# User’s Guide to an Open Source Reliability Analysis software (ORA)

**Xiamen University; GuangZhou University; University of California, Berkeley**

**Introduction：**

An open source reliability analysis software (ORS) is developed for structural reliability analysis, which enhances an existing reliability framework in OpenSees, an open system for earthquake engineering simulation. The ORS is able to perform traditional time invariant or time variant reliability analyses, e.g., first order reliability analysis (FORM), second order reliability analysis (SORM), Monte Carlo simulation (MCS) and mean upcrossing rate analysis. It also includes some advanced reliability analysis method, e.g. subset simulation method. The ORS can be combined with other finite element analysis software (e.g., OpenSees or Abaqus) for reliability analysis of complicated structural, geotechnical or soil-structure interaction (SSI) problems. In this Manu, the commands for reliability analysis are illustrated in Chapter 1, and a set of examples for these analyses are shown in Chapter 2.

1. Commands for Reliability Analysis

The commands for reliability analyses are based on a tool command language，TCL [1]. Detailed explanation of some reliability commands can be found in the dissertations of Dr. Quan Gu and Dr. Terje Haukaas [2]-[3].

1.1. Uncertainty Modeling

A set of independent random variables are used to define the uncertainty in a reliability model. A random variable object can be created using ***randomVariable*** command, while the format may be slightly different for different type of random variables. A simple format is as following (note that a variable beginning with *$* character, and input values are to be an integer, floating point number or variable, and optional input values are identified in enclosing <> braces.):

***randomVariable $tag $type $mea $stdv <$startPt>***

The parameter ***$tag***is an integer tag of the random variable. The tags must be ordered sequentially beginning from 1. The parameters ***$mean*** and ***$stdv*** are the mean and standard deviation of the random variable respectively. The parameter ***$type*** indicates the distribution type of random variable, which may be normal, lognormal, uniform, chiSquare, exponential, gamma, gumbel, laplace, pareto, rayleigh, shiftedExponential, shiftedRayleigh, type1LargestValue, type1SmallestValue, type2LargestValue, type3SmallestValue, weibull and beta. Some distributions have more parameters than mean and the standard deviation, which may be described later. The ***$startPt*** argument allows the user to specify an initial value of the random variable to be used as the start point in the design point (DP) search algorithm. By default the ***$startPt*** is set to the mean.

Correlations between random variables may be specified in one of the following ways:

***correlation $rv1 $rv2 $correlation***

***correlationGroup $firstRV $lastRV $correlation***

The first command ***correlation*** specifies a correlation coefficient *$****correlation*** between random variables with tags ***$rv1*** and ***$rv2***. The second one ***correlationGroup*** specifies that every pairs of random variables between random variable with tag ***$firstRV*** and random variable with tag ***$lastRV*** are equi-correlated with a correlation coefficient of *$****correlation***.

1.2. Performance Functions

The performance functions, or limit state function (LSF) define the “safe” or “failure” states by using command ***performanceFunction***.

***performanceFunction $tag "$expression"***

The *$****tag*** is an integer tag of LSF. The ***$expression*** should be enclosed by double-quotes, which is an analytical expression of TCL and able to be evaluated by the Tcl interpreter [1]. Various quantities may be used in the expression, e.g., random variables, response quantities from the finite element analysis, the parameters defined in the Tcl, or data in a specified file. These quantities, however, must be defined in TCL and updated by the reliability analysis algorithm, e.g., during the iterations of DP search. The syntax shown below should be used. Note that the curly braces are mandatory and that the finite element responses are provided in a file named PerformanceU.txt and PerformanceV.txt if the LSF need the displacement and velocity of the structural node. The data in the file are calculated and recorded by the other finite element analysis software (e.g., OpenSees in this paper [4]). A process of reading data and assigning values to the expression is to trigger before LSF is performed.

**• *{x\_5}: random variable number 5.***

***• {u\_5\_2}: displacement of node 5 along dof number 2.,***

***• {ud\_5\_2}: velocity of node 5 along dof number 2.***

1.3. Analysis conditions

Before a reliability analysis (e.g., FORM or SORM) is performed, a set of commands need to be used to specify the computational methods for each sub-step of the reliability analysis, e.g., probability transformation method, evaluation of LSF, gradient of LSF, DP search. These commands need to be used in a specified sequence, since some commands depend on others.

A ***probabilitytransformation*** object specifies how to transform random variables between the original and standard normal spaces. Currently, the Nataf model is the available [3]. The command is:

***probabilityTransformation Nataf <-print $flag>***

The optional print ***$flag*** is 0 by default but can be set to 1 to print the random variables to screen.

A ***gFunEvaluator*** object is used to specify how to evaluate (i.e., compute the value of) the performance functions (also denoted as LSFs or g-functions) for a given realization of the random variables. The following three methods are available:

***gFunEvaluator Basic***

***gFunEvaluator OpenSees Timeinvariant -file $filename***

***gFunEvaluator OpenSees Timevariant -analyze $numIncr $dt -file $filename***

If the ***Basic*** evaluator is used (i.e., the first command), only random variables and variables in TCL can be included in the performance function. When ***OpenSees*** is specified as g-function evaluator (i.e., in the second and third commands), the OpenSees.exe will be called each time when the performance function need to be evaluated. In this process, the values of random variable are created by ORA, recorded in a file named RV.txt, and assigned to the parameters of FE analysis. The response quantities from FE analysis may be included in the performance function. ***Timeinvariant*** or ***Timevariant*** denotes that the reliability analysis is time invariant or time variant. The user should provide a file with name ***$filename*** for detailed TCL commands of FE analysis. For time variant reliability analysis, the user may specify a time when the time variant reliability analysis, or more precisely, the mean-upcrossing rate, is to be performed. The time is the ***numIncr*** multiplied ***dt***, denoting the number of step multiplied by the analysis time step ***∆t***.

The **gradGEvaluator** command is used to specify how to calculate the gradients of the performance functions. Currently only forward finite difference method is available as,

***gradGEvaluator FiniteDifference <-pert $arg1> <-check>***

The optional **<-pert *$arg1*>** specifies the perturbation size as **1/***$****arg1***, and the default perturbation size is 1/1000. The ***-check*** flag is available to have the gradient vector printed to the screen.

A **searchdirection** object specify the algorithm to get the search direction for searching the design point (DP), which may be

***searchDirection iHLRF***

***searchDirection PolakHe***

***searchDirection SQP $c\_bar $e\_bar***

***searchDirection GradientProjection***

The iHLRF algorithm is an improved HLRF algorithm by adding a line search scheme [3]. PolakHe is algorithms for constrained optimization problems with inequality constraints [3]. The Sequential Quadratic Programming (SQP) algorithm is presented for particular optimization problem [3] with two parameters of the SQP algorithm ***$c\_bar*** and ***$e\_bar***. The GradientProjection is another algorithm for search direction [3].

