The Journal of Open Source Software

Fortuna.jl: Structural and System Reliability Analysis in Julia

Damir Akchurin [1¶]

1 Department of Civil and Systems Engineering, Johns Hopkins University, Baltimore, MD ¶ Corresponding author

DOI: 10.21105/joss.06967

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @baxmittens
- @rafaelorozco

Submitted: 26 June 2024
Published: 05 August 2024

Summary

In any field of engineering, addressing uncertainties is crucial to ensure the adequate reliability - the ability to function without failure – of structures and systems, such as residential and commercial buildings, bridges, nuclear power plants, energy distribution networks, transportation networks, and even electric circuits. In these systems, uncertainties are ubiquitous and present in their geometric and material properties, the demands acting on them, and the measurements used to assess their condition. Moreover, uncertainties can also be present in the probabilistic models used to quantify them. As a result of the irreducibility of some of these uncertainties, it is impossible to guarantee the absolute reliability and serviceability of a system at any given point in time with certainty. However, it is possible to define and quantify measures of reliability, such as the failure probability of a system under given demands, using probabilistic methods developed within the field of structural and system reliability.

`Fortuna.jl` is an open-source general-purpose package for structural and system reliability analysis purely written in the Julia programming language (Bezanson et al., 2017). It implements a wide suite of commonly used reliability analysis methods to solve reliability, inverse reliability, and sensitivity problems. `Fortuna.jl` leverages Julia's high-performance capabilities to efficiently solve even the most challenging reliability problems. At the same time, `Fortuna.jl` is designed to be user-friendly and flexible, making it suitable for both research and teaching settings. It is intended that `Fortuna.jl` can serve as a platform for the development and implementation of new rapidly emerging reliability analysis methods.

Statement of Need

Due to the largely numerical nature of almost all reliability analysis methods, which often require computation of gradient vectors and Hessian matrices, the use of computational resources is necessary for all but the most trivial problems. As a result, a large number of both open-source and commercial software packages have been developed in different programming languages over the past 30 years, such as `UQpy` (Python) (Olivier et al., 2020), `Pystra` (Python), `FERUM` (MATLAB) (Bourinet et al., 2009), and `CalREL` (FORTRAN) (Der Kiureghian et al., 2006).

The development of `Fortuna.jl` was mainly motivated by the absence of packages for structural and system reliability analysis written in Julia. Additionally, it aimed to achieve an improved balance between user experience, ease of implementing new reliability analysis methods, computational efficiency, and interoperability with external finite element (FE) modeling software not directly available through Julia.

A key distinguishing feature of `Fortuna.jl` is that it is capable of performing differentiation of the limit state function, which defines the failure criterion for a given reliability problem, required by most reliability analysis methods using automatic differentiation techniques provided

by `ForwardDiff.jl` package (Revels et al., 2016), significantly speeding up the analysis process. This contrasts with other software packages for structural and system reliability analysis, which typically rely on variants of finite difference approximations. As it was mentioned before, `Fortuna.jl` also allows to easily define limit state functions using external FE modeling software, such as `Abaqus`, `OpenSees` (McKenna et al., 2010), and `SAFIR` (Franssen & Gernay, 2017). `Fortuna.jl` can also easily work with surrogate models of limit state functions that are computationally expensive to evaluate, such as when a limit state function is defined in terms of a complex FE model, using the functionality provided by `Surrogates.jl` package (Bessi et al., 2024). Lastly, `Fortuna.jl`'s ability to work with random variables is based on a widely adopted `Distributions.jl` package (Besançon et al., 2021), enabling seamless integration with other software packages for probabilistic analysis written in Julia, such as `Copulas.jl` (Laverny & Jimenez, 2024), `RxInter.jl` (Bagaev et al., 2023), and `Turing.jl` (Ge et al., 2018), which provide all the functionality needed for the development and implementation of more modern reliability analysis methods based on Bayesian inference.

`Fortuna.jl` has already been successfully used to compare the levels of safety achieved by the Allowable Strength Design (ASD) and Load and Resistance Factor Design (LRFD) methods, two design frameworks used to design individual component in structures (Akchurin et al., 2024; Sabelli et al., 2024). `Fortuna.jl` is also being actively used to develop and implement a next-generation design framework that can account for and optimize system behavior in the design of steel structures, which cannot be achieved using the current methods such as the ASD and LRFD.

Example

Consider a simple reliability problem from Echard et al. (2013) with the limit state function given by

$$g(U_1, U_2) = \frac{1}{2}(U_1 - 2)^2 - \frac{3}{2}(U_2 - 5)^3 - 3, \tag{1}$$

where U_1 and U_2 are two independent standard normal random variables. The failure domain defined by this limit state function is shown in Figure 1. The reference geometric reliability index β and failure probability P_f obtained using the first-order reliability method (FORM) are 3.93 and 4.21×10^{-5}, respectively. These results can be easily recreated using `Fortuna.jl`:

```
# Preamble:
using Fortuna

# Define the random vector and its correlation matrix:
U = [randomvariable("Normal", "M", [0, 1]),
     randomvariable("Normal", "M", [0, 1])]
ρ = [1 0; 0 1]

# Define the limit state function:
g(u::Vector) = 0.5 * (u[1] - 2) ^ 2 - 1.5 * (u[2] - 5) ^ 3 - 3

# Define the reliability problem and solve it using the FORM:
Problem = ReliabilityProblem(U, ρ, g)
Solution = solve(Problem, FORM())
println("Geometric reliability index: ", Solution.β)
println("Failure probability: ", Solution.PoF)
# Geometric reliability index: 3.932419
# Failure probability: 4.204761E-5
```

As shown in the code above, the results obtained using `Fortuna.jl` are consistent with the reference values.

