FP7 2013 Cooperation Work Programme
Theme 6: Environment (Including Climate Change)

Novel indicators for identifying critical *INFRA*structure at *RISK* from Natural Hazards

Deliverable D6.2

Stress Test Framework for Systems

Primary Author	Pieter van Gelder, Noel van Erp/ Probabilistic Solutions Consult and Training (PSCT)
WP	6
Submission Date	29/11/2016
Primary Reviewer	Alan O'Connor/Roughan & O'Donovan Ltd. (ROD)
Dissemination Level	PU

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 603960.

Project Information

Project Duration: 1/10/2013 - 30/09/2016

Project Coordinator: Professor Eugene O' Brien
Roughan & O' Donovan Limited
eugene.obrien@rod.ie

Work Programme: 2013 Cooperation Theme 6:
Environment (Including Climate Change).

Call Topic: Env.2013.6.4-4 Towards Stress Testing of Critical Infrastructure Against Natural Hazards-FP7-ENV-2013-two stage.

Project Website: www.infrarisk-fp7.eu

Partners:

Roughan & O' Donovan Limited, Ireland

Eidgenössische Technische Hochschule Zürich, Switzerland.

Dragados SA, Spain.

Gavin and Doherty Geosolutions Ltd., Ireland.

Probabilistic Solutions Consult and Training BV, Netherlands.

Agencia Estatal Consejo Superior de Investigaciones Científicas, Spain.

University College London, United Kingdom.

PSJ, Netherlands.

Stiftelsen SINTEF, Norway.

Ritchey Consulting AB, Sweden.

University of Southampton (IT Innovation Centre), United Kingdom.

Document Information

Version	Date	Description	Primary Author
Rev01	November 2015	Deliverable D6.2: Development of framework, design and optimization of stress tests in infrastructural systems	H.R.N. van Erp, R.O. Linger, P. Prak, P.H.A.J.M. van Gelder.
Rev02	March 2016	Deliverable D6.2: Revised based on review by UCL and ROD	H.R.N. van Erp, R.O. Linger, P. Prak, P.H.A.J.M. van Gelder.
Rev03	July 2016	Revised based on review by ROD	H.R.N. van Erp, R.O. Linger, P. Prak, P.H.A.J.M. van Gelder.
Rev04	September 2016	Revised based on review by ROD	H.R.N. van Erp, R.O. Linger, P. Prak, P.H.A.J.M. van Gelder

Executive Summary

In the INFRARISK project, infrastructural systems (e.g. collections of roads, bridges, train-tracks, etc.) are modelled as probabilistic systems where individual components are represented by way of individual stochastics which express if these individual components are compromised damage-wise for a given hazard load. Probabilistic representations of a given infrastructural system are subjected to virtual shocks corresponding with some stress scenario. This allows us to gauge how the efficacy of the probabilistic representation has been compromised by the applied shocks and come to some estimate on the effects of possible stress scenarios that have not yet materialized.

In this deliverable, a general stress test framework is offered up in which a stress test is a special instance of a risk assessment, where instead of marginalizing over the entire possible stress scenarios one specific stress scenario is chosen instead for which to gauge its potential effects.

This stress test framework is simple enough on the conceptual side. On the practical side, however, when one wishes to implement this framework, things can quickly become non-trivial. For example, if our probabilistic system consists of N components, each of these components having M possible states, then the total density of states of the system as a whole will consist of N^M possible states. This is an instance of the well-known “curse of dimensionality” which will necessitate the use of sampling techniques the state probability distribution on the system level. In this deliverable, sampling methods are discussed by which stress tests may be evaluated on probabilistic systems which consist of a large number of stochastics.

If the probabilistic system consists of stochastically dependent components then it is recommended to use the novel Nested Sampling algorithm in order to evaluate the dependencies between these stochastic components, whereas traditional Monte Carlo sampling method will suffice if the stochastics within a probabilistic system are independent. Also, a specific type of a stress scenarios are cascading effects scenarios. For Task 6.2 there has been developped the Probability Sort algorithm which allows one to model temporal and spatial uncertainties in cascading effect risk scenarios.

The research results of this deliverable are relevant for all those, be they infrastructural managers or not, that wish to evaluate the state probability distribution of probabilistic systems that have been constructed as systems of stochastic components that are spatially and/or temporally dependent.

Table of Contents

1.0 INTRODUCTION

In this deliverable we give a general stress test definition as well as a discussion of a suite of algorithms that will allow one to apply this general stress test definition to probabilistic systems that consist of a large number of stochastic components.

In Chapter 2 a general stress definition is given. This stress test definition is explicitly linked to the overarching risk assessment methodology proposed in work package 4 of the INFRARISK project.

In order to illustrate the relevancy of the work presented in Chapters 4 through 8, a small case sudy is given in Chapter 3 for a system of 5 bridges of two different types, where an ensemble of fragility curves is derived for each bridge type, by taking not only into account (artificial) sampling uncertainty but also the (non-trivial) parameter uncertainty that invariable will occur in these problems. In this small case study Nested Sampling is used to take into account the dependencies between the fragility parameters. Probability maps for each bridge in the system may then be obtained by taking an ensemble of the conditional bridge probability maps over all the probalble fragility parameter combinations. Using these probability maps, which constitute a system of independent components/damage state probability distributions, we may use traditional Monte Carlo (MC) sample to come to a representative (i.e. having the greatest multiplicity) sample of damage state vectors.

In Chapters 4, 5, and 6, it is recommended that for large systems of dependent components/ parameters the Nested Sampling be used to take into account these depedendencies. For large systems of independent compoents it is recommended to use traditional MC methods.

Finally, a novel Probability Sort algorithm is presented in Chapter 7 and Appendix A, by which cascading effects may be modelled with a high degree of accuracy, as is demonstrated in Chapter 8.

2.0 A STRESS TEST DEFINITION

2.1 Introducing Stress Tests

A scenario is a possible future environment, either at a point in time or over a period of time. A projection of the effects of a scenario over the time period studied can address any kind of system, from local regions, to countries, to entire continents. To determine the relevant aspects of a possible future environment, one or more events or changes in circumstances may be forecast, possibly through identification or simulation of several risk factors. The effect of these events or changes in circumstances in a scenario can be generated from a shock to the system resulting from a sudden change in a single variable or risk factor. Scenarios can also be complex, involving changes to and interactions among many factors over time, as, for example, in cascading events (IAA, 2013).

It can be helpful in scenario analysis to provide a narrative (story) behind the scenario, including the risks (i.e. triggering events) that generated the scenario. Because the future is uncertain, there are many possible scenarios. In addition, there may be a range of effects on a particular system arising from each scenario. The projection of the financial effects during a selected scenario will likely differ from those seen using the modeller's best expectation of the way the current state of the world is most likely to evolve. Nevertheless, an analysis of alternative scenarios can provide useful information to involved stakeholders (IAA, 2013).

Sensitivity considers the effect of a set of alternative assumptions regarding some scenario. This alternative scenario can be the result of a single or several alternative risk factors, occurring either over a short or long period of time. A scenario used for sensitivity testing usually represents a relatively small change in these risk factors or their likelihood of occurrence. A scenario with significant or unexpected adverse consequences is referred to as a stress scenario.

A stress test is a projection of a particular system under a specific set of severe to moderately adverse conditions (i.e. stress scenarios). The forecast in the figure below represents the best case projection, deviations from which would constitute the effect of sensitivities and stress scenarios (where sensitivities are those scenarios that are close to the projected forecast scenario), Figure 2.1.

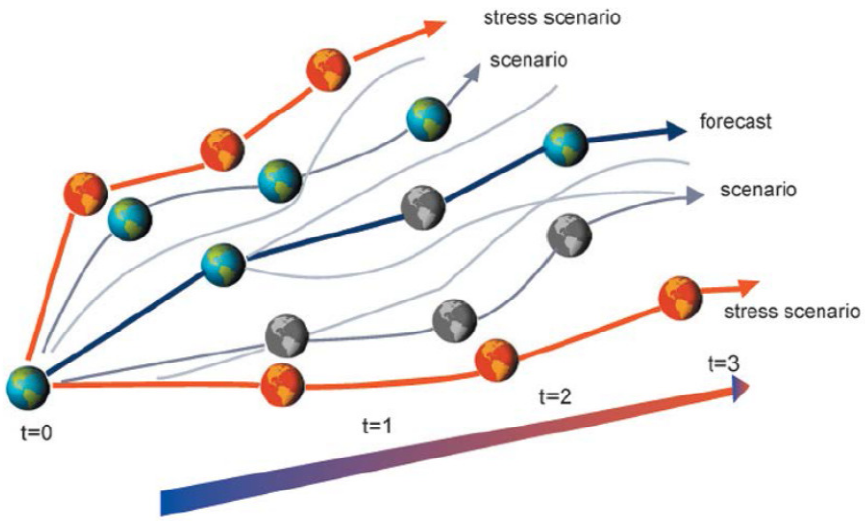

Figure 2.1: Stress Scenarios (IAA, 2013)

While the study of the effect of likely scenarios is useful for business planning and for the estimation of expected profits or losses, the assessing of the impact of rare and/or catastrophic future events, or even moderately adverse scenarios can enhance a risk culture, as it can alert decision makers to potentially inconvenient truths and provide a framework to enable them to base their operational strategies and risk mitigation activities on a range of scenarios, rather than a single best-estimate projected forecasts. In the actuarial sciences stress testing is considered to be essential tools for effective risk management and prudential oversight (IAA, 2013).

2.2 Why Perform Stress Tests?

The purpose of stress testing is not so much to predict future events, but rather to stimulate stakeholders to be prepared in case a disruptive event should occur (IAA, 2013). Stated differently, stress tests are done to get an answer to the question “How much could be lost?” in some worst case scenario, rather than the question “How much is likely to be lost?” under the current status quo, (Blaschke et al., 2001). Stress tests help to show us the parts of the system that need to be strengthened, or, if this is not feasible, as the stress scenario is too severe to build against (e.g. "the big one", along the San Andreas fault), the post-disaster contingencies we need to prepared for should our stress scenario materialize.

For an example of a successful stress test, in 2003 the regulators in Australia (APRA) conducted a stress test focusing on the domestic exposure of their banks and mortgage insurance companies to a decline in housing prices, leading to an equity shock. Based on the results of this stress test, APRA was able to implement regulatory safeguards that led them to avoid the issues faced by most jurisdictions following the collapse of the American mortgage market at the start of the financial crises in 2008-2009.

For an example of a missed stress test opportunity, the Fukushima Daiichi plant levees were build to withstand typhoon related 5.7m wave heights, while the exceedance return period of 10m tsunamis was only a mere 36 years, as 14 tsunamis of at least 10m have been observed in the last 500 years on the Japanese west coast (Krauß and Berg, 2011). Had the Fukushima Daiichi event been studied by way of a tsunami stress testing exercise prior to it happeningthe event occuring, then one of the possible outcomes with probability greater than zero would have been the outcome that we are currently witnessing; the outcome where a large part of Northern Japan has been radioactively contaminated.

In light of the fact that Fukushima Daiichi levees were only designed for typhoons, we find it plausible that a Fukushima Daiichi stress test would have compelled the relevant stakeholders to install also tsunamis typhoons, or, at the very least, move the power back-up from the cellar to the roof. Moreover, it is a distinct possibility that the Fukushima Daiichi incident escalated to a nationwide calamity, only because of the absence of simple emergency protocols, as a more timely depressurisation of the reactor coolant systems by the Fukushima Daiichi staff would have allowed for coolant injection by way of fire pumps, which might have helped to prevent cataclysmic core damage (Krauß and Berg, 2011). So it can be argued that a stress test exercise prior to the 2011 tsunami might have given the Fukushima Daiichi stakeholders a timely warning to put such emergency protocols in play at their nuclear plant.

2.3 Stress Testing for Infrastructural Risk Managers

In analogy with the actuarial stress test definition, an infrastructure stress test is defined as an analysis conducted under an unfavorable scenario, which is designed to determine whether there are inacceptable infrastructure related risks for a given unfavorable scenario. Stress tests may help detect objects that if "strengthened" through the execution of preventive interventions will greatly decrease the infrastructure related risk (i.e. hard engineering measures) and/or determine those non-engineering mitigation actions that will minimize the detrimental effects of a loss of functionality of the infrastructure (i.e. "soft" policy measures).

In this deliverable we will propose a general framework by which to quantify stress tests[1]. The stress test framework, at its highest level of abstraction, is a sub-set of the general risk assessment framework which is discussed below. In a stress test one assumes that some extreme hazard event has actually occurred, whereas in a risk assessment the effects of all hazard scenarios are accounted for and weighted by their respective probabilities of occurrence. In what follows we will first give a general discussion of Infrarisk's general risk assessment framework, as outlined in D4.2 (Heckl et al., 2016), after which we will turn to stress tests, as a special class of risk assessments.

2.3.1 The General Risk Assessment Framework

The general risk management process proposed in Infrarisk's D4.2 includes different sub-processes (Adey et al. 2014): the (1) problem identification process, (2) the system definition processes, (3) the risk identification process, (4) the risk analysis process, (5) the risk evaluation process, and (6) risk treatment.

In the problem identification step the information need of the relevant stakeholders is specified, as a first outline is given of the type of hazards that are to be studied in a given area and their effects on the objects of interest. So, an example of a risk assessment problem identification would be the general question: 'What dangers do earthquakes (type of hazards) pose to Bologna's (area) ten-T road network (object of interest)?'

In the system definition step a system representation is constructed. A system representation is a model of some relevant part of reality and consists of all the relevant realisations of the stochastic (i.e. uncertain) processes within the investigated time period. It includes the consideration of assumptions as to how the system will react in certain specific situations, and drawing fixed system boundaries, where it is clear that the things outside the system being modelled are not being modelled; that is, a system representation includes sufficiently good representations of the hazards, objects of interest, and consequences, as well as the interaction between them, so that it can be reasonably certain that there is an appropriate understanding of the system and all the possible scenarios that might arise in the case of some hazard event.

[1] Note that our commitment to the quantitative over the qualitative should not be read as a dismissal of qualitative stress tests; far from it, for qualitative stress tests may serve their purpose very effectively. It is just that a choice had to be made, and the choice fell naturally on quantification, as mathematics is the common mode of discourse in engineering risk research.

So, a system representation typically will include the natural environment (e.g. amount of rain, amount of water in rivers), the physical infrastructure (e.g. the behaviour of a bridge when subjected to high water levels), and human behaviour (e.g. traffic patterns when a bridge is no longer functioning). Also, as it is necessary to model the system over time it will often be necessary to specifically model the spatial and temporal dependencies between events and activities with the investigated time period. An important example of spatially and temporally dependent events are the so-called cascading events (see Chapter 8).

In WP4's general risk assessment framework 'risks' are equated with outcome scenarios. All relevant outcome scenarios are enumerated, by way of the system representation which was constructed in the previous step, in the risk identification step. These outcome scenarios are then assigned probabilities and consequences, again, by way of the system representation which was constructed in the previous step, in the risk analysis step.

In WP6's stress testing framework the 'risk identification' and 'risk analysis' steps equate with one singular 'construction the outcome probability distribution' step. Outcome probability distributions are the information carriers of our quantitative risk analyses, as for a given outcome metric (e.g. costs of physical repairs to the network, delay times for network users, restoration durations for the network, etc.) outcome probability distributions (1) enumerate all possible outcomes , and (2) given a plausibility (i.e. probability) to each of these possible outcomes. Stated differently, outcome probability distributions are (very important) risk assessment outputs. For an example of a risk assessment output in a physical repair cost outcome metric, conditional on some hypothetical adverse river discharge scenario, see Figure 3.3 in the next Chapter.

In the risk evaluation step the risk present in the outcome probability distribution, which was constructed in the preceding risk analysis step is communicated to and verified with the stakeholders. It is at this point in the general risk assessment process that stakeholders (i.e. decision makers) will be most likely to indicate whether or not the risk analysis needs to be redone with more detailed system representations, more sophisticated models and/or improved assumptions

In the risk treatment step involves the selection of the best way to modify the system which is being analysed. The best way to modify the system may be comprised of one or more interventions. These interventions can include physical changes to the infrastructure, alteration of the natural environment, and/or activities to alter the human behaviour during or following a hazard event. The selection of the best way to modify the system involves balancing of costs and effort of implementation against the benefits derived, taking into consideration constraints such as legal, regulatory, and other requirements such as social responsibility and the protection of the environment.

WP4's risk evaluation and (the cost benefit part of the) risk treatment steps are the subject of D6.3.

2.3.2 The General Stress Test Framework

As already stated in the previous section, in a quantitative/probabilistic analysis the outcome probability distribution is the information carrier that tells us all we ever would want to know, as it is

that in the outcome probability distribution all the possible outcomes are (1) enumerated, and (2) given a plausibility (i.e. probability). Conditional outcome probability distributions are the information carriers in stress tests, whereas unconditional, marginalized outcome probability distributions are the information carriers in risk analyses.

2.3.2.1 Stress Test Outputs

A stress test output is a (range of) projected value(s) of some outcome metric under the assumption that some stress scenario actually will occur with certainty. If there is only the one possible outcome, then this outcome has a probability of 1.0. But if there is more than one projected outcome value, then, in principle, a probability distribution may be assigned over these outcomes. The stress output will then consist of, in probability theoretical terms, an outcome probability distribution which is conditional on the proposed stress scenario; i.e. a conditional outcome distribution.

In a stress test we just construct the one outcome probability distribution for some given adverse scenario S, say,

$$p\left(O_{i^{(S)}} \mid S, A^{(0)}\right) \tag{2.1}$$

where the $O_{i^{(S)}}$ are the outcomes, for $i^{(S)} = 1, \ldots, n^{(S)}$ and $A^{(0)}$ is the action to keep the status quo. Whereas in a risk analysis we first construct outcome probability distribution for all the possible scenarios, be they adverse, neutral, or positive, say,

$$p\left(O_{i^{(S_j)}} \mid S_j, A^{(0)}\right) \tag{2.2}$$

where the $O_{i^{(S_j)}}$ are the outcomes, for $i^{(S_j)} = 1, \ldots, n^{(S_j)}$, for scenarios S_j, where $j = 1, \ldots, m$, and where S in (2.1) is a member of the set of possible scenarios $\{S_1, \ldots, S_m\}$ and $A^{(0)}$ is the action to keep the status quo. After which we weigh these conditional outcome probability distributions by the probabilities of their corresponding scenarios, in order to obtain the marginalized probability distribution:

$$p\left(O_{i^{(S_j)}} \mid A^{(0)}\right) = \sum_{j=1}^{m} p\left(S_j\right) p\left(O_{i^{(S_j)}} \mid S_j, A^{(0)}\right) \tag{2.3}$$

For a simple example of a stress test output (2.1), see Figure 3.3, where a system of five bridges is stress tested, in terms of the physical repair costs, for some stress scenario which leads to an increase of flow (m^3/s) around these bridges.

2.3.2.2 Selecting Stress (Test) Scenarios

Stress tests can involve estimating the impact of a change in a single risk factor (a sensitivity test), or the effect of a simultaneous move in a group of risk factors (a scenario analysis). Stress scenarios can be based on historical scenarios, employing shocks that occurred in the past, or can be based on hypothetical/synthetic scenarios, constructed to take account of plausible changes in circumstances that have no historical precedent. Two other techniques that are often included under the rubric of stress testing are extreme value theory, which applies statistical analysis to the tails of return distributions, and the maximum loss approach, which estimates the combination of factors that would cause the largest loss to the system under consideration (Blaschke et al., 2001).

The definition of an appropriate stress scenarios is a difficult task in that requires multiple persons bringing together their opinions and feelings into multiple coherent questions to be answered. The process of creating different stress scenarios is arguably the most difficult and controversial aspect of stress testing, as an ideal stress test needs to be relevant to the system under consideration. This requirement can impose significant resource costs, and involve a great deal of expertise and judgment by the parties involved (Blaschke et al., 2001).

So, risk managers need to represent their stress scenarios as plausible, being clear as to the extent of "invention" being applied, and to have a forthright discussion of the boundaries of conditions and events that should be anticipated. The aim is for stakeholders to be active participants and to consider risk to the system under consideration when making strategic and tactical decisions. The (implied) stress level should not be so severe as to merit a dismissal as being too alarmist. But neither should it be so mild as to desensitize the stakeholders to the potential risks (IAA, 2013).

Formulating a convincing and believable narrative or story may, on the one hand, be crucial to achieve buy-in from stakeholders into the stress scenarios, and, on the other hand, be helpful for the risk managers, as the formulation of a stress scenario becomes equivalent to the telling of a story (IAA, 2013):

> "And when it comes to influencing decisions and prompting action, the power of a 'story' should never be underestimated. A 'plausible model or reality' is exactly that, a 'story' that connects a variety of visible and readily understandable inputs to more or less extreme outcomes." (*Coherent Stress Testing*, Ricardo Rebonato)

Structured brainstorming sessions, such as conducted in general morphological analyses (Ritchey, 1998) as well as in WP4 of the InfraRisk project, may be used to elicit this expert knowledge based narrative from the relevant experts and stakeholders. One possible instrument by which to structure a brainstorming session is the use of Delphi panels (see Appendix B) and Similarity Judgment. Both methodologies are used in WP8 of the InfraRisk project. Such a combined approach has been used for example to prioritize objects within the National Alert System in the Netherlands to prioritise the most vulnerable locations related to terrorism (Prak, 2009).

It is also often useful to scan large databases, such as the one proposed in (Gavin and Martinovic, 2014), to have an idea as to what scenarios could be of particular interest. Descriptions of what can

be found in such databases in the case of infrastructure related risk due to natural hazards can be found in (Cheng and Taalab, 2014). Also, when using the extreme value theory approach, particular thought needs to be given here to the levels at which the stress tests need to be conducted; e.g. do you want to have a cumulative risk due to both floods and earthquakes below a threshold value, or do you want to have the risk due to floods below one threshold value, and the risk due to earthquakes below another threshold value, or both.

2.3.2.3 Risk Acceptability and the Choosing Amongst Alternatives

At this point it is worthwhile to explain, at least generally, what an acceptable level of risk is, beyond that it is one where the infrastructure manager is not required to execute interventions to reduce risk. The level of risk that is considered acceptable will typically varies from situation to situation, as risk acceptability is inversely related to both the number of cost efficient alternative courses of action that may reduce the risk and the extent of that cost efficiency; that is, the (un)acceptability of some current risk is something that has to be demonstrated by all parties involved.

If there are no feasible (i.e. cost-efficient) alternatives to protect us from some risk, like for example a meteorite strike that wipes out all life on the Northern Hemisphere, then we simply will accept that risk, even if its consequences are off the scale. But if we learn, for example, that the Fukushima Daiichi core breach could have been prevented with either some relatively minor investments in tsunami levees, or some small engineering adjustments in which the coolant power supply was moved to the roof of the nuclear plant, or the instalment of some simple emergency protocols, then it will be felt by most that unacceptable risks were taken by the Fukushima Daiichi stakeholders.

So, risk acceptability depends on whether there are possibilities to reduce the risk and how costly these are. This concept is sometimes referred to as the economically optimal level of risk, and was first proposed in the safety science domain by (van Danzig, 1956).

In addition to the method of optimizing the economical level of risk, others, such as (Jonkman et al., 2006), have proposed to determine acceptable risk levels by comparing the actual risk with norms on individual and societal risk, where individual risk indicates the distribution of the risk over the potentially affected individuals, and societal risk describes the relationship between frequency and the number of people suffering from a specified level of harm. The acceptable level of risk is considered to be one that is below that described in norms.

In D6.3 there is presented an decision making protocol which takes its cue from the economics (i.e. cost-benefit) based risk approach, where risks are implicitly deemed unacceptable if the costs for a safer system are less than the estimated benefit in terms of risk reduction. If we let risk be some function of both consequences, $\mathbf{x} = \{x_1, \ldots, x_n\}$, and the probabilities of these consequences, $\mathbf{p} = \{p_1, \ldots, p_n\}$; that is,

$$\text{Risk} = f(\mathbf{x}, \mathbf{p}) \tag{2.4}$$

Then it is argued in D6.3 that the risk function $f(\mathbf{x},\mathbf{p})$ may interpreted as a position measure on the corresponding outcome probability distribution:

$$p(\mathbf{x}) = \begin{cases} p_1, & x_1, \\ p_2, & x_2, \\ \quad \vdots & \\ p_n, & x_n. \end{cases} \tag{2.5}$$

where the $\mathbf{x} = \{x_1, \ldots, x_n\}$ are mapped on the *x*-axis and the $\mathbf{p} = \{p_1, \ldots, p_n\}$ are mapped on the *y*-axis.

