The Journal of Open Source Software

# PDOPT: A Python library for Probabilistic Design space exploration and OPTimisation

**Andrea Spinelli** [1¶] **and Timoleon Kipouros** [1]

**1** Centre for Propulsion and Thermal Power, Cranfield University, MK430AL, UK ¶ Corresponding author

**DOI:** 10.21105/joss.06110

**Software**

- Review
- Repository
- Archive

**Editor:** Kyle Niemeyer

**Reviewers:**

- @e-dub
- @jbussemaker

**Submitted:** 23 October 2023
**Published:** 04 March 2024



## Summary

Contemporary engineering design is characterised by products and systems with increasing complexity coupled with tighter requirements and tolerances. This leads to high epistemic uncertainty due to numerous possible configurations and a high number of design parameters. Set-Based Design is a methodology capable of handling these design problems, by exploring and evaluating as many alternatives as possible, before committing to a specific solution.

The Python package `PDOPT` aims to provide this capability without the high computational cost associated with the factorial-based design of experiments methods. Additionally, `PDOPT` performs the requirement mapping without explicit rule definition. Instead, it utilizes a probabilistic machine learning model to identify the areas of the design space most promising for user-provided requirements. This yields a plethora of feasible design points, assisting designers in understanding the system behaviour and selecting the desired configurations for further development.

## State of the field

Engineering design is a process where quantitative and qualitative needs are transformed into a set of specifications (e.g., geometry, materials, component list) such that they can be sufficiently satisfied. The procedure involves performing several analyses of the system under design through suitable modelling tools, which map the design parameters to the quantified needs, and carrying out decisions to narrow and identify the range of desirable parameters. To aid this procedure, the designer relies on computer simulations through optimisation methods or the design of experiments in the design space.

Historically, optimisation has been used in the late stages of design, to refine a solution already in the ballpark of the requirements. However, with the development of Multi-disciplinary Optimisation (MDO), there has been increasing interest in bringing optimisation in earlier stages of the design process. Searching for optimal designs with concurrent analysis of different disciplines allows one to obtain better results than with sequential optimisation alone, reducing the time and cost of the development cycle (Martins & Lambe, 2013). In the conceptual phase, optimisation can be employed to understand which combination of input design parameters enables to satisfy the requirements. The algorithm acts by changing the input quantities iteratively to find the ideal combination(s) that minimise/maximise quantities of interest (often performance targets) without violating the problem constraints (which can be technological, geometrical, economical, and so on). Population-based solvers such as Genetic Algorithm (K. Deb et al., 2002) or Particle Swarm (Kennedy & Eberhart, 1995) are best suited for this task as they evolve multiple design points in parallel, rather than iteratively improving a single solution (Kalyanmoy Deb, 2008). The result is a family of points, covering the design space which are optimal. Surrogate modelling (Forrester & Keane, 2009) is often introduced to

reduce the computational cost of each functional evaluation and correct with high-fidelity data fast low-fidelity models often used in the conceptual design phase (Kontogiannis et al., 2020).

Software libraries developed for optimisation in Python are the OpenMDAO framework, which focuses on multi-disciplinary optimisation (Gray et al., 2019), the Object-Oriented `pyOpt` (Perez et al., 2012), the `DEAP` evolutionary optimisation library (Fortin et al., 2012), and `pymoo` multi-objective optimisation library (Blank & Deb, 2020). These libraries are open-source and general-purpose.

While effective in finding the desired set of design points, optimisation methods require to properly set up the problem to deliver meaningful results. This may be difficult at the beginning of the design procedure, as needs may be not fully defined, and the behaviour of the model not be fully understood. Design of Experiments (DoE) methods can assist in initiating this process through sampling and evaluation of the design space to obtain an understanding of the system response. Traditionally this approach is carried out through factorial sampling, where all possible combinations of extreme parameter values are tested (Montgomery, 2019). A response surface would then be built for analysing the response of the system inside the interval. This approach is feasible when the number of variables is low, as the number of experiments to be carried out grows exponentially. In the computational engineering field, quasi-random sampling is preferred, with Latin Hypercube sampling and Sobol sequences being most used (Yondo et al., 2018). These allow for training surrogate models for design exploration with a lower number of points in cases where factorial design would require an unfeasible amount.

