OPENCOSSAN 2.0: an efficient computational toolbox for risk, reliability and resilience analysis

Edoardo Patelli¹, Silvia Tolo¹, Hindolo George-Williams¹, Jonathan Sadeghi¹, Roberto Rocchetta¹, Marco de Angelis¹ and Matteo Broggi²

¹ Institute for Risk and Uncertainty, Chadwick Building, University of Liverpool, Peach Street, Liverpool L69 7ZF, United Kingdom
e-mail: edoardo.patelli@liverpool.ac.uk
² Institut für Risiko und Zuverlässigkeit, Leibniz Universität Hannover, Germany, EU
e-mail: broggi@bauinf.uni-hannover.de

Keywords. Software, Uncertainty Quantification, Simulation, Resilience, Reliability, Networks

Abstract. Many complex phenomena and the analysis of large and complex system and network can only be studied adopting advanced computational methods. In addition, in many engineering fields virtual prototypes are used to support and drive the design of new components, structures and systems. Uncertainty quantification is a key requirement and challenge for a realistic and reliable numerical modelling and prediction that spans across various disciplines and industry as well.

The treatment of uncertainty required the availability of efficient algorithms and computational techniques able to reduce the computational cost required by the non-deterministic analysis and to interface with opensource and commercial model (e.g. FE/CFD) and libraries. In order to satisfy these requirements and allowing the inclusion of non-deterministic analyses as a practice standard routing in scientific computing, a general purpose software for uncertainty quantification and risk assessment, named COSSAN, is under continuous development.

This paper presents an overview of the main capabilities of the recent release of the Matlab open source toolboxes OPENCOSSAN. The new release includes interfaces with 3rd party libraries allowing to couple OPENCOSSAN with the state-of-the-art tools in Machine Learning and Meta-modelling. In addition, new toolboxes for reliability and resilient analysis of system and network are also presented. OPENCOSSAN is released under the Lesser GNU licence. It is therefore freely available. It is also be package as a Python or Java library for distribution to end users who do not need MATLAB.

1 INTRODUCTION

In the last decades, simulation technologies have advanced rapidly and many smart and computational cost saving strategies proposed. These technologies allows nowadays the virtual prototyping of many complex and interconnected components and systems. Different disciplines are already taken advantage of the digital revolution that allows engineers to design products faster by reducing the number of expensive and destructive tests necessary to qualify new products. This allows to speed-up the development cycle of new products and coping with an increasingly competitive market.

However, the success of rapid prototyping through computational models and numerical simulations can only be achieved with the availability of high-fidelity and reliable simulations. Reliable simulations can only be obtained taking into account all the uncertainties that can significantly affect the performance of the components and systems. For this reason, reliable model prediction can only be obtained when different sources of uncertainty are explicitly included in the analysis [4]. Hence, uncertainty quantification is a key requirement and challenge across various disciplines. Stochastic analysis allows not only to obtain more reliable simulation but can also be used to assess e.g. the robustness and resilience of the system, scheduling efficient maintenance activities, and taking informed decisions.

These tasks are challenging and requires the propagation of aleatory and epistemic uncertainty through each system to quantify the quantities of interest. Although the continuous growth of computational power available allows simulating the effect of the uncertainties with increased precision, traditional approaches such as plain Monte Carlo simulation approaches might not be applicable due to their computational requirements in particular when a detailed deterministic model is used. In addition, novel concepts of imprecise probability (also known as generalised probabilistic approaches), including interval analysis, random sets, probability boxes, fuzzy methods etc. [20, 3, 1] can be adopted to deal with model uncertainties, errors in modelling and measurements, limited and scarce data. The downside of such approaches is the associated with the computational cost required by the analysis. In fact, soon or later these approaches rely on some kind of global optimisation techniques to identify the bounds associated with the propagation of the uncertainty (see Figure 1).
Despite the advantages offered by the stochastic analysis, industrial design methods are still predominantly deterministic. One of the reasons is the lack of proper software for uncertainty quantification and stochastic analysis. This paper presents the recent advancements on the OpenCossaN software, an open-source general purpose software for risk, reliability and resilient analysis. The toolbox is designed to satisfy the requirements of industry, academics, and researchers as well for teaching purpose on non-deterministic analysis. The software incorporates the knowledge, understanding and intellectual property from more than 30 years of research in the field of computational stochastic analysis.