For example, if we take as our risk function $f(\mathbf{x},\mathbf{p})$ the expectation value:

$$f(\mathbf{x},\mathbf{p}) = \sum_{i=1}^{n} x_i \; p_i = E(X) \tag{2.6}$$

then we have that our risk index is a measure of the position of the most-likely scenario (of losses). Now, given the iniquitousness of (2.6) as a definition for risk, there must be some merit in taking the most-likely loss-scenario as our risk measure. An alternative, more cautious position is taken by the return period methodology, which takes as its risk index the measure the position of an unlikely (to be on the safe side of things) worst-case scenario:

$$f(\mathbf{x},\mathbf{p}) = E(X) + k \; std(X) \tag{2.7}$$

where

$$E(X) = \sum_{i=1}^{n} x_i \; p_i, \qquad\qquad std(X) = \sqrt{\sum_{i=1}^{n} x_i^2 \; p_i - \left(\sum_{i=1}^{n} x_i \; p_i \right)^2} \tag{2.8}$$

and k is the sigma-level that will give us the desired upper percentile value. Now, in D6.3 it is proposed that the position measure that takes into account the worst, most-likely, and best case scenarios:

$$f(\mathbf{x},\mathbf{p}) = \frac{LB(X) + E(X) + UB(X)}{3} \tag{2.9}$$

is the most all-round risk measure. Now, there are as of yet no guiding mathematical principles by which to choose between the alternative risk indices (2.6), (2.7), and (2.9). We have only general common sense principles, like those expounded in D6.3, to guide us when it comes to this decision theoretical degree of freedom (van Erp et al., 2016*a*).

Now, in practice we will have that a stress test is done order to check if alternative courses of actions are to be taken. More specifically, an infrastructure stress test is an analysis conducted under unfavorable scenarios which is designed to determine whether there are inacceptable infrastructure related risks. These tests are meant to detect system objects that if "strengthened" through the execution of (hard engineering and/or soft policy) preventive interventions will greatly decrease the infrastructure related risk.

So the strengthening of one or more infrastructural system objects are the alternative actions which are open to the road manager, relative to a status quo where he only performs regular maintenance. Moreover, the road manager will have some budget constraint under which he has to decide whether or not to strengthen additional infrastructural objects or not. If we enumerate all the possible actions that a road manager might take as the set

$$\{A_1, A_2, \ldots, A_m\}. \tag{2.10}$$

Then we have that each action A_k will map to a specific outcome probability distribution:

$$p(\mathbf{x} \mid A_k) = \begin{cases} p_1, & x_1, \\ p_2, & x_2, \\ \quad \vdots & \\ p_{n_k}, & x_{n_k}. \end{cases} \tag{2.11}$$

where n_k is the number of possible outcomes under the *k*th action A_k. Now, we may compute for each of these outcome probability distributions (2.11), depending on our risk appetite, any of the risk indices (2.6), (2.7), and (2.9). For a given choice of risk index, the road manager then chooses that decision

$$A_k \in \{A_1, A_2, \ldots, A_m\} \tag{2.12}$$

which has the lowest risk (index) value of its corresponding outcome probability distribution (2.11).

So, in this straightforward decision theoretical approach (Jaynes, 2003), no threshold values are needed. All that needs to be done is:

1. An enumeration of all the possible action (2.10),
2. The construction of the corresponding outcome probability distributions (2.11),
3. A commitment to one of the risk indices either (2.6), (2.7), or (2.9),
4. A minimization of the chosen risk index over the set of possible actions (2.10).

For an actual demonstration of this approach, see D6.3.

So, if we make explicit the fact that the decision of whether or not to accept a certain risk must be against the backdrop of some set of alternative actions, we may bypass the threshold definition

problem. Moreover, by doing so, we have at our disposal an approach to risk-based decision making for critical infrastructures.

3.0 QUANTIFYING THE EFFECT OF STRESS SCENARIOS IN TRANSPORT SYSTEMS

We provide here a simple example of how a stress test may be evaluated for a simple system of bridges. This chapter acts as a step-by step illustration of the more technical material that will follow in Chapters 4, 5, and 6.

3.1 A System Description

Let us consider the following road network with 5 components (bridges over a river), Figure 3.1.

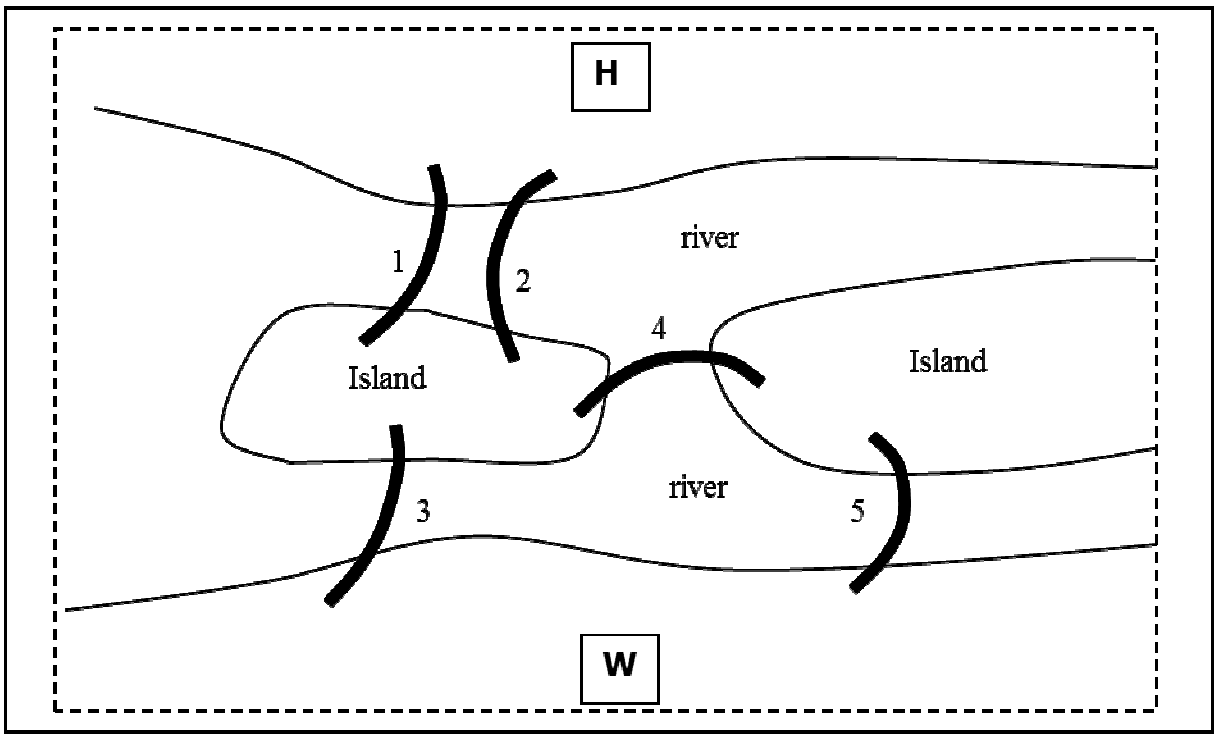

Figure 3.1: Bridge System

We assume that bridges 1, 2, and 3 are bridges of type I and bridges 4 and 5 of type II. Each type of bridge has its own characteristics and, as a consequence, will behave differently under different scour conditions; that is, bridges of type I are considered to be more resistant to scour than those of type II.

The scour load for a given bridge is considered to be some limit state function of some discharge value Q measured in the vicinity of the bridge (e.g. upstream, downstream, etc.). By specifying a limit state function for Q, we may determine for (increasing) discharge values Q_j thecorresponding scour load probability distribution (Gehl and D'Ayala, 2015; D'Ayala and Gehl, 2015). If these scour load probability distributions are connected to some damage state model by way of loading thresholds, then we may compute (on the assumptions that our limit state function and damage state model are valid) for a given discharge value Q_j a corresponding probability of a bridge being in either one of the damage states.

So, if there are $(n+1)$ damage states, of which the zeroth state is the "undamaged"-state, $i = 0,1,\ldots,n$, and if we have m discharge values Q_j, for $j = 1,\ldots,m$, then we may determine, by way of a limit state function the probability p_{ij} of being in damage state i given the discharge value Q_j. If we do not have direct access to the probabilities p_{ij}, then, alternatively, we may also sample the limit state function, in order find the number of realisations of bridge damage Z_{ij} in N_{ij} independent samples (Gehl and D'Ayala, 2015; D'Ayala and Gehl, 2015). Note that this later approach will introduce additional sampling uncertainty, as we determine p_{ij} by way of a sampling approach frequency and not analytically. But this additional sampling uncertainty may be removes if we let $N_{ij} \to \infty$.

3.2 The Probability Model

As the basis for our failure probability model we may take probit functions for each of the $i = 1,2,3$ actual damage states:

$$P(i \mid \alpha_i, \beta, Q) = \Phi\left[\frac{\ln(Q/\alpha_i)}{\beta}\right], \qquad \text{for } i = 1,2,3, \tag{3.1}$$

where the parameters α_1, α_2, α_3, and β are the so-called fragility parameters. The probability of being in (the un)damage state $i = 0$ then is

$$\pi(i = 0 \mid \alpha_1, \beta, Q) = 1 - \Phi\left[\frac{\ln(Q/\alpha_1)}{\beta}\right]; \tag{3.2a}$$

the probability of being in damage state $i = 1$ is

$$\pi(i = 1 \mid \alpha_1, \alpha_2, \beta, Q) = \Phi\left[\frac{\ln(Q/\alpha_1)}{\beta}\right] - \Phi\left[\frac{\ln(Q/\alpha_2)}{\beta}\right]; \tag{3.2b}$$

the probability of being in damage state $i = 2$ is

$$\pi(i = 2 \mid \alpha_2, \alpha_3, \beta, Q) = \Phi\left[\frac{\ln(Q/\alpha_2)}{\beta}\right] - \Phi\left[\frac{\ln(Q/\alpha_3)}{\beta}\right]; \tag{3.2c}$$

the probability of being in damage state $i = 3$ is

$$\pi(i = 3 \mid \alpha_3, \beta, Q) = \Phi\left[\frac{\ln(Q/\alpha_3)}{\beta}\right]. \tag{3.2d}$$

In order to evaluate the probabilities of being in one of the damage states for a given river discharge Q, we need to determine the fragility parameters α_1, α_2, α_3, and β.

3.3 Estimating Fragility Parameters

3.3.1 Fragility Parameter Estimation for Type I Bridges

Let us assume that for the bridge of type I we have 10 discharge values Q_j, for $j = 1,\ldots,10$. Let us also assume that we sample the limit state function for each discharge value and each damage state, starting from damage state $i = 1$, $N = 100$ times in order to determine each time the number of realizations Z_{ij} that are in the pertinent damage states $i = 1$, $i = 2$, and $i = 3$, respectively. In Table 3.1 we give a possible realization of such a sampling exercise.

	Q_j	Z_{1j}	Z_{2j}	Z_{3j}
j = 1	10	0	0	0
j =2	39	10	0	0
j =3	78	30	3	0
j =4	156	60	10	0
j =5	312	100	30	3
j =6	625	100	60	10
j =7	1250	100	100	30
j =8	2500	100	100	60
j =9	5000	100	100	100
j =10	10000	100	100	100

Table 3.1: Type I Bridge (discharge values and associated number of damage state realisations)

Based on this data, we can specify the fragility-parameter likelihood model (Shinozuka et al., 2003):

$$L(\alpha_1,\alpha_2,\alpha_3,\beta) = \prod_{i=1}^{3}\prod_{j=1}^{10} \Phi\left[\frac{\ln(Q_j/\alpha_i)}{\beta}\right]^{Z_{ij}} \left(1-\Phi\left[\frac{\ln(Q_j/\alpha_i)}{\beta}\right]\right)^{N-Z_{ij}}, \qquad (3.3)$$

where Φ is the symbol of the cumulative standard normal distribution. If we assign the following non-informative prior to the fragility-parameters (Jaynes, 1968)

$$p(\alpha_1,\alpha_2,\alpha_3,\beta) \propto \frac{1}{\alpha_1\alpha_2\alpha_3\beta}, \qquad (3.4)$$

Then we may combine (3.1) and (3.2) into the posterior probability distribution (Jaynes, 2003)

$$p(\alpha_1,\alpha_2,\alpha_3,\beta \mid D) \propto \frac{1}{\alpha_1\alpha_2\alpha_3\beta}\prod_{i=1}^{3}\prod_{j=1}^{10}\Phi\left[\frac{\ln(Q_j/\alpha_i)}{\beta}\right]^{Z_{ij}}\left(1-\Phi\left[\frac{\ln(Q_j/\alpha_i)}{\beta}\right]\right)^{N-Z_{ij}} \quad (3.5)$$

where the data D consists of the set of inputted flow discharges $\{Q_j\}$ and the set of observed number of failure realisations $\{Z_{ij}\}$ in $N=100$ trials, for $i=1,2,3$ and $j=1,\ldots,10$, as shown in Table 3.1.

By way of the Nested Sampling algorithm, we may obtain a univariate representation for the fragility parameter probability distribution (3.5) which allows us to evaluate the mean and standard deviation vectors, and the correlation-matrices of the fragility parameters (see Chapters 4, 5, and 6):

$$\mu_{\alpha_1}=107.99,\quad \mu_{\alpha_2}=421.62,\quad \mu_{\alpha_3}=1651.18,\quad \mu_b=0.7008, \quad (3.6a)$$

$$\sigma_{\alpha_1}=5.81,\quad \sigma_{\alpha_2}=21.54,\quad \sigma_{\alpha_3}=66.11,\quad \sigma_b=0.0273, \quad (3.6b)$$

And

$$\text{corr}=\begin{bmatrix} 1 & -0.02 & -0.01 & 0.03 \\ -0.02 & 1 & 0.10 & 0.05 \\ -0.01 & 0.10 & 1 & 0.04 \\ 0.03 & 0.05 & 0.04 & 1 \end{bmatrix}. \quad (3.6c)$$

From the correlation matrix (3.6*c*) it can be seen that the fragility parameters values are only somewhat correlated with each other, where we note that uncorrelatedness does not imply independence, as the correlation measure is a *linear* dependence measure; i.e. there are all kinds of non-linear depdencies conceivable which have a correlation measure of zero.

As the probabability distribution (3.5) cannot be easily factorized in the product of four independent probability dsitributions, one will need to use the univariate Nested Sampling representation of (3.5), say,

$$p_{NS}(\alpha_1,a_2,\alpha_3,\beta \mid D_1,\text{Type I}), \quad (3.7)$$

where D_1 is as in Table 1 and (3.7) itself is a collection of probability weighted fragility parameter vectors, in order to take into account the fragility parameter uncertainty, (see Chapter 6).

3.3.2 Fragility Parameter Estimation for Type II Bridges

In out hypothetical problem, we have that the type II are more vulnerable to scour. For the bridge of type II, we use the same 10 discharge values Q_j that were used in Table 1, where we sample from the same limit state function for each discharge value and each damage state, in order to determine

each time the number of realizations Z_{ij} that are in the pertinent damage states $i=1$, $i=2$, and $i=3$, respectively. In Table 3.2 we give a possible realization of such a sampling exercise. Note that the difference in the number of Z_{ij} realizations, relative to Table 3.1, are due to the fact that the damage state model for a type II bridge will set all the damage state thresholds lower, as these types of bridges are more vulnerable to scour loading.

	Q_j	Z_{1j}	Z_{2j}	Z_{3j}
j = 1	10	10	0	0
j =2	39	30	3	0
j =3	78	60	10	0
j =4	156	100	30	3
j =5	312	100	60	10
j =6	625	100	100	30
j =7	1250	100	100	60
j =8	2500	100	100	100
j =9	5000	100	100	100
j =10	10000	100	100	100

Table 3.2: Type II Bridge (discharge values and associated number of damage state realisations)

By way of the Nested Sampling algorithm, we may obtain a univariate representation for the fragility parameter probability distribution (3.5) which allows us to evaluate the mean and standard deviation vectors, and the correlation-matrices of the fragility parameters (see Chapter 6):

$$\mu_{\alpha_1}=48.73,\quad \mu_{\alpha_2}=210.01,\quad \mu_{\alpha_3}=861.58,\quad \mu_b=0.7626, \tag{3.8a}$$

$$\sigma_{\alpha_1}=2.88,\quad \sigma_{\alpha_2}=11.80,\quad \sigma_{\alpha_3}=44.91,\quad \sigma_b=0.0323, \tag{3.8b}$$

And

$$\text{corr}=\begin{bmatrix} 1 & -0.03 & -0.04 & -0.05 \\ -0.03 & 1 & -0.06 & 0.05 \\ -0.04 & -0.06 & 1 & 0.04 \\ -0.05 & 0.05 & 0.04 & 1 \end{bmatrix}. \tag{3.8c}$$

Again, from the correlation matrix (3.8*c*) it can be seen that the fragility parameters values are only somewhat correlated with each other.

As the probabability distribution (3.5) cannot be easily factorized in the product of four independent probability dsitributions, one will need to use the univariate Nested Sampling representation of (3.5), say,

$$p_{NS}\left(\alpha_1,a_2,\alpha_3,\beta \mid D_2,\text{Type II}\right), \tag{3.9}$$

where D_2 is as in Table 3.2 and (3.9) itself is a collection of probability weighted fragility parameter vectors, in order to take into account the fragility parameter uncertainty, (see Chapter 6).

3.4 Connecting Fragility Parameter Estimates to Damage State Probabilities

Using the Nested Sampling proxies (3.7) and (3.9), we may take into account, by way of the Law of Total Probability and the fragility parameter uncertainty in (3.2):

$$\begin{aligned}\pi(i \mid Q, D_1, \text{Type I}) &= \sum_{(\alpha_1,a_2,\alpha_3,\beta)} \pi(i, \alpha_1, a_2, \alpha_3, \beta \mid Q, D_1, \text{Type I}) \\ &= \sum_{(\alpha_1,a_2,\alpha_3,\beta)} \pi(i \mid \alpha_1, \alpha_2, \alpha_3, \beta, Q)\; p_{NS}(\alpha_1, a_2, \alpha_3, \beta \mid D_1, \text{Type I})\end{aligned} \tag{3.10a}$$

and, likewise,

$$\begin{aligned}\pi(i \mid Q, D_2, \text{Type II}) &= \sum_{(\alpha_1,a_2,\alpha_3,\beta)} \pi(i, \alpha_1, a_2, \alpha_3, \beta \mid Q, D_2, \text{Type II}) \\ &= \sum_{(\alpha_1,a_2,\alpha_3,\beta)} \pi(i \mid \alpha_1, \alpha_2, \alpha_3, \beta, Q)\; p_{NS}(\alpha_1, a_2, \alpha_3, \beta \mid D_2, \text{Type II})\end{aligned} \tag{3.10b}$$

3.5 Computing a Damage State Probability Map for a Stress Scenario

So, going back to Figure 3.1, we now will proceed to evaluate some stress scenario that will lead to elevated flood discharges throughout the river; that is, elevated flood discharges are predicted in the vicinity of the bridge, Figure 3.2. The fragility parameter weighted damage state probabilities for the type I bridges are given as, Figure 3.2 and (3.10*a*),

$$\pi(i \mid q_1 = 500, D_1, \text{Type I}) = [0.0147 \quad 0.3885 \quad 0.5522 \quad 0.0446], \tag{3.11a}$$
$$\pi(i \mid q_2 = 450, D_1, \text{Type I}) = [0.0212 \quad 0.4410 \quad 0.5056 \quad 0.0322], \tag{3.11b}$$
$$\pi(i \mid q_3 = 700, D_1, \text{Type I}) = [0.0040 \quad 0.2304 \quad 0.6547 \quad 0.1109], \tag{3.11c}$$

and the fragility parameter weighted damage state probabilities for the type II bridges are given as, Figure 3.2 and (3.10*b*),

$$\pi(i \mid q_4 = 300, D_2, \text{Type II}) = [0.0089 \quad 0.3106 \quad 0.5966 \quad 0.0840], \tag{3.11d}$$
$$\pi(i \mid q_5 = 550, D_2, \text{Type II}) = [0.0008 \quad 0.1027 \quad 0.6176 \quad 0.2789], \tag{3.11e}$$

where the damage state probabilities are ordered as $i = 0,1,2,3$; that is, for the stress scenario that gives us river discharge values as in Figure 3.2, all the bridges have the largest probability to be in damage state 2. The resulting probability map is given in Table 3.3.

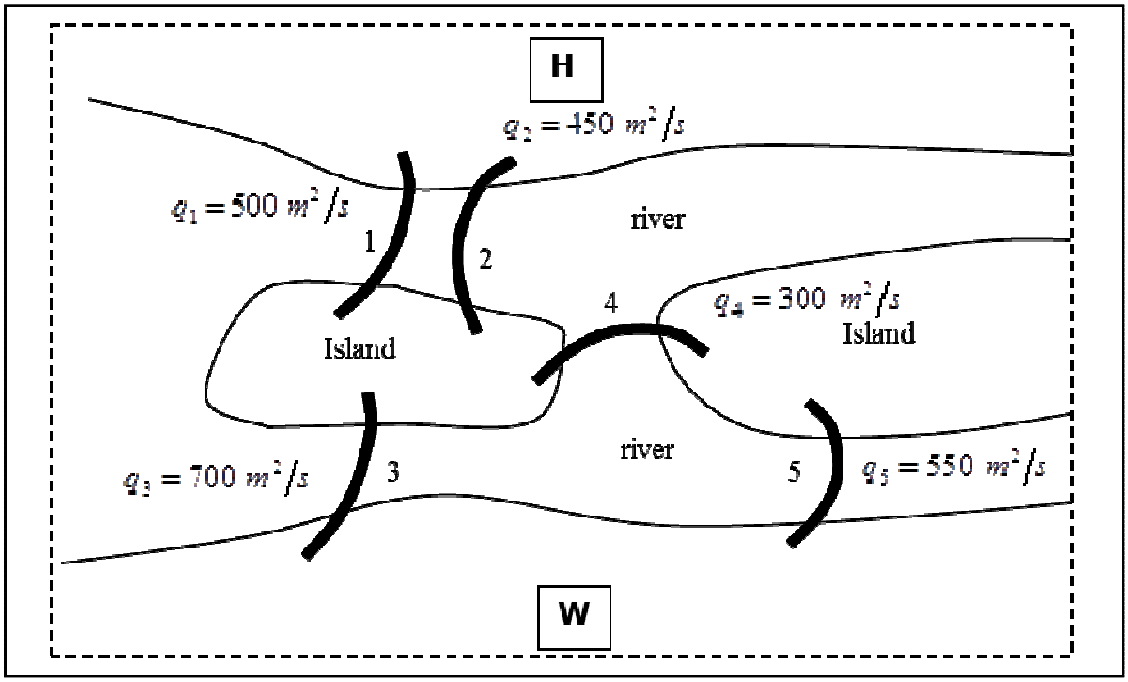

Figure 3.2: River Discharge Stress Scenario

	i = 0	*i* = 1	*i* = 2	*i* = 3
Bridge 1	0.0147	0.3885	0.5522	0.0446
Bridge 2	0.0212	0.4410	0.5056	0.0322
Bridge 3	0.0040	0.2304	0.6547	0.1109
Bridge 4	0.0089	0.3106	0.5966	0.0840
Bridge 5	0.0008	0.1027	0.6176	0.2789

Table 3.3: Damage State Probability Map of Bridge System under Stress Scenario in Figure 3.2

3.6 Assigning Probabilities to Damage State Vectors

With the parameter weighted damage state probabilities in (3.11) we may assign probabilities to all the $4^5 = 1024$ possible damage state vectors of the bridge system in Figure 3.1. For example, the probability of the damage state where the bridges 1, 2, and 3 are in damage state $i = 2$ and the bridges 4 and 5 are in damage state $i = 3$, that is,

$$\mathbf{x}^{(1)} = (2 \quad 2 \quad 2 \quad 3 \quad 3),$$

is found by taking the product of the probabilities, Table 3.3,

$$p\left(\mathbf{x}^{(1)} \mid \text{stress scenario}\right) = (0.5522)(0.5056)(0.6547)(0.0840)(0.2789) = 0.0043. \quad (3.12a)$$

The most likely damage state vector is the system state where all the bridges are in the damage state $i = 2$,

$$\mathbf{x}^{(2)} = (2 \quad 2 \quad 2 \quad 2 \quad 2),$$

which has a probability of, Table 3.3,

$$p\left(\mathbf{x}^{(2)} \mid \text{stress scenario}\right) = (0.5522)(0.5056)(0.6547)(0.5966)(0.6176) = 0.0673. \quad (3.12b)$$

The least likely damage state vector is the system state is the system state where all the bridges are in the undamaged state $i = 0$,

$$\mathbf{x}^{(3)} = (0 \quad 0 \quad 0 \quad 0 \quad 0),$$

which has a probability of, Table 3.3,

$$p\left(\mathbf{x}^{(3)} \mid \text{stress scenario}\right) = (0.0147)(0.0212)(0.0040)(0.0089)(0.0008) = 8.88\times10^{-12}. \quad (3.12c)$$

Note that given the chosen stress scenario, the system state probabilities are computed as the product of independent probability components; that is, given the values of the bridge relevant river discharge values $q_1, q_2, \ldots, q_N$, Figure 3.2, the probability of a damage state vector consisting of N components is the product of the probabilities of the N components in that state vector (3.12).

