Reliability index. From Figure 3.2 we ca observe that β is the perpendicular distance from the limit-state function to the origin in the standard normal domain.

$$\beta = \frac{(\mu_R - \mu_S)}{\sqrt{\sigma_R^2 + \sigma_R^2}} \tag{3.4}$$

The reliability of the system is given by,

$$R = 1 - p_f = 1 - \Phi(\beta) \tag{3.5}$$

Most of the assumptions up to this point hold true when:

1. The random variables involved in the limit state function are independent.
2. The random variables involved in the limit state function follow a normal distribution.
3. The limit state function is linear in nature.

This is not the case in most engineering or design problems. For this reason the simplified method discussed about cannot be used reliability, however it does give us a good approximation for the system reliability.

3.2 Mean Value First Order Second Moment method.

The Mean Value First Order Second Moment (MVFOSM) method, is a very simple method to compute the reliability of a limit-state function. It is derived from the first-order Taylor expansion of the limit-state function at the mean values of the basic variables.

The computation is carried out for the reliability index β^* assuming the variables are uncorrelated normal variables. First, the mean value of the limit-state function is computed by using the mean values of the basic variables.

$$\mu_{\tilde{g}} = g(\mu_{x^*}) \tag{3.6}$$

Next, The standard deviation is computed using Equation 3.7

$$\sigma_{\tilde{g}} = \sqrt{\sum g'(\mu_{x^*})^2 \times Var(x^*)} \tag{3.7}$$

Finally, the estimate of β^* is given my Equation 3.8.

$$\beta^* = \frac{\mu_{\tilde{g}}}{\sigma_{\tilde{g}}} \tag{3.8}$$

This is simplest analytical solution to find the reliability of a limit-state function, The assumptions going in to this computation does not reflect real world problems, yet it provides a good approximation that can be used as a starting point for more complex reliability computation methods.

3.3 Hasofer Lind - Rackwitz Fiessler Method

To address the assumptions of MVFOSM, the Hasofer Lind - Rackwitz Fiessler method (HL-RF) method is introduced. This is an iterative approach for non-linear limit-state functions with non-normal random variables. Since the limit-state function is no longer a linear equation, perpendicular distance is no longer available for a single point. We can rewrite the reliability index as,

$$\beta = \min_{U \epsilon g(U)=0} (U^T U)^{1/2} \tag{3.9}$$

Now, β is the minimum distance from the origin to the limit-state function in the standard normal domain. A simple representation of this is shown in Figure 3.3. Here the two β values are the approximations based on linearization of the limit-state functions.

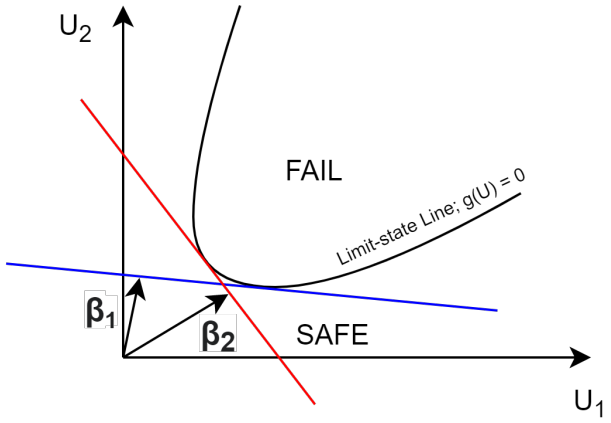

Figure 3.3: Non-linear limit-state concept in the standard normal domain.

To be able to handle the non-normal variables it will be transformed into equivalent normal variables after which the computation of the reliability is carried out. The transformation of into equivalent normal variables is a two step process:

1. The CDF of the actual variable and the equivalent variable are matched at the points of interest x^*.

$$\begin{aligned} F_X(x^*) &= \Phi\left(\frac{x^* - \mu_X^N}{\sigma_X^N}\right) \\ \mu_X^N &= x^* - \Phi^{-1}(F_X(x^*))\sigma_X^N \end{aligned} \tag{3.10}$$

   Here, $F_X(x^*)$ is the CDF of the non-normal distribution at point x^*.