2 COSSAN SOFTWARE

2.1 Introduction

The development of the COSSAN software was initiated by Prof. G.I Schueller from the Institute for Engineering Mechanics, University of Innsbruck, Austria [32] with the aim to make stochastic analysis as standard procedure in engineering practice and bridge the gap between academia and industry. The original purpose of the software was to perform stochastic structural analysis as indicated by its name (COmputation Stochastic Structural ANalysis). However, since 2006 the code has been completely rewritten and developed by the Cossan working group as a general purpose software which provides generic functionalities that can be changed and expanded through user-defined routines and additional codes. The COSSAN software is now intended for a wider range of applications, which includes optimization analysis, life-cycle management, reliability and resilience analysis, sensitivity, optimization, and design under uncertainty [24].

2.2 OpenCossaN

OpenCossaN represents the computational core of the COSSAN software. Programmed in Matlab environment since 2006, it has been released under the Lesser GNU licence [8] in 2012 (see Figure 2). OpenCossaN can be used for free, redistributed, and modified under the terms of the GNU General Public License. The source code is available at the web address https://www.cossan.co.uk. The Apache Subversion system is used for software versioning and revision control system distributed to keep track of changes and contribution from different developers.

The OpenCossaN project aims to promote learning and understanding of non-deterministic analysis through the availability of a free and open computational toolbox in Matlab environment [21, 19]. Programmed in object oriented fashion, it provides an intuitive, flexible, expandable and powerful toolbox for dealing with uncertainty. The combination of various algorithms with specific solution sequences permits the analysis of different engineering and scientific problems.

OpenCossaN software is developed collaboratively with contribution from different researchers and practitioners and supported by different universities and research institutes. The Institute for Risk and Uncertainty at University of Liverpool is hosting the development of the software since 2011 [19, 21]. In 2015 the Institut für Risiko und Zuverlässigkeit from the Leibniz Universität Hannover (DE) and in 2017 the Shanghai Institute of Disaster Prevention and Relief, Tongji University, China also join the development of the software.

Figure 1: Computational costs of the analysis with imprecise probability.

Figure 2: Development of OpenCossaN, release chart and main academic partners.
OPENCOSSAN, is coded exploiting the object-oriented Matlab programming environment, where it is possible to define specialized solution sequences, which include reliability methods, sensitivity analysis, optimization strategies, surrogate models and parallel computing strategies.

The computational framework is organized in packages. A package is a namespace for organizing classes and interfaces in a logical manner, which makes large software project OPENCOSSAN easier to manage. A class describes a set of objects with common characteristics such as data structures and methods. Objects, that are instances of classes can be aggregated forming more complex objects and proving solutions for practical problems in a compact, organized and manageable format. OPENCOSSAN provides intuitive, clear, well documented and human readable interfaces to the classes. No acronyms are used to define methods and properties. The structure of OPENCOSSAN allows for extensive modularity and efficient code re-utilization. Objects (instances of a class) can be aggregated forming more complex objects with methods providing solutions for practical problem in a compact, organized and manageable format. Hence, different objects and methods can be combined by the users to solve specific problems, including uncertainty quantification, sensitivity analysis, reliability analysis and robust design. Such problems can be solved adopting traditional probabilistic approaches as well as by generalised probabilistic methods. Thanks to the modular nature of OPENCOSSAN, it is possible to define specialized solution sequences including any reliability method, optimization strategy and surrogate model or parallel computing strategy to reduce the overall cost of the computation without loss of accuracy.

The availability of an extended documentation, tutorials and examples is of a key importance for the usability of a software. The COSSAN documentation is written collaboratively by COSSAN developers and end-users using MediaWiki tool where with rare exceptions, articles can be edited by anyone.

2.3 OpenCossan 2.0

After 6 years from the initial release of OPENCOSSAN, the version 2.0 of the software has been released. This version contains not only new capabilities and toolboxes as summarised in Table 1 but significant changes in the code structure. Those changes were necessary to simplify the user interaction, and the maintainability of the software. Significant efforts were dedicated to the development of automatic unit test and good test coverage. This ensures that developers are immediately notified after any code change if the modification introduces any unforeseen defects. This allows to improve the code quality and the overall reliability of the software [5].