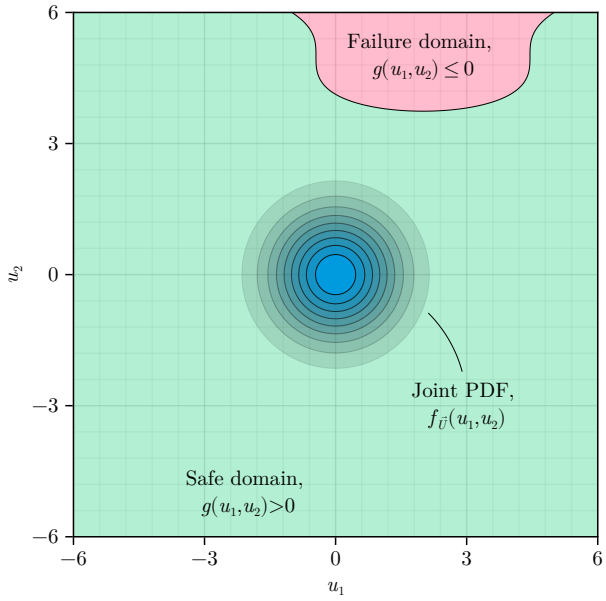

Figure 1: Failure domain defined by the limit state function in Equation 1[1].

Acknowledgements

The author would like to thank academic and industrial partners of the "*Reliability 2030: Design of Steel as a System*" initiative for the financial support. The author also like to thank the American Institute of Steel Construction for the financial support of the "*System Reliability for Structural Steel*" project.

References

Akchurin, D., Sabelli, R., Ziemian, R. D., & Schafer, B. W. (2024). ASD and LRFD: Reliability Comparison for Designs Subjected to Wind Loads. *Journal of Constructional Steel Research*, *213*, 108327. https://doi.org/10.1016/j.jcsr.2023.108327

Bagaev, D., Podusenko, A., & De Vries, B. (2023). RxInfer: A Julia Package for Reactive Real-Time Bayesian Inference. *Journal of Open Source Software*, *8*(84), 5161. https://doi.org/10.21105/joss.05161

Besançon, M., Papamarkou, T., Anthoff, D., Arslan, A., Byrne, S., Lin, D., & Pearson, J. (2021). Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem. *Journal of Statistical Software*, *98*(16). https://doi.org/10.18637/jss.v098.i16

Bessi, L., Rackauckas, C., Vikram, Singh Rathaur, R., Bhagavan, S., Marks, T., Anantharaman, R., marcoq, Strouwen, A., Cognolato, A., Foiles, D., Latawiec, P., Bharambe, A., michiboo, Yalburgi, S., Baker, F., Fuhrmann, J., st–, Thazhemadam, A., … Sarnoff, J. (2024). *SciML/Surrogates.jl: v6.10.0*. Zenodo. https://doi.org/10.5281/ZENODO.12571718

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bourinet, J.-M., C'ecile, M., & Dubourg, V. (2009, September). *A Review of Recent Features and Improvements Added to FERUM Software*. ISBN: 0-415-47557-0

[1]The figure is created using `Makie.jl` package (Danisch & Krumbiegel, 2021).

Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible High-Performance Data Visualization for Julia. *Journal of Open Source Software*, *6*(65), 3349. https://doi.org/10.21105/joss.03349

Der Kiureghian, A., Haukaas, T., & Fujimura, K. (2006). Structural Reliability Software at the University of California, Berkeley. *Structural Safety*, *28*(1-2), 44–67. https://doi.org/10.1016/j.strusafe.2005.03.002

Echard, B., Gayton, N., Lemaire, M., & Relun, N. (2013). A Combined Importance Sampling and Kriging Reliability Method for Small Failure Probabilities with Time-Demanding Numerical Models. *Reliability Engineering & System Safety*, *111*, 232–240. https://doi.org/10.1016/j.ress.2012.10.008

Franssen, J.-M., & Gernay, T. (2017). Modeling Structures in Fire with SAFIR®: Theoretical Background and Capabilities. *Journal of Structural Fire Engineering*, *8*(3), 300–323. https://doi.org/10.1108/JSFE-07-2016-0010

Ge, H., Xu, K., & Ghahramani, Z. (2018). *Turing: A Language for Flexible Probabilistic Inference*. 1682–1690.

Laverny, O., & Jimenez, S. (2024). Copulas.jl: A Fully Distributions.jl-Compliant Copula Package. *Journal of Open Source Software*, *9*(94), 6189. https://doi.org/10.21105/joss.06189

McKenna, F., Scott, M. H., & Fenves, G. L. (2010). Nonlinear Finite-Element Analysis Software Architecture Using Object Composition. *Journal of Computing in Civil Engineering*, *24*(1), 95–107. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000002

Olivier, A., Giovanis, D. G., Aakash, B. S., Chauhan, M., Vandanapu, L., & Shields, M. D. (2020). UQpy: A General-Purpose Python Package and Development Environment for Uncertainty Quantification. *Journal of Computational Science*, *47*, 101204. https://doi.org/10.1016/j.jocs.2020.101204

Revels, J., Lubin, M., & Papamarkou, T. (2016). *Forward-Mode Automatic Differentiation in Julia*. arXiv. https://doi.org/10.48550/ARXIV.1607.07892

Sabelli, R., Ziemian, R. D., & Schafer, B. W. (2024). ASD and LRFD Lateral Load Combinations: Comparison of Required Strength and Reliability for Design of Structural Steel. *Journal of Constructional Steel Research*, *212*, 108210. https://doi.org/10.1016/j.jcsr.2023.108210