2. The PDF of both the variables are also matched.

$$\begin{aligned} f_X(x^*) &= \frac{1}{\sigma_X^N}\phi\left(\frac{x^* - \mu_X^N}{\sigma_X^N}\right) \\ \sigma_X^N &= \frac{\phi(\frac{x^* - \mu_X^N}{\sigma_X^N})}{f_X(x^*)} \\ \sigma_X^N &= \frac{\phi(\Phi^{-1}(F_X(x^*))}{f_X(x^*)} \end{aligned} \tag{3.11}$$

Here, $f_X(x^*)$ is the PDF of the non-normal distribution at point x^*.

From Equation 3.10 and Equation 3.11, we can compute the equivalent mean μ_X^N and standard deviation σ_X^N at the point x^*. These will be used to carry out the computation of the reliability when the input variables are non-normal.

Since we have established the required formulations, the algorithm followed to implement the HL-RF method is discussed below:

1. Compute an initial guess for the reliability index β^* assuming the variables are uncorrelated normal variables using the Mean Value First Order Second Moment (MVFOSM) method discussed in Section 3.2.

2. Using the current design point x^* (the mean value for the first iteration), the equivalent normal mean and standard deviation is computed using Equation 3.10 and Equation 3.11. next the correlated random variables are decoupled using Equation 3.12.

$$\bar{X} = [\sigma_{x^*}^N][T]\bar{y} + \bar{\mu}_{x^*}^N \tag{3.12}$$

   where, $[T]$ is the eigen vector matrix of the correlation matrix and $\bar{y}$ is unknown.

3. After finding a decoupled random variables at the point of interest we can find the value of the limit-state function as a function of the unknown $\bar{y}$. The gradient of the limit-state function is also computed with respect to the unknown $\bar{y}$.

$$g(X) \Rightarrow g(y) \Rightarrow \frac{\partial g}{\partial y} \tag{3.13}$$

4. Next, using the gradient found in Equation 3.13, the direction cosines α_i at the design point x^* is computed using Equation 3.14

$$\alpha = \frac{\frac{\partial g}{\partial y} \times \sqrt{\lambda}}{\sqrt{\sum \left(\frac{\partial g}{\partial y}^2 \times \lambda \right)}} \tag{3.14}$$

   Where, λ are the eigen values of the correlation matrix.

5. The new design point is now computed in the independent normal domain using Equation 3.15

$$\bar{y}^* = -\alpha\sqrt{\lambda}\beta \tag{3.15}$$

6. Finally the new design point is transformed back into the physical domain using Equation 3.16

$$\bar{X}^* = [\sigma_{x^*}^N][T]\bar{y}^* + \bar{\mu}_{x^*}^N \tag{3.16}$$

7. The change in α is checked with the previous iteration to be within the user defined specification else steps 2 - 6 are repeated until there is convergence.

8. once the value of α converges, The new value of β is computed by:

   (a) Setting β as an unknown in Equation 3.15.

   (b) Substituting the resulting $\bar{y}^*$ in Equation 3.12. This results in a new design point x^* as a function of unknown β.

(c) This new design point is used to compute the value of the limit-state function $g(x^*)$.

(d) Finally the new value of β is computed by substituting $g(x^*) = 0$, this can be done by using any solver.

9. Using the new value of β, the probability of failure, reliability of the system are calculated based on Equation 3.5.

10. The change in the value of β is computed with respect to the previous iteration. if the values is below the convergence criteria the computation is stopped else steps 2 - 9 are repeated.