One of the key feature of this software version is its improved ability to connect with 3rd party libraries (see Section 3.3). In addition, OPENCOSSAN is also available as compiled Java or Python library. This allows royalty-free integration of OPENCOSSAN with other programming languages. Users can take advantages of OPENCOSSAN algorithms and capability without the need of MATLAB.

3 SELECTED NEW FEATURES

This section presents some selected new features and toolboxes available in OPENCOSSAN 2.0. Table 1 summarised the main components added to the software.

<table>
<thead>
<tr>
<th>New toolboxes</th>
<th>Key features</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imprecise probability</td>
<td>Modelling uncertainty and imprecision using Interval, Probability Boxes, Dempster-Shafer structures, Fuzzy Variables. Propagate uncertainty using double loop approach, $\alpha$-cuts single Monte Carlo analysis.</td>
<td>[20, 3, 6, 1, 2, 27]</td>
</tr>
<tr>
<td>Credal Network</td>
<td>Traditional Bayesian Network, Enhanced Bayesian Network and Credal Network. Allows to consider discrete, probabilistic, bounded and hybrid nodes. Reduction of Credal Networks and Enhanced Bayesian Network to traditional Bayesian Network. Inference computation on the reduced network.</td>
<td>[35, 33, 34]</td>
</tr>
<tr>
<td>Meta modelling</td>
<td>Interval predictive model, Polynomial Chaos Expansion (via UQ-Lab), Robust Neural Network, interface with FANN, Tensorflow and Neural Network toolbox from Mathworks.</td>
<td>[22, 29, 30, 18]</td>
</tr>
<tr>
<td>System and Network</td>
<td>Power flow simulation,</td>
<td>[14, 13, 12, 11, 7, 23]</td>
</tr>
<tr>
<td>Power Grid</td>
<td>Reliability and Resilient analysis of power grids. Cascade analysis, Interface with MATPOWER and</td>
<td>[26, 28]</td>
</tr>
</tbody>
</table>
3.1 Credal Networks toolbox

Bayesian Belief Networks, more commonly known as Bayesian Networks, are a probabilistic graphical model based on the use of directed acyclic graphs, integrating graph theory with the robustness of Bayesian statistics. The graphical framework of such models consists of nodes, representing the variables of the problem of interest, connected to each other by edges, generally arrows, that depict the dependency link existing between two nodes. Bayesian Belief Networks assumes local Markov propriety and therefore each variable is conditionally independent of its non-descendants given its parent variables.

Each node of the network is linked to a Conditional Probability Distribution that, according to the local Markov propriety, defines the strength of the probabilistic dependency existing between each individual node and its parents. When a node has no parents, i.e. it is a root of the network, a marginal probability distribution is associated with it.

Credal Networks are a generalization of Bayesian Belief Networks. In Bayesian Belief Networks the conditional probabilities are either crisp values or continuous probabilistic distributions. In Credal Networks imprecise probabilities can be used to represent the relationships among variables enhancing significantly the robustness and accuracy of the approach. This because it allows to capture the uncertainty in input and to propagate it within the model avoiding the introduction of enhanced significantly the robustness and accuracy ofthe approach since allows to capture imprecision due to limited data and avoiding the introduction of biases and unjustified assumptions.

Exact inference algorithms adopting an analytical approach exists for Bayesian Belief Networks but they are almost exclusively limited to the use of crisp probability values. This often leads to the adoption of discretization procedure which impoverish the quality of the information available. On the other hand, approximate inference algorithms allow the use of continuous probabilistic variables but at the cost of lower accuracy, and they can result often inefficient or have unknown rates of convergence. The limitations of the approximate inference approach result even more significant when considering the field of risk analysis and decision making, or more generally those applications where rare events and near-real-time computation play a crucial role [33]. Adopting structural reliability methods, Credal Networks and Bayesian Belief Networks with continuous nodes can be reduced to traditional Bayesian Belief Networks containing only discrete nodes. The elimination of the continuous nodes can be obtained by means of numerical integration methods commonly used in the field of structural reliability.