This independence of the damage state probabilities of the separate components will greatly facilitate the computational effort needed to come to a set of representative samples, by which we may evaluate the consequences of the stress scenario in Figure 3.2, as this independence allows us to bypass the need for Nested Sampling and revert to traditional MC sampling, where the 5 probability distributions (3.11) are sampled in sequentially in order to come to representative (i.e. having the greatest multiplicity) damage state vectors .

3.7 Assigning Direct Repair Costs to Damage State Vectors

Both type of bridges will have their own associated repair costs once they reach one of the damage states $i \geq 1$, Table 3.4.

	i = 0	*i* = 1	*i* = 2	*i* = 3
Bridge 1	0	10.000	50.000	1.000.000
Bridge 2	0	10.000	50.000	1.000.000
Bridge 3	0	10.000	50.000	1.000.000
Bridge 4	0	6.000	24.000	480.000
Bridge 5	0	6.000	24.000	480.000

Table 3.4: Damage State Repair Cost Map of Bridge System under Stress Scenario in Figure 3.2

With the damage state repair costs we may assign a repair cost to all the $4^5 = 1024$ possible damage state vectors of the bridge system in Figure 3.1. For example, the repair cost of the damage state where the bridges 1, 2, and 3 are in damage state $i = 2$ and the bridges 4 and 5 are in damage state $i = 3$, that is,

$$\mathbf{x}^{(1)} = (2 \quad 2 \quad 2 \quad 3 \quad 3),$$

is found by taking the sum of the repair costs, Table 3.4,

$$\begin{aligned} c(\mathbf{x}^{(1)} \mid \text{stress scenario}) &= 50.000 + 50.000 + 50.000 + 480.000 + 480.000 \\ &= 1.210.000. \end{aligned} \tag{3.13a}$$

The most likely damage state vector is the system state where all the bridges are in the damage state $i = 2$,

$$\mathbf{x}^{(2)} = (2 \quad 2 \quad 2 \quad 2 \quad 2),$$

which has a probability of, Table 3.3,

$$\begin{aligned} c(\mathbf{x}^{(2)} \mid \text{stress scenario}) &= 50.000 + 50.000 + 50.000 + 24.000 + 24.000 \\ &= 198.000. \end{aligned} \tag{3.13b}$$

The least likely damage state vector is the system state is the system state where all the bridges are in the undamaged state $i = 0$,

$$\mathbf{x}^{(3)} = (0 \quad 0 \quad 0 \quad 0 \quad 0),$$

which has a probability of, Table 3.3,

$$c\left(\mathbf{x}^{(3)} \mid \text{stress scenario}\right) = 0+0+0+0+0 \qquad (3.13c)$$
$$= 0.$$

3.8 Evaluating the Repair Cost Probability Distribution

The mean and standard deviations of the repair costs of the separate bridges are given in Table 3.5, as computed by way of the values in Tables 3.3 and 3.4.

Bridge	1	2	3	4	5
mean repair cost	$\mu_1 = 76095$	$\mu_2 = 61890$	$\mu_3 = 145939$	$\mu_4 = 56502$	$\mu_5 = 149310$
std. repair cost	$\sigma_1 = 200570$	$\sigma_2 = 172270$	$\sigma_3 = 302100$	$\sigma_4 = 128510$	$\sigma_5 = 205730$

Table 3.5: Mean and Standard Deviation of Repair Costs of Respective Bridges in Figure 3.2

The mean and standard deviation of the repair cost of all the bridges then is given as, Table 3.5,

$$\mu_{total} = \sum_{i=1}^{5} \mu_i = 489740, \qquad \sigma_{total} = \sqrt{\sum_{i=1}^{5} \sigma_i^2} = 469050. \qquad (3.14)$$

For comparison, if we take a mere 100 MC samples, then we find the sample estimates

$$\overline{X}_{total} = 515200, \qquad S_{total} = 462340; \qquad (3.15a)$$

if we take 1000 MC samples, then we find the sample estimates

$$\overline{X}_{total} = 483174, \qquad S_{total} = 463410; \qquad (3.15b)$$

and if we take 10.000 MC samples, then we find the sample estimates

$$\overline{X}_{total} = 486380, \qquad S_{total} = 467050. \qquad (3.15c)$$

It can be seen that the mean and standard deviation of the MC sampled total repair costs, (3.15), quickly converge to the true population values (3.14). The total repair cost histogram of 1.000.000 MC samples is given as in Figure 3.3. The frequency distribution in Figure 3.3 has a mean and standard deviation of

$$\overline{X}_{total} = 488960, \qquad S_{total} = 468900. \qquad (3.16)$$

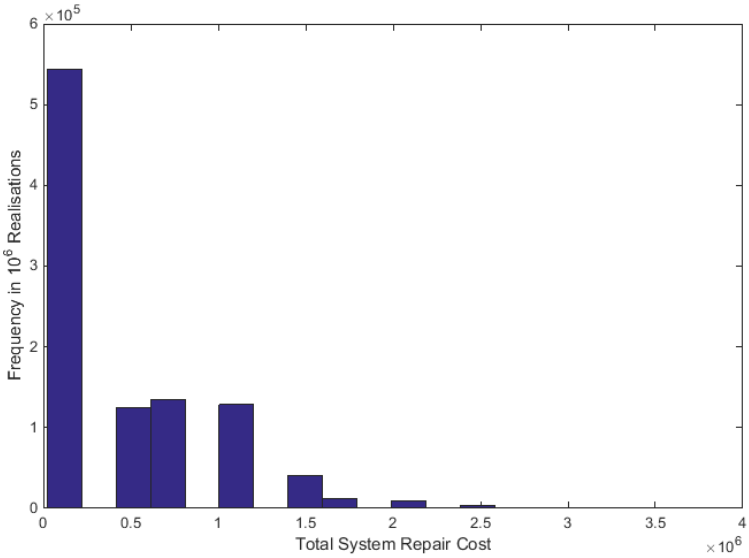

Figure 3.3: Frequency Distribution of Total Repair Costs under the Stress Scenario in Figure 3.2

4.0 UNIVARIATE REPRESENTATIONS OF MULTIVARIATE PROBABILITY DISTRIBUTIONS

We now discus how to represent highly multivariate probability distribution functions on the two dimensional plane (Skilling, 2004), as this will lay some of the groundwork for the upcoming discussion of the Probability Sort and Nested Sampling algorithms. The latter algorithm was instrumental in the evaluation of the case study in Chapter 3, whereas the former algorithm will allow one to model cascading effects stress scenarios.

4.1 Univariate Representations

Say we wish to numerically evaluate the integral of the bivariate normal distribution $MN(\boldsymbol{\mu}, \Sigma)$ where

$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \text{and} \qquad \Sigma = \begin{pmatrix} 1.96 & -1.37 \\ -1.37 & 1.96 \end{pmatrix}, \tag{4.1a}$$

or, equivalently,

$$p(x, y) = \frac{\sqrt{1-(0.7)^2}}{2\pi} \exp\left[-\frac{1}{2}\left(x^2 + 1.4xy + y^2\right) \right], \tag{4.1b}$$

where $-5 \le x, y \le 5$, Figure 4.1.

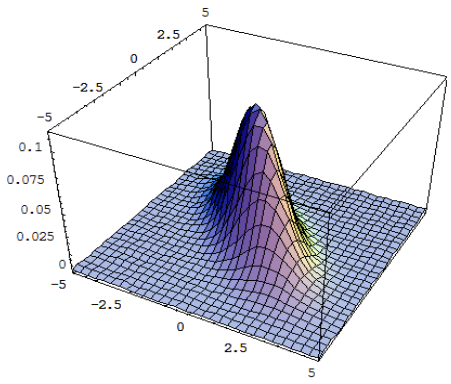

Figure 4.1: Graph of *p*(*x*, *y*)

Then the total volume under the curve $p(x, y)$ in Figure 4.1 is given by the integral

$$\int_{-5}^{5}\int_{-5}^{5} \frac{\sqrt{1-(0.7)^2}}{2\pi} \exp\left[-\frac{1}{2}\left(x^2 + 1.4xy + y^2\right) \right] dx\, dy = 0.9993\,. \tag{4.2}$$

We may evaluate the integral (4.2) through brute force. We partition the x, y -plane in little squares with area $dxdy$, then define the centre of these areas as $(\tilde{x}_j, \tilde{y}_k)$, for $j = 1,\ldots,20$, $k = 1,\ldots,20$, and compute the strips of volume V_{jk} as

$$V_{jk} = p(\tilde{x}_j, \tilde{y}_k)dxdy. \tag{4.3}$$

In Figure 4.2 we give all the volume elements V_{jk} together:

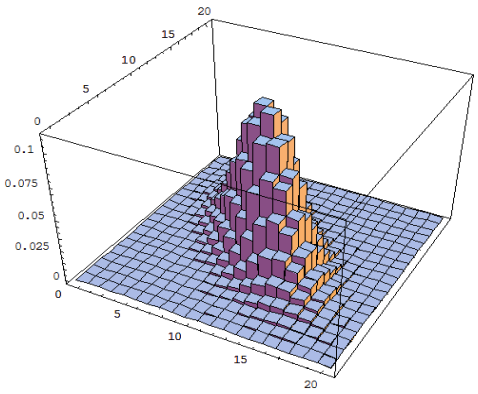

Figure 4.2: Volume Elements of *p*(*x*, *y*)

The total volume under the curve $p(x, y)$ may be approximated as

$$volume = \sum_{j=1}^{20}\sum_{k=1}^{20} V_{jk} = 0.9994. \tag{4.4}$$

Now, we may map these 3-dimensional volume elements V_{jk} to corresponding 2-dimensional area elements A_i. This is easily done by introducing the following notation

$$dw = dxdy, \qquad p(w_i) = p[(\tilde{x}, \tilde{y})_i] = p(\tilde{x}_j, \tilde{y}_k), \tag{4.5}$$

where index i is a function of the indices j and k:

$$i \equiv (j-1)20 + k \tag{4.6}$$

and $i = 1,\ldots,400$. Using (4.5), we may rewrite (4.3) as

$$A_i = p(w_i)dw = p[(\tilde{x}, \tilde{y})_i]dw. \tag{4.7}$$

In Figure 4.3 we give all the 400 are elements A_i together:

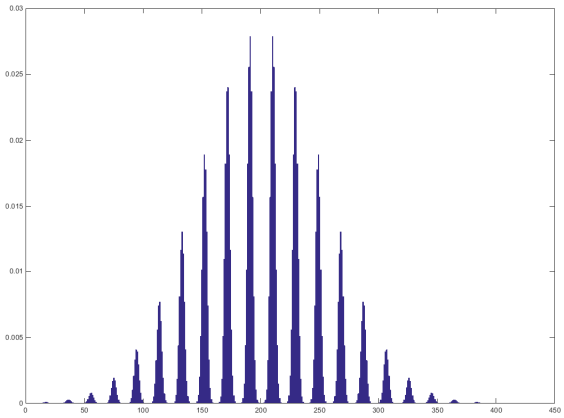

Figure 4.3: Area Elements of *p*(*x*, *y*)

Since (4.7) is equivalent to (4.3), we have that the mapping of the 3-dimensional volume elements V_{jk} to their corresponding 2-dimensional area elements A_i has not led to any loss of information; that is,

$$area = \sum_{i=1}^{400} A_i = \sum_{j=1}^{20}\sum_{k=1}^{20} V_{jk} = volume\,. \tag{4.8}$$

We now may, trivially, rearrange the elements A_i in Figure 4.3 in descending order, so we obtain Figure 4.4.

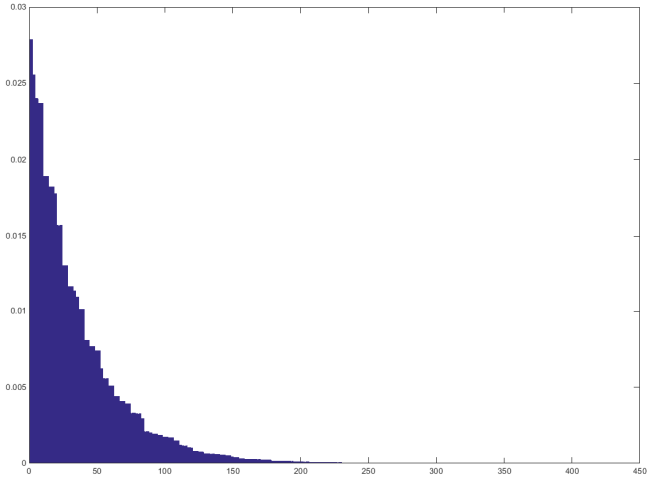

Figure 4.4: Ordered Area Elements of *p*(*x*, *y*)

Note that the horizontal axis of Figure 4.4 is non-dimensional. This is because we are looking at a collection of rectangular area elements ordered in one of many possible configurations. Now all these rectangular elements have a base of $dw = dx\,dy = 0.25$, being that there are 400 area elements we might view Figure 4.4 as a representation of some monotonic descending function $g(w)$, where $0 \le w \le 100$, as displayed in Figure 4.5.

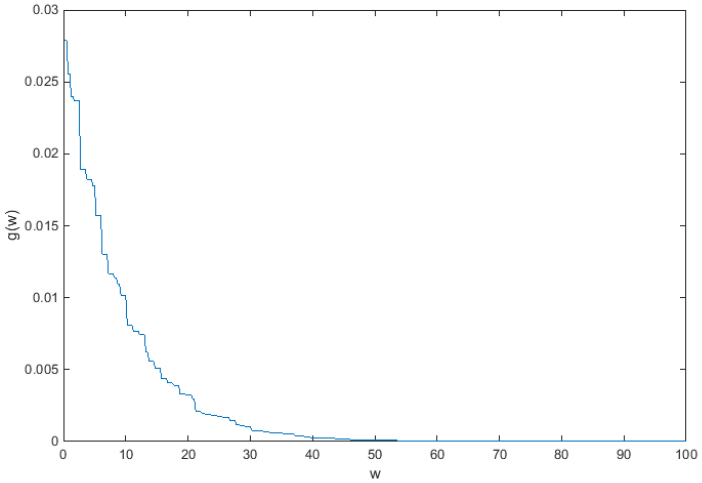

Figure 4.5: Function *g*(*w*); Ordered Univariate Representation of *p*(*x*, *y*)

What we have accomplished is that we have mapped 3-dimensional volume elements, (Figure 4.2), of the bivariate probability distribution p, (Figure 4.1), to 2-dimensional area elements, (Figure 4.3), after which we have rearranged these area elements in descending order, (Figure 4.4), so as to get a monotonic descending 'function' g, (Figure 4.5), and in going from Figure 4.1 to Figure 4.5 all the pertinent probability density information is retained, as every point on Figure 4.5's *w*-axis corresponds with a (x, y)-coordinate. Any *k*-variate function probability distribution $p(\mathbf{x})$ may be thus reduced to its corresponding monotonic descending univariate representation $g(w)$ (Skilling, 2004), where it is understood that every point on the univariate *w*-axis corresponds with some point in the multivariate $\mathbf{x}$-domain.

4.2 Retention of Pertinent Probability Density Information

If we have some function f which takes as its arguments x and y, which have been assigned the probability distribution (4.1), then the q^{th} order moment of the function f may be evaluated as, (4.3)-(4.8),

$$
\begin{aligned}
E\left\{ [f(x,y)]^q \right\} &= \iint [f(x,y)]^q \; p(x,y)\,dx\,dy \\
&\approx \sum_j \sum_k \left[f(\tilde{x}_j, \tilde{y}_k) \right]^q \; V_{jk} \\
&= \sum_i \left\{ f\left[(\tilde{x}, \tilde{y})_i \right] \right\}^q \; A_i \\
&= \int [f(w)]^q \; g(w)\,dw,
\end{aligned}
\tag{4.9}
$$

where it is understood that every w in (4.9) points to (i.e. is a placeholder for) a specific (x, y)-coordinate.

Now, if the function f takes as its inputs arguments that admit a probability distribution, then one may map this uncertainty regarding the input arguments to an uncertainty regarding the corresponding f values. For example, the expectation value f (i.e., $q = 1$) is given as (4.9)

$$\mu_f = E[f(x, y)], \tag{4.10}$$

whereas the standard deviation of f is given as (4.9)

$$\sigma_f = \sqrt{E\{[f(x, y)]^2\} - \{E[f(x, y)]\}^2}\ . \tag{4.11}$$

So, by going from the standard multivariate probability distribution $p(x, y)$, Figure 4.1, to its univariate representation $g(w)$, Figure 4.5, all the pertinent probability density information in the former is retained in the latter.

4.3 Generating Representative Samples

4.3.1 MC-Sampling

If we want to obtain a representative sample of n realizations from (4.1), Figure 4.1, then we may do this by simply Monte Carlo (MC) sampling the cumulative distribution function of the probability sorted univariate representation $g(w)$ in Figure 4.6 for n consecutive times.

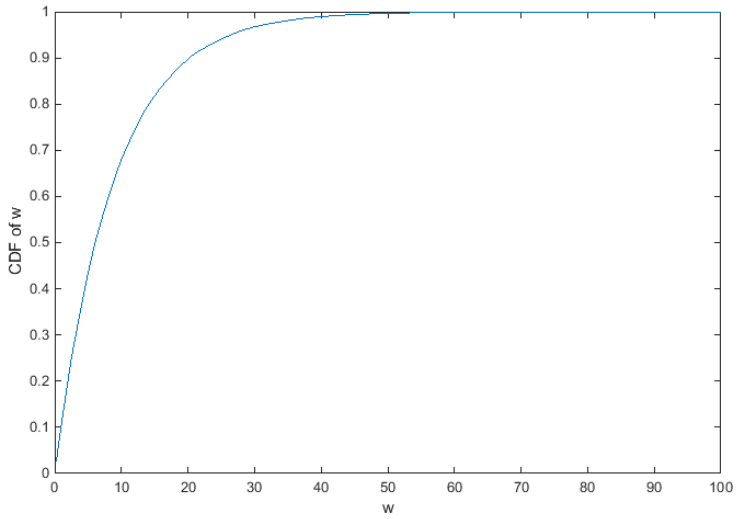

Figure 4.6: Cumulative Distribution Function (CDF) of *w*

A MC sample is obtained as follows. Let u be random realisation from the unit uniform probability distribution $U(0,1)$. Then, for a given u on the *y*-axis of Figure 4.6, we look up the corresponding w_i, $i=1,2,\ldots,400$, value on the *x*-axis, where it is understood that this w_i value is a placeholder for the probability sorted coordinate $(\tilde{x},\tilde{y})_i$. This coordinate $(\tilde{x},\tilde{y})_i$ is the representative MC-sample we are looking for.

4.3.2. Staircase Sampling

Alternatively, if we want to control for sampling the same coordinate $(\tilde{x},\tilde{y})_i$ more than once, we may construct the cumulant staircase (4.7):

$$\begin{aligned} T_k &= u + n\sum_{i=1}^{k} g(w_i)dw \\ &= u + n\sum_{i=1}^{k} A_i, \end{aligned} \tag{4.12}$$

for $k=1,2,\ldots,400$, and where u is again some random realisation from the unit uniform probability distribution $U(0,1)$. If the staircase T_k rises for the first time above either of the integer values, 1, 2, 3, … , n, then record the value k and take aside the corresponding coordinate vector $(\tilde{x},\tilde{y})_k$; see (3.3). Also, in order to avoid a repeated sampling of the same coordinate, we let

$$n \le \frac{1}{\max_i(A_i)}. \tag{4.13}$$

This will leave us with a representative Monte Carlo sample of (x,y)-coordinates (Sivia and Skilling, 2006). For example, if we set $n=35$, then we may obtain from Figure 4.7 the staircase sampler (4.12) which is displayed in Figure 4.7.

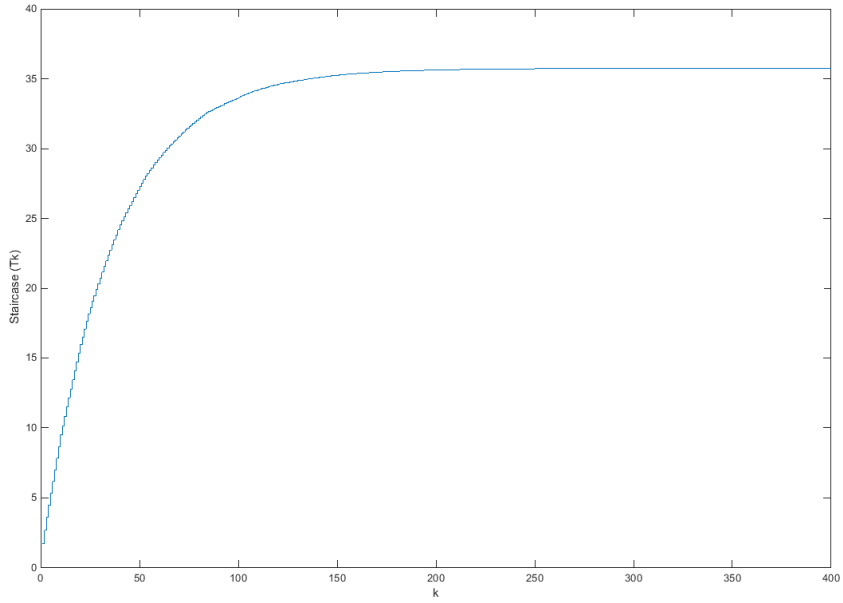

Figure 4.7: Staircase Sampler

If we zoom in to $k = 1,2,\ldots,30$, then we may see the staircase sampler in action for the fifth, tenth, fifteenth, twentieth sample points, which map respectively, to the probability sorted coordinates $(\tilde{x},\tilde{y})_5$, $(\tilde{x},\tilde{y})_{11}$, $(\tilde{x},\tilde{y})_{19}$, and $(\tilde{x},\tilde{y})_{29}$, Figure 4.8.

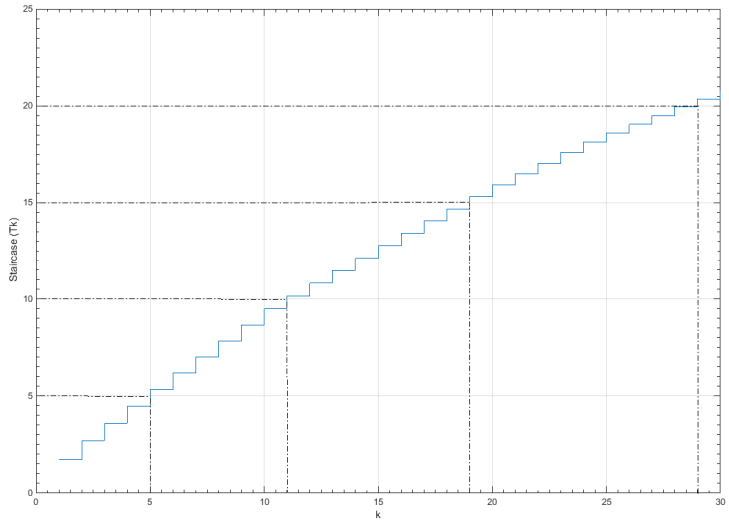

Figure 4.8: Staircase Sampler (Zoom-in)

By way of the staircase sampler, we then obtain a set of $n = 35$ representative samples. These samples are plotted together with the contourplot of (4.1), Figure 4.9.