The steps involved in the HL-RF method are shown as a flow diagram in Figure 3.4

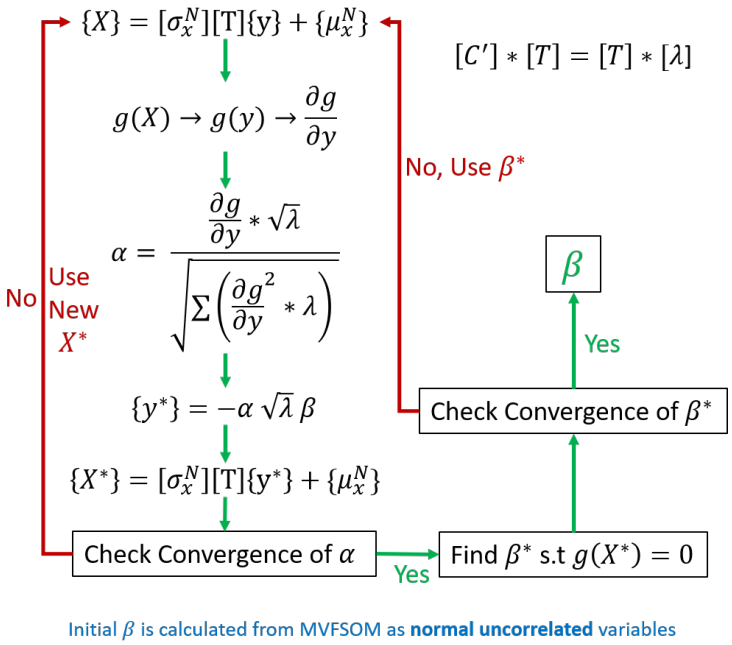

Figure 3.4: Flowchart of the steps followed in computing the reliability usign HL-RF method.

3.4 Monte Carlo Simulation Method

There are several methods at various degrees of complexity that can be used to estimate the reliability of a limit-state function. Most of these methods rely on some sort of approximation in terms of a linearizartion limit-state function or estimation of the random variables as to a random normal variables. The simplest form of computing the reliability of a system is to generate random variable and individually test the limit-state function, This method is commonly referred to as the Monte Carlo Simulation (MCS) method.

The current implementation of the MCS method assumes that the properties of the variable distributions are encoded within the covariance matrix $[C]$. The random design point $\bar{x}$ is computed as,

$$\bar{x} = [L]\bar{y} + \bar{\mu_X} \tag{3.17}$$

Here, $[L]$ is the lower triangle Cholesky decomposition of $[C]$ such that, $[C] = [L][L]^T$. $\bar{y}$ is a random sample taken from a standard normal distribution. Note, the realtion between the covariance matric and the coerrelation matrix is given by Equation 3.18.

$$C_{ij} = C'_{ij} \times \sigma_i \times \sigma_j \tag{3.18}$$

where, i, j go from 1 to the total number of variables N.

The assumptions that the distributions properties are encoded into the covariance matrix does have its drawbacks in terms of distributions that are not similar in nature to the normal distribution. Future releases of ReliaComp will address implementing a strong, correlated distribution independent sampling technique.

Chapter 4

Examples

This chapter goes through a few examples that have been setup to help the user navigate ReliaComp. These examples can be used as a reference to setup the users own limit-state function. Table 4.1 summarizes the different examples and what capabilities of ReliaComp they can be used with. further details of these examples are described in the following sections.

Table 4.1: Table of examples provided with ReliaComp, and the modeules they work with

Eample	**HL-RF**	**Monte Carlo**	**MVFOSM**
Simple structural problem	Yes	Yes	Yes
Blade deflection using simple beam model	No	Yes	Yes
Blade deflection using TANA approximation	Yes	Yes	Yes

4.1 Simple structural problem

This example follows the one on page 93 of Reliability Assessment using Stochastic Finite Element Anmalysis by Halder and Mahadevan [Haldar, 2000]. A steel structural beam is subjected to a bending moment of 1140kip-in. The yield stress of steel Fy is $36ksi$, and the nominal plastic modulus of the section Z is $54in^3$. Consider Fy follows a log-normal distribution with COV of 10% and Z follows normal distribution with a COV of 5%. Let the correlation between the variables by 0.3. Based on stress on a beam in bending we can write the limit state function as,

$$g() = F_y Z - 1140 = 0 \tag{4.1}$$

The user can locate this example in the "TextBookExample" folder and by loading the *MahadevanFuncP93.mat* file. This example is also used to validate the methods implemented in ReliaComp.

4.2 Blade deflection using simple beam model

In this example we look at a deflection of a wind turbine blade under static loading at a test facility. This example has been taken from a real life bade that was developed, manufactured and tested [Yao et al., 2020, Kaminski et al., 2020]. Parts of this work are still in the process of being published hence a few functions are saved as content-obscured p-files.

The blade is represented as a Euler beam finite element model [Przemieniecki, 1985] and PreComp [Bir, 2006] is used to compute the sectional properties of the blade from a detailed NuAMD [Berg and Resor, 2012] model. Figure 4.1 shows the coordinates systems used to define the problem as well as the location of the loads applied during static testing of the blade. The deflection is analyzed at the tip of the blade at the trailing edge.