The **stepSizeRule** command calculates the step size along a search direction in the line search algorithm. Two selection are available at this time.

***stepSizeRule Fixed $stepSize***

***stepSizeRule Armijo -maxNum $arg1 -base $arg2 <-print $arg3> <-initial $b0 $numSteps> <-sphere $radius $dist $evol>***

The ***Fixed*** selection (i.e., the first command) specifies a constant stepsize as ***$stepSize*** given by users. An alternative one is the ***Armijo*** line search algorithm, i.e., the second command. where ***arg1*** represents the maximum number of step size reductions before the errors are accepted; ***arg2*** represents the base number *b* in the step size value , where *k* ≥ 0 is the smallest integer satisfying the merit function check. The optional ***-initial*** and ***-sphere*** flags are available to avoid trial steps too far out in the failure domain. With the ***-initial*** option the user specifies the value b0 = 1.0 after ***numSteps*** trial steps. The alternative ***–sphere*** option allows the user to define a hyper-***sphere***, within which the trial steps are restricted to stay. The sphere is defined in the standard normal space and its radius as ***$radius*** given by users. The evolution of the radius is happened based on the input argument ***$evol*** and ***$dist*** ifthe design point is not be located inside the hyper-sphere.

The default values of the parameters for the Armijo algorithm are as follows: arg1 = 10, arg2 = 0.5, b0 = 1.0, numSteps = 2, radius = 50.0, dist = 0.1 and evol = 0.5. The user may consider using other parameters in typical finite element reliability problems.

**A root finding algorithm** is used to find the optimum point, i.e., the DP, by taking advantages of the previous defined objects of gradient and limit-state function.

***rootFinding type -maxIter $arg1 -tol $arg2 -maxStepLength $arg3***

Currently, the ***type*** may be ***Secant***, denoting the secant method for the rooting finding algorithm. The maximum iteration number is ***$arg1*** and ***$arg2*** denotes the target tolerance measure on the performance function value, scaled by the initial value. The argument ***$arg3*** is be used to constrain the length of the steps.

A **meritFunctionCheck** command defines amerit function used to determine the suitability of a step size:

***meritFunctionCheck AdkZhang -add $arg1 -multi $arg2 -factor $arg3***

***meritFunctionCheck PolakHe -factor $arg1***

***meritFunctionCheck SQP -factor $arg1***

***meritFunctionCheck criteriaReduction***

The ***AdkZhang*** selection (i.e., the first command) uses the arguments ***$arg1*** and ***$arg2*** to compute the factor *c* so that by the following equation. Default values are ***arg2***=2 and ***arg1***=10. The argument **-factor** (e.g., in the first, second, third command) is used in the equation  (old and new trial point) ≤ factor (...), which is a typical format of a merit function check. The **criteriaReduction** type requires that all convergence criteria be improved to accept the step size.

**A reliabilityConvergenceCheck** command determines whether the reliability analysis has converged by checking the convergence criteria and target tolerance (see Section 1.4):

***reliabilityConvergenceCheck Standard -e1 $arg1 -e2 $arg2***

The first convergence check ***e1*** determines the closeness of the DP to the limit-state surface (i.e.,). The second convergence check ***e2*** determines how closely the gradient vector points towards the origin in the standard normal space ((i.e., ).

A ***startPoint*** is used to specify the initial value used for the DP search algorithm, which may be the mean of the random variables, origin (i.e., the zero point in the standard normal space), a specified initial DP vector or data in a file, corresponding to the following commands, respectively,

***startPoint Mean***

***startPoint Origin***

***startPoint Given***

***startPoint -file filename***

An **findDesignPoint** command specify the method to search the DP, currently a step-by-step search scheme ***StepSearch*** is available, i.e.,

***findDesignPoint StepSearch -maxNumIter $arg1 <print option>***

The ***$arg1*** is the maximum number of iterations for DP search. The print option enables the user to have the trial points or the design point printed, e.g., to a file named ***filename***:

***-printAllPointsX filename***

***-printAllPointsY filename***

***-printDesignPointX filename***

***-printDesignPointY filename***

***-printCurrentPointX filename***

***-printCurrentPointY filename***

Where the suffix X and Y denote the vector in the original space and standard normal space respectively.

After the DP is obtained, a **findCurvatures** command can be used to record the principal curvature [3], e.g., to get the first principal curvature:

***findCurvatures* firstPrincipal**

A ***transformXtoU*** command transforms a point in the physical space (saved in pointX.out) into the standard normal space (output to the file pointu.out). And a ***transformUtoX*** command transforms a point in the standard normal space (saved in the file pointu.out) to the physical space (output to pointX.out).

***transformXtoU -fileX PointX.out -fileU pointu.out***

***transformUtoX -fileX PointX.out -fileU pointu.out***

A ‘***probabilityTransformation***’ command must be previously defined before this command is used.

It is first necessary to compute the Hessian matrix at the design point. The Hessian is computed by perturbation of the sensitivity results. i.e., .

The Tcl command is:

***computeHessian -FDM -file $fileName -designPoint $designPtFile –perturbation $perturbSize***

Where ***$fileName*** is the name of the file in which the hessian matrix will be stored. ***$designPtFile*** is provided by the user and contains the previously computed design point in a single column (i.e., each value of RVs saved in one line). ***$perturbSize*** is the size of the perturbation, default value is 1e-5.

1.4. Analysis and output

There are different types of analyses available in ORA. A ***runFORMAnalysis*** command performs the first order reliability analysis:

***runFORMAnalysis outputfilename <-relSens $arg1>***

The optional ***-relSens*** describes whether the reliability sensitivities (i.e., the sensitivity of the reliability index and estimated probability of failure with respect to means and standard deviations) need to be calculated and the flag ***$arg1*** is set to 1 if yes. Before the FORM analysis (and other analyses explained later), necessary analysis conditions need to be specified, e.g., methods of probability transformation and a DP search must be created.

The FORM analysis results in the output file are self-explanatory (i.e., users can understand the data from file). The vectors *x\**, *u\**, *alpha*, *gamma*, *delta* and *eta* are:

• *x\** is the design point of the random variables in the original space.

• *u\** is the design point of the random variables in the standard normal space.

• *alpha* is the negative normalized gradient of the limit-state function at the design point in the standard normal space.

• *gamma* is the importance vector for the original random variables when correlation is present. In the case of independent random variables gamma=alpha.

• *delta* is the sensitivity of the reliability index *β* with respect to (w.r.t.) the mean of each random variable, scaled by the corresponding standard deviation.

