Simulation Decomposition in Python

Pamphile T. Roy ⓘ [1]¶ **and Mariia Kozlova** ⓘ [2]

1 Consulting Manao GmbH, Vienna, Austria **2** LUT Business School, LUT University, Lappeenranta, Finland ¶ Corresponding author

Summary

Uncertainties are everywhere. Whether you are developing a new Artificial Intelligence (AI) system, running complex simulations or making an experiment in a lab, uncertainties influence the system. Therefore, an approach is needed to understand how these uncertainties impact the system's performance.

SimDec offers a novel visual way to understand the intricate role that uncertainties play. A clear Python Application Programming Interface (API) and a no-code interactive web dashboard make uncertainty analysis with SimDec accessible to everyone.

Statement of need

From real life experiments to numerical simulations, uncertainties play a crucial role in the system under study. With the advent of AI and new regulations such as the AI Act or the *Better Regulation Guideline* (European Commission, 2021), there is a growing need for explainability and impact assessments of systems under uncertainties.

Traditional methods to analyse the uncertainties focus on quantitative methods to compare the importance of factors, there is a large body of literature and the field is known as: Sensitivity Analysis (SA) (Saltelli et al., 2007). The indices of Sobol' are a prominent example of such methods (Sobol, 1993).

Simulation Decomposition or SimDec moves the field of SA forward by supplementing the computation of sensitivity indices with the visualization of the type of interactions involved, which proves critical for understanding the system's behavior and decision-making (Kozlova et al., 2024). In short, SimDec is a hybrid uncertainty-sensitivity analysis approach that reveals the critical behavior of a computational model or an empirical dataset. It decomposes the distribution of the output (target variable) by automatically forming scenarios that reveal the most critical behavior of the system. The scenarios are formed out of the most influential input variables (defined with variance-based sensitivity indices) by breaking down their numeric ranges into states (e.g. *low* and *high*) and creating an exhaustive list of their combinations (e.g. (i) *low A* & *low B*, (ii) *low A* & *high B*, (iii) *high A* & *low B*, and (iv) *high A* and *high B*). The resulting visualization shows how different output ranges can be achieved and what kind of critical interactions affect the output—as seen in Figure 1. The method has shown value for various computational models from different fields, including business, environment, and engineering, as well as an emerging evidence of use for empirical data and AI.
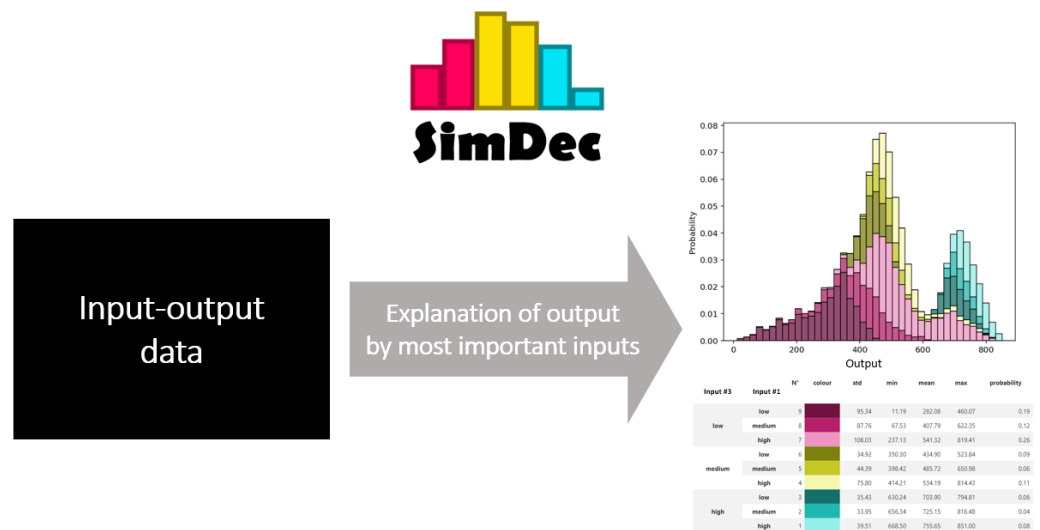
Figure 1: SimDec: explanation of output by most important inputs. A simulation dataset of a structural reliability model with one key output variable and four input variables is used for this case. Inputs 3 and 1 have the highest sensitivity indices and thus are automatically chosen for decomposition. The most influential input 3 divides the distribution of the output into three main states with distinct colors. Input 1 further subdivides them into shades. From the graph, it becomes obvious that input 1 influences the output when input 3 is low, but has a negligible effect when input 3 is medium or high.

Besides proposing a comprehensive yet simple API through a Python package available on PyPi, SimDec is also made available to practitioners through an online dashboard at https://simdec.io. The project relies on powerful variance-based sensitivity analysis methods from SALib (Herman & Usher, 2017) and SciPy (Roy et al., 2023; Virtanen et al., 2020)—notably the Quasi-Monte Carlo capabilities with `sp.stats.qmc` and in the future sensitivity indices with `sp.stats.sensitivity_indices`. The dashboard is made possible thanks to Panel (Rudiger et al., 2024).

Acknowledgements

References

European Commission. (2021). *Better Regulation Toolbox*. https://ec.europa.eu/info/law/law-making-process/planning-and-proposing-law/better-regulation-why-and-how/better-regulation-guidelines-and-toolbox_en

Herman, J., & Usher, W. (2017). SALib: An open-source Python library for sensitivity analysis. *The Journal of Open Source Software*, *2*(9). https://doi.org/10.21105/joss.00097

Kozlova, M., Moss, R. J., Yeomans, J. S., & Caers, J. (2024). Uncovering heterogeneous effects in computational models for sustainable decision-making. *Environmental Modelling & Software*, *171*, 105898. https://doi.org/10.1016/j.envsoft.2023.105898

Roy, P. T., Owen, A. B., Balandat, M., & Haberland, M. (2023). Quasi-monte carlo methods in Python. *Journal of Open Source Software*, *8*(84), 5309. https://doi.org/10.21105/joss.05309

Rudiger, P., Madsen, M. S., Hansen, S. H., Liquet, M., Andrew, Artusi, X., Bednar, J. A., B, C., Stevens, J.-L., Deil, C., Roumis, D., Signell, J., Paprocki, M., Wu, J., Mease, J., Arne, Coderambling, Amanieu, H.-Y., thuydotm, … TBym. (2024). *Holoviz/panel: Version 1.4.3* (Version v1.4.3). Zenodo. https://doi.org/10.5281/zenodo.11261266

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., & Tarantola, S. (2007). *Global Sensitivity Analysis. The Primer* (pp. 237–275). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470725184

Sobol, I. M. (1993). Sensitivity analysis for non-linear mathematical models, originally "sensitivity estimates for non-linear mathematical models." *Math Model Comput Exp*, *1*, 407–414.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., & Bright, J. et al. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2