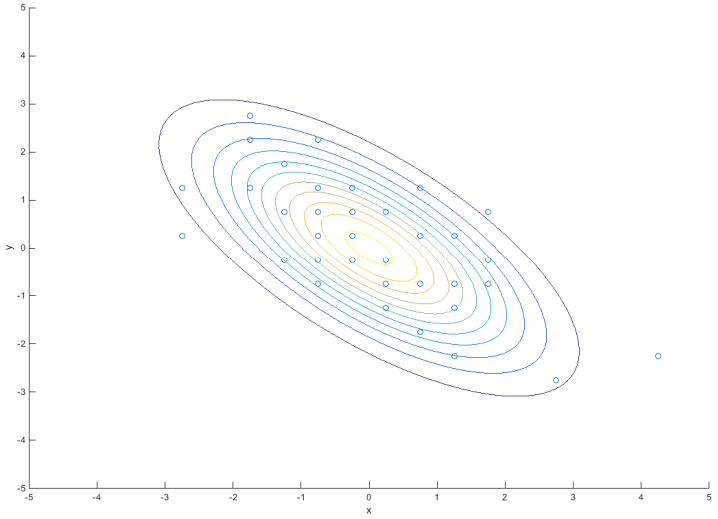

Figure 4.9: Contour Plot of (4.1) with 35 Representative Samples (20-by-20 grid)

The mean and covariance matrix esitmates of this representative sample compare favourable with (3.1):

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}0.04\\ 0.11\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}2.18 & -1.28\\ -1.28 & 1.67\end{bmatrix}.$$

For a 100-by-100 partition of the (x,y)-domain in Figure 4.1 and a set of $n=879$ representative samples, we obtain Figure 4.10.

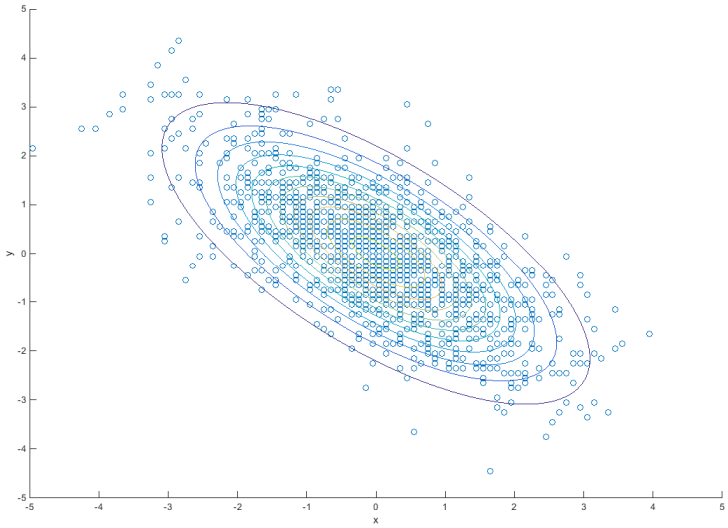

Figure 4.10: Contour Plot of (4.1) with 879 Representative Samples (100-by-100 grid)

The mean and covariance matrix esitmates of (4.1) of the representative samples in Figure 4.10 are, respectively,

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}0.02\\ -0.01\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}1.99 & -1.38\\ -1.38 & 1.93\end{bmatrix}.$$

For a 200-by-200 partition of the (x, y)-domain in Figure 4.1 and a set of $n=3519$ representative samples, we obtain Figure 4.11.

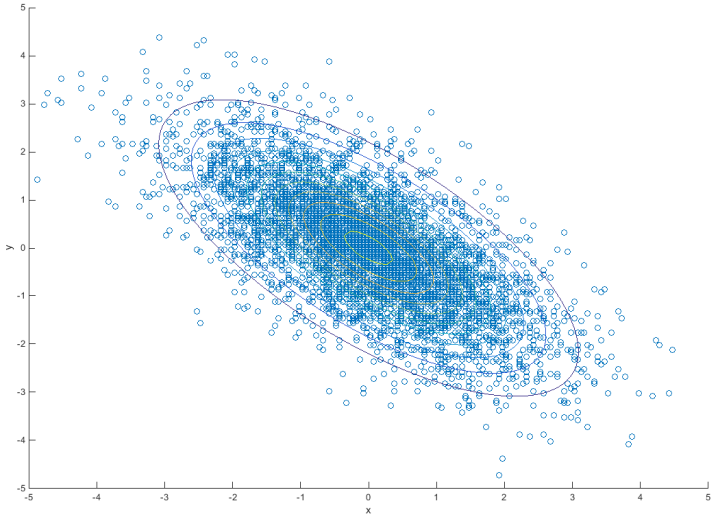

Figure 4.11: Contour Plot of (4.1) with 3529 Representative Samples (200-by-200 grid)

The mean and covariance matrix esitmates of (4.1) of the representative samples in Figure 4.11 are, respectively,

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}-0.01\\ -0.03\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}1.96 & -1.32\\ -1.32 & 1.87\end{bmatrix}.$$

5.0 REPRESENTATIVE SAMPLES FROM SYSTEMS OF INDEPENDENT COMPONENTS

In this chapter we show how for systems of independent components/variables, the MC- or staircase-sampling from the cumulative distribution function of probability sorted univariate representation of the multivariate system state probability distribution may be short-circuited by a simple MC-sampling from the cumulative distribution functions of the independent compenents seperately. This observation will lead to the recommendation that for indenpedent compenents/variables simple MC-sampling is to be used, as is also done in the case study presented in Chapter 3.

5.1 Sampling from Probability Sorted Total System Representations

Say we wish to numerically evaluate the integral of the bivariate normal distribution $MN(\boldsymbol{\mu},\Sigma)$ where

$$\boldsymbol{\mu}=\begin{pmatrix}0\\0\end{pmatrix}, \qquad \text{and} \qquad \Sigma=\begin{pmatrix}1.96 & 0\\0 & 1.96\end{pmatrix}, \tag{5.1a}$$

or, equivalently,

$$p(x,y)=\frac{1}{2\pi(1.96)}\exp\left[-\frac{1}{2(1.96)}\left(x^2+y^2\right)\right], \tag{5.1b}$$

where $-5\le x,y\le 5$, Figure 5.1.

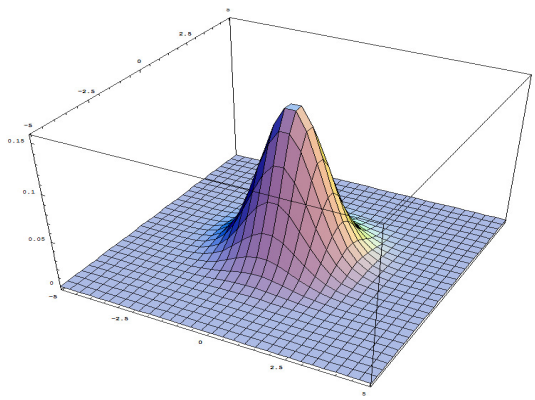

Figure 5.1: Graph of $p(x, y)$

Then we may discretize (5.1) into a collection of volume elements V_{jk} (4.3), Figure 5.2.

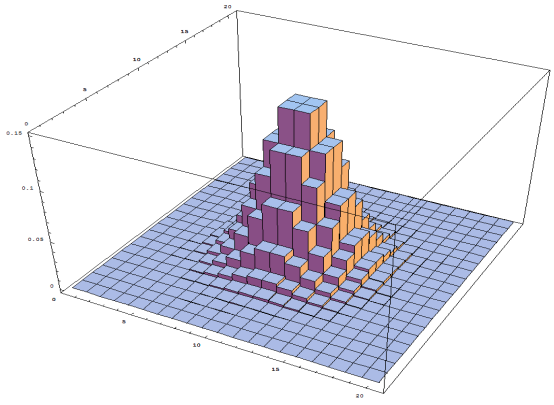

Figure 5.2: Volume Elements of $p(x, y)$

The volume elements in Figure 5.2 may be transformed to corresponding area elements A_i (4.5)-(4.7), Figure 5.3.

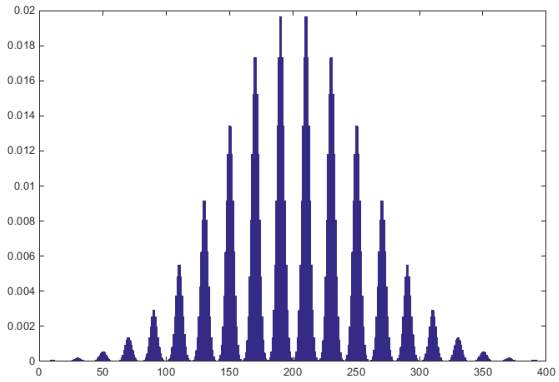

Figure 5.3: Area Elements of $p(x, y)$

The elements A_i in Figure 5.3 are then rearranged in descending order, Figure 5.4.

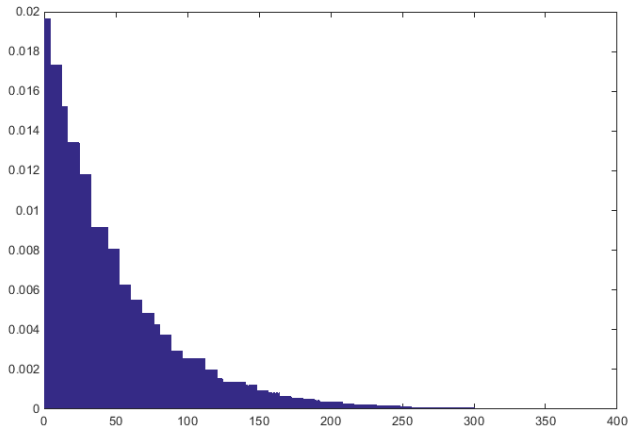

Figure 5.4: Ordered Area Elements of $p(x, y)$

Figure 5.4 is a representation of some monotonic descending function $g(w)$, Figure 5.5.

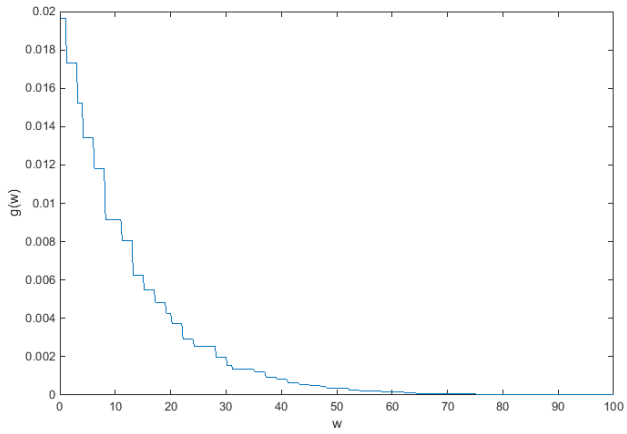

Figure 5.5: Function *g*(*w*); Ordered Univariate Representation of *p*(*x*, *y*)

Now, if we want to obtain a representative sample of n realizations from (5.1) in Figure 5.1, then we may do this by MC-sampling the cumulative distribution function of the probability sorted univariate representation $g(w)$, Figure 5.6, or by way of a stair case sampler (4.12).

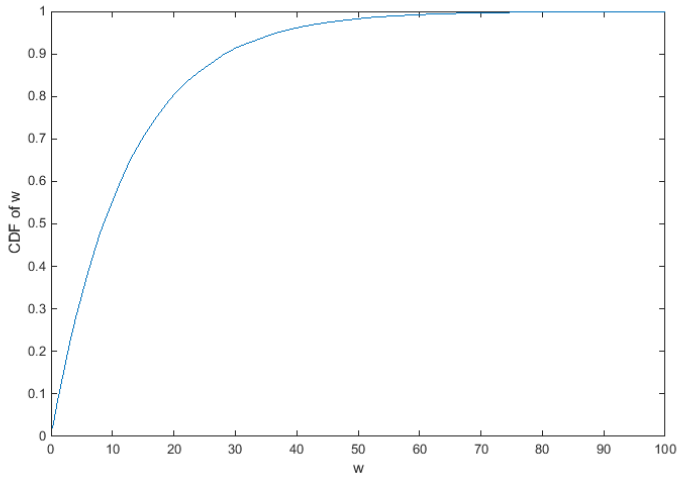

Figure 5.6: Cumulative Distribution Function (CDF) of *w*

By way of staircase sampling, as we wish to guard against a repeated sampling of the same coordinate $(\tilde{x}, \tilde{y})_i$, we then obtain a set of $n = 50$ unique representative samples. These samples are plotted together with the contourplot of (5.1), Figure 5.7.

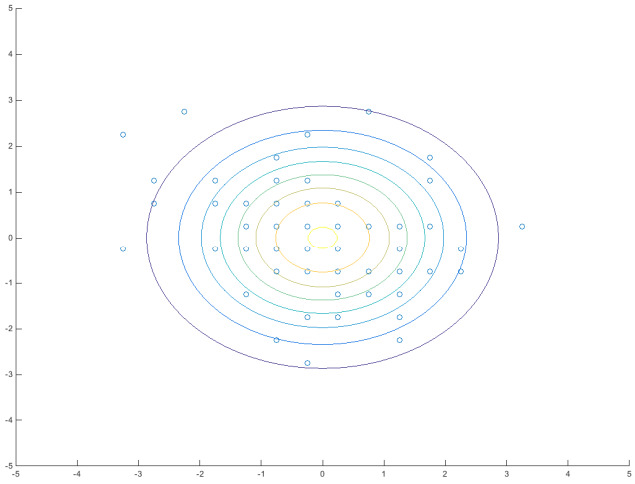

Figure 5.7: Contour Plot of (5.1) with 50 Representative Samples (20-by-20 grid)

The mean and covariance matrix esitmates of this representative sample compare favourable with (5.1):

$$\begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} -0.10 \\ 0.03 \end{pmatrix}, \qquad \text{and} \qquad S = \begin{bmatrix} 2.11 & -0.51 \\ -0.51 & 1.61 \end{bmatrix}.$$

For a 100-by-100 partition of the (x, y)-domain in Figure 5.1 and a set of $n = 1232$ representative samples, we obtain Figure 5.8.

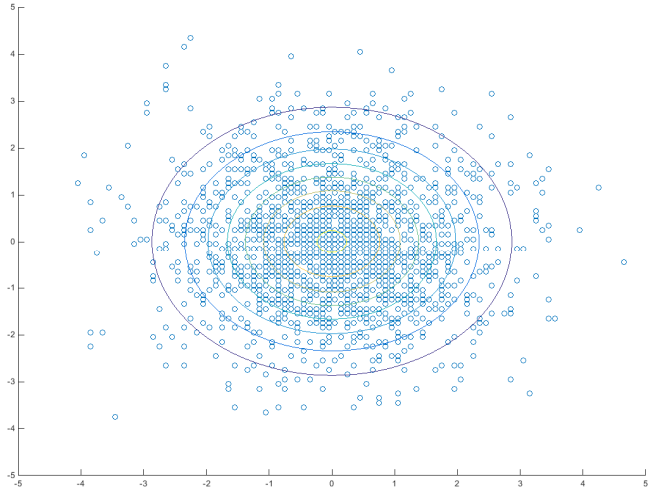

Figure 5.8: Contour Plot of (5.1) with 1232 Representative Samples (100-by-100 grid)

The mean and covariance matrix esitmates of (5.1) of the representative samples in Figure 5.8 are, respectively,

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}0.02\\ -0.07\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}1.94 & -0.07\\ -0.07 & 1.94\end{bmatrix}.$$

5.2 MC-Sampling from Independent System Components

Taking a closer look at (5.1), we see that this bivariate probability distribution can be factored as a product of two normal probability distributions $N(\mu,\sigma^2)$, each having a mean of $\mu=0$ and a variance of $\sigma^2=1.96$:

$$p(x,y)=p(x)p(y), \tag{5.2}$$

where

$$p(x)=\frac{1}{\sqrt{2\pi(1.96)}}\exp\left(-\frac{x^2}{2(1.96)}\right) \tag{5.3}$$

and

$$p(y)=\frac{1}{\sqrt{2\pi(1.96)}}\exp\left(-\frac{y^2}{2(1.96)}\right). \tag{5.4}$$

In order to come to a MC sample from (5.1), we may make use of the factorization in (5.2), by first taking a MC-sample from (5.3). The cdf of (5.3) is given as

$$\int_{-\infty}^{x}p(v)dv=\Phi\left(\frac{x}{\sqrt{1.96}}\right)=\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x}{\sqrt{2(1.96)}}\right)\right], \tag{5.5}$$

where the $\operatorname{erf}(x)$ function is part of the MATLAB library, Figure 5.9.

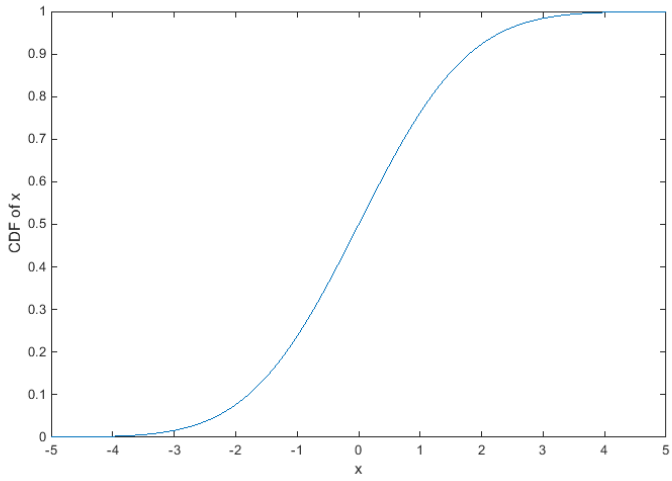

Figure 5.9: Cumulative Distribution Function (CDF) of *x*

Let u be random realisation from the unit uniform probability distribution $U(0,1)$:

$$u \sim U(0,1). \tag{5.6}$$

Then the x-value MC-realisation for the first factorization is given as as the solution of the equality, (5.5) and (5.6),

$$\frac{1}{2}\left[1+\operatorname{erf}\left(\frac{x_{MC}}{\sqrt{2(1.96)}}\right)\right]=u\,. \tag{5.7}$$

It follows from (5.7) that for (5.3) the x-value MC-realisation can be given analytically as

$$x_{MC}=\sqrt{2(1.96)}\,\operatorname{erf}^{-1}(2u-1). \tag{5.8}$$

where the inverse error function $\operatorname{erf}^{-1}(x)$ is part of the MATLAB library. Likewise, as (5.2) and (5.3) are equal, apart from their variable labelling, we have that for a new realisation of (5.6) we obtain the y-value MC-realisation

$$y_{MC}=\sqrt{2(1.96)}\,\operatorname{erf}^{-1}(2u-1). \tag{5.9}$$

Let u_1 and u_2 be two separate realisations from (5.6), then a single MC-realisation from (5.1) may be obtained as

$$(x_{MC},y_{MC})=\left[\sqrt{2(1.96)}\,\operatorname{erf}^{-1}(2u_1-1),\sqrt{2(1.96)}\,\operatorname{erf}^{-1}(2u_2-1)\right]. \tag{5.10}$$

By way of this simple MC-sampling, we then may obtain a set of $n=50$ representative samples. These samples are plotted together with the contourplot of (5.1), Figure 5.10.

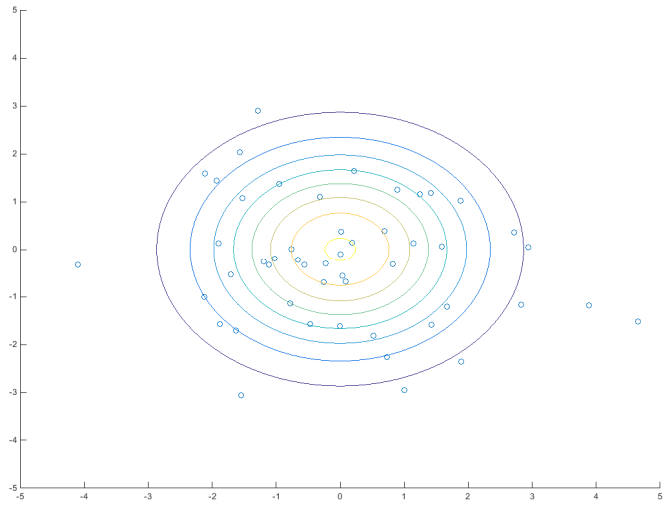

Figure 5.10: Contour Plot of (5.1) with 50 Representative Samples

The mean and covariance matrix esitmates of this representative sample are given as

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}0.06\\ -0.26\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}2.84 & -0.39\\ -0.39 & 1.65\end{bmatrix}.$$

Now, if we sample a set of $n=1232$ representative MC-realisations, we obtain Figure 5.11.

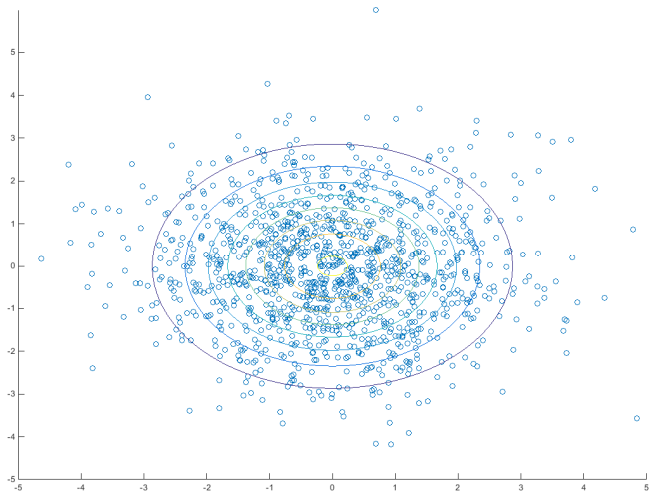

Figure 5.11: Contour Plot of (5.1) with 1232 Representative Samples

The mean and covariance matrix esitmates of (5.1) of the representative samples in Figure 5.11 are, respectively,

$$\begin{pmatrix}\bar{x}\\ \bar{y}\end{pmatrix}=\begin{pmatrix}-0.03\\ -0.08\end{pmatrix}, \qquad \text{and} \qquad S=\begin{bmatrix}2.12 & -0.08\\ -0.08 & 1.90\end{bmatrix}.$$

5.3 Sampling Recommendation

If a multivariate probability distribution can be factorised in a product of several probability distributions of lesser dimensionality, as is the case in (5.1), via (5.2)-(5.4) , then no univariate representation is needed for the system as a whole, as one can sample the factorisations separately. Moreover, if these factorisations allow for simple MC-sampling, as is the case for (5.3) and (5.4), then one may obtain very easily and large numbers representative of samples from factorised system of independent components. A non-trivial example of a system of independent components is the system of damage state probabilities for the bridges under the stress scenario in Figure 3.2, as we have that for a given river discharge scenario the damage state probabilities for each bridge in the system are independent of (i.e. unconditional on) the damage states of the other bridges, (3.11).

If a multivariate probability distribution cannot be factorised in a product of several probability distributions of lesser dimensionality, as is the case in (4.1), then some kind of univariate representation is needed for the system as a whole, as this will allow for either a simple MC-sampling or a somewhat more involved staircase sampling from this univariate representation (see Figures 4.6-4.11). A non-trivial example of a system of 'dependent' components is the 'system' of fragility parameters values in the joint probability distribution (3.5), as we have that this joint probability distribution cannot be factorised in probability distributions of the respective fragility parameters α_1, α_2, α_3, and β. In the next chapter we will present the Nested Sampling algorithm (Skilling, 2004), by which we may obtain univariate representations of any probability distribution which defined on a system of depedent components and/or joint probability distribution which cannot be factorized to a product of univariate probability distribution.

So our recommendation is as follows. For a probability distribution which is defined on a system of independent components, use simple MC-sampling or a variation thereof for each of the independent components. For a probability distribution which is defined on a system of dependent components, first obtain a univariate representation of that system probability distribution, either by brute force evalution (see Chapter 4) or by way of Nested Sampling (see Chapter 6), and then use simple MC-sampling or a variation thereof on that univariate representation.

In closing, the actual ordering of the area elements are not pertinent to either sampling method; i.e. from a sampling point of view a monotonic increasing function is , say, $h(w)$ is just as good as a descending one. What is key, however, is that the multivariate distribution (Figures 4.1 and 5.1) has been reduced to an univariate one (4.5 and 5.5), which then allows us to sample that univariate representation by constructing its corresponding cumulative distribution function (Figures 4.6 and 5.6) or, alternatively, its staircase sampler (Figures 4.7 and 4.8).

It is only because Nested Sampling takes advantage of the fact that it has *constructed* $g(w)$ to be monotonic descending (see Chapter 6) that the monotonic descending form, as displayed in Figures 4.5 and 5.5, is prefererred over any other.