• *eta* is the sensitivity of the reliability index *β* with respect to the standard deviation of each random variable, scaled by the standard deviation.

A ***runFOSMAnalysis*** command is used to performthe first-order second-moment (FOSM) reliability analysis and the results in the output file **outputfilename** are self-explanatory. A DP search (e.g., FORM) must be performed before this analysis.

***runFOSMAnalysis* outputfilename**

A ***runSORMAnalysis*** command performs the second order reliability analysis and output the results to a self-explanatory file. A DP search (e.g., FORM) must be performed before this analysis:

***runSORMAnalysis* outputfilename**

There are several sampling methods available in ORA [2], including Importance Samplint (IS), Orthogonal Plane Sampling (OPS), and crude Monte Carlo Sampling (MCS), and the corresponding analysis commands are illustrated in the following:

***runImportanceSamplingAnalysis $filename -type failureProbability -variance $var -maxNum $maxNum -targetCOV $cov -print $printflag***

The analysis results will be recorded in the output file ***$filename***. Where ***–variance*** denotes the standard deviation of the sampling distribution is ***$var*** (default = 1.0), ***-maxNum*** denotes the maximum number of simulations is ***$maxNum***, ***-targetCOV*** denotes the target coefficient of variation of the estimate is ***$cov*** (default = 0.05), ***-print*** is a print flag with the following meaning: 0, the status of the sampling analysis is not printed to the screen or file; 1, the status after each sample is printed only to the screen; 2, the status after each sample is printed only to the screen, while necessary information is printed into a file as well such that the sampling analysis may be restarted (i.e., analysis will continue from current point at the next run). The selection 2 is very useful for computational cases that need a large number of FE simulations.

***runOrthogonalPlaneSamplingAnalysis -fileName $fileName -maxNum $maxNum -type $type -targetCOV $cov -print $printFlag < -funcTol $funTol -varTol $varTol -max-Iter $NumIter -littleDt $little\_dt>***

The definition ***-fileName***, ***-maxNum***, ***-targetCOV***, ***-print*** are the same as those in the ‘runImportanceSamplingAnalysis’ command. Where***–type*** denotes the reliability type is ***$type***, which may be ‘failureProbability’ or ‘outCrossing’, corresponding to the time invariant (failure probability computation) and time variant (mean upcrossing rate computation) reliability analysis, respectively. ***-funcTol*** denotes the tolerance of limit-state function (LSF) is ***$funTol***, such that the zerofinding algorithm stops when **|G|<*$funTol***. ***–varTol*** denotes the tolerance criterion between two neighboring *x* is ***$varTol***: when <*$varTol*, the zerofinding algorithm will also stop. ***-max-Iter*** denotes the maximum number of iterations in the zerofinding algorithm is ***$NumIter***. The default values for ***$funTol***, ***$varTol***, and ***$NumIter*** are 1e-5, 1e-3, and 20 respectively. ***-littleDt*** denotes the small time increment between the two LSF is ***$little\_dt* ()**.

***runMonteCarloResponseAnalysis -outPutFile $fileName -maxNum $maxNum -print $printFlag -tclFileToRun $tclfile***

The definition ***-outPutFile***, ***-maxNum***, ***-print*** are the same as before, while ***-tclFileToRun*** specifies the name of a tcl file that specifies the analysis is ***$tclfile***.

A ***runSubsetSimulationAnalysis*** command is used to perform subset simulation analysis and the results in the output file **outputfilename** are self-explanatory.

***runSubsetSimulationAnalysis -outPutFile $fileName -SamplingType $arg1 <$arg2> <-NumSeedSamples $arg3> -print $printFlag***

The definition ***–fileName*** and ***-print*** are the same as those in the previous command. ***–SamplingType*** denotes the sampling method in subset simulation process is ***$arg1***, which may be ‘MH’ and ‘HMC’, corresponding to Metropolis Hasting and Hamiltonian Monte Carlo method [5]. ***$arg2*** is the special input argument when MH method is chosen. The value of the ***$arg2*** with the following meaning: 1, the proposal distribution in MH sampling method is a standard normal distribution; 2, the proposal distribution is a uniform distribution with mean zero and width two. The optional ***-NumSeedSamples*** denotesthe number of samples in each subset is ***$arg3*** (default=1000).

**A *runOutCrossingAnalysis*** command is created to perform dynamic reliability analysis by the mean upcrossing rate and the corresponding results are recorded in the output file named ***$outputfilename.***

***runOutCrossingAnalysis $outputfilename -results*** ***$stepsToStart $stepsToEnd $freq $sampleFreq -littleDt $little\_dt $Type***

***$stepsToStart*** and ***$stepsToEnd*** specify the start and end time steps at which the mean upcrossing rate is performed. ***$freq*** determines at which time point the mean upcrossing rate is evaluated between the start and end time steps. These time points are: ***$stepsToStart***, ***$stepsToStart***+***$freq***, ***$stepsToStart***+***2\*$freq***, ... until ***$stepsToEnd.*** ***$sampleFreq*** is the time interval ∆t between two neighboring impulses. The ***$little\_dt*** (*δt*) is the small time increment by which the second design point excitation is obtained through shifting the first design point, i,e,. . ***$Type*** can be specified as either -Koo or -twoSearches. The latter option implies that two design point searches will be performed for each evaluation point, while the former employs the method developed. It is assumed that the user has specified a discretized random process as time series for at least one load pattern and that the corresponding random variable positioners have been created to map random variables into this time series object.

A ‘***runMultiDimVisualPrinPlane***’ command uses to visualize the LSF in the first two principal planes. The command may take advantage of the hessian matrix obtained by ‘computeHessian’ command. In case that hessian matrix is not available, it will call the ‘computeHessian’ command implicitly to obtain the hessian matrix.

***runMultiDimVisualPrinPlane -funSurf function -designPt designPointX.out -ndir $n -output vis.out <-gridInfo {0 $minY $maxY $nPts0 1 $minX1 $maxX1 $nPts1 2 $minX2 $maxX2 $nPts2 ...} -timeVariant -littleDt $little\_dt> <-saveHessian $filename>***

where the previously computed design point is stored in the file designpointx.out. ***–ndir*** denotes the number of principal axes direction is ***$n***. the first principal axes and the design point direction are used to fit the response surface. The visualization results of LSF will be saved in the file vis.out. And the output data format inside vis.out is: ‘ x y G(x,y) ’ for each point in each line. An optional ***-gridInfo*** specifies the information of grid point. Along the design point axis, there are ***$nPts0*** points between [***$minY $maxY***]; Along the first principal axis, there are ***$nPts1*** points between [***$minX1 $maxX1***]; Along the second principal axis, there are ***$nPts2*** points between [***$minX2 $maxX2***]. ***-littleDt*** denotes the small time increment between the two LSF is ***$little\_dt* ()**.Where ***$fileName*** is the name of the file in which the hessian matrix will be stored, in case the Hessian matrix is not provided by the user, the command comupteHessian is called;

The ***runDP\_RSM\_SimTimeInvariantAnalysis*** command is used to compute the failure probability for time invariant reliability analysis.