Currently there is not a specific Python library to perform DoE-based design space exploration. Unlike an optimisation problem, there is no fixed procedure to obtain the mapping of the requirements on the input parameters. The designer is free to play with the model, with the aid of visualisation methods to understand the problem at hand. Examples of libraries useful for performing DoE sampling are the Quasi-Monte Carlo sampling module `scipy.qmc` and the Surrogate Modeling Toolbox (SMT) (Saves et al., 2024). SMT also provides methods to build response surfaces from the sampled data, using both radial-basis function (RBF) interpolation and Kriging (Gaussian Process Regression). Alternatively, scipy provides a RBF interpolation function in `scipy.interpolate.Rbf`, while Kriging models can be built using `scikit-learn.gaussian_process` module (Pedregosa et al., 2011) or with the library `GPpy` (GPy, since 2012).

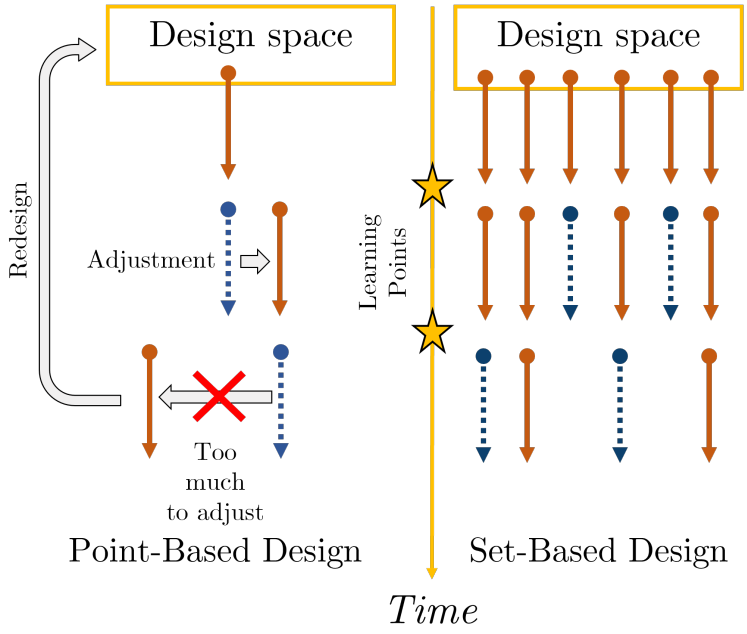


**Figure 1:** Comparison between Set-Based Design and traditional design.

The framework presented in this paper attempts to combine the two approaches in a single framework through the application of the principles of Set-Based Design (SBD) and Bayesian probability. SBD is a design practice that focuses on narrowing down the input design space by eliminating the candidate designs that do not satisfy the needs and requirements (Singer et al., 2009). It was originally developed by Ward (Ward & Seering, 1993) and it is commonly associated with the Toyota Production System (Sobek & Ward, 1996). Figure 1 compares SBD with the traditional iterative design process. The traditional method selects an initial candidate design to be refined through iterations as the problem is better understood over time. This approach is suited mainly for evolutionary designs, where the starting point is often already in the ballpark of the requirements. However, if the initial point is not close to the desired solution, local adjustements may not be enough to bring it to the required satisfaction, prompting a complete rework of the design. The SBD approach mitigates this by analysing many candidates in parallel and eliminating those who are found to be unfeasible, gradually reducing the pool to the desired solution. This approach has been shown to produce a more robust design cycle (McKenney et al., 2011).

SBD is used in `PDOPT` for performing an initial assesment of the design space and restriction to the most promising portions, which are then evaluated with local MDO problems. Georgiades previously developed a framework combining SBD and MDO named `ADOPT` (Georgiades et al., 2019), of which `PDOPT` is a development. The difference is in the set-elimination process. `ADOPT` used expert-defined rules that mapped the input parameters to the quantities of interest. This approach is robust in case of well understood design problems, but limited for unconventional systems, of which there is no best practices to draw from. `PDOPT` overcomes this limitation by applying Bayesian probability (Bernardo & Smith, 2009) as a selection criterion and assuming the underlying MDO model is a source of knowledge for the set elimination process. By casting the requirements in a probabilistic statement (i.e., "What is the probability it is satisfied?") and sampling in each set using surrogate models, it is possible to estimate the likelihood a set can satisfy all the requirements simultaneously and, therefore, worthy of further analysis. The advantage of this methodology is not having to rely on additional hardcoded rules on top of the implicit assumptions in the MDO model.