3.2 Interaction with UQ-Lab

Stochastic analysis requires the repeated evaluation of a (detailed) numerical model. The analysis time can be reduced significantly by using meta-models, which approximate the quantities of interest at low computational costs. The meta-model toolbox allows to replace computational original models (such as FE or CFD models) with mathematical functions (meta-models) with negligible computational cost which mimic the behaviour of the original model. Such metamodels are in general constructed (trained) using a number of deterministic solutions each corresponding to a sample point in random space. A number of different meta-models are available in OPENCOSAN including response surface, Kriging or Gaussian process, Artificial neural networks etc. Metamodel techniques are fundamental to reduce the cost of the stochastic analysis.

OPENCOSAN 2.0 allows to use the state of the art technique of polynomial chaos expansion available in UQ-Lab [16, 31]. This toolbox requires the proprietary software from UQ-Lab, which is currently freely available for academic use at www.uqlab.com/. With this development OPENCOSAN users can now integrate UQ-Lab metamodels into their existing workflow, whilst maintaining the ease of use and syntax which they are used to.

In this section, the mechanism by which UQ-Lab was integrated into OPENCOSAN is briefly described. A metamodel class is created in OPENCOSAN to interface with the polynomial chaos expansion (PCE) libraries of UQ-Lab. A constructor method is used to specify properties of the PCE, for example the maximum polynomial degree of the PCE, and the input and output names. To calibrate the metamodel OPENCOSAN uses provided training data or creates samples from a simulator with a sampling scheme defined by the user. The calibrate method calls UQ-Lab and converts all of

![Figure 3: Simplified workflow of OPENCOSAN connection with PCE libraries of UQ-Lab.](image-url)
Proceedings of the joint ICVRAM ISUMA UNCERTAINTIES conference
Florianópolis, SC, Brazil, April 8-11, 2018

Figure 4: Example of OpenCossan code used to create PCE based on UQ-Lab libraries.

The information stored in the OpenCossan metamodel class into a UQ-Lab input. UQ-lab finds the coefficients of the PCE using least squares. Then the metamodel is created and stored in the OpenCossan class called PolynomialChaosUQLab. The original OpenCossan simulator can then be replaced with the metamodel, which permits the analytic computation of the Sobol’ Sensitivity Indices. Figure 3 shows the COSSAN workflow required to use PCE from UQ-Lab. Figure 4

3.3 Artificial Neural Networks

One of the most used meta-modelling technique is based on artificial neural networks. Artificial Neural networks are actively developed in most research centres and as consequence a different libraries are now available.

OpenCossan originally supported only on Fast Artificial Neural Network library [17]. In the new version of OpenCossan an abstract class called ArtificialNeuralNetwork has been introduced. This class provides a common interface to OpenCossan for different libraries. Dedicated subclasses have been created providing the link to the most popular Artificial Neural Network libraries. As shown in Figure 5, the Neural Network Toolbox from Mathworks and the library NNSYSID and connected using a Matlab interface. FANN library is connected via a MEX interface while TensorFlow is connected using a Python interface.

The advantage of this approach is to provide a common interface to use different libraries. It also provides the possibility for the user to use its own favourite library.

Figure 5: Meta-modelling based in OpenCossan and link to 3rd-party libraries providing access to the state-of-the-art of Artificial Neural Network and Machine Learning.
### 3.4 Power Grids and interconnected critical infrastructures

The COSSAN software was originally developed to perform uncertainty quantification analysis on computational models for structural engineering and linked to a variety of software for finite element analysis. Recently, others linkage capabilities have been integrated into the main software body. In particular, OPEnCOSSAN can now interact with tools dedicated to the analysis of power grids.

Two Matlab-based computational tools for Power Grid have been connected to OPEnCOSSAN. The first is the well known open source MATPOWER software [36] (also available in Python programming language as PYPOWER). MATPOWER is widely used tool among researchers, this is due to the many available power grid cases and its flexibility. It can be used to solve the power flow and optimal power flow problems using different models (AC, DC, AC-Optimal Power Flow, etc.) and additional constraints can be easily integrated to create tailored user-defined optimizations. The second tool is a realistic cascade analysis simulator developed by the Energy group at ETH Zurich [15]. This tool can be used to perform cascading failures evaluations. A variety of models can be selected (DC, AC, linearized-AC, SCOPF) and for sake of adherence to reality, many post contingency correction schemes are considered (e.g. frequency control, island detection, generators ramp up, spinning reserve). The tool is very efficient and well represent realistic power grid operations.