***runDP\_RSM\_SimTimeInvariantAnalysis -designPt designPointX.out -output results.out -ndir $n <-experimentalPointRule Uniform -gridInfo {-1 minY maxY nPts 0 minY maxY nPts0 1 minX1 maxX1 nPts1 2 minX2 maxX2 nPts2 ...}> -saveHessian hession.out <-surfaceDesign $type -simulation $SamplingMethod -tarCOV $cov -numSimulation $maxNum >***

where designPointX.out contains the previously computed design point; results.out saves the output; ***–ndir*** denotes the number of principal axes direction is ***$n*** (e,g,. ‘***$n*** =2’ indicates that the first 2 principal axes and the design point direction are used to fit the response surface (i.e., along these three directions, LSF is nonlinear, while along all other axes, the LSF is considered as linear plane), thus the number of grid planes is computed by. Note that the design point direction is an extra direction used to fit the response surface; ‘***Uniform***’ means that the grid point is uniformly distributed along each axis; ***-gridInfo*** and ***-saveHessian*** command are the same as in the runMultiDimVisualPrinPlane command; ***$type*** specifies how to fit the response surface, currently ‘UnivariateDecomposition’ and BivariateDecomposition’ are available; ***$samplingMethod*** describes which sampling method is used to get the failure probability after the response surface is obtained, currently only ‘ImportanceSampling’ is available; ***$cov***, ***$maxNum*** are the same as those in the runOrthogonalPlaneSamplingAnalysis command. The command will work only after the following commands are used:

***performanceFunction***

***probabilityTransformation***

***gFunEvaluator***

***randomNumberGenerator***

The ***runDP\_RSM\_SimTimeVariantAnalysis*** command is used to compute the mean upcrossing rate for time variant reliability analysis.

***runDP\_RSM\_SimTimeVariantAnalysis -designPt dp.out -output results.out -ndir $n <-experimentalPointRule Uniform -gridInfo {-1 minY maxY nPts 0 minY maxY nPts0 1 minX1 maxX1 nPts1 2 minX2 maxX2 nPts2 ...}> -saveHessian hession.out <-surfaceDesign $type -simulation $SamplingMethod -tarCOV $cov -numSimulation $maxNum -littleDt $little\_dt -ImpulseInterval $delt\_t >***

Most of the parameters in this command are the same as for the command ***runDP\_RSM\_SimTimeVariantAnalysis***, while two more parameters are necessary: ***$little\_dt*** is the same as that in ***runOrthogonalPlaneSamplingAnalysis***. ***$delt\_t*** is the time interval between two sequential random variables (or impulse in white noise excitation case).

2. Numerical Examples

A number of examples are presented to demonstrate the new features and analysis capabilities of ORA. The presentation starts with simple problems and proceeds to a comprehensive example involving structural reliability analysis. The chapter includes traditional time invariant or time variant reliability cases, and the available analysis types and corresponding execution commands in ORA are summarized in Table1.

Table1: Available analysis types and corresponding execution commands of reliability analysis in ORA

|  |  |
| --- | --- |
| Analysis types | Executing commands |
| 1. FORM Analysis | runFORMAnalysis |
| 1. SORM Analysis | runSORMAnalysis |
| 1. Importance Sampling Analysis | runImportanceSamplingAnalysis |
| 1. Orthogonal Plane Sampling Analysis | runOrthogonalPlaneSamplingAnalysis |
| 1. Monte Carlo simulation | runMonteCarloResponseAnalysis |
| 1. Subset Simulation Analysis | runSubsetSimulationAnalysis |
| 1. Mean Upcrossing Rate Analysis | runOutCrossingAnalysis |

2.1. FORM

The first example model is termed “FORM”. The performance function is defined explicitly in terms of the basic random variables. Three random variables are present. The first is assigned a lognormal probability distribution with mean 500 and standard deviation 100. The second is also lognormal but with mean 2000 and standard deviation 400. The last random variable is assigned the uniform probability distribution with mean 5 and standard deviation 0.5. The correlation structure between the random variables is specified in terms of the following correlation matrix:

 (2.1)

The performance function reads:

 (2.2)

where *par1* = 1.0 is a deterministic parameter.

First a FORM reliability analysis is performed. Initially, The Tcl commands are as follows:

1. randomVariable 1 lognormal 500.0 100.0 500.0
2. randomVariable 2 lognormal 2000.0 400.0 2000.0
3. randomVariable 3 uniform 5.0 0.5 5.0
4. correlate 1 2 0.3
5. correlate 1 3 0.2
6. correlate 2 3 0.2
7. set a "{x\_2}/(1000.0\*{x\_3})"
8. set b "{x\_1}/(200.0\*{x\_3})"
9. performanceFunction 1 "1.0 - $a - $b\*$b"
10. probabilityTransformation Nataf -print 0
11. reliabilityConvergenceCheck Standard -e1 1.0e-3 -e2 1.0e-3 -print 1
12. gFunEvaluator Basic
13. gradGEvaluator FiniteDifference -pert 1000
14. rootFinding Secant -maxIter 50 -tol 1.0e-2 -maxStepLength 1.0
15. stepSizeRule Fixed -stepSize 1.0
16. searchDirection GradientProjection
17. startPoint Mean
18. findDesignPoint StepSearch -maxNumIter 100
19. runFORMAnalysis FORMoutput.out

As seen, the search for the DP is performed by the improved HL-RF algorithm and it starts at the mean point.

The search for the design point converges in 8 iterations. The following records are printed to the screen during the analysis:

FORM Analysis is running ...

Limit-state function number: 1

Limit-state function value at start point, g=0.35

STEP #0: check1=(1.000e+00), check2=(6.798e-02), dist=0.12684013810289

STEP #1: Armijo trial point rejected; reducing step size...

.......: check1=(4.168e-01), check2=( 8.174e-02), dist=1.16929159031478

STEP #2: check1=(3.782e-02), check2=( 1.197e-01), dist=1.83214258152454

STEP #3: check1=(6.233e-03), check2=( 7.790e-02), dist=1.76750010896384

STEP #4: Armijo trial point rejected; reducing step size...

.......: check1=(3.693e-03), check2=( 1.696e-02), dist=1.76760726868615

STEP #5: check1=(1.165e-04), check2=( 1.087e-02), dist=1.77230400319642

STEP #6: Armijo trial point rejected; reducing step size...