6.0 NESTED SAMPLING

By reducing any *k*-variate probability distribution p to a corresponding monotonic descending univariate function g, and by using order statistics, the univariate representation of any *k*-variate probability distribution p may be evaluated using a Monte Carlo sampling scheme called Nested Sampling (Skilling, 2004; Skilling, 2006). Nested Sampling is used in the case study in Chapter 3 to come to an estimate of the joint probability distribution of the fragility parameters (3.5).

6.1 Sampling Abcissa's

Say, we have some multivariate probability distribution $p(\mathbf{x})$ for which we want to obtain an univariate representation, say,

$$g(w), \qquad \text{for } 0 < w \le W. \tag{6.1}$$

By construction, we may let g be some montonic descending function of w. Let $\mathbf{x}^{(w)}$ be some some point in the parameter space of p, then $p(\mathbf{x}^{(w)})$ will correspond, *by construction*, with some value $g(w)$. Now, if we have a value of the ordinate $g(w)$ (i.e. the "*y-value*") without knowing the corresponding abscissa value w (i.e. the "*x-value*). Then the only thing we know about w is that it take on a value somewhere in the range

$$0 < w \le W, \tag{6.2}$$

where W is the (hyper-)volume that spans the parameter space of $\mathbf{x} = (x_1, x_2, \ldots, x_N)$; that is,

$$W = R_{x_1} R_{x_2} \cdots R_{x_N}, \tag{6.3}$$

where R_{x_k} is the range that spans the domain of parameter x_k; e.g. in Figure 4.5 $W = (10)(10) = 100$. The (uninformed) state of knowledge (6.2) translates directly to the state of knowledge that w is uniformly distributed as

$$p(w) = \frac{1}{W}, \qquad \text{for } 0 < w \le W, \tag{6.4}$$

with a mean of

$$E(w) = \frac{W}{2}, \tag{6.5}$$

and a standard deviation of

$$std(w) = \frac{W}{2\sqrt{3}}, \tag{6.6}$$

Now, suppose that we sample n values of $g(w)$, that is, we have sampled $g(w_1),\ldots,g(w_n)$, and though we still do not know the values of $w_1,\ldots,w_n$, the one thing we now do know is that the smallest realisation of $g(w)$ must correspond with the greatest value of w. This is because function $g(w)$ is, *by construction*, a monotonic descending function. It follows that we may use an order distribution for the unknown value $w_{\max}$:

$$p(w_{\max}) = n\left(\frac{w_{\max}}{W}\right)^{n-1}\frac{1}{W}, \qquad \text{for } 0 < w_{\max} \le W, \tag{6.7}$$

with mean of

$$E(w_{\max}) = \frac{n}{n+1}W, \tag{6.8}$$

and a standard deviation

$$std(w_{\max}) = \sqrt{\frac{n}{n+2}}\,\frac{W}{n+1} \to \frac{W}{n+1}, \qquad \text{as } n >> 1, \tag{6.9}$$

Note that the standard deviation, that is, our uncertainty regarding the unknown value of $w_{\max}$, falls of with a factor of approximately n . It will be seen that (6.8) and (6.9) form the backbone of the Nested Sampling algorithm.

6.2 The Basic Nested Sampling Algoritm

In this discussion of the Nested Sampling algorithm we will not protect against under- and overflow. We will just focus here on the basic philosophy which underlies Nested Sampling.

Step 1.

Find n random values $\mathbf{x}_i$ in the $\mathbf{x}$-plane, where all the states $\mathbf{x}_i$ are assumed to be equally probable and greater than zero

$$p(\mathbf{x}_i) = P_i > 0, \qquad \text{for } i = 1,2,\ldots,n. \tag{6.10}$$

It holds trivially that the n values of $p(\mathbf{x}_i)$ also correspond with n values of its univariate representation $g(w)$, as we always may perform the steps as shown in Figures 4.1- 4.5 and Figures 5.1- 5.5.

In the absence of an explicit sorting of the volume/area elements, we cannot map the $\mathbf{x}$-coordinates to the corresponding w-coordinate explicitly. But the thing we can do is use (6.8) to statistically approximate this corresponding w-coordinate for the $\mathbf{x}_i$ that gives the smallest observed P_i (6.10), and thus get our first coordinate (w_1, g_1) of the unknown function $g(w)$, where

$$w_1 = \frac{n}{n+1}W, \qquad g_1 = \min_i(P_i) = \min_i[p(\mathbf{x}_i)], \tag{6.11}$$

where the error of our estimated w_1 will fall of with a factor n, as can be seen in (6.9). We now approximate the integral right of w_1 (6.11) as

$$A_1 = \int_{w_1}^{W} g(w)dw \approx (W - w_1)g_1 = \frac{W}{n+1}g_1, \tag{6.12}$$

and we set

$$Z_1 = A_1. \tag{6.13}$$

Step 2.

We again find n random values $\mathbf{x}_j$ in the $\mathbf{x}$-plane, but now we constrain these random values to be equal or greater than the value of the minimum of the last iterate, that is, we sample $\mathbf{x}_j$ under the constraint (6.11)

$$p(\mathbf{x}_j) = P_j \geq g_1, \qquad \text{for } j = 1,2,\ldots,n, \tag{6.14}$$

where all the states $\mathbf{x}_j$ which adhere to this constraint are assumed to be equally probable. Let

$$w_1 = w^*, \tag{6.15}$$

then we may rewrite the constraint (6.14) as the equivalently constraint

$$g(w_j) \geq g(w^*), \qquad \text{for } j = 1,2,\ldots,n, \tag{6.16}$$

as it holds trivially that the n values of $p(\mathbf{x})$ must correspond with n values of $g(w)$. Now, since $g(w)$ is by construction a monotonic descending function, and since w_1 is the coordinate that is associated with the lowerbound g_1 (6.11), it follows that the equivalent sampling constraints (6.14) and (6.16) imply for the unknown w_j the constraint

$$0 < w_j \le w^* . \tag{6.17}$$

So, again by way of (6.8), but now replacing W with w_1, the second coordinate (w_2, g_2) of the unknown function $g(w)$ may be estimated as

$$w_2 = \frac{n}{n+1} w_1 , \qquad g_2 = \min_j (P_j) = \min_j \left[p(\mathbf{x}_j) \right]. \tag{6.18}$$

We now approximate the area integral from w_2 to w_1 as

$$A_2 = \int_{w_2}^{w_1} g(w) dw \approx \frac{w_1 - 1}{n+1} g_2 , \tag{6.19}$$

and we approximate the area integral from w_2 tot W as

$$Z_2 = \int_{w_2}^{W} g(w) dw \approx A_1 + A_2 . \tag{6.20}$$

Step *t*.

For the t^{th} iterate we find n random values $\mathbf{x}_k$ in the $\mathbf{x}$-plane under the constraint

$$p(\mathbf{x}_k) = P_k \ge g_{t-1} , \qquad \text{for } k = 1,2,\ldots,n , \tag{6.21}$$

where all the states $\mathbf{x}_j$ which adhere to this constraint are assumed to be equally probable. The ordinate of the t^{th} coordinate (w_t, g_t) of the unknown function $g(w)$ may be estimated as

$$g_t = \min_k (P_k) = \min_k \left[p(\mathbf{x}_k) \right], \tag{6.22}$$

and its corresponding abscissa, by way of the order statistic (6.8), is estimated as

$$w_t = \frac{n}{n+1} w_{t-1} . \tag{6.23}$$

We now approximate the area integral from w_t to w_{t-1} as

$$A_t = \int_{w_t}^{w_{t-1}} g(w)dw \approx \frac{w_{t-1}-1}{n+1} g_t \,, \tag{6.24}$$

And we approximate the area integral from w_t tot W as

$$Z_t = \int_{w_t}^{W} g(w)dw \approx \sum_{i=1}^{t} A_i \,. \tag{6.25}$$

Termination Step.
We have that $\lim_{t\to\infty} w_t = 0$, because of the identity:

$$w_t = \left(\frac{n}{n+1}\right)^t W \,. \tag{6.26}$$

So, if we want to find the iteration T at which we need to terminate the Nested Sampling run we may solve

$$\left(\frac{n}{n+1}\right)^T W = w_T \tag{6.27}$$

for T, which gives

$$T = \frac{\log\left(\frac{w_T}{W}\right)}{\log\left(\frac{n}{n+1}\right)} \,. \tag{6.27}$$

where w_T is the point on the w-axis where we stop to evaluate the function g, W is the (hyper-) volume of the parameter space (6.3), and n is the number of samples which are sampled uniformly with the likelihood constraint at each iteration step (i.e. n is number of 'Nested Sampling objects'). For example, we may let the Nested Sampling algorithm run T iterations until $w_T = 0.1$ or, if we wish more precision, until $w_T = 0.01$, as we have that $0 < w \leq W$.

6.3 Issues of Computational Efficiency

In the Nested Sampling algorithm we need at each iteration t to obtain n equiprobable samples $\mathbf{x}_k$ under the constraint (6.21)

$$p(\mathbf{x}_k) \geq g_{t-1}, \qquad \text{for } k = 1, \ldots, n,$$

where $g_0 = 0$. One computationally wasteful way to obtain these n equiprobable samples is to just draw random samples until we have obtained the necessary n samples that adhere to (6.21). Another, more efficient way is to realize that at iteration step $(t-1)$ we already had $(n-1)$ objects that satisified both the constraint (7.22) as well as the desideratum of equiprobability; see (6.10), (6.14), and (6.21). If we keep these $(n-1)$ objects, then we only need to sample one aditional object in order to obtain our needed sample of n objects.

In the words of Skilling (2004): "After each iteration t we discard one object from our sample of n objects. This discarded object becomes the lowerbound g_t. The $(n-1)$ surviving objects are taken to the next iteration and an additional object $\mathbf{x}_n$ is sampled under the constraint $p(\mathbf{x}_n) \geq g_t$. This implementation reduces the computational costs with an order of magnitude of n."

The generating of an additional object $\mathbf{x}_n$ under:

(a) the constraint $p(\mathbf{x}_n) \geq g_t$,

(b) the desideratum of equiprobability,

at time step $(t+1)$ is where the computational overhead of Nested Sampling lies. The constraint (a) is simple enough to enforce. All proposals $\mathbf{x}_n$ with probabilities $p(\mathbf{x}_n) < g_t$ are simply rejected. However, the desideratum of equiprobability is more difficult to fullfill. Equiprobability means that all the states $\mathbf{x}_i$, for which we have $p(\mathbf{x}_i) \geq g_t$, must have the same probability of being sampled. For the implementation of the Nested Sampling algorithm in the InfraRisk, we use a simple MCMC sampler (Sivia and Skilling, 2006; Section 9.6.3).

7.0 THE PROBABILITY SORT ALGORITHM

We now discuss the concept/background behind the Probability Sort algorithm for the case where we have only two damage states $M = 2$; that is, the states damaged and not-damaged. The MATLAB code, together with a pseudo-code for the general case of arbitrary M is given in Appendix A.

For the case where $M = 2$, the number of possible damage states will be 2^N. The Probability Sort algorithm goes from the most likely damage state $\mathbf{x}_i^{(1)}$, to the next likely damage state $\mathbf{x}_i^{(2)}$, to the next likely damage state $\mathbf{x}_i^{(3)}$, and so on, such that

$$p\left(\mathbf{x}_i^{(s)}\right) \geq p\left(\mathbf{x}_i^{(t)}\right), \qquad \text{for } s < t \ . \tag{7.1}$$

The selection of the $\mathbf{x}_i^{(s)}$ is done so efficiently that there are no rejections in the damage state proposals. Moreover, the selection itself only takes $O(N)$ time.

For the specific case $M = 2$, the state vectors $\mathbf{x}_i$, for $i = 1,2,\ldots,2^N$, may be constructed as vector consisting of 0 and 1's. The probabilities of an element x_k in $\mathbf{x}_i$ being either 0 or 1 is dependent on the PGA value which is associated with that element

$$p\left(x_{j_{(k)}} \mid PGA_k\right), \qquad \text{for } j_k = 1,2, \tag{7.2}$$

and $k = 1,2,\ldots,N$. So, the probability which is associated with a given $\mathbf{x}_i$ may be computed as

$$p(\mathbf{x}_i) = \prod_{k=1}^{N} p\left(x_{j_{(k)}} \mid PGA_k\right). \tag{7.3}$$

Now, let

$$p_k^{\max} = \max\left[p\left(x_{j_{(k)}} = 1 \mid PGA_k\right), p\left(x_{j_{(k)}} = 0 \mid PGA_k\right)\right] \tag{7.4}$$

be the maximum possible damage state probability for component k, and let

$$x_k^{\max} \in \{0,1\} \tag{7.5}$$

be the damage state which corresponds with this maximum probability. Then the damage state vector with maximum probability,

$$P^{\max} = \prod_{k=1}^{N} p_k^{\max}, \tag{7.6}$$

is given as

$$\mathbf{x}^{\max} = \left\{x_1^{\max}, x_2^{\max}, \ldots, x_N^{\max}\right\}. \tag{7.7}$$

Now, it stands to reason that this 'Most Likelihood' damage state vector $\mathbf{x}^{\max}$ should be the first and foremost of all the possible damage state scenarios of which the consequences should be evaluated; that is,

$$\mathbf{x}_i^{(1)} = \mathbf{x}^{\max}, \tag{7.8}$$

where, by construction,

$$p\left(\mathbf{x}_i^{(1)}\right) = p\left(\mathbf{x}^{\max}\right) = P^{\max}. \tag{7.9}$$

If we follow this line of reasoning, then the second best damage state proposal would be that damage state vector which has the second highest probability.

Now, the minimum possible probability for a damage state for component k is given as (7.4):

$$p_k^{\min} = 1 - p_k^{\max} = \min\left[\ p\left(x_{j_{(k)}} = 1 \mid PGA_k\right), p\left(x_{j_{(k)}} = 0 \mid PGA_k\right)\ \right], \tag{7.10}$$

with corresponding damage state (7.5):

$$x_k^{\min} = 1 - x_k^{\max} \in \{0,1\}. \tag{7.11}$$

Let

$$\mathbf{p}^{\min} = \left\{p_1^{\min}, p_2^{\min}, \ldots, p_N^{\min}\right\} \tag{7.12}$$

be the vector with the minimum probabilities for the N components. Then the damage state $x_q^{\min}$ which corresponds with the maximum of the minimum vector (7.12)

$$p_q^{\min} = \max\left(\mathbf{p}^{\min}\right) \tag{7.13}$$

is the only possible candidate for as state switch (7.11):

$$\mathbf{x}_i^{(2)} = \left\{x_1^{\max}, x_2^{\max}, \ldots, x_{q-1}^{\max}, x_q^{\min}, x_{q+1}^{\max}, \ldots, x_N^{\max}\right\} \tag{7.14}$$

So, the probability which corresponds with this second best proposal is (7.4), (7.6) and (7.13):

$$p\left(\mathbf{x}_i^{(2)}\right)=\frac{p_q^{\min}}{p_q^{\max}}P^{\max}. \tag{7.15}$$

as the *q*th state probability has switched from its maximum probability state to its minimum.

Now, the third most probable damage state vector necessarily will also be of the form where only one damage state, say x_u, is being switched, as we reset x_q to its original damage state value in (7.7):

$$\mathbf{x}_i^{(3)}=\left\{x_1^{\max},x_2^{\max},\ldots,x_{u-1}^{\max},x_u^{\min},x_{u+1}^{\max},\ldots,x_{q-1}^{\max},x_q^{\max},x_{q+1}^{\max},\ldots,x_N^{\max}\right\}. \tag{7.16}$$

But as we come to the fourth most probable damage state vector, then we find that we bifurcate into the possibility of either both x_u and x_q being switched,

$$\mathbf{x}_i^{(4a)}=\left\{x_1^{\max},x_2^{\max},\ldots,x_{q-1}^{\max},x_q^{\min},x_{q+1}^{\max},\ldots,x_{u-1}^{\max},x_u^{\min},x_{u+1}^{\max},\ldots,x_N^{\max}\right\}, \tag{7.17}$$

or x_u being reset to its original damage state value in (7.7), as we switch some other element, say x_w:

$$\mathbf{x}_i^{(4b)}=\left\{x_1^{\max},x_2^{\max},\ldots,x_q^{\max},\ldots,x_u^{\max},\ldots,x_{w-1}^{\max},x_w^{\min},x_{w+1}^{\max},\ldots,x_N^{\max}\right\}. \tag{7.18}$$

In Appendix A the Probability Sort switching algorithm is given which produces scenario proposals ordered by their probabilities.

8.0 THE MODELLING OF CASCADING EFFECTS

The INFRARISK project is concerned with the behaviour of critical infrastructures, such as road and rail networks, when subjected to natural hazards such as landslides, floods, earthquakes or a combination of all three. These natural hazards, as well as the consequent behaviour of the infrastructural elements, vary both spatially and temporally.

For example, the closer an infrastructural objects is to the epicentre of some seismic event, the greater will be its tendency to be in a damaged state. Moreover, if the damage state of one infrastructural objects is dependent on the damage state of another, then as the latter infrastructural object is damaged and time progresses the greater will be the probability of the former infrastructural object to be in a damaged state.

One example of such a system of temporally related systems of interdependent damage states is, say, an infrastructural system where the levee damage states are dependent upon the damage state of the electrical infrastructure (flooding influences functionality of power generators), and vise versa (levees are powered by electricity). Another example is, say, a system consisting of pressurized fuel storage tanks, where an exploded damage state of one or more storage tanks will be of influence, by way of initial overpressure and subsequent heat radiation, on the damage states of the surrounding storage tanks.

In this chapter we will discuss the modelling of temporal dependencies between systems of interdependent damage states, by way of the latter fuel storage field example, as this is the archetypical example of a cascading effect scenario.

8.1 The 'Physics' Behind the Probability Map

For our fuel storage field it is assumed that the explosion of a given pressurized fuel storage tank generates a heat radiation of, say, 200 kW/m^2 which falls of, say, as the inverse of the distance. Moreover, it is assumed that the total heat radiation for multiple explosions is a superposition of the heat radiation of the separate explosions.

For example, if we have K exploded fuel tanks, having coordinates (X_k, Y_k), for $k = 1, \ldots, K$. Then the total heat radiation R which is experienced by an intact fuel tank having coordinates (x, y) is given as

$$R(x,y) = \sum_{k=1}^{K} \frac{200}{\sqrt{(x-X_k)^2 + (y-Y_k)^2}}. \qquad (8.1)$$

The corresponding (probit) probability of being damaged is given as

$$P(x,y) = \frac{1}{2}\left(1 + erf\left[\frac{-4.7534 + R(x,y)}{\sqrt{2}}\right]\right), \qquad (8.2)$$

which for a heat radiation of $R(x_0, y_0) = 0$ will give a corresponding base-line damage probability of $P(x_0, y_0) = 10^{-6}$.

8.2 Some Example Probability Maps

Say we have $N = 25$ fuel storage objects arranged in a 5-by-5 grid with, say, a distance of 50 meters between horizontally and vertically adjacent objects and a distance of

$$70.71 = \sqrt{50^2 + 50^2}$$

meters between diagonally adjacent objects. If we let the fuel storage objects with coordinates $(150, 150)$ and $(150, 100)$ explode, then we obtain the state matrix in Table 8.1.

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	0	0

Table 8.1: State Matrix 1

The corresponding probability map may be constructed by way of (8.1) and (8.2), Table 8.2.

0.0129	0.0446	0.0778	0.0446	0.0129
0.0605	0.4459	0.8937	0.4459	0.0605
0.1674	0.9810	1	0.9810	0.1674
0.1674	0.9810	1	0.9810	0.1674
0.0605	0.4459	0.8937	0.4459	0.0605

Table 8.2: Explosion Probability Map for State Matrix 1

Alternatively, if we let the fuel storage objects with coordinates $(150, 150)$, $(150, 200)$, $(100, 150)$ explode, then we obtain the state matrix in Table 8.3.

0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
0	0	0	0	0
0	0	0	0	0

Table 8.3: State Matrix 2

The corresponding probability map may be constructed by way of (10.1) and (10.2), Table 8.4.

0.5943	0.9688	0.9988	0.8994	0.3296
0.9688	1.0000	1	0.9999	0.6180
0.9988	1	1	1.0000	0.6438
0.8994	0.9999	1.000	0.9508	0.3877
0.3296	0.6180	0.6438	0.3877	0.1313

Table 8.4: Explosion Probability Map for State Matrix 2

It may be glanced from Tables 8.2 and 8.4 that the superposition of heat radiation in (8.1) in all likelihood will lead to a cascade of explosions.

8.3 Probability Sort Analysis of Cascading Effects

Say we have $N = 25$ fuel storage objects arranged in a 5-by-5 grid with, say, a distance of 50 meters between horizontally and vertically adjacent objects. The primary initiating event, at time step $t = 0$, is the event where the centre fuel storage object with coordinates $(150, 150)$ has exploded, Table 8.5.

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Table 8.5: State Matrix of Primary Event at *t* = 1

The corresponding probability map may be constructed by way of (10.1) and (10.2), Table 8.6.

0.0004	0.0015	0.0029	0.0015	0.0004
0.0015	0.0271	0.2256	0.0271	0.0015
0.0029	0.2256	1	0.2256	0.0029
0.0015	0.0271	0.2256	0.0271	0.0015
0.0004	0.0015	0.0029	0.0015	0.0004

Table 8.6: Explosion Probability Map for Primary Event at *t* = 1

The number of damage states is $M = 2$, the number of objects is $N = 25$, and the number of elements in a damaged (i.e. exploded) state is $K = 1$. So the total state space which corresponds with the explosion probability map in Table 8.6 is

$$M^{N-K} = 2^{25-1} = 2^{24} = 1.68 \times 10^7 . \qquad (8.3)$$

It follows that following the primary event in Table 8.5, we will have 2^{24} possible event scenarios at each time step. Among these higher order event scenarios are the scenarios in Tables 8.1, 8.3, and 8.5, with corresponding probability maps Tables 8.2, 8.3, and 8.6.

As we have an irreversible process (i.e exploded fuel storage tanks cannot 'unexplode'), the total number of scenario routes at a given time step may be modelled by way of a 2^{24}-by-2^{24} Markovian transition matrix, having

$$2^{24} \times 2^{24} = 2.81 \times 10^{14} \tag{8.4}$$

elements. Now, a state matrix with K explosions will map to possible 2^{25-K} end points. So, our hypothetical 2^{24}-by-2^{24} Markovian transition matrix has

$$2^{24} + \sum_{i=1}^{24} \frac{24!}{i!\,(24-i)!} 2^{24-i} = 2.82 \times 10^{11} \tag{8.5}$$

non-zero probability elements. In other words, at a given time step $t > 0$ there are 2.82×10^{11} admissible routes in which we go from one of the 1.68×10^{7} possible starting scenarios to some admissible subset of the total scenario space, with subsets ranging from 1.68×10^{7} scenarios to 1 scenario.

This overwhelming number of admissible routes (8.5) notwithstanding, it is found that the Probability Sort algorithm will give very decent probability coverages over the time steps for given probability cut-offs for the primary event in Table 8.5, with a probability map 'physics' of (8.1) and (8.2), Table 8.6. In Table 8.7 these probability coverages are given together with the number of active probability components at each time step.

Time Step	Cut-off = 10^{-6}		Cut-off = 10^{-7}	
	coverage	# components	coverage	# components
1	0.9995	1094	0.9999	2459
2	0.9177	33100	0.9754	111430
3	0.8745	16104	0.9608	61476
4	0.8529	7069	0.9527	32864
5	0.8426	2417	0.9484	15976
6	0.8382	651	0.9463	7045
7	0.8365	126	0.9452	2373

Table 8.7: Probability Coverage and Number of Active Probability Components

The probability cut-offs in Table 8.7 are enforced such that the probability for a given scenario, (8.3), at a given time step is not smaller than that cut-off.

It may be glanced from the time progression of the number of active probability components in Table 8.7 that the primary event in Table 8.5, together with (8.1) and (8.2), will lead us from an initial low-entropic state, to an intermediate higher-entropic state, back to a final low entropic state. This may be explained as follows.

Initially, we only expect the fuel storage objects which are horizontally and vertically adjacent to the exploded object to reach an exploded state, Table 8.6. Because of the superposition of heat radiation we expect (see Tables 8.2 and 8.4) the fuel storage tank field to cascade as time progresses to a total conflagration state. But we are uncertain as to the route that will take us from the initial low entropic state to this final low entropic state. This uncertainty translates to an intermediate higher entropic state where the probabilities are more spread out over the total state space and, consequently, more active probability components are in play.