These tools are extremely flexible and open source. However, these tools have a very basic approach for dealing with uncertainty. For instance, in MATPOWER the reliability and security assessment methods are not directly available as well as the uncertainty quantification of dynamic cascade analysis. Hence, OPEnCOSSAN can now be employed to overcome those limitations.

In addition to the new linkage capabilities to those libraries, new methods for power grids analysis and for assessment of critical infrastructures have been developed and integrated. The methods dedicated to power grid analysis primarily focus on vulnerability reliability and resilience assessments [26]. The developed toolbox allows the integration of renewable generators, the coupled analysis between power grids and heating district networks [25] as well the coupled analysis of weather conditions grids performance [28].

Both power grid structural and operational data can be stored into dedicated objects and loaded from existing database (e.g. case study from the MATPOWER toolbox). Users can freely modify system structure and create their own case study. All the developed methods for power grid analysis can use classical and generalized uncertainty quantification tools available in OPEnCOSSAN. Concerning the vulnerability assessment methods, different approaches have been developed. The methods focus on critical contingencies identification and ranking (see [26] for further details on the framework). The following functionalities can be identified: 1) flow-based methods focusing on operational weaknesses 2) pure topological methods 3) hybrid topological methods. The methods in 2 and 3 use graph theoretical approaches and the difference is that methods 2 use only the topological data of the grid. Different metrics for vulnerability evaluation can be defined and ranking easily compared. Generalised reliability assessment capabilities have been also included which can be used when the available data is scarce or of poor quality [27]. These focus on the calculation of reliability indices such as e.g. the energy not supplied, demand not supplied, average interruptions frequency indices.

### 3.5 Complex and interconnected systems

#### 3.5.1 General remarks

Failure is inherent in every mechanical and electronic component, consequent of either its operating environment, human interaction, dependence on other components, or its natural ageing process. This, inevitably renders everyday systems susceptible to failures, some of which could be catastrophic, imposing severe economic and human losses on communities. The need, therefore, for the robust evaluation of the reliability of these systems, cannot be overemphasised. When we set out to evaluate the reliability of a realistic system, there are four key determinants of the ease and accuracy with which we can do this. The first challenge relates to the topology and size of the system. Traditional approaches like Fault trees and Markov Chains, require the combination of component failures that result in system failure. A key step, therefore, in their use for system reliability analysis, is the manual identification of these combinations, prior to system analysis. This is not too critical a problem for simple series-parallel topologies but a serious one for complex topologies and large series-parallel systems. Consequently, the analyst requires a lot of effort and expertise to model such systems, which naturally renders the process error-prone. The second challenge stems from the multi-state attributes of the components of the system. Traditional techniques are best suited to binary-state systems, that is, systems that are either fully functional or completely failed. However, there are instances when one or more intermediate states exist between the two extremes. Such systems are termed multi-state systems. For these, traditional techniques like fault-trees and Markov chains get tedious and sometimes, completely intractable. The third challenge is attributable to complex interdependencies, restrictive operational requirements/strategies, and non-exponentially distributed transition times of components. Finally, the lack of data or sufficient knowledge about the failure, repair, and operational attributes of the system and its components pose a significant challenge to the reliable computing of its reliability and performance indices.

The system reliability analysis package provides an effective means of modelling the reliability, availability, and performance of multi-state systems of any topology. This powerful tool is a Monte Carlo methods and complex network theory hybrid, as such, it inherits their desirable attributes. With Monte Carlo methods, realistic system attributes like
Proceedings of the joint ICVRAM ISUMA UNCERTAINTIES conference
Florianópolis, SC, Brazil, April 8-11, 2018

Figure 6: Package architecture of the network and system toolbox.