.......: check1=(6.924e-05), check2=( 2.166e-03), dist=1.77230381951648

STEP #7: check1=(6.385e-06), check2=( 1.268e-03), dist=1.77239263554812

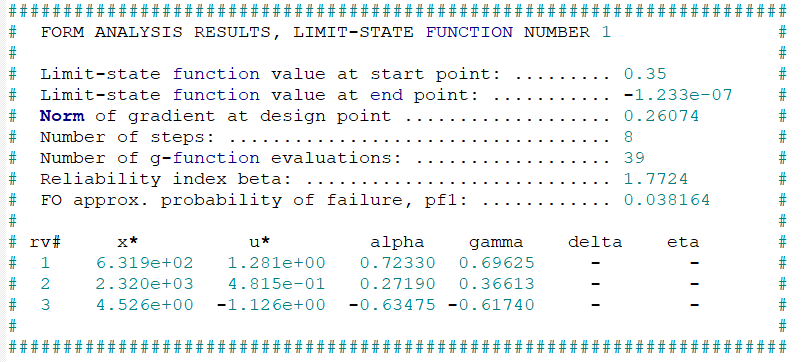
STEP #8: check1=(3.523e-07), check2=( 7.677e-04), dist=1.77239977866749

Design point found!

Done analyzing limit-state function 1, beta=1.7724

FORMAnalysis completed.

Upon convergence the following output is stored in the output file:



The 39 evaluations of the performance function include 9 evaluations of the performance function and its gradient by finite difference at the start point and the 8 trial points (9\*4 = 36) plus 3 evaluations during line searches that required reduction of the step size.

It is interest to investigate the performance of the alternative search algorithms for this problem. First the Gradient Projection algorithm is employed. To this end, the meritFunctionCheck object is removed from the input file and a root-finding algorithm is introduced. In addition, the step size and search direction algorithms are modified:

**rootFinding Secant -maxIter 50 -tol 1.0e-2 -maxStepLength 1.0**

**stepSizeRule Fixed -stepSize 1.0**

**searchDirection GradientProjection**

The following output is displayed during the analysis:

FORM Analysis is running ...

Limit-state function number: 1

Limit-state function value at start point, g=0.35

STEP #0: check1=(1.000e+00), check2=( 6.798e-02), dist=0.12684013810289

STEP #1: check1=(4.451e-03), check2=( 2.223e-01), dist=1.79890011702173

STEP #2: check1=(1.298e-05), check2=( 1.258e-01), dist=1.78187196916584

STEP #3: check1=(5.475e-03), check2=( 7.073e-02), dist=1.76772624405465

STEP #4: check1=(2.294e-07), check2=( 4.237e-02), dist=1.77341814511813

STEP #5: check1=(6.778e-04), check2=( 2.515e-02), dist=1.77182536326680

STEP #6: check1=(1.032e-06), check2=( 1.520e-02), dist=1.77252983416550

STEP #7: check1=(9.100e-05), check2=( 9.144e-03), dist=1.77232511546465

STEP #8: check1=(1.233e-04), check2=( 5.563e-03), dist=1.77225102131038

STEP #9: check1=(1.294e-04), check2=( 3.361e-03), dist=1.77222359347395

STEP #10: check1=(1.415e-04), check2=( 2.039e-03), dist=1.77221358252083

STEP #11: check1=(1.386e-04), check2=( 1.233e-03), dist=1.77220990030124

STEP #12: check1=(1.377e-04), check2=( 7.478e-04), dist=1.77220855261669

Design point found!

Done analyzing limit-state function 1, beta=1.77221

FORMAnalysis completed.

The number of steps with the Gradient Projection algorithm is comparable to the iHLRF algorithm. However, the Gradient Projection algorithm requires 67 evaluations of the performance function compared to 39 for the iHLRF algorithm. This is due to the root-finding procedure of the Gradient Projection algorithm to project the trial points onto the limit-state surface. It is verified in the output file that the same design point is found.

Next, the Polak-He algorithm is employed. For this purpose the root-finding algorithm is removed and the Armijo step size rule from the iHLRF analysis is re-introduced. Furthermore, the search direction and the merit function check are specified as:

**searchDirection PolakHe**

**meritFunctionCheck PolakHe -factor 0.5**

In addition, the performance function must be scaled due to the properties of the Polak-He algorithm discussed. As seen above, the value of the performance function at the mean point is 0.35. Based on this information, it is selected to multiply the performance function by a factor of 15.0, that is:

**performanceFunction 1 "15.0\*(1.0 - $a - $b\*$b)"**

Keeping all other input the same, the following analysis progress is now observed:

FORM Analysis is running ...

Limit-state function number: 1

Limit-state function value at start point, g=5.25

STEP #0: check1=(1.000e+00), check2=( 6.798e-02), dist=0.12684013810289

STEP #1: check1=(2.451e-01), check2=( 2.803e-01), dist=2.14340869599745

STEP #2: check1=(8.753e-02), check2=( 1.006e-01), dist=1.89519004591045

STEP #3: check1=(3.075e-02), check2=( 1.551e-02), dist=1.81369442648341

STEP #4: check1=(9.815e-03), check2=( 6.251e-03), dist=1.78557615897402

STEP #5: check1=(3.084e-03), check2=( 1.602e-03), dist=1.77653925453772

STEP #6: check1=(9.655e-04), check2=( 4.771e-04), dist=1.77369409062779

Design point found!

Done analyzing limit-state function 1, beta=1.77369

FORMAnalysis completed.

In this case the total number of evaluations of the performance function is only 28 (4 per trial point due to the use of finite difference to compute the gradient). However, the scaling of the performance function has an important influence on the convergence of the Polak-He algorithm. The result shows the number of trial points for this algorithm with different start values of the performance function. It is clear that proper scaling of the performance function is essential when using the Polak-He algorithm.

Finally, the SQP algorithm is applied to the problem. The same input as the previous case is used, except for the following modifications:

**searchDirection SQP -c\_bar 200.0 -e\_bar 0.5**

**meritFunctionCheck SQP -factor 0.5**

This leads to the following display during the analysis:

FORM Analysis is running ...

Limit-state function number: 1

Limit-state function value at start point, g=0.35

STEP #0: check1=(1.000e+00), check2=(6.798e-02), dist=0.12684013810289

STEP #1: Armijo trial point rejected; reducing step size...

.......: check1=(4.168e-01), check2=( 8.174e-02), dist=1.16929159031478

STEP #2: Armijo trial point rejected; reducing step size...

.......: check1=(1.991e-01), check2=( 8.555e-02), dist=1.49956881044696

STEP #3: Armijo trial point rejected; reducing step size...