## Statement of need

PDOPT, short for Probabilistic Design and OPTimisation, is a Python package for design space exploration of systems under design. It implements a set-based approach for mapping the requirements to the design space, using a probabilistic surrogate model trained on the provided design model. This procedure ensures to identification of the best candidate areas of the design space with the minimum number of assumptions of the design of the system. The framework is designed to handle both continous parameters (represented as ranges) and discrete parameters (represented as a list of integers to be interpreted by the evaluation function).

The API of PDOPT was designed as a library with class-based interfaces between the components of the framework. This ensures both flexibility and transparency, as the user can inspect the main data structure between the phases of the framework. A full PDOPT analysis consists of two phases: the Exploration phase and the Search phase, shown in Figure 2.

The first phase surveys the design space to isolate the areas that are most likely to satisfy the constraints over the quantities of interest of the model. It does not seek to identify individual design points, but looks at "sets" (i.e., portions of the design space defined by parameter ranges) such to already eliminate candidates that would not produce feasible design points. The second phase introduces a multi-objective optimisation problem in each surviving set for recovering the individual design points. These are run with the input parameters bounded by the limits of each set, thus thoroughly finding the local optimal designs, satisfying the constraints. The result is multiple local Pareto fronts, one for each set.

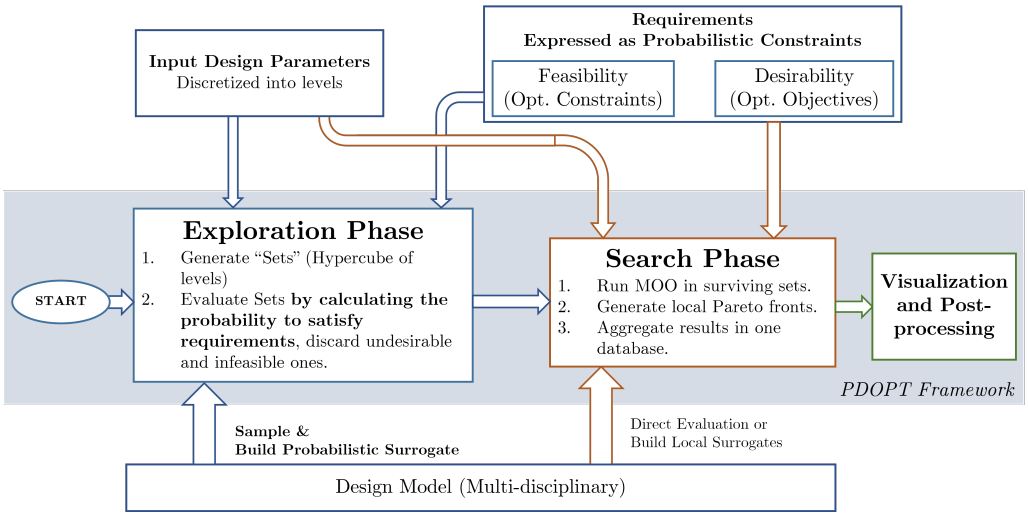


**Figure 2:** Overall architecture of PDOPT.

The fundamental idea behind the Exploration phase is to map the requirements without rule elicitation, instead relying on the design model provided for the Search phase. The assumption is the design model contains implicitly the knowledge necessary to map the quantities of interest to the input parameters. The mapping procedure mathematically is equivalent to identifying the domain where each constraint is true. The edge of this domain is the decision boundary of that constraint (Figure 3).