8.3.1 Time Evolving Marginal Damage State Probabilities

If, for the cut-off of 10^{-7}, we weigh the damage state 'matrices' $\mathbf{x}^{(s)}$ by the normalized probabilities

$$\tilde{P}^{(s)} = \frac{P^{(s)}}{\sum_{s=1}^{S} P^{(S)}}\ , \tag{8.6}$$

where $P^{(s)}$ is the probability of $\mathbf{x}^{(s)}$ and S is the total number of active probability components, or, equivalently, probability sort scenario proposals, then we obtain the following expected marginal probabilities, say, $E(\boldsymbol{\theta})$, where

$$E(\boldsymbol{\theta}) = \sum_{s=1}^{S} \tilde{P}^{(s)}\, \mathbf{x}^{(s)}\ , \tag{8.7}$$

of being in an exploded state, Tables 8.8-8.14.

.0004	0.0015	0.0029	0.0015	0.0004
0.0015	0.0271	0.2256	0.0271	0.0015
0.0029	0.2256	1.000	0.2256	0.0029
0.0015	0.0271	0.2256	0.0271	0.0015
0.0004	0.0015	0.0029	0.0015	0.0004

Table 8.8: Estimated Explosion Probability Map at *t* = 1

0.1426	0.2902	0.3703	0.2902	0.1426
0.2902	0.5637	0.7353	0.5637	0.2902
0.3703	0.7353	1.000	0.7353	0.3703
0.2902	0.5637	0.7353	0.5637	0.2902
0.1426	0.2902	0.3703	0.2902	0.1426

Table 8.9: Estimated Explosion Probability Map at *t* = 2

0.7318	0.7776	0.8024	0.7776	0.7318
0.7776	0.8624	0.9161	0.8624	0.7776
0.8024	0.9161	1.000	0.9161	0.8024
0.7776	0.8624	0.9161	0.8624	0.7776
0.7318	0.7776	0.8024	0.7776	0.7318

Table 8.10: Estimated Explosion Probability Map at t = 3

0.9178	0.9315	0.9390	0.9315	0.9178
0.9315	0.9571	0.9736	0.9571	0.9315
0.9390	0.9736	1.000	0.9736	0.9390
0.9315	0.9571	0.9736	0.9571	0.9315
0.9178	0.9315	0.9390	0.9315	0.9178

Table 8.11: Estimated Explosion Probability Map at t = 4

0.9754	0.9794	0.9815	0.9794	0.9754
0.9794	0.9868	0.9918	0.9868	0.9794
0.9815	0.9918	1.000	0.9918	0.9815
0.9794	0.9868	0.9918	0.9868	0.9794
0.9754	0.9794	0.9815	0.9794	0.9754

Table 8.12: Estimated Explosion Probability Map at t = 5

0.9929	0.9939	0.9945	0.9939	0.9929
0.9939	0.9960	0.9974	0.9960	0.9939
0.9945	0.9974	1.000	0.9974	0.9945
0.9939	0.9960	0.9974	0.9960	0.9939
0.9929	0.9939	0.9945	0.9939	0.9929

Table 8.13: Estimated Explosion Probability Map at t = 6

0.9981	0.9983	0.9984	0.9983	0.9981
0.9983	0.9988	0.9992	0.9988	0.9983
0.9984	0.9992	1.000	0.9992	0.9984
0.9983	0.9988	0.9992	0.9988	0.9983
0.9981	0.9983	0.9984	0.9983	0.9981

Table 8.14: Estimated Explosion Probability Map at t = 7

It may be glanced from Tables 8.8-8.14, that the marginal probabilities of being in an exploded state will increase in magnitude as time progresses. Also note that the estimated marginal probabilities of being in an exploded stated at time step 1, Table 8.8, are the same as the analytical probability map

in Table 8.6, which was obtained by way of the primary event in Table 8.5 and the probability map model (8.1) and (8.2).

8.3.2 Time evolving ML-damage states

We now will focus on the change in probabilities of four representative fixed damage state scenarios, Tables 8.15-8.18.

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Table 8.15: State Matrix 1

In Table 22 we have the total containment scenario, where no additional fuel storage objects explode.

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	0	0

Table 8.16: State Matrix 2

0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
0	0	0	0	0
0	0	0	0	0

Table 8.17: State Matrix 3

In Tables 8.16 and 8.17 we have limited spill-off scenarios, where, respectively, one and two additional fuel storage objects have exploded.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Table 8.18: State Matrix 4

In Table 8.18 we have the total destruction scenario, where all the storage objects have exploded. We now take a look at the progression of the probabilities of these damnage states as time progresses, where we put the Most Likelihood (ML) probabilities in boldface, Table 8.19.

Time Step	*P*(State matrix 1)	*P*(State matrix 2)	*P*(State matrix 3)	*P*(State matrix 4)
1	**0.3140**	0.0915	0.0267	9.17×10^{-56}
2	**0.0986**	0.0287	0.0084	0.0282
3	0.0310	0.0090	0.0026	**0.6703**
4	0.0097	0.0028	0.0008	**0.8650**
5	0.0031	0.0009	0.0003	**0.9226**
6	0.0010	0.0003	8.14×10^{-5}	**0.9390**
7	0.0003	8.78×10^{-5}	2.56×10^{-5}	**0.9433**

Table 8.19: Probabilities of State Matriices in Tables 8.15-8.18

At both time steps 1 and 2 the total containment scenario is the ML scenario. From time step 3 onwards, the total destruction scenario becomes the ML scenario. At time step 1 there is still a considerable likelihood that there is either full containment or limited spill-off:

$$0.3140+\binom{4}{1}(0.0915)+\binom{4}{2}(0.0267)=0.8402, \tag{8.8}$$

where the combinatorial factors result from the switching symmetries present in Table 8.6. At time step 2 this likelihood has dropped off dramatically:

$$0.0986+\binom{4}{1}(0.0287)+\binom{4}{2}(0.0084)=0.2638. \tag{8.9}$$

At time step 3 the likelihood of either full containment or limited spill-off has dwindled to a mere

$$0.0310+\binom{4}{1}(0.0090)+\binom{4}{2}(0.0026)=0.0826, \tag{8.10}$$

while the probability of the total destruction scenario is a hefty 0.6703, and as time progresses this probability approaches certainty. Especially so, if we take into account that total probability coverage has not been achieved (i.e., compare the right hand probability coverages in Table 8.7 with the State Matrix 4 probabilities in Table 8.19).

9.0 CONCLUSIONS

This deliverable contains a general stress test framework. In this framework stress tests are just a special instance of a risk assessment, where instead of marginalizing over the entire possible stress scenarios one specific stress scenario is chosen instead for which to gauge its potential effects.

If we wish to conduct a stress test on large probabilistic systems consisting of many stochastic components then, as a matter of practical implementation, the evaluation of the densitity of states will necessitate the use of sampling techniques. If the stochastic components in the probabilistic system under consideration are independent then MC-sampling may be used, if the stochastic components are dependent then Nested Sampling is recommended, and if temporal and spatial correlations (i.e. cascading effects) are to be evaluated on a system of stochastic components, then the Probability Sort algorithm is recommended.

REFERENCES

Adey, B.T., Hackl, J., Heitzler, M. & Iosifescu., I. (2014) Preliminary Model, Methodology and Information Exchange. *Deliverable 4.1, Novel indicators for identifying critical, INFRAstructure at RISK from Natural Hazards, grant agreement No. 603960 July, 93 pages*.

Blaschke, W., Jones, M.T., Majoni, G. & Peria, S.M. (2001) Stress Testing of Financial Systems: An Overview of Issues, Methodologies, and FSAP Experiences. *IMF Working Paper*.

Boje, D. & Murnighan J. (1982) Group confidence pressures in iterative decisions. *Management Science*, 28, 1187-1196.

Brockhoff K. (1975) The perfomance of forecasting groups in computer dialogue and face to face discussions. In Linstone H. & Turoff M. (eds.) *The Delphi Method: Techniques and Applications*. London: Addison-Wesley.

Cheng, T. & Taalab, K. (2014) Integrated Spatio-Temporal Database. *Deliverable 5.1, Novel indicators for identifying critical, INFRAstructure at RISK from Natural Hazards, grant agreement No. 603960, 35 pages*.

D'Ayala D. & Gehl P. (2015) Fragility Functions Matrix. *Deliverable 3.2, Novel indicators for identifying critical, INFRAstructure at RISK from Natural Hazards, grant agreement No. 603960*.

Delbecq, A.L., Van de Ven, A.H. & Gustafson, D.H. (1975) Group Techniques for Program Planning: A Guide to Nominal and Delphi Processes. Scott, Foresman and Co., Glenview, IL.
Delphi Method: Techniques and Applications (2002) Editors: Harold A. Linstone and Murray Turoff . (web edition at http://is.njit.edu/pubs/delphibook/)

Gavin, K. & Martinovic, K. (2014) Critical Infrastructure Database. *Deliverable 2.1, Novel indicators for identifying critical, INFRAstructure at RISK from Natural Hazards, grant agreement No. 603960, 19 pages*.

Gehl, P. & D'Ayala D. (2015) Integrated Multi-Hazard Framework for the Fragility Analysis of Roadway Bridges. *Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering* (ICASP12), Vancouver, Canada, July 12-15.

Gibbs, W., Graves, P. R., & Bernas, R. S. (2001) Evaluation guidelines for multimedia courseware. *Journal of Research on Technology in Education*, 34(1), 2-17.

Jürgen, H., Magnus, H., Juan Carlos, L., Bryan A. & Lorenz H. (2016) Final Model, Methodology and Information Exchange. *Deliverable 4.2, Novel indicators for identifying critical, INFRAstructure at RISK from Natural Hazards, grant agreement No. 603960, 210 pages*.

International Actuarial Association (IAA) Insurance Regulation Committee (2013) Stress Testing and Scenario Analysis.

Jaynes, E.T. (2003) *Probability Theory: The Logic of Science*. Cambridge University Press.

Jaynes, E.T. (1968) Prior Probabilities. *IEEE Trans. Systems Sci. Cybernetics* SSC-4 (3), 227-241.

Jonkman, S.N., Vrijling, J.K. & Van Gelder, P.H.A.J.M. (2006) A Generalized Approach for Risk Quantification and the Relationship Between Individual and Societal Risk. In Guedes Soares & Zio (Eds.), *Safety and reliability for managing risk* (pp. 1051-1059). London: Taylor and Francis Group.

Krauß, M. & Berg H.P. (2011) New Evaluation of External Hazards in the Light of the Fukushima Accident. *9th International Probabilistic Workshop*, Budelmann H., Holst A. and Proske D. (Eds.), Braunschweig, Germany.

Murphy, M.K., Black, N., Lamping, D.L., McKee, C.M., Sanderson, C.F.B., and Askham J. (1998) Consensus development methods and their use in clinical guideline development. *Health Technology Assessment* 2(3).

Prak, P. (2009) Onderzoek Naar de Toepassing van Similarity Judgment Bij het Vaststellen van Alerteringslocaties Binnen de Spoorsector. *SSM thesis*, TU Delft.

Powell, C. (2003). The Delphi technique: Myths and realities. *Methodological Issues in Nursing Research*, 41 (4), 376-382.

Ritchey, T. (1998) General Morphological Analysis: A General Method for Non-Quantified Modelling. *16th EURO Conference on Operational Analysis*, Brussels.

Rowe, G. & Wright G. (2001) Expert opinions in forecasting: role of the Delphi technique. In: Armstrong, editor. *Principles of forecasting: a handbook of researchers and practitioners*. Boston: Kluwer Academic Publishers.

Shinozuka, M., Feng, M. Q., Kim, H., Uzawa, T., and Ueda T. (2003). Statistical analysis of fragility curves. *MCEER University at Buffalo*, State University of New York, Buffalo.

Sivia, D.S. & Skilling, J. (2006) Data Analysis: A Bayesian Tutorial. *Oxford University Press Inc*., New York.

Skilling, J. (2004) Nested Sampling. *In Maximum entropy and Bayesian methods in science and engineering* (ed. G. Erickson, J.T. Rychert, C.R. Smith), AIP Conf. Proc. 735, 395-405.

Skilling, J. (2006): Nested Sampling for Bayesian Computations. *Proceedings Valencia/ISBA 8th World Meeting on Bayesian Statistics*.

Van Danzig, D. (1956) Economic Decision Problems for Flood Prevention. *Econometrica 24*, p. 276 - 287.

Van Erp H.RN., Linger R.O., & Van Gelder P.H.A.J.M., (2016) An Outline of the Bayesian Decision Theory. *In Bayesian Inference and Maximum Entropy Methods in Science and Engineering-35th International Workshop*, Potsdam, New York.

APPENDIX A: THE PROBABILITY SORT ALGORITHM

A.1 Algorithmic Outline

Step 1: Permutate the stateMatrix and the probMatrix into their desired base-line states.

Step 2: Define the function undoPermutate(.) which undoes these base-line permutations in the final probability sorted damage state vectors.

Step 3: Set up the output list probabilitySort and the intermediate Proposals list.

Step 4: enter a While-loop, until the desired number of probability sorted damage state vectors, desiredNumber, has been obtained, or until all possible damage state vectors have been passed through, whichever comes first.

Step 5: take that damage state vector entry from the Proposals list which has the

maximum probability, make the components of that entry available within the While-loop, clean up the Proposals list, update the probabilitySort list by way of these components, and update the total probability coverage variable sumProb.

Step 6: replenish the Proposals list which with a maximum of three new proposals. These new proposals guarantee that all the remaining leaves of the event tree of the damage state space may still be explored, and that the next best probability is always in the updated Proposals list.

Step 7: Print the sumProb probability coverage value and terminate the algorithm. The probabilitySort list consisting of the probability sorted damage state vectors and their corresponding probabilities is now available for the user.

A.2 Pseudo-Code

INPUT

stateMatrix: State matrix/list of the N components under consideration.
probMatrix: State probability matrix/list of the N components under consideration.
desiredNumber: desired number of probability sorted damage state vectors.

OUTPUT

probabilitySort: list consisting of damage state proposals $\mathbf{x}^{(s)}$ with corresponding probabilities $P^{(s)}$, ordered in descending order by way of the probabilities $P^{(s)}$;
sumProb: the total probability density covered by the probabilities of the damage state vectors in the list probabilitySort

ALGORITHM

Step 1.a

If we have M possible damage states for each of the N possible infrastructural elements, then we may define the *N*-by-*M* matrix

$$\text{stateMatrix} = \begin{bmatrix} 1 & 2 & \cdots & M \\ 1 & 2 & \cdots & M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & M \end{bmatrix} \quad (1)$$

the corresponding probability matrix which has its rows the damage state pdf of the corresponding infrastructural element may be given as the *N*-by-*M* matrix ,

$$\text{probMatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1M} \\ \theta_{21} & \theta_{22} & \cdots & \theta_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{N1} & \theta_{N2} & \cdots & \theta_{NM} \end{bmatrix} \quad (2)$$

We then do a Sort over the rows of probMatrix so that per rows the probabilities are in descending order, from large to small:

permutatedProbMatrix =

$$\begin{bmatrix} \max(\theta_{11},\theta_{12},\ldots,\theta_{1M}) & \text{nextMax}(\theta_{11},\theta_{12},\ldots,\theta_{1M}) & \cdots & \min(\theta_{11},\theta_{12},\ldots,\theta_{1M}) \\ \max(\theta_{21},\theta_{22},\ldots,\theta_{2M}) & \text{nextMax}(\theta_{21},\theta_{22},\ldots,\theta_{2M}) & \cdots & \min(\theta_{21},\theta_{22},\ldots,\theta_{2M}) \\ \vdots & \vdots & \ddots & \vdots \\ \max(\theta_{N1},\theta_{N2},\ldots,\theta_{NM}) & \text{nextMax}(\theta_{N1},\theta_{N2},\ldots,\theta_{NM}) & \cdots & \min(\theta_{N1},\theta_{N2},\ldots,\theta_{NM}) \end{bmatrix}. \quad (3)$$

where we track the permutations of each of the rows that take us from probMatrix to permutatedProbMatrix. These permutations are then applied to the corresponding rows in stateMatrix. A possible realization of the resulting permutatedStateMatrix may be, say

$$\text{permutatedStateMatrix} = \begin{bmatrix} 2 & M & \cdots & 1 \\ M & 2 & \cdots & M-1 \\ \vdots & \vdots & \ddots & \vdots \\ M & 1 & \cdots & 2 \end{bmatrix}. \quad (4)$$

The permutatedStateMatrix allows us to keep track of which probabilities in the rows of permutatedProbMatrix point to which damage state.

The first column of the permutatedStateMatrix then gives the damage state vector that has the highest probability of occurring, with a probability of

P = 1;

For[i =1, $i \leq N$,
$P = P \times$ permutatedProbMatrix(i, 1);
i++]

N.B.: Instead of the specific case of a N-by-M matrix, we alternatively, and more generally, may have a list of length N, say, probList, with in each row of that list a discrete probability distribution over M_i damage states, where $1 \leq i \leq N$, which are given in the corresponding rows of the list, say, stateList. For this more general case we may compute a permutatedProbList and a permutatedStateList. The first column of the permutatedStateList then also will give the damage state vector that has the highest probability of occurring, with a probability of P.

Step 1.b

then do a row Sort over the entire permutatedProbMatrix such that in its second column the probabilities are arranged in descending order from large to small. This results in, say, for short, the doublePermutatedProbMatrix, where

second column of doublePermutatedProbMatrix =

$$\begin{pmatrix} \max\left[\text{nextMax}(\theta_{11},\theta_{12},\ldots,\theta_{1M}),\ \text{nextMax}(\theta_{21},\theta_{22},\ldots,\theta_{2M}),\ldots,\text{nextMax}(\theta_{N1},\theta_{N2},\ldots,\theta_{NM})\right] \\ \text{nextMax}\left[\text{nextMax}(\theta_{11},\theta_{12},\ldots,\theta_{1M}),\ \text{nextMax}(\theta_{21},\theta_{22},\ldots,\theta_{2M}),\ldots,\text{nextMax}(\theta_{N1},\theta_{N2},\ldots,\theta_{NM})\right] \\ \vdots \\ \min\left[\text{nextMax}(\theta_{11},\theta_{12},\ldots,\theta_{1M}),\ \text{nextMax}(\theta_{21},\theta_{22},\ldots,\theta_{2M}),\ldots,\text{nextMax}(\theta_{N1},\theta_{N2},\ldots,\theta_{NM})\right] \end{pmatrix} \quad (5)$$

Step 2

Keeping track of the permutations that take us from the permutatedProbMatrix to the doublePermutatedProbMatrix, we may construct the corresponding doublePermutatedStateMatrix. For example, if we have the index vector

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix} \quad (6)$$

of the original row ordering in permutatedProbMatrix, then the corresponding row ordering in both the doublePermutatedProbMatrix and doublePermutatedStateMatrix may be, say,

$$\begin{bmatrix} 4 & 1 & 5 & 2 & 8 & 6 & 10 & 3 & 7 & 9 \end{bmatrix} \quad (7)$$

Now let undoPermutate be that function that rearranges the index vector (7) back the original index vector, or, equivalently, (3) and (4)

undoPermutate[doublePermutatedProbMatrix] = permutatedProbMatrix

(8)

undoPermutate[doublePermutatedStateMatrix] = permutatedStateMatrix.

Step 3.a

Set the vector stateVector as the first column of the doublePermutatedStateMatrix:

stateVector = doublePermutatedStateMatrix(1, :); (9)

or, equivalently, depending on the programming language used,

stateVector = doublePermutatedStateMatrix(1, All);

Likewise, set

P = 1;
For[i =1, $i \leq N$,
$P = P \times$ doublePermutatedProbMatrix(i,1); (10)
i++]

Store both the probability

$$P^{(1)} = P \quad (11)$$

and the unsorted stateVector, see (8),

$$\mathbf{x}^{(1)} = \text{undoPermutate[stateVector]} \quad (12)$$

in a list $\left\{P^{(1)}, \mathbf{x}^{(1)}\right\}$ and insert that list entry into the list probabilitySort

$$\text{probabilitySort} = \left\{ \left\{P^{(1)}, \mathbf{x}^{(1)}\right\} \right\}. \quad (13)$$

Step 3.b

Now the stateVector in (9) gives the damage state vector that has the highest probability of occurring, while being arranged such that that the switching of the first damage state to the damage state of the second entry in the first row of the doublePermutatedStateMatrix will have the next highest probability; that is,

stateVector(1) = doublePermutatedStateMatrix(1, 2); (14)

has a corresponding next best probability of (10)

$$P = [\, P / \text{doublePermutatedProbMatrix}(1,1)\,] \times \text{doublePermutatedProbMatrix}(1,2); \quad (15)$$

In order to reflect the switch operation (14) we initialize the switchVector as the base-line vector

$$\text{switchVector} = \text{zeros}(N, 1)\ ; \quad (16a)$$

which gives

$$\text{switchVector} = [0 \quad 0 \quad \ldots \quad 0]; \quad (16b)$$

after which we switch the first entry of this vector from 0 to 1, so as to reflect the switch operation in (14):

$$\text{switchVector}(1) = 1\ ; \quad (17a)$$

which gives

$$\text{switchVector} = [1 \quad 0 \quad \ldots \quad 0]; . \quad (17b)$$

Also, we set the active switch location as

$$\text{activeSwitch} = 1. \quad (18)$$

Store the adjusted probability (15), the adjusted state vector (14), the switch vector (17*b*), and the active switch location (18) in a list

$$\{\, P\ , \text{stateVector}, \text{switchVector}, \text{activeSwitch}\, \} \quad (19)$$

and insert that list entry into the list Proposals

$$\text{Proposals} = \{\ \{\, P\ , \text{stateVector}, \text{switchVector}, \text{activeSwitch}\, \}\ \}. \quad (20)$$

N.B.: Instead of performing multiplications and divisions on the probabilities in (10) and (15), we also may perform summations and subtractions from the corresponding log-probabilities; this will guard against the potential underflow of the product of N probabilities for large N .

Step 4

We now have come to the core of the Probability Sort algorithm. This core consists of a While-loop which runs until the desired number desiredNumber of probability sorted damage state vectors has

been obtained or until the Proposals list is empty, signifying that the total state space has been explored.

count = 1;

While[(Length[Proposals] > 0) OR (count < desiredNumber)

Repeat ***Steps 5*** and ***6;*** (21)

count++]

Step 5.a
In each iteration of this While-loop the current Proposals list is updated by taking the list entry which has the greatest path probability *P* ; that is, take that list

{ *P*, stateVector, switchVector, activeSwitch }. (22)

in Proposals where *P* is maximal.

Step 5.b
We then make available the entities in the list (18) for the algorithmic steps that will follow, by setting

work*P* = *P* ;
workStateVector = stateVector ;
workSwitchVector = switchVector ; (23)
workActiveSwitch = activeSwitch ;

Step 5.c
After which we remove the list entry (18) from the Proposals list.