.......: check1=(9.661e-02), check2=( 3.124e-02), dist=1.64119272286979

STEP #4: check1=(2.840e-03), check2=( 4.043e-03), dist=1.77622127716843

STEP #5: check1=(1.187e-06), check2=( 8.522e-04), dist=1.77240501332517

Design point found!

Done analyzing limit-state function 1, beta=1.7724

FORMAnalysis completed.

As seen, 5 steps and 3 step size reductions are used in the SQP algorithm to find the design point; 6 \*4 + 3 = 27 calls were made to evaluate the performance function.

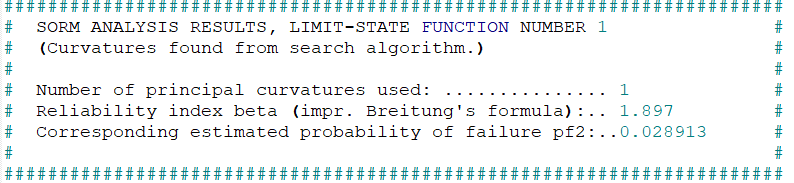
In summary, for this example the SQP algorithm is most efficient followed closely by the Polak-He algorithm when the performance-function is properly scaled. Next in efficiency comes the iHLRF algorithm while the Gradient Projection algorithm ranks last in efficiency. Obviously, one example is not sufficient to draw conclusions regarding the behavior of the different search algorithms. Results of further testing are presented in the following sections.

2.2. SORM

A SORM analysis is performed and the corresponding Tcl commands are as follows:

1. randomVariable 1 lognormal 500.0 100.0 500.0
2. randomVariable 2 lognormal 2000.0 400.0 2000.0
3. randomVariable 3 uniform 5.0 0.5 5.0
4. correlate 1 2 0.3
5. correlate 1 3 0.2
6. correlate 2 3 0.2
7. set a "{x\_2}/(1000.0\*{x\_3})"
8. set b "{x\_1}/(200.0\*{x\_3})"
9. performanceFunction 1 "1.0 - $a - $b\*$b"
10. probabilityTransformation Nataf -print 0
11. reliabilityConvergenceCheck Standard -e1 1.0e-3 -e2 1.0e-3 -print 1
12. gFunEvaluator Basic
13. gradGEvaluator FiniteDifference -pert 1000
14. rootFinding Secant -maxIter 50 -tol 1.0e-2 -maxStepLength 1.0
15. stepSizeRule Fixed -stepSize 1.0
16. searchDirection GradientProjection
17. startPoint Mean
18. findDesignPoint StepSearch -maxNumIter 100
19. runFORMAnalysis FORMoutput.out
20. findCurvatures firstPrincipal –exe
21. runSORMAnalysis SORMoutput.out

The current SORM implementation requires a FORM analysis to have been executed previously in the same session. And an algorithm to compute curvatures must have been created before the SORM object is created. The output file reads:



2.3. Importance Sampling Analysis

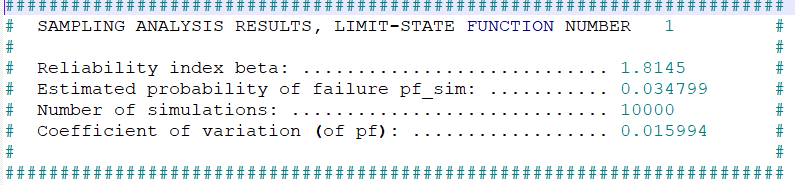
An importance sampling analysis around the design point is now performed by executing the following command (on one line):

*runSamplingAnalysis file.out -type failureProbability -variance 1.0 -maxNum 10000 -targetCOV 0.01 -print 0*

The corresponding Tcl commands are as follows:

1. randomVariable 1 lognormal 500.0 100.0 500.0
2. randomVariable 2 lognormal 2000.0 400.0 2000.0
3. randomVariable 3 uniform 5.0 0.5 5.0
4. correlate 1 2 0.3
5. correlate 1 3 0.2
6. correlate 2 3 0.2
7. set a "{x\_2}/(1000.0\*{x\_3})"
8. set b "{x\_1}/(200.0\*{x\_3})"
9. performanceFunction 1 "1.0 - $a - $b\*$b"
10. probabilityTransformation Nataf -print 0
11. reliabilityConvergenceCheck Standard -e1 1.0e-3 -e2 1.0e-3 -print 1
12. gFunEvaluator Basic
13. gradGEvaluator FiniteDifference -pert 1000
14. rootFinding Secant -maxIter 50 -tol 1.0e-2 -maxStepLength 1.0
15. stepSizeRule Fixed -stepSize 1.0
16. searchDirection GradientProjection
17. startPoint Mean
18. findDesignPoint StepSearch –maxNumIter 100 -printDesignPointX designPoint1.txt
19. runFORMAnalysis FORMoutput.out
20. startPoint -file designPoint1.txt
21. runImportanceSamplingAnalysis SIMULATIONoutput1.out -type failureProbability -variance 1.0 -maxNum 10000 -targetCOV 0.01 -print 0

The output file reads:



It is seen that after 10,000 samples the probability estimate is obtained with a coefficient of variation of 1.6%. The slightly higher reliability index of 1.81 from the sampling analysis as compared to the FORM value of 1.77 suggests that the limit-state surface is curved away from the origin around the design point.

2.4. Orthogonal Plane Sampling Analysis

An orthogonal plane sampling analysis is performed for single degree of freedom system. The truss element with steel01 material is used to model the system. Tcl commands of the reliability analysis are as follows:

1. randomVariable 1 lognormal 210000.0 [expr 0.05\*210000.0] 210000.0
2. randomVariable 2 lognormal 355.0 [expr 0.05\*355.0] 355.0
3. randomVariable 3 lognormal 337500.0 [expr 0.05\*337500.0] 337500.0
4. performanceFunction 1 "1.85-{u\_2\_1}"
5. probabilityTransformation Nataf -print 0
6. randomNumberGenerator CStdLib
7. reliabilityConvergenceCheck Standard -e1 1.0e-3 -e2 1.0e-3 -print 1
8. gFunEvaluator OpenSees Timeinvariant -file test9.tcl
9. gradGEvaluator FiniteDifference -pert 1000
10. searchDirection iHLRF
11. meritFunctionCheck AdkZhang -multi 2.0 -add 10.0 -factor 0.5
12. stepSizeRule Armijo -maxNum 50 -base 0.5 -initial 1.0 2 -print 1 -sphere 1000.0 1.0 1.0
13. startPoint Mean
14. findDesignPoint StepSearch -maxNumIter 100 -printDesignPointX designPoint1.txt
15. runFORMAnalysis FORMoutput.out
16. startPoint -file designPoint1.txt
17. runOrthogonalPlaneSamplingAnalysis -fileName Orthogonalplaneoutputoutput.out -maxNum 10000 -type failureProbability -targetCOV 0.01 -print 1

Tcl commands of the structural response analysis in OpenSees are as follows (test9.tcl), and detail described of Tcl commands for the structural response analysis can be found in [4].