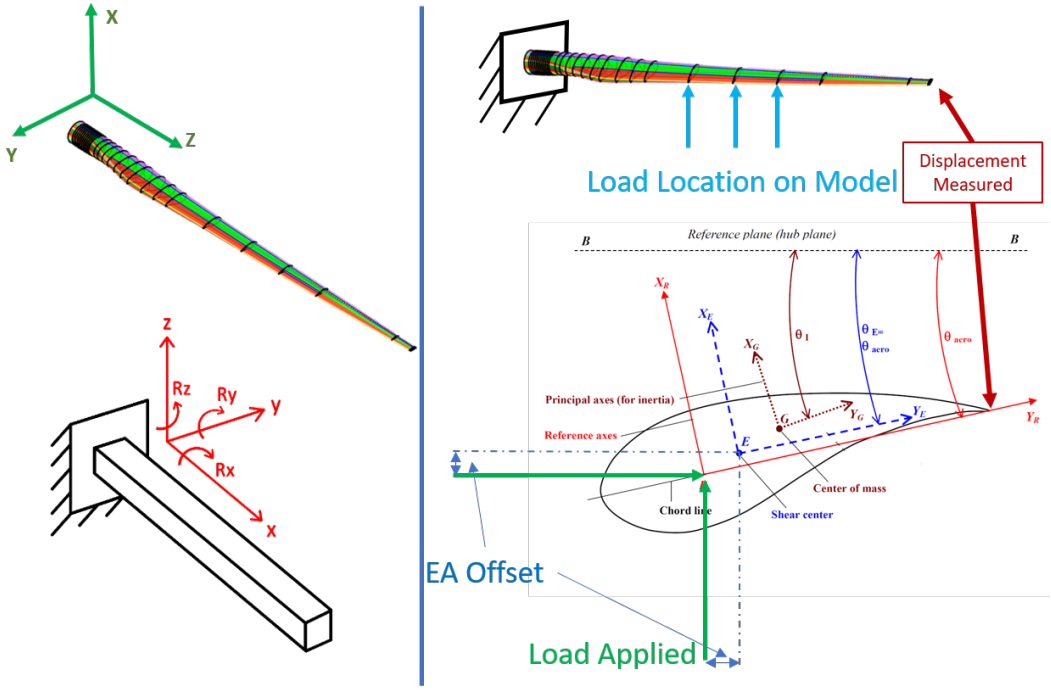

Figure 4.1: The left part shows the two frames of references used in this example, The right shows the loading on the blade as well a the point at which the tip deflection is measured.

For this example we assumed that the blade is made of a single material and that the material properties are the input variables. The limit-state function as defined such that the blade tip is not allowed to deflect more than 0.8m. The problem is deifined as below.

$$\begin{aligned} E11 &\sim log - normal(354910, \delta = 0.05)MPa \\ E22 &\sim log - normal(14330, \delta = 0.05)MPa \\ E12 &\sim log - normal(3510, \delta = 0.05)MPa \\ g() &= 0.8 - Disp(Tip) \end{aligned} \tag{4.2}$$

$$[C'] = \begin{bmatrix} 1 & 0.1 & 0.15 \\ 0.1 & 1 & 0.05 \\ 0.15 & 0.05 & 1 \end{bmatrix} \tag{4.3}$$

The user can locate this example in the "BladeModelFEM" folder and by loading the *BladeModelFEM.mat* file. Due to the current limitation of ReliaComp, this example can only be run using the Monte Carlo and MVFOSM method.

4.3 Blade deflection using TANA approximation

Since the previous example cannot be used in the HL-RF method due to limitation in finding analytical gradients. We use a functional approximation of the previous example using Two Point Adaptive Nonlinearity Approximation (TANA) [Fadel et al., 1990]. The formulation of the problem follows the same as above. Figure 4.2, shows the approximation of the FEM solution about the point of interest.