complex interdependencies, complex maintenance strategies, non-exponential transitions, and maintenance optimisation in the presence of multiple external dynamics can be modelled. Network theory, on the other hand, allows the intuitive representation of complex system topologies, from which an intelligent algorithm computes system performance. Unlike most similar tools, the package is applicable to multi-state, multi-output systems with competing loads. It also employs a bottom-up approach, where the analyst is only required to model the individual components of the system, and efficient algorithms return the expected outcome. The entire package is coded according to an object-oriented programming paradigm and allows most of the input to be defined as matrices or arrays. These attributes make it intuitive and, therefore, usable even by non-experts on the system being analysed. The package is also equipped to accommodate imprecision in the failure and repair characteristics of system components. It, therefore, addresses all the major problems normally encountered in the reliability evaluation of realistic engineering systems. In summary, the package exhibits the following merits:

- It allows the topology of the system is defined by a matrix of 1’s and 0’s called an adjacency matrix. With this, the analyst does not have to know, in advance, the combinations of component failures leading to system failure.
- Given the adjacency matrix and the stochastic processes of the components, the rest of the analysis is done by inbuilt algorithms requiring no assistance from the user. It’s intuitive!
- It’s applicable to systems with flow losses. Consequently, it can be used to analyse power networks susceptible to line, generator, and transformer losses or a network of pipelines susceptible to a partial pipeline rupture. This single attribute distinguishes it from the other system reliability evaluation techniques.
- It is applicable to systems with limited maintenance teams and complex operational dynamics.

A detailed review of its advantages over existing techniques is available in [11]. The package has been applied to inter-dependent systems [12], nuclear power plant station blackout risk analysis [9, 10], and the maintenance optimisation of complex power systems [13].

3.5.2 Underlying Modelling Formalism

In the package, a component is defined as a semi-Markov stochastic process, to account for its multiple states, as well as the characteristics of the transitions between these states. The structure of the system is then defined as a graph/network in which the components are represented as nodes connected by arcs that can be unreliable or inefficient. A Monte Carlo Simulation algorithm is invoked to recreate the component transitions. Following each component transition, a load-flow algorithm is used to deduce the performance of the system from the ensuing performance levels of the components.

Appreciating the high computational costs of Monte Carlo Simulation, the package invokes the Survival Signature approach [7, 23] to moderate the computation time for binary-state systems. With Survival Signature, the system topology does not need to be analysed on every evaluation of the reliability of the system. This proves useful for sensitivity, uncertainty analysis, and maintenance optimisation. For multi-state systems, a mimicry of the Survival Signature is used. Here, the system performance corresponding to each possible components states combination is derived prior to system analysis. By this, flow is calculated only once for each components states combination [12].
Simulation is not always necessary for some systems. This is the case for systems with no forced or conditional transitions [13]. With this in mind, the package is provided with an option to analyse the reliability of the system using an analytical approach [11], resulting in immense gains in computation time and accuracy.

Finally, the package uses a probability-box approach, firstly to characterise the epistemic uncertainty in the transition time distributions of the components and then an efficient technique to derive the bounds on the performance and reliability of the system. This technique computes the exact reliability and performance bounds of coherent binary-state systems from just two model evaluations [14].

### 3.5.3 Package Architecture

The package consists of three sequentially linked modules, as shown in Figure 6. Module 1 is a user-accessible Matlab script that allows the analyst to define the properties of each of the components of the system, the system topology, the maintenance strategy, and the system object. Each component is defined as an object of the component class, which is made up of methods to perform specific operations on the data contained in the component object. The second module reads the information contained in the system object and selects the appropriate approach (simulation or analytical) to analyse the system and in the case of simulation, saves the simulation history as an object of a third class. It is important to note that the computational requirements of the same system vary with its operational requirements and the reliability indices required by the analyst. For instance, the system with a limited number of maintenance teams is more computationally intensive than the same system with no restriction on the number of parallel maintenance actions. It is, therefore, imperative that the appropriate simulator is used, if unnecessary computational costs should be avoided. Module 3 uses the output of module 2 to compute the required reliability and performance indices. For each desired index, the module returns both an on-screen display (interactive plots where applicable) and a savable data. The output of each module can be saved, so the analyst could always return to where they left off after an interruption.

The simulation workflow is summarised in Figure 7 where the main objects required by the analysis are shown.