1. model basic -ndm 2 -ndf 2
2. set GMScale\_sample [open RV.txt "r"];
3. set GMScale\_data [read $GMScale\_sample];
4. close $GMScale\_sample;
5. set E [lindex $GMScale\_data 0];
6. set fy [lindex $GMScale\_data 1];
7. set P [lindex $GMScale\_data 2];
8. node 1 0.0 0.0
9. node 2 5000.0 0.0
10. uniaxialMaterial Steel01 1 $fy $E 0.02
11. element truss 1 1 2 1000.0 1
12. fix 1 1 1
13. fix 2 0 1
14. pattern Plain 1 Linear {
15. load 2 $P 0.0
16. }
17. recorder Node nodeDisp.out -time -node 2 -dof 1 disp
18. constraints Plain
19. numberer RCM
20. test NormUnbalance 1.0e-6 25 0
21. integrator LoadControl 0.025 1 0.025 0.025
22. algorithm Newton
23. system ProfileSPD
24. integrator LoadControl 0.02 3 0.02 0.02
25. analysis Static
26. analyze 10
27. set u21 [nodeDisp 2 1]
28. set fo [open PerformanceU.txt w] ;
29. foreach different\_content {u21} {
30. puts $fo [set $different\_content]
31. }
32. close $fo

In this example, the value of random variables are sequentially recorded in a RV.txt file based on the tag of the variables (i.e., the first value is randomVariable 1). Before starting each finite element analysis, an updating process for the value of random variables are performed. After finishing each structural analysis, a displacement response value used to compute the LSF is recorded in a PerformanceU.txt file. The failure probability and the coefficient of variation are Pf = 0.0232153, c.o.v = 0.0100005, respectively.

2.5. Monte Carlo Sampling (MCS)

A Monte Carlo sampling analysis is performed for single degree of freedom system. The material and geometric parameters of the elastic element are determined. The elastic modulus is  and the size of the cross section is. A horizontal random load is applied to the top node.

Tcl commands of the reliability analysis are as follows:

1. randomVariable 1 normal 5000.0 [expr 0.05\*5000.0] 5000.0
2. performanceFunction 1 "0.13-{u\_2\_1}"
3. probabilityTransformation Nataf -print 0
4. randomNumberGenerator CStdLib
5. gFunEvaluator OpenSees Timeinvariant -file test12.tcl
6. runMonteCarloResponseAnalysis -outPutFile m.out -maxNum 3000 -print 1

Tcl commands of the structural response analysis in OpenSees are as follows (test12):

1. model basic -ndm 2 -ndf 3
2. node 1 0.0 0.0
3. node 2 0.0 6000.0
4. fix 1 1 1 1
5. fix 2 0 1 1
6. geomTransf Linear 1
7. element elasticBeamColumn 1 1 2 562500 3.0E+04 2.636719E+010 1
8. set F\_channel [open RV.txt "r"];
9. set FData [read $F\_channel];
10. close $F\_channel;
11. set F [lindex $FData 0];
12. pattern Plain 1 Linear {load 2 $F 0.0 0.0}
13. recorder Node -file DisplacementUnderDeadLoad.txt -time -node 2 -dof 1 disp
14. constraints Lagrange
15. numberer Plain
16. system BandGeneral
17. test EnergyIncr 1.0e-006 200
18. algorithm Newton
19. integrator LoadControl 0.5
20. analysis Static
21. analyze 2
22. set u21 [nodeDisp 2 1]
23. set fo [open PerformanceU.txt w] ;
24. foreach different\_content {u21} {
25. puts $fo [set $different\_content]
26. }
27. close $fo

The failure probability and the coefficient of variation are Pf = 0.00366667, c.o.v = 0.300958, respectively.

2.6. Subset Simulation Analysis

A subset simulation analysis is performed for single degree of freedom system. Tcl commands of the reliability analysis are as follows:

randomVariable 1 normal 5000.0 [expr 0.05\*5000.0]

performanceFunction 1 "0.13-{u\_2\_1}"

probabilityTransformation Nataf -print 0

randomNumberGenerator CStdLib

gFunEvaluator OpenSees Timeinvariant -file test12.tcl

runSubsetSimulationAnalysis -outPutFile m.out -SamplingType HMC -NumSeedSamples 1000 -print 1

Tcl commands of the structural response analysis in OpenSees are as before section 2.5 (test12). The failure probability and the coefficient of variation are Pf = 0.00169, c.o.v = 0.129, respectively.

Next, another analysis method of subset simulation is introduced. The same input as the previous case is used, except for the following modifications:

*runSubsetSimulationAnalysis -outPutFile m.out -SamplingType MH 2 -NumSeedSamples 5000 -print 1*

The failure probability and the coefficient of variation are Pf = 0.00197, c.o.v = 0.130, respectively.

2.7. Mean Upcrossing Rate

In this section, a time variant reliability analysis, the FORM approximation of the mean upcrossing rate is computed by using the following command:

*runOutCrossingAnalysis $filename -results $stepsToStart $stepsToEnd $freq $sam-pleFreq -littleDt $little\_dt –Koo*

The corresponding Tcl commands of reliability analysis are as follows:

1. set totalTime 3.0
2. set numPulses 10
3. set numTimeSteps [expr $numPulses\*10]
4. set pi 3.14159265358979
5. set phi0 0.25
6. set deltat [expr $totalTime/$numPulses]
7. for { set i 1 } { $i <= [expr $numPulses] } { incr i } {