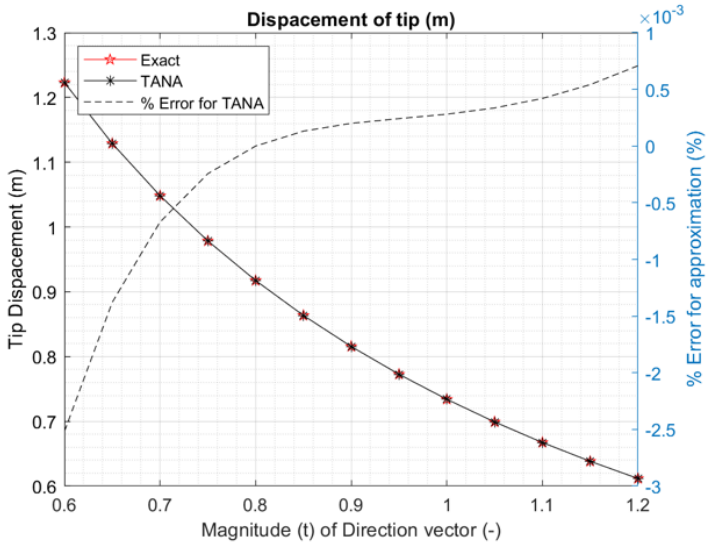

Figure 4.2: TANA approximation of blade tip deflection using FEM.

The user can locate this example in the "BladeModelTANA" folder and by loading the *BladeModelTANA.mat* file. A simple script to fit a TANA and TANA2 [Grandhi and Wang, 1998] approximation is provided.

Chapter 5

Future Work

ReliaComp is an easy to use tool to calculate the reliability of a given limit-state function. Currently it preforms three different methods of reliability computation for variables with normal, log normal and uniform distributions. The tool has a few limitation that will be addressed in future releases. Few of the future improvements to the tool are:

1. Allow the users to provide numerical limit-state functions while using HL-RF method.
2. Allow users to provide analytical gradient in the limit-state functions.
3. Allow non-normal sampling of correlated random variables for Monte Carlo Simulation method.
4. Add Second Order Reliability Methods.
5. General code and speed improvements to the code.

Bibliography

[Berg and Resor, 2012] Berg, J. C. and Resor, B. R. (2012). Numerical manufacturing and design tool (numad v2. 0) for wind turbine blades: User's guide. *Sandia National Laboratories Technical Report, SAND2012-7028*.

[Bir, 2006] Bir, G. S. (2006). User's guide to precomp (pre-processor for computing composite blade properties). Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States).

[Fadel et al., 1990] Fadel, G., Riley, M., and Barthelemy, J. (1990). Two point exponential approximation method for structural optimization. *Structural optimization*, 2(2):117–124.

[Fiessler and Rackwitz, 1978] Fiessler, B. and Rackwitz, R. (1978). Structural reliability under combined random load sequences. *Comput Struct*, 9(5):489–94.

[Grandhi and Wang, 1998] Grandhi, R. V. and Wang, L. (1998). Reliability-based structural optimization using improved two-point adaptive nonlinear approximations. *Finite Elements in Analysis and Design*, 29(1):35 – 48.

[Haldar, 2000] Haldar, A. (2000). *Reliability assessment using stochastic finite element analysis*. John Wiley & Sons, New York ;.

[Kaminski et al., 2020] Kaminski, M., Noyes, C., Loth, E., Damiani, R., Hughes, S., Bay, C., Chetan, M., Griffith, D. T., Johnson, K., and Martin, D. (2020). Gravo-aeroelastic scaling of a 13-mw downwind rotor for 20% scale blades. *Wind Energy*.

[Przemieniecki, 1985] Przemieniecki, J. S. (1985). *Theory of matrix structural analysis*. Courier Corporation.

[Yao et al., 2020] Yao, S., Griffith, D. T., Chetan, M., Bay, C. J., Damiani, R., Kaminski, M., and Loth, E. (2020). A gravo-aeroelastically scaled wind turbine rotor at field-prototype scale with strict structural requirements. *Renewable Energy*, 156:535 – 547.

Appendix A

Limit-state Function Example

Listing A.1: Exmaple for a limit-state function.

```
function val = LimitStateFunc(x)
%   This function is used calculate the value of the performance function
%   based on the example on page 93 of [1].
%
%   Input:
%           x         = the value about which to evaluate the the function
%                       [(n,1) vector]
%   Output:
%           val       = The value of the performance function at x
%                       [Scalar]
%
% [1] Haldar, Achintya, and Sankaran Mahadevan. Reliability assessment
%     using stochastic finite element analysis. John Wiley & Sons, 2000.
**************************************************************************Fy = x(1);Z = x(2);val = Fy .*Z - 1140;end
```