### 3.5.4 The Component Class

This class allows a component or more generally, a node of the system to be defined. It contains properties that capture the attributes of the component/node relevant to the analysis. The most basic of these being, the mission time, the time step (for time-dependent systems), the capacity of the component in each state, its transition map, and the probability distribution associated with each transition. Other properties become necessary depending on the type of system being analysed. For instance, the dependency matrix is required if any of the component’s possible transitions can affect the performance of 1 or more components. Similarly, the replacement probability matrix is required if the likelihood of the use of spares is different for each failure state. There are also a few properties not required by the user at object creation, they are used only for caching during system analysis.

To define the probability distribution associated with each component transition, the package is equipped with three classes for this. The RandomVariable class is used for precise distributions, the ImpreciseRandomVariable class, for distributions with parameter imprecision, and the HazardVariable class, for the case when only the hazard function is known. In the latter, the user has the option to use the hazard function directly without first creating a HazardVariable object. In this case, the Component class defines the object during initialisation. In fact, if the hazard function is known for all the transitions, the class automatically computes the state probabilities of the component and mark the component as eligible for analytical reliability evaluation. Generally, a component is eligible for analytical reliability evaluation if none of its transitions is forced or conditional. Similarly, a system is eligible for analytical reliability evaluation if all its components are eligible for analytical reliability evaluation. Shown in Figure 8 is the sample input script for a 3-state component.
3.5.5 The System Class

This class requires four basic properties; the set of component objects making up the system (Xcomponent), the maintenance policy (where necessary), the set of common-cause groups (Xccg), and the network model defining the physical connection between the components. Like the Component class, this class contains properties not required by the user. The maintenance policy is defined as an object of the class MaintenancePolicy, and contains such details as the number of maintenance groups, their relative sizes, their assigned components, and the maintenance strategies they employ. The network model on the other hand, is defined as an object of the NetworkModel class. Since this class requires certain details like the maximum capacity and name of each component, the set of components is first passed through the function getMaxNodeCapacity, to extract the relevant information. This information is then used in conjunction with the adjacency matrix to create the NetworkModel object, as shown in Figure 9.

Common-Cause Groups are defined as objects of the CommonCauseGroup class. Each object contains information about the components in the group, the Common-Cause Failure (CCF) probabilities, the common failure mode on occurrence of the CCF event, and the state a component has to be in to be affected by the CCF event.

With the System object created, all is set for system analysis. If the system is eligible for analytical reliability evaluation, the evaluateReliability method computes the reliability of the system and save the results in specific properties of the original System object. If for any reason a second reliability analysis of the same system is required, the new updated System object can be used with the evaluateReliabilityWithCutSets method. This ensures that a full analysis is not performed on the system for the second time, thereby enhancing the computational efficiency. If, however, the system is ineligible for analytical reliability evaluation, the simulate method analyses the system using an efficient event-driven Monte Carlo Simulation and save the result as an object of the SystemData class. Special methods in this class can be invoked to extract the system reliability indices from the object.

4 CONCLUSION

The availability of an intuitive and easy to use general purpose software is a key for making the non-deterministic analysis a common practice in computational models and numerical simulations. In fact, uncertainty and imprecision are unavoidable and they must be accounted for in any analyses. Only implementing stochastic analysis, digital (or virtual) design will be credible and applicable to different sectors and fields. Digital design allows to design fast and better, reducing the design costs and provide cost-effective and feasible engineering solutions.

This paper presented some selected features introduced in OPENCOSSAN 2.0. OPENCOSSAN represents the engine of the COSSAN software. It is an opensource project freely available making the software more accessible and its development more sustainable. This thanks to the continuously integration of code provided by users and developers from different institutions. It is an excellent collaborative tools for researchers and academics encouraging the cross-discipline utilisation of stochastic analysis and increasing the knowledge transfer between academia and industry.

OPENCOSSAN is designed to be open, avoiding duplication of existing excellent libraries but instead providing simple tools and approaches to integrate them in a simple and coherent toolbox.
**ACKNOWLEDGEMENTS**

Part of this work has been supported by EPSRC (Grant number EP/M018415/1) and by the EPSRC-ESRC Centre for Doctoral Training on Quantification and Management of Risk & Uncertainty in Complex Systems & Environments Grant number (EP/L015927/1).

**REFERENCES**


**RESPONSIBILITY NOTICE**

The authors are the only responsible for the printed material included in this paper.