randomVariable $i normal 0.0 [expr pow(2\*$pi\*$phi0/$deltat,0.5)] 0.0

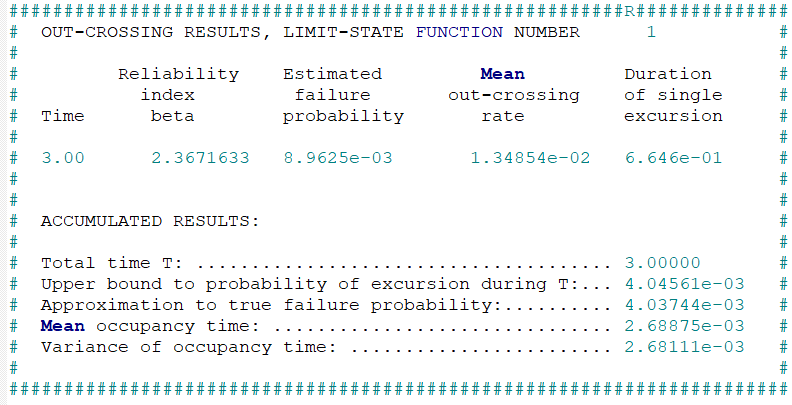
}

1. performanceFunction 1 "0.0048 - {u\_2\_1}"
2. probabilityTransformation Nataf -print 0
3. reliabilityConvergenceCheck Standard -e1 1.0e-6 -e2 1.0e-6 -print 1
4. gFunEvaluator OpenSees Timevariant -analyze $numTimeSteps [expr $totalTime/$numTimeSteps] -file reliability\_3\_aux.tcl
5. gradGEvaluator FiniteDifference -pert 1000
6. searchDirection iHLRF
7. meritFunctionCheck AdkZhang -multi 2.0 -add 10.0 -factor 0.5
8. stepSizeRule Armijo -maxNum 50 -base 0.5 -initial 1.0 2 -print 0
9. startPoint Given
10. findDesignPoint StepSearch -maxNumIter 100 -printDesignPointX designPointX.out
11. runOutCrossingAnalysis OutCross.out -results $numTimeSteps $numTimeSteps $numTimeSteps [expr $numTimeSteps/$numPulses] -littleDt 1.0e-6 -Koo;

Tcl commands of the structural response analysis in OpenSees are as follows:

1. model basic -ndm 2 -ndf 2
2. set zzzzero 0;
3. set GMScale\_sample [open RV.txt "r"];
4. set GMScale\_data [read $GMScale\_sample];
5. close $GMScale\_sample;
6. set fo1 [open RV2.txt w] ;
7. foreach different\_content {"zzzzero" "GMScale\_data"} {
8. puts $fo1 [set $different\_content]
9. }
10. close $fo1
11. set M 28.8e3
12. node 1 0.0 0.0
13. node 2 1.0 0.0 -mass $M $M
14. set K 40560.0e3
15. set pi 3.14159265358979
16. uniaxialMaterial Hardening 1 $K 734.0e3 0.0 2.1347e+006
17. set omega [expr pow($K/$M,0.5)]
18. element truss 1 1 2 1.0 1
19. fix 1 1 1
20. fix 2 0 1
21. set accelSeries "Series -dt 0.3 -filePath RV2.txt -factor 1";
22. pattern UniformExcitation 1 1 -accel $accelSeries
23. recorder Node -file nodeDisp\_1.out -time -node 2 -dof 1 disp
24. recorder Element -file axial.out -time -element 1 axialForce
25. recorder Element -file axial.out -time -element 1 axialDeformation
26. # STRUCTURAL ANALYSIS MODEL
27. system BandSPD
28. constraints Plain
29. test NormDispIncr 1.0e-16 50
30. algorithm Newton
31. numberer RCM
32. integrator Newmark 0.5 0.25 [expr 2\*0.02\*$omega] 0.0 0.0 0.0
33. analysis Transient
34. set totalTime 3.0
35. set numPulses 10
36. set numTimeSteps [expr $numPulses\*10]
37. set phi0 0.25
38. # UNCERTAINTY CHARACTERIZATION
39. set deltat [expr $totalTime/$numPulses]
40. analyze $numTimeSteps [expr $totalTime/$numTimeSteps]
41. set u21 [nodeDisp 2 1]
42. set fo [open PerformanceU.txt w] ;
43. foreach different\_content {u21} {
44. puts $fo [set $different\_content]
45. }
46. close $fo

The output file reads:



Next, another method of mean upcrossing rates is performed. The same input as the previous case is used, except for the following modifications:

*runOutCrossingAnalysis OutCross.out -results $numTimeSteps $numTimeSteps $numTimeSteps [expr $numTimeSteps/$numPulses] -littleDt 1.0e-6 -twoSearches;*

Tcl commands of the structural response analysis in OpenSees are the same as before. And a velocity output structural response analysis in OpenSees of is introduced and corresponding Tcl commands are as follows:

set V21 [nodeVel 2 1]

set foV [open PerformanceV.txt w] ;

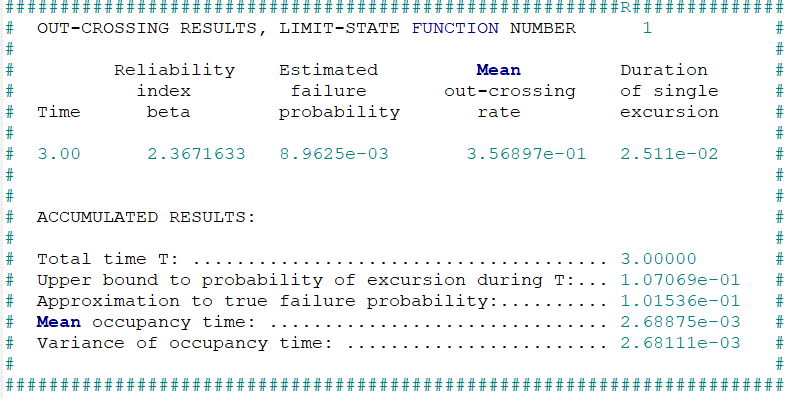
foreach different\_content {V21} {

puts $foV [set $different\_content]

}

close $foV

After finishing each structural analysis, a displacement response value and a velocity response value used to compute the LSF are recorded in PerformanceU.txt and PerformanceV.txt file, respectively. The output file of reliability analysis reads:



References

1. Welch B B. Practical programming in Tcl and Tk. Prentice Hall, Upper Saddle River, New Jersey, 3rd edition, 2000.
2. Gu Q. Finite element response sensitivity and reliability analysis of soil-foundation-structure-interaction (SFSI) systems. University Of California, San Diego, CA, 2008.
3. Haukaas T, Der Kiureghian A. User’s Guide for Reliability and Sensitivity Analysis in OpenSees, Pacific Earthquake Engineering Research Center, University of California, Berkeley, CA, 2003.
4. McKenna F, Fenves G L, et al. OpenSees (Open System for Earthquake Engineering Simulation) Users Manual, Version 1.6.0, 2004.
5. Wang Z, Macro B, et al. Hamiltonian Monte Carlo methods for Subset Simulation in reliability analysis. Structural Safety 2019; 72:65-73.

Table: Available types of reliability analysis in application framework

|  |
| --- |
| 1. FORMAnalysis |
| 1. SORMAnalysis |
| 1. ImportanceSamplingAnalysis |
| 1. OrthogonalPlaneSampling Analysis |
| 1. MonteCarloSampling |
| 1. SubsetSimulationAnalysis |
| 1. OutCrossingAnalysis |