Step 5.d
We then set

$$P^{(\text{count}+1)} = P \tag{24}$$

and the unsorted stateVector, see (8),

$$\mathbf{x}^{(\text{count}+1)} = \text{undoPermutate[stateVector]} \tag{25}$$

in a list $\left\{P^{(\text{count}+1)}, \mathbf{x}^{(\text{count}+1)}\right\}$ and insert that list entry at the back of the list probabilitySort, so we obtain the updated list:

$$\text{probabilitySort} = \left\{ \left\{ P^{(1)},\, \mathbf{x}^{(1)} \right\}, \left\{ P^{(2)},\, \mathbf{x}^{(2)} \right\}, \ldots, \left\{ P^{(\text{count}+1)},\, \mathbf{x}^{(\text{count}+1)} \right\} \right\} . \quad (26)$$

Step 5.e

Finally, we update the total probability coverage variable:

$$\text{sumProb} = \text{sumProb} + P^{(\text{count}+1)} \; ; \quad (26)$$

Step 6

Now, the candidate with the greatest path probability P , that is, (18), is allowed to generate offspring before it gets moved to the probSort list. Each candidate can get a maximum of three 'children'. As these children take the place of their progenitor in the Proposals list, they guarantee that

a) all the remaining leaves of the event tree of the damage state space may still be explored, and

b) that the next best probability is always in the updated Proposals list,

Offspring may be produced as follows:

Step 6.a

Flip active switch one layer deeper, to a more improbable state, if permissible given maximum layer depth, and set that switch as the active switch and update the corresponding probability P and stateVector; that is,

% first determine the number of possible damage states
% for the infrastructural element under consideration:

q = workActiveSwitch ;
M = length(doublePermutatedProbMatrix(q, :) ;

% elements in the switchVector take on values
% from 0 (base-line damage state with the highest probability)
% to M – 1 (damage state the lowest probability)
% So we have below that $0 \le r \le M - 1$.

r = workSwitchVector(q) ;
If[r < M – 1,

% Set
P = workP ;
stateVector = workStateVector ;

switchVector = workSwitchVector
activeSwitch = workActiveSwitch ;

%Then update
P = [*P/* doublePermutatedProbMatrix(q, r)] × doublePermutatedProbMatrix(q, r + 1);
stateVector(q) = doublePermutatedStateMatrix(q, r + 1) ;
switchVector(q) = r + 1;

%Store the list
offSpring1 = { *P*, stateVector, switchVector, workActiveSwitch } ;

% anywhere in the Proposals list,
Proposals = Insert [Proposals, offSpring1] ;

%and close the If-statement.
];

Step 6.b

If active switch is a first layer switch, then de-activate switch and position switch one step forward if permissible given (a) row length or (b) a zero spot being available at that position, and activate that switch for that forward position.

q = workActiveSwitch ;

If[(workSwitchVector(q) == 1) AND (q + 1 ≤ *N*) AND (workSwitchVector(q + 1) == 0),

% Set
P = workP ;
stateVector = workStateVector ;
switchVector = workSwitchVector
activeSwitch = workActiveSwitch ;

%Then update
P = [*P/* doublePermutatedProbMatrix(q, 2)] × doublePermutatedProbMatrix(q, 1);
P = [*P* / doublePermutatedProbMatrix(q + 1, 1)] × doublePermutatedProbMatrix(q + 1, 2);

stateVector(q) = doublePermutatedStateMatrix(q, 1) ;
stateVector(q + 1) = doublePermutatedStateMatrix(q + 1, 2) ;

switchVector(q) = 0;
switchVector(q + 1) = 1;

```
        activeSwitch = q + 1;

%Store the list
        offSpring2  =  { P, stateVector, switchVector, activeSwitch } ;

% anywhere in the Proposals list,
        Proposals = Insert [Proposals, offSpring2] ;

%and close the If-statement.
];
```

Step 6.c

If the first entry in the switchVector is an available zero spot, then set that zero spot to a first level active switch, and update the corresponding probability *P* and stateVector; that is,

```
If[ switchVector(1) == 0,

% Set
        P  = workP ;
        stateVector  = workStateVector ;
switchVector = workSwitchVector
        activeSwitch = workActiveSwitch ;

%Then update
        P = [ P/ doublePermutatedProbMatrix(1,1) ] × doublePermutatedProbMatrix(1,2);
        stateVector(1) = doublePermutatedStateMatrix(1, 2) ;
        switchVector(1) = 1;
        activeSwitch = 1;

%Store the list
        offSpring3  =  { P, stateVector, switchVector, activeSwitch } ;

% anywhere in the Proposals list,
        Proposals = Insert [Proposals, offSpring3] ;

%and close the If-statement.
];
```

Step 7:

Print the probability coverage value sumProb and STOP.
The list probabilitySort is now available to the user.

N.B.: We may set some probability criterion, say, crit, which the probabilities of the state vectors in the probabilitySort list must exceed. This criterion then has to be enforced by way of an If-statement in ***Steps* 6.*a***, **6.*b***, and **6.*c***, as the Proposals list gets replenished by new offspring.

Moreover, this probability criterion crit may also prohibit the elements of the switch vector from the $(N - m)$th element onwards to leave their optimal base-line states of 0, as the probabilities of switch state 1 from the $(N - m)$th element onwards puts the probability of the probability P of the adjusted base-line state vector

$P = 1;$
For[$i = 1, i \leq N,$
$P = P \times$ permutatedProbMatrix(i, 1);
i++]

below the admissible threshold; that is,

$$Pm < \ldots < P2 < P1 < \text{crit}$$

where

$P1 = [\ P/\ \text{doublePermutatedProbMatrix}(N - m + 1,1)\ ] \times \text{doublePermutatedProbMatrix}(N - m + 1, 2);$
$P2 = [\ P/\ \text{doublePermutatedProbMatrix}(N - m + 2,1)\ ] \times \text{doublePermutatedProbMatrix}(N - m + 1, 2);$

…

$Pm = [\ P/\ \text{doublePermutatedProbMatrix}(N,1)\ ] \times \text{doublePermutatedProbMatrix}(N, 2);$

So, under a probability criterion crit we may set the length of the switch vector in (16) to be of the reduced length $(N - m)$, rather than the full length N.

A.3 MATLAB-Code

```
% INPUTS

% total number of desired scenario proposals
desiredNumber = 10^6;

% probability cut-off
crit = log(10^-6);

% begin explosion(s)
input = [13];

% over-pressure value
P = 200;
```

```
% grid definition
nX = 5;
nY = 5;

% INSERT PROBABILITY MAP
totalProbMap4 = probabilityMapUCL

% DO DOUBLE SORT ON PROBABILITY MAP



% total number of objects
N = nX * nY;

% sort within the rows
damageState = zeros(N, 4);
probability = zeros(N, 4);
for i = 1 : N
    [sortProb, index] = sort(totalProbMap4(i,:),'descend');
    probability(i,:) = sortProb;
    damageState(i,:) = index;
end

% sort over the rows using the third column
[sortedProb, index] = sortrows(probability,-3);
sortedState = damageState(index,:);

% indexRevert reverts second sort
% probability - sortedProb(indexRevert, :)
[~, indexRevert] = sort(index);

% SET-UP PROPOSAL AND STORAGE LISTS

% logarithm matrix of double sorted probabilities
logSortedProb = log(sortedProb);

% INITIALIZE PROPOSAL LIST
% 1. probability
firstProb = logSortedProb(:,1);
% 2. ordered state vector
firstState = sortedState(:,1);
orderedState = firstState(indexRevert);

% proposal probabilities; pre-allocate memory for better performance
proposalProb = zeros(desiredNumber, 1);
proposalProb(1) = exp(sum(firstProb));
% proposal state vectors; pre-allocate memory for better performance
proposalState = zeros(desiredNumber, N);
proposalState(1,:) = orderedState;



% INITIALIZE STORAGE LIST
```

```
% 1. updated probability
logProb = sum(firstProb) - logSortedProb(1,1) + logSortedProb(1,2);
% 2. switch vector
switchVector = zeros(N,1);
switchVector(1) = 1;
% 3. active switch
activeSwitch = 1;
% 4. updated state vector
stateVector = firstState;
stateVector(1) = sortedState(1,2);

% storage; pre-allocate memory for better performance
% probabilities
storageProb = zeros(desiredNumber, 1);
storageProb(1) = exp(logProb);
% switch vector
storageSwitchVector = zeros(desiredNumber, N);
storageSwitchVector(1,:) = switchVector;
% active switch
storageActiveSwitch = zeros(desiredNumber, 1);
storageActiveSwitch(1) = activeSwitch;
% state vector
storageStateVector = zeros(desiredNumber, N);
storageStateVector(1,:) = stateVector;


% CORE OF ALGORITHM

sumProb = exp(sum(firstProb));
count = 2;

oldPointer = [];
pointer = 2;

flag = 1;

while flag == 1

    % Choose optimal proposal from storage list
    [prob, index] = max(storageProb);
    switchVectorWork = storageSwitchVector(index,:);
    activeSwitchWork = storageActiveSwitch(index);
    stateVectorWork = storageStateVector(index,:);

    % update proposals list
    proposalProb(count) = prob;
    orderedState = stateVectorWork(indexRevert);
    proposalState(count,:) = orderedState;

    % update probability coverage measure
    sumProb = sumProb + prob;
```

```
if mod(count,1000) == 0
    count
    sumProb
end


% break out of while-loop at the next iteration if necessary
if count == desiredNumber
    flag = 0;
end


% update counter
count = count + 1;

% 'delete' optimal proposal from storage list
storageProb(index) = 0;

% set-up overwrite in storage list of the optimal proposal
oldPointer = [oldPointer; index];

activeSwitch = activeSwitchWork;
switchVector = switchVectorWork;

q = activeSwitch;
m = switchVector(q);

% switches go from 0 to M-1
if m < 3

    % additional working material
    stateVector = stateVectorWork;

    % update prob
    logProb = log(prob);
    logProb = logProb - logSortedProb(q,m+1) + logSortedProb(q,m+2);

    % update switch vector
    switchVector(q) = switchVector(q) + 1;

    % update state vector
    stateVector(q) = sortedState(q,m+2);

    if ~isempty(oldPointer)
        index2 = oldPointer(1);
        drop = ones(length(oldPointer),1);
        drop(1) = 0;
        drop = logical(drop);
        oldPointer = oldPointer(drop);
```

```
        else
            index2 = pointer;
            pointer = pointer + 1;
        end

        % fill storage matrices

        % probability
        storageProb(index2) = exp(logProb);
        % switch vector
        storageSwitchVector(index2,:) = switchVector;
        % active switch
        storageActiveSwitch(index2) = activeSwitch;
        % state vector
        storageStateVector(index2,:) = stateVector;

    end

    % restore switchVector
    switchVector = switchVectorWork;

    if (m == 1) && (q < N) && (switchVector(q + 1) == 0)

        % additional working material
        stateVector = stateVectorWork;

        % update prob
        logProb = log(prob);
        logProb = logProb - logSortedProb(q,2) + logSortedProb(q,1);
        logProb = logProb - logSortedProb(q+1,1) + logSortedProb(q+1,2);

        % update switch vector
        switchVector(q) = 0;
        switchVector(q+1) = 1;

        % update active switch
        activeSwitch = q + 1;

        % update state vector
        stateVector(q) = sortedState(q,1);
        stateVector(q+1) = sortedState(q+1,2);

        if ~isempty(oldPointer)
            index2 = oldPointer(1);
            drop = ones(length(oldPointer),1);
            drop(1) = 0;
            drop = logical(drop);
            oldPointer = oldPointer(drop);
        else
            index2 = pointer;
            pointer = pointer + 1;
```

```
        end

        % fill storage matrices

        % probability
        storageProb(index2) = exp(logProb);
        % switch vector
        storageSwitchVector(index2,:) = switchVector;
        % active switch
        storageActiveSwitch(index2) = activeSwitch;
        % state vector
        storageStateVector(index2,:) = stateVector;

    end

    % restore switchVector
    switchVector = switchVectorWork;

    if switchVector(1) == 0

        % additional working material
        stateVector = stateVectorWork;

        % update prob
        logProb = log(prob);
        logProb = logProb - logSortedProb(1,1) + logSortedProb(1,2);

        % update switch vector
        switchVector(1) = 1;

        % update active switch
        activeSwitch = 1;

        % update state vector
        stateVector(1) = sortedState(1,2);


         if ~isempty(oldPointer)
            index2 = oldPointer(1);
            drop = ones(length(oldPointer),1);
            drop(1) = 0;
            drop = logical(drop);
            oldPointer = oldPointer(drop);
        else
            index2 = pointer;
            pointer = pointer + 1;
        end


        % fill storage matrices
```

```
        % probability
        storageProb(index2) = exp(logProb);
        % switch vector
        storageSwitchVector(index2,:) = switchVector;
        % active switch
        storageActiveSwitch(index2) = activeSwitch;
        % state vector
        storageStateVector(index2,:) = stateVector;

    end

end
```

A.4 A Pen and Paper Algorithmic Run

We now give a pen and paper algorithmic run of the proposed Probability Sort algorithm for state vectors having probabilities greater or equal to crit = 10^{-6}, for the most simple non-trivial case where we have $N=3$ elements each having $M=3$ possible damage states. This in order to give the reader/programmer a concrete sense of the here proposed algorithm.

Let

$$\text{probMatrix} = \begin{bmatrix} \frac{90}{100} & \frac{9}{100} & \frac{1}{100} \\ \frac{9900}{10000} & \frac{99}{10000} & \frac{1}{10000} \\ \frac{999000}{1000000} & \frac{999}{1000000} & \frac{1}{1000000} \end{bmatrix} \quad (1)$$

and let the state matrix be such that its column is equivalent to the switch vector and its subsequent switching layers:

$$\text{stateMatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{bmatrix} \quad (2)$$

Then ML proposal $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{(1)}$ has a probability of (1) and (2)

$$P^{(1)} = \frac{90}{100} \times \frac{9900}{10000} \times \frac{999000}{1000000} = \frac{890109 \times 10^6}{10^{12}}$$

So, we set the ProbabilitySort list as

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, \begin{bmatrix}0 & 0 & 0\end{bmatrix}^{(1)}\right\}\right\}$$

The next best state vector in terms of being the next most probable state vector then is given as:

$$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^{(1)} \rightarrow \begin{bmatrix}\underline{\mathbf{1}} & 0 & 0\end{bmatrix}^{(2)}, \quad \frac{890109\times10^{5}}{10^{12}},$$

as (1) and (2)

$$P^{(2)} = \frac{9}{100}\times\frac{9900}{10000}\times\frac{999000}{1000000} = \frac{890109\times10^{5}}{10^{12}},$$

And where the bold face underlined switch state points to the active switch position. So, we set the Proposals list as

$$\text{Proposals} = \left\{\left\{\frac{890109\times10^{5}}{10^{12}}, \begin{bmatrix}\underline{\mathbf{1}} & 0 & 0\end{bmatrix}^{(2)}\right\}\right\}$$

In the first iteration of the While-loop we then insert the most probable of the proposals in the probability sort list

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, \begin{bmatrix}0 & 0 & 0\end{bmatrix}^{(1)}\right\}, \left\{\frac{890109\times10^{5}}{10^{12}}, \begin{bmatrix}1 & 0 & 0\end{bmatrix}^{(2)}\right\}\right\}$$

After we clean up the Proposals list as

$$\text{Proposals} = \{\ \}$$

We then go through the three offspring checkpoints:

$$\begin{bmatrix}\underline{\mathbf{1}} & 0 & 0\end{bmatrix}^{(2)} \rightarrow \begin{array}{ll}\begin{bmatrix}\underline{\mathbf{2}} & 0 & 0\end{bmatrix}^{(3)}, & \frac{98901\times10^{5}}{10^{12}} \\ \begin{bmatrix}0 & \underline{\mathbf{1}} & 0\end{bmatrix}^{(4)}, & \frac{890109\times10^{4}}{10^{12}} \\ \begin{bmatrix}(1,\underline{1}) & 0 & 0\end{bmatrix}^{(reject)}, & \text{n.a.}\end{array}$$

So as we store the proposals $\begin{bmatrix}\underline{\mathbf{2}} & 0 & 0\end{bmatrix}^{(3)}$ and $\begin{bmatrix}0 & \underline{\mathbf{1}} & 0\end{bmatrix}^{(4)}$ into the Proposals list with the corresponding probabilities

$$\text{Proposals} = \left\{ \left\{ \frac{98901 \times 10^5}{10^{12}},\ [\underline{\mathbf{2}}\quad 0\quad 0]^{(3)} \right\},\ \left\{ \frac{890109 \times 10^4}{10^{12}},\ [0\quad \underline{\mathbf{1}}\quad 0]^{(4)} \right\} \right\}$$

In the next While-iteration we then have the most probable state vector $[\underline{\mathbf{2}}\quad 0\quad 0]^{(3)}$ and add it together with its probability to the probability sort list, as we remove that entry from the proposals list:

$$\text{ProbabilitySort} = \begin{array}{l} \left\{ \left\{ \frac{890109 \times 10^6}{10^{12}}, [0\quad 0\quad 0]^{(1)} \right\}, \right. \\ \left\{ \frac{890109 \times 10^5}{10^{12}}, [1\quad 0\quad 0]^{(2)} \right\}, \\ \left. \left\{ \frac{98901 \times 10^5}{10^{12}},\ [2\quad 0\quad 0]^{(3)} \right\} \right\} \end{array}$$

$$\text{Proposals} = \left\{ \left\{ \frac{890109 \times 10^4}{10^{12}},\ [0\quad \underline{\mathbf{1}}\quad 0]^{(4)} \right\} \right\}$$

We have in this While-iteration

$$[\underline{\mathbf{2}}\quad 0\quad 0]^{(3)} \rightarrow \begin{array}{l} [\underline{\mathbf{3}}\quad 0\quad 0]^{(reject)} \\ [0\quad \underline{\mathbf{1}}\quad 0]^{reject} \\ [(2,\underline{1})\quad 0\quad 0]^{(reject)} \end{array}$$

So, we have at the end of the While-iteration proposals list is not replenished.

$$\text{Proposals} = \left\{ \left\{ \frac{890109 \times 10^4}{10^{12}},\ [0\quad \underline{\mathbf{1}}\quad 0]^{(4)} \right\} \right\}$$

In the next While-iteration we then have the most probable state vector $[0\quad \underline{\mathbf{1}}\quad 0]^{(4)}$ and add it together with its probability to the probability sort list, as we remove that entry from the proposals list:

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, [0\quad 0\quad 0]^{(1)}\right\}, \left\{\frac{890109\times10^{5}}{10^{12}}, [1\quad 0\quad 0]^{(2)}\right\}, \left\{\frac{98901\times10^{5}}{10^{12}}, [2\quad 0\quad 0]^{(3)}\right\}, \left\{\frac{890109\times10^{4}}{10^{12}}, [0\quad 1\quad 0]^{(4)}\right\}\right\}$$

$$\text{Proposals} = \{\quad\}$$

We have in this While-iteration the parent $[0\quad \underline{\mathbf{1}}\quad 0]^{(4)}$, which begets the offspring

$$[0\quad \underline{\mathbf{1}}\quad 0]^{(4)} \rightarrow \begin{matrix}[0\quad \underline{\mathbf{2}}\quad 0]^{(5)}, & \frac{8991\times10^{4}}{10^{12}} \\ [0\quad 0\quad \underline{\mathbf{1}}]^{(6)}, & \frac{890109\times10^{3}}{10^{12}} \\ [\underline{\mathbf{1}}\quad 1\quad 0]^{(7)}, & \frac{890109\times10^{3}}{10^{12}}\end{matrix}$$

So, we have at the end of the While-iteration the replenished proposals list

$$\text{Proposals} = \left\{\left\{\frac{8991\times10^{4}}{10^{12}}, [0\quad \underline{\mathbf{2}}\quad 0]^{(5)}\right\}, \left\{\frac{890109\times10^{3}}{10^{12}}, [0\quad 0\quad \underline{\mathbf{1}}]^{(6)}\right\}, \left\{\frac{890109\times10^{3}}{10^{12}}, [\underline{\mathbf{1}}\quad 1\quad 0]^{(7)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector a choice between $[0\quad 0\quad \underline{\mathbf{1}}]^{(6)}$ and $[\underline{\mathbf{1}}\quad 1\quad 0]^{(7)}$. We may choose either of these proposals and add it together with its probability to the probability sort list, as we remove that entry from the proposals list, say

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, [0\quad 0\quad 0]^{(1)}\right\}, \left\{\frac{890109\times10^{5}}{10^{12}}, [1\quad 0\quad 0]^{(2)}\right\}, \left\{\frac{98901\times10^{5}}{10^{12}}, [2\quad 0\quad 0]^{(3)}\right\}, \left\{\frac{890109\times10^{4}}{10^{12}}, [0\quad 1\quad 0]^{(4)}\right\}, \left\{\frac{890109\times10^{3}}{10^{12}}, [0\quad 0\quad 1]^{(6)}\right\}\right\}$$

$$\text{Proposals} = \left\{\left\{\frac{8991\times10^{4}}{10^{12}}, [0\quad \underline{\mathbf{2}}\quad 0]^{(5)}\right\}, \left\{\frac{890109\times10^{3}}{10^{12}}, [\underline{\mathbf{1}}\quad 1\quad 0]^{(7)}\right\}\right\}$$

We have in this While-iteration the parent $[0\quad 0\quad \underline{\mathbf{1}}]^{(6)}$, which begets the offspring

$$[0\quad 0\quad \underline{\mathbf{1}}]^{(6)} \rightarrow \begin{matrix} [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}, & \frac{891\times10^{3}}{10^{12}} \\ [0\quad 0\quad 0\quad \underline{\mathbf{1}}]^{(reject)}, & \text{n.a.} \\ [\underline{\mathbf{1}}\quad 0\quad 1]^{(9)}, & \frac{890109\times10^{2}}{10^{12}} \end{matrix}$$

So, we have at the end of the While-iteration the replenished proposals list

$$\text{Proposals} = \left\{\left\{\frac{8991\times10^{4}}{10^{12}}, [0\quad \underline{\mathbf{2}}\quad 0]^{(5)}\right\}, \left\{\frac{890109\times10^{3}}{10^{12}}, [\underline{\mathbf{1}}\quad 1\quad 0]^{(7)}\right\}, \left\{\frac{891\times10^{3}}{10^{12}}, [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}\right\}, \left\{\frac{890109\times10^{2}}{10^{12}}, [\underline{\mathbf{1}}\quad 0\quad 1]^{(9)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[\underline{\mathbf{1}} \quad 1 \quad 0]^{(7)}$. We add it together with its probability to the probability sort list, as we remove that entry from the proposals list:

$$\text{ProbabilitySort} = \begin{Bmatrix} \left\{\frac{890109\times10^{6}}{10^{12}}, [0 \quad 0 \quad 0]^{(1)}\right\}, \\ \left\{\frac{890109\times10^{5}}{10^{12}}, [1 \quad 0 \quad 0]^{(2)}\right\}, \\ \left\{\frac{98901\times10^{5}}{10^{12}}, [2 \quad 0 \quad 0]^{(3)}\right\}, \\ \left\{\frac{890109\times10^{4}}{10^{12}}, [0 \quad 1 \quad 0]^{(4)}\right\}, \\ \left\{\frac{890109\times10^{3}}{10^{12}}, [0 \quad 0 \quad 1]^{(6)}\right\}, \\ \left\{\frac{890109\times10^{3}}{10^{12}}, [1 \quad 1 \quad 0]^{(7)}\right\} \end{Bmatrix}$$

$$\text{Proposals} = \begin{Bmatrix} \left\{\frac{8991\times10^{4}}{10^{12}}, [0 \quad \underline{\mathbf{2}} \quad 0]^{(5)}\right\}, \\ \left\{\frac{891\times10^{3}}{10^{12}}, [0 \quad 0 \quad \underline{\mathbf{2}}]^{(8)}\right\}, \\ \left\{\frac{890109\times10^{2}}{10^{12}}, [\underline{\mathbf{1}} \quad 0 \quad 1]^{(9)}\right\} \end{Bmatrix}$$

We have in the next While-iteration the parent $[\underline{\mathbf{1}} \quad 1 \quad 0]^{(7)}$, which begets the offspring

$$[\underline{\mathbf{1}} \quad 1 \quad 0]^{(7)} \rightarrow \begin{matrix} [\underline{\mathbf{2}} \quad 1 \quad 0]^{(10)}, & \frac{98901\times10^{3}}{10^{12}} \\ [0 \quad (1,\underline{1}) \quad 0]^{(reject)}, & \text{n.a.} \\ [(1,\underline{1}) \quad 1 \quad 0]^{reject}, & \text{n.a.} \end{matrix}$$

So, we have at the end of the While-iteration the replenished proposals list

$$\text{Proposals} = \left\{\left\{\frac{8991\times10^4}{10^{12}},\ [0\quad \underline{\mathbf{2}}\quad 0]^{(5)}\right\},\ \left\{\frac{891\times10^3}{10^{12}},\ [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}\right\},\ \left\{\frac{890109\times10^2}{10^{12}},\ [\underline{\mathbf{1}}\quad 0\quad 1]^{(9)}\right\},\ \left\{\frac{98901\times10^3}{10^{12}},\ [\underline{\mathbf{2}}\quad 1\quad 0]^{(10)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[\underline{\mathbf{2}}\quad 1\quad 0]^{(10)}$. We add it together with its probability to the probability sort list,

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^6}{10^{12}}, [0\quad 0\quad 0]^{(1)}\right\},\ \left\{\frac{890109\times10^5}{10^{12}}, [1\quad 0\quad 0]^{(2)}\right\},\ \left\{\frac{98901\times10^5}{10^{12}},\ [2\quad 0\quad 0]^{(3)}\right\},\ \left\{\frac{890109\times10^4}{10^{12}}, [0\quad 1\quad 0]^{(4)}\right\},\ \left\{\frac{890109\times10^3}{10^{12}},\ [0\quad 0\quad 1]^{(6)}\right\},\ \left\{\frac{890109\times10^3}{10^{12}},\ [1\quad 1\quad 0]^{(7)}\right\},\ \left\{\frac{98901\times10^3}{10^{12}},\ [2\quad 1\quad 0]^{(10)}\right\}\right\}$$

as we remove that entry from the proposals list. We have in this While-iteration the parent $[\underline{\mathbf{2}}\quad 1\quad 0]^{(10)}$, which begets no offspring

$$[\underline{\mathbf{2}}\quad 1\quad 0]^{(10)} \rightarrow \begin{matrix}[\underline{\mathbf{3}}\quad 1\quad 0]^{(reject)}\\ [0\quad (1,\underline{1})\quad 0]^{(reject)}\\ [(2,\underline{1})\quad 1\quad 0]^{reject}\end{matrix}$$

So, we have at the end of the While-iteration the updated proposals list

$$\text{Proposals} = \left\{\left\{\frac{8991\times10^{4}}{10^{12}},\ [0\quad \underline{\mathbf{2}}\quad 0]^{(5)}\right\},\ \left\{\frac{891\times10^{3}}{10^{12}},\ [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}\right\},\ \left\{\frac{890109\times10^{2}}{10^{12}},\ [\underline{\mathbf{1}}\quad 0\quad 1]^{(9)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[0\quad \underline{\mathbf{2}}\quad 0]^{(5)}$. We add it together with its probability to the probability sort list,

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, [0\quad 0\quad 0]^{(1)}\right\},\ \left\{\frac{890109\times10^{5}}{10^{12}}, [1\quad 0\quad 0]^{(2)}\right\},\ \left\{\frac{98901\times10^{5}}{10^{12}},\ [2\quad 0\quad 0]^{(3)}\right\},\ \left\{\frac{890109\times10^{4}}{10^{12}}, [0\quad 1\quad 0]^{(4)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [0\quad 0\quad 1]^{(6)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [1\quad 1\quad 0]^{(7)}\right\},\ \left\{\frac{98901\times10^{3}}{10^{12}},\ [2\quad 1\quad 0]^{(10)}\right\},\ \left\{\frac{8991\times10^{4}}{10^{12}},\ [0\quad 2\quad 0]^{(5)}\right\}\right\}$$

as we remove that entry from the proposals list. We have in this While-iteration the parent $[0\quad \underline{\mathbf{2}}\quad 0]^{(5)}$, which begets the offspring

$$[0\quad \underline{\mathbf{2}}\quad 0]^{(5)} \rightarrow \begin{matrix} [0\quad \underline{\mathbf{3}}\quad 0]^{(reject)}, & \text{n.a.} \\ [0\quad 0\quad \underline{\mathbf{1}}]^{(reject)}, & \text{n.a.} \\ [\underline{\mathbf{1}}\quad 2\quad 0]^{(11)}, & \frac{8991\times10^{3}}{10^{12}} \end{matrix}$$

So, we have at the end of the While-iteration the updated and replenished proposals list

$$\text{Proposals} = \left\{\left\{\frac{891\times10^{3}}{10^{12}},\ [0\ \ 0\ \ \underline{\mathbf{2}}]^{(8)}\right\},\ \left\{\frac{890109\times10^{2}}{10^{12}},\ [\underline{\mathbf{1}}\ \ 0\ \ 1]^{(9)}\right\},\ \left\{\frac{8991\times10^{3}}{10^{12}},\ [\underline{\mathbf{1}}\ \ 2\ \ 0]^{(11)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[\underline{\mathbf{1}}\ \ 0\ \ 1]^{(9)}$. We add it together with its probability to the probability sort list, as we remove that entry from the proposals list:

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, [0\ \ 0\ \ 0]^{(1)}\right\},\ \left\{\frac{890109\times10^{5}}{10^{12}}, [1\ \ 0\ \ 0]^{(2)}\right\},\ \left\{\frac{98901\times10^{5}}{10^{12}},\ [2\ \ 0\ \ 0]^{(3)}\right\},\ \left\{\frac{890109\times10^{4}}{10^{12}}, [0\ \ 1\ \ 0]^{(4)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [0\ \ 0\ \ 1]^{(6)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [1\ \ 1\ \ 0]^{(7)}\right\},\ \left\{\frac{98901\times10^{3}}{10^{12}},\ [2\ \ 1\ \ 0]^{(10)}\right\},\ \left\{\frac{8991\times10^{4}}{10^{12}},\ [0\ \ 2\ \ 0]^{(5)}\right\},\ \left\{\frac{890109\times10^{2}}{10^{12}}, [1\ \ 0\ \ 1]^{(9)}\right\}\right\}$$

We have in this While-iteration the parent $[\underline{\mathbf{1}}\ \ 0\ \ 1]^{(9)}$, which begets the offspring

$$[\underline{\mathbf{1}}\ \ 0\ \ 1]^{(9)} \rightarrow \begin{matrix} [\underline{\mathbf{2}}\ \ 0\ \ 1]^{(12)}, & \frac{98901\times10^{2}}{10^{12}} \\ [0\ \ \underline{\mathbf{1}}\ \ 1]^{(13)}, & \frac{890109\times10}{10^{12}} \\ [(1,\underline{1})\ \ 0\ \ 1]^{(reject)}, & \text{n.a.} \end{matrix}$$

So, we have at the end of the While-iteration the updated and replenished proposals list

$$\text{Proposals} = \left\{\left\{\frac{891\times10^{3}}{10^{12}},\ [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}\right\},\ \left\{\frac{8991\times10^{3}}{10^{12}},\ [\underline{\mathbf{1}}\quad 2\quad 0]^{(11)}\right\},\ \left\{\frac{98901\times10^{2}}{10^{12}},\ [\underline{\mathbf{2}}\quad 0\quad 1]^{(12)}\right\},\ \left\{\frac{890109\times10^{1}}{10^{12}},\ [0\quad \underline{\mathbf{1}}\quad 1]^{(13)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[\underline{\mathbf{2}}\quad 0\quad 1]^{(12)}$. We add it together with its probability to the probability sort list,

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}}, [0\quad 0\quad 0]^{(1)}\right\},\ \left\{\frac{890109\times10^{5}}{10^{12}}, [1\quad 0\quad 0]^{(2)}\right\},\ \left\{\frac{98901\times10^{5}}{10^{12}},\ [2\quad 0\quad 0]^{(3)}\right\},\ \left\{\frac{890109\times10^{4}}{10^{12}}, [0\quad 1\quad 0]^{(4)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [0\quad 0\quad 1]^{(6)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [1\quad 1\quad 0]^{(7)}\right\},\ \left\{\frac{98901\times10^{3}}{10^{12}},\ [2\quad 1\quad 0]^{(10)}\right\},\ \left\{\frac{8991\times10^{4}}{10^{12}},\ [0\quad 2\quad 0]^{(5)}\right\},\ \left\{\frac{890109\times10^{2}}{10^{12}}, [1\quad 0\quad 1]^{(9)}\right\},\ \left\{\frac{98901\times10^{2}}{10^{12}},\ [2\quad 0\quad 1]^{(12)}\right\}\right\}$$

as we remove that entry from the proposals list. We have in this While-iteration the parent $[\underline{\mathbf{2}} \quad 0 \quad 1]^{(12)}$, which begets no offspring

$$[\underline{\mathbf{2}} \quad 0 \quad 1]^{(12)} \rightarrow \begin{matrix} [\underline{\mathbf{3}} \quad 0 \quad 1]^{(reject)} \\ [0 \quad \underline{\mathbf{1}} \quad 1]^{(reject)} \\ [(2,\underline{1}) \quad 0 \quad 1]^{(reject)} \end{matrix}$$

So the updated proposals list at the end of the While-iterations is

$$\text{Proposals} = \begin{matrix} \left\{\frac{891\times10^{3}}{10^{12}}, \quad [0 \quad 0 \quad \underline{\mathbf{2}}]^{(8)}\right\}, \\ \left\{\frac{8991\times10^{3}}{10^{12}}, \quad [\underline{\mathbf{1}} \quad 2 \quad 0]^{(11)}\right\}, \\ \left\{\frac{890109\times10^{1}}{10^{12}}, \quad [0 \quad \underline{\mathbf{1}} \quad 1]^{(13)}\right\} \end{matrix}$$

In the next While-iteration we then have as the most probable state vector $[\underline{\mathbf{1}} \quad 2 \quad 0]^{(11)}$. We add it together with its probability to the probability sort list,

$$
\begin{aligned}
\text{ProbabilitySort} = \Bigg\{ & \left\{\frac{890109\times10^{6}}{10^{12}}, [0 \quad 0 \quad 0]^{(1)}\right\}, \\
& \left\{\frac{890109\times10^{5}}{10^{12}}, [1 \quad 0 \quad 0]^{(2)}\right\}, \\
& \left\{\frac{98901\times10^{5}}{10^{12}}, [2 \quad 0 \quad 0]^{(3)}\right\}, \\
& \left\{\frac{890109\times10^{4}}{10^{12}}, [0 \quad 1 \quad 0]^{(4)}\right\}, \\
& \left\{\frac{890109\times10^{3}}{10^{12}}, [0 \quad 0 \quad 1]^{(6)}\right\}, \\
& \left\{\frac{890109\times10^{3}}{10^{12}}, [1 \quad 1 \quad 0]^{(7)}\right\}, \\
& \left\{\frac{98901\times10^{3}}{10^{12}}, [2 \quad 1 \quad 0]^{(10)}\right\}, \\
& \left\{\frac{8991\times10^{4}}{10^{12}}, [0 \quad 2 \quad 0]^{(5)}\right\}, \\
& \left\{\frac{890109\times10^{2}}{10^{12}}, [1 \quad 0 \quad 1]^{(9)}\right\}, \\
& \left\{\frac{98901\times10^{2}}{10^{12}}, [2 \quad 0 \quad 1]^{(12)}\right\}, \\
& \left\{\frac{8991\times10^{3}}{10^{12}}, [1 \quad 2 \quad 0]^{(11)}\right\}\Bigg\}
\end{aligned}
$$

as we remove that entry from the proposals list. We have in this While-iteration the parent $[\underline{\mathbf{1}} \quad 2 \quad 0]^{(11)}$, which begets the offspring

$$
\begin{aligned}
& & [\underline{\mathbf{2}} \quad 2 \quad 0]^{(14)}, & \quad \frac{999\times10^{3}}{10^{12}} \\
[\underline{\mathbf{1}} \quad 2 \quad 0]^{(11)} \rightarrow & [0 \quad (2,\underline{1}) \quad 0]^{(reject)}, & & \text{n.a.} \\
& [(1,\underline{1}) \quad 2 \quad 0]^{(reject)}, & & \text{n.a.}
\end{aligned}
$$

So the updated proposals list at the end of the While-iterations is

$$\text{Proposals} = \left\{\left\{\frac{891\times10^{3}}{10^{12}},\ [0\quad 0\quad \underline{\mathbf{2}}]^{(8)}\right\},\ \left\{\frac{890109\times10^{1}}{10^{12}},\ [0\quad \underline{\mathbf{1}}\quad 1]^{(13)}\right\},\ \left\{\frac{999\times10^{3}}{10^{12}},\ [\underline{\mathbf{2}}\quad 2\quad 0]^{(14)}\right\}\right\}$$

In the next While-iteration we then have as the most probable state vector $[0\quad \underline{\mathbf{1}}\quad 1]^{(12)}$. We add it together with its probability to the probability sort list,

$$\text{ProbabilitySort} = \left\{\left\{\frac{890109\times10^{6}}{10^{12}},[0\quad 0\quad 0]^{(1)}\right\},\ \left\{\frac{890109\times10^{5}}{10^{12}},[1\quad 0\quad 0]^{(2)}\right\},\ \left\{\frac{98901\times10^{5}}{10^{12}},\ [2\quad 0\quad 0]^{(3)}\right\},\ \left\{\frac{890109\times10^{4}}{10^{12}},[0\quad 1\quad 0]^{(4)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [0\quad 0\quad 1]^{(6)}\right\},\ \left\{\frac{890109\times10^{3}}{10^{12}},\ [1\quad 1\quad 0]^{(7)}\right\},\ \left\{\frac{98901\times10^{3}}{10^{12}},\ [2\quad 1\quad 0]^{(10)}\right\},\ \left\{\frac{8991\times10^{4}}{10^{12}},\ [0\quad 2\quad 0]^{(5)}\right\},\ \left\{\frac{890109\times10^{2}}{10^{12}},\ [1\quad 0\quad 1]^{(9)}\right\},\ \left\{\frac{98901\times10^{2}}{10^{12}},\ [2\quad 0\quad 1]^{(12)}\right\},\ \left\{\frac{8991\times10^{3}}{10^{12}},\ [1\quad 2\quad 0]^{(11)}\right\},\ \left\{\frac{890109\times10^{1}}{10^{12}},\ [0\quad 1\quad 1]^{(13)}\right\}\right\}$$

as we remove that entry from the proposals list. We have in the next While-iteration the parent $[0 \quad \underline{\mathbf{1}} \quad 1]^{(13)}$, which begets the offspring

$$[0 \quad \underline{\mathbf{1}} \quad 1]^{(13)} \rightarrow \begin{matrix} [0 \quad \underline{\mathbf{2}} \quad 1]^{(15)}, & \frac{8991\times10}{10^{12}} \\ [0 \quad 0 \quad (1,\underline{1})]^{(reject)}, & \text{n.a.} \\ [\underline{\mathbf{1}} \quad 1 \quad 1]^{(16)}, & \frac{890109}{10^{12}} \end{matrix}$$

So, we have at the end of the While-iteration the updated and replenished proposals list

$$\text{Proposals} = \begin{matrix} \left\{\left\{\frac{891\times10^{3}}{10^{12}}, \quad [0 \quad 0 \quad \underline{\mathbf{2}}]^{(8)}\right\}\right., \\ \left\{\frac{999\times10^{3}}{10^{12}}, \quad [\underline{\mathbf{2}} \quad 2 \quad 0]^{(14)}\right\}, \\ \left\{\frac{8991\times10}{10^{12}}, \quad [0 \quad \underline{\mathbf{2}} \quad 1]^{(15)}\right\}, \\ \left.\left\{\frac{890109}{10^{12}}, \quad [\underline{\mathbf{1}} \quad 1 \quad 1]^{(16)}\right\}\right\} \end{matrix}$$

All the remaining proposals have probabilities smaller than 10^{-6}, which is why we (arbitrarily) terminate this pend-and-paper run.

Note that we build in the (arbitrary) If-statement for crit = 10^{-6} into the algorithmic ***Step 6.a***, ***6.b***, and ***6.c***, then the above Proposals list would have been empty and the algorithm would have terminated automatically at this point.

We give below the rest of the event-tree coverage without the corresponding probabilities.

$$[0 \quad 0 \quad \underline{\mathbf{2}}]^{(8)} \rightarrow \begin{matrix} [0 \quad 0 \quad \underline{\mathbf{3}}]^{(reject)} \\ [0 \quad 0 \quad 0 \quad \underline{\mathbf{1}}]^{(reject)} \\ [\underline{\mathbf{1}} \quad 0 \quad 2]^{(17)} \end{matrix}$$

$$[\underline{\mathbf{2}} \quad 2 \quad 0]^{(14)} \rightarrow \begin{matrix} [\underline{\mathbf{3}} \quad 2 \quad 0]^{(reject)} \\ [0 \quad (2,\underline{1}) \quad 0]^{(reject)} \\ [(2,\underline{1}) \quad 2 \quad 0]^{(reject)} \end{matrix}$$

$$[0 \quad \underline{\mathbf{2}} \quad 1]^{(15)} \rightarrow \begin{matrix} [0 \quad \underline{\mathbf{3}} \quad 1]^{(reject)} \\ [0 \quad 0 \quad (1,\underline{1})]^{(reject)} \\ [\underline{\mathbf{1}} \quad 2 \quad 1]^{(18)} \end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{2}} & 1 & 1]^{(19)} \\ [\underline{\mathbf{1}} & 1 & 1]^{(16)} \rightarrow [0 & (1,\underline{1}) & 1]^{(reject)} \\ & [(1,\underline{1}) & 1 & 1]^{(reject)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{2}} & 0 & 2]^{(20)} \\ [\underline{\mathbf{1}} & 0 & 2]^{(17)} \rightarrow [0 & \underline{\mathbf{1}} & 2]^{(21)} \\ & [(1,\underline{1}) & 0 & 2]^{(reject)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{2}} & 2 & 1]^{(22)} \\ [\underline{\mathbf{1}} & 2 & 1]^{(18)} \rightarrow [0 & (2,\underline{1}) & 1]^{(reject)} \\ & [(1,\underline{1}) & 2 & 1]^{(reject)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{3}} & 1 & 1]^{(reject)} \\ [\underline{\mathbf{2}} & 1 & 1]^{(19)} \rightarrow [0 & (1,\underline{1}) & 1]^{(reject)} \\ & [(2,\underline{1}) & 1 & 1]^{(reject)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{3}} & 0 & 2]^{(reject)} \\ [\underline{\mathbf{2}} & 0 & 2]^{(20)} \rightarrow [0 & \underline{\mathbf{1}} & 2]^{(reject)} \\ & [(2,\underline{1}) & 0 & 2]^{(reject)}\end{matrix}$$

$$\begin{matrix}[0 & \underline{\mathbf{2}} & 2]^{(23)} \\ [0 & \underline{\mathbf{1}} & 2]^{(21)} \rightarrow [0 & 0 & (2,\underline{1})]^{(reject)} \\ & [\underline{\mathbf{1}} & 1 & 2]^{(24)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{3}} & 2 & 1]^{(reject)} \\ [\underline{\mathbf{2}} & 2 & 1]^{(22)} \rightarrow [0 & (2,\underline{1}) & 1]^{(reject)} \\ & [(2,\underline{1}) & 2 & 1]^{(reject)}\end{matrix}$$

$$\begin{matrix}[0 & \underline{\mathbf{3}} & 2]^{(reject)} \\ [0 & \underline{\mathbf{2}} & 2]^{(23)} \rightarrow [0 & 0 & (2,\underline{1})]^{(reject)} \\ & [\underline{\mathbf{1}} & 2 & 2]^{(25)}\end{matrix}$$

$$\begin{matrix}[\underline{\mathbf{2}} & 1 & 2]^{(26)} \\ [\underline{\mathbf{1}} & 1 & 2]^{(24)} \rightarrow [0 & (1,\underline{1}) & 2]^{(reject)} \\ & [(1,\underline{1}) & 1 & 2]^{(reject)}\end{matrix}$$

$$\begin{matrix}\left[\underline{\mathbf{1}} & 2 & 2\right]^{(25)}\end{matrix} \rightarrow \begin{matrix}\left[\underline{\mathbf{2}} & 2 & 2\right]^{(27)} \\ \left[0 & (2,\underline{1}) & 2\right]^{(reject)} \\ \left[(1,\underline{1}) & 2 & 2\right]^{(reject)}\end{matrix}$$

$$\begin{matrix}\left[\underline{\mathbf{2}} & 1 & 2\right]^{(26)}\end{matrix} \rightarrow \begin{matrix}\left[\underline{\mathbf{3}} & 1 & 2\right]^{(reject)} \\ \left[0 & (1,\underline{1}) & 2\right]^{(reject)} \\ \left[(2,\underline{1}) & 1 & 2\right]^{(reject)}\end{matrix}$$

$$\begin{matrix}\left[\underline{\mathbf{2}} & 2 & 2\right]^{(27)}\end{matrix} \rightarrow \begin{matrix}\left[\underline{\mathbf{3}} & 2 & 2\right]^{(reject)} \\ \left[0 & (2,\underline{1}) & 2\right]^{(reject)} \\ \left[(2,\underline{1}) & 2 & 2\right]^{(reject)}\end{matrix}$$

And we see that the algorithm covers all of the $M^N = 3^3 = 27$ leaves in our event tree.

APPENDIX B: DELPHI PANELS

A stress testing Delphi panel should lead to the formulation of a set of stress scenarios for road/rail networks. Comprehensive identification of relevant scenarios is critical, because scenarios excluded in this task will not be included in further analysis and may result in an incorrect estimation of risk. To minimise the possibility of this happening it is important that experts in each area are involved, as argued by Adey et al. (2016). In this appendix, some background about Delphi panels, its composition, its size, its working methods, etc. is given.

The Delphi method is a method for knowledge elicitation which relies on a panel of independent experts (Rowe G., Wright G., 2001). The basic principle of Delphi is that forecasts from a structured group of experts are more accurate than those from unstructured groups or individuals. The properties of the different group communication techniques are presented in Table B.1 (The Delphi Method: Techniques and Applications, 2002). Depending on the chosen technique and number of items addressed the survey can take in average from 1 week to 6 months and may cost from 1.000 to 1.5 million euro.

There are few key characteristics of Delphi: regular feedback of individual contributions of information and knowledge; assessment of the group judgment or' view; opportunity for individuals to revise views; and anonymity of the participants (The Delphi Method: Techniques and Applications, 2002). There are different ways how Delphi can be applied in the framework of these main characteristics, ranging from qualitative to quantitative, to mixed.

The advantages of using Delphi:

- Allows freely expressions of opinions and ideas by a large number of participants.
- Discussions and results are not influenced by one leader.
- It is convenient as it allows participants who are geographically distanced, to work from own home or office.
- Participants have an opportunity to think deeper and gather more information on issue between the rounds.
- Allows long-term thinking (gives view on future) and in the same time orientated towards actions.
- It shows well performance when the issue is complex.
- Highlight a consensus decision (if it is reached or not).
- Provides a transparent and democratic technique.

Disadvantages:

- It takes time for organisers and can be expensive to run the survey.
- May be difficult to motivate participants.
- Some participants may drop out during the process (especially after the first round).
- Results may be influenced by the set of participants involved.
- Will not build relationships or generate a dialogue between participants.
- There is a danger of ignoring a single opinion that might have of special value.

Rowe and Wright (2001) suggested to use the following principal to receive a better result:

- Knowledgeable experts
- Heterogeneous experts (who focus in the field of research area)
- Groups of 5 to 20 experts
- Mean or median estimates of the panel and rationales of these estimate for feedback.
- Rounds should continue until stable results arrive (usually 3 rounds are enough).

Gibbs and others (2001) suggested to invite as experts the participants who fulfill such criteria as: Participants have published articles in the last five years on the topic of research; Participants have taught courses about these topics; or Participants' primary employment responsibilities are related to these areas. According to the some authors (Delbecq et al., 1975; Murphy et al., 1998) this gives a wide perspective and range of alternatives that will lead to better performance. The number of experts which can give the best accuracy is, however, uncertain. Some of the publications suggested that there is no clear difference in accuracy (Brockhoff, 1975; Boje, Murnighan, 1982; Powell, 2003). Participants may have different arguments of their opinions even if the opinions are similar. It can give additional information and therefore should be included together with the feedback from estimates outside the quartiles. Rowe and Wright (2001) pointed out that the first round is more valuable as it is unstructured. It gives the experts the possibility to specify key issues and formulate relevant and balanced sets of questions. However, a lot of studies use structured questionnaires in the first stage.

Group Communication Techniques					
	Conference Telephone Call	Committee Meeting	Formal Conference or Seminar	Conventional Delphi	Real-Time Delphi
Effective Group Size	Small	Small to Medium	Small to Large	Small to Large	Small to Large
Occurrence of Interaction by Individual	Coincident with group	Coincident with group	Coincident with group	Random	Random
Length of Interaction	Short	Medium to Long	Long	Short to Medium	Short
Number of Interactions	Multiple, as required by group	Multiple, necessary time delays between	Single	Multiple, necessary time delays between	Multiple, as required by individual
Normal Mode Range	Equality to chairman control (flexible)	Equality to chairman control (flexible)	Presentation (directed)	Equality to monitor control (structured)	Equality to monitor control or group control and no monitor (structured)
	Conference Telephone Call	Committee Meeting	Formal Conference or Seminar	Conventional Delphi	Real-Time Delphi
Principal Costs	Communications	• Travel • Individual's Time	• Travel • Individual's Time • Fees	• Monitor Time • Clerical • Secretarial	• Communications • Computer Usage
	Time-urgent considerations	Forced delays		Forced delays	Time-urgent considerations
Other Characteristics	• Equal flow of information to and from all • Can maximize psychological effects		Efficient flow of information from few to many	• Equal flow of information to and from all • Can minimize psychological effects • Can minimize time demanded of respondents or conferees	

Table B.1: The Delphi Method: Techniques and Applications