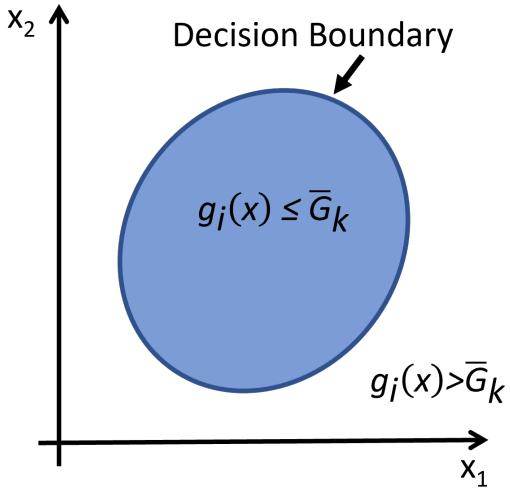


**Figure 3:** Decision boundary and domain of a generic requirement.

The set-based procedure consists of breaking down the design space into discrete portions to be evaluated. Sets crossed by the decision boundary are difficult to evaluate in a boolean way. This is avoided by reformulating the statement (i.e., the inequality must be true) as a probabilistic one to be interpreted in a Bayesian way (i.e., "What is the probability the inequality is true?"). This allows a fuzzy margin around the decision boundary and enables to inclusion of sets that would otherwise be hard to select or discard, as shown in Figure 4 and Figure 5.

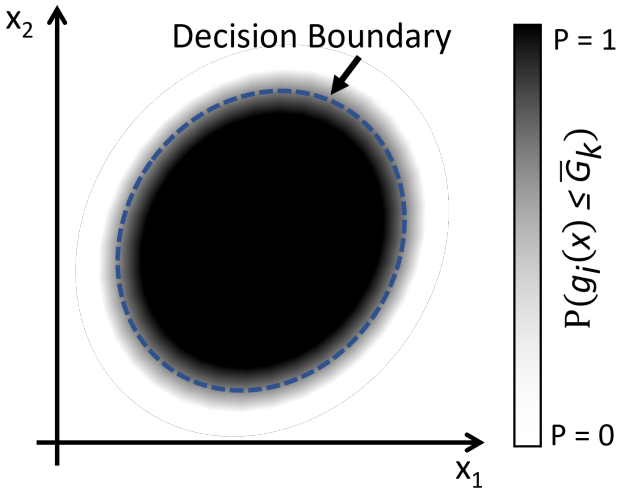


**Figure 4:** Probabilistic decision boundary.

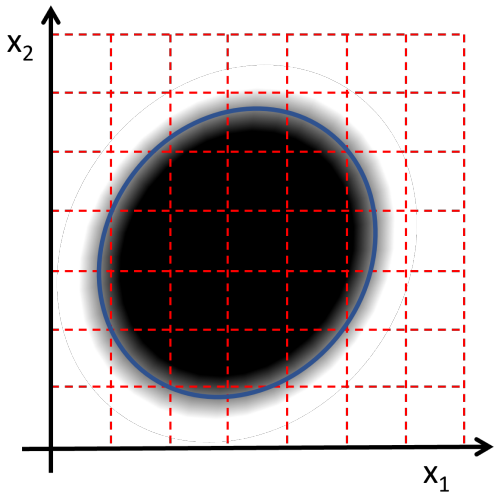


**Figure 5:** Probabilistic decision boundary and set boundaries.

Casting the requirements in probabilistic form also enables requirement mapping without the need for explicit rules. Instead, by sampling the set and evaluating the points with a Gaussian Process Regressor, it is possible to estimate the probability of requirement satisfaction. Each evaluated point has a mean and variance of the quantity of interest subject to the constraint. The k-th point probability of satisfaction of the i-th requirement is then calculated as:

$$P^k(y_i < \overline{g_i}) = \Phi\left(\frac{\overline{g_i} - \mu_i^k}{\sigma_i^k}\right)$$

where $\overline{g_i}$ is the value of the value of the decision boundary. This calculation is visualised in Figure 6.

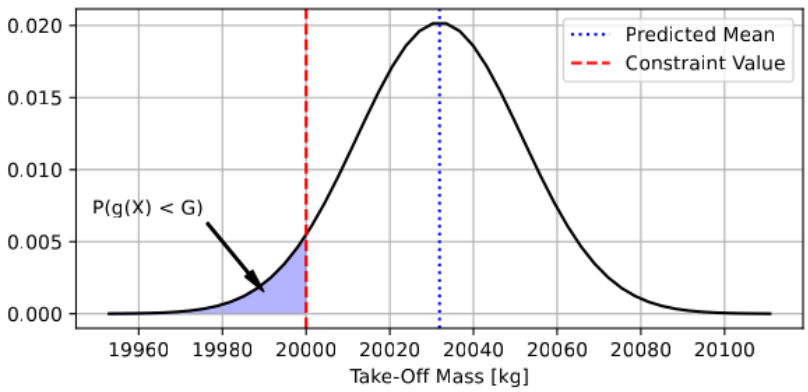


**Figure 6:** Probability of satisfying a constraint $\overline{g_i}$ for a sampled point $X_k$.

The probability of the whole set is calculated by counting how many points were able to satisfy the requirement over the total number of samples:

$$P_i = \frac{n_{i,sat}}{N_{samples}}$$

Multiple requirements are aggregated by assuming conditional independence, thus multiplying them together. Sets are then discarded if their overall probability is lower than the threshold for acceptance. Surviving sets are passed to the Search phase for local MDO. The optimisation algorithm adopted is the U-NSGA3 (Seada & Deb, 2016) implementation in `pymoo`, a gradient-free genetic algorithm with the flexibility of handling from single to many-objective problems. Furthermore it is capable of handling a non-smooth evaluation function, assuming the simulation model would capture any exceptions in its execution. The framework is intended to be used for large number of input paramters, with the option to run locally trained surrogate models to speed up the optimisation analysis.

The design points obtained from the local MDO problems yield both the global Pareto front and the feasible suboptimal points. Interactive visualisation tools can be used to analyse the results and proceed with design selection. Thanks to the probabilistic mapping of the requirements to the design space, the computational cost for design space exploration can be reduced by up to 80% (Spinelli, Anderson, et al., 2022), as the unfeasible sets are evaluated with the multi-disciplinary optimisation code.

`PDOPT` is intended to be used by researchers and engineers alike in developing complex engineering systems. It has been developed within the FutPrint50 project (*FutPrInt50*, 2020) and released as open-source software under the MIT license. The software has been used in several scientific publications regarding the design of hybrid-electric aircraft (Spinelli, Enalou, et al., 2022), and the effects of operating conditions (Spinelli, Krupa, Kipouros, Berseneff, et al., 2023) and technological uncertainty (Spinelli, Krupa, & Kipouros, 2023) on the design.

## Availability

`PDOPT` can be found on GitHub (Spinelli, 2023) and is compatible with the latest Python release. The release includes a PDF manual as a user guide and API reference. An example test set-up is also provided in the GitHub repository. Dependencies include the standard Python scientific stack (`numpy`, `scipy`, `pandas`, `matplotlib`) with the addition of the `scikit-learn` machine learning library (Pedregosa et al., 2011), the `pymoo` multi-objective optimisation framework (Blank & Deb, 2020), and the `joblib` parallelisation library. As an optional feature, `plotly` can be installed to take advantage of the prototypical decision-making environment packaged with the library.

## Acknowledgements

The authors thank the FutPrInt50 researchers for their collaboration and contributions to the software and test case applications. This project has received funding from the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No 875551.

## References

Bernardo, J. M., & Smith, A. F. (2009). *Bayesian theory* (Vol. 405). John Wiley & Sons.

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access*, *8*, 89497–89509.

Deb, Kalyanmoy. (2008). Introduction to evolutionary multiobjective optimization. In J. Branke, K. Deb, and K Miettinen, & R. Słowiński (Eds.), *Multiobjective optimization: Interactive and evolutionary approaches* (pp. 59–96). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-88908-3_3

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

https://doi.org/10.1109/4235.996017

Forrester, A. I. J., & Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, *45*(1), 50–79. https://doi.org/https://doi.org/10.1016/j.paerosci.2008.11.001

Fortin, F.-A., Rainville, F.-M. D., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*(70), 2171–2175. https://doi.org/https://doi.org/10.5555/2503308.2503311

*FutPrInt50*. (2020). https://www.futprint50.eu/

Georgiades, A., Sharma, S., Kipouros, T., & Savill, M. (2019). ADOPT: An augmented set-based design framework with optimisation. *Design Science*, *5*, e4. https://doi.org/10.1017/dsj.2019.1

GPy. (since 2012). *GPy: A gaussian process framework in python*. http://github.com/SheffieldML/GPy.

Gray, J. S., Hwang, J. T., Martins, J. R. R. A., Moore, K. T., & Naylor, B. A. (2019). OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, *59*(4), 1075–1104. https://doi.org/10.1007/s00158-019-02211-z

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Kontogiannis, S. G., Demange, J., Savill, A. M., & Kipouros, T. (2020). A comparison study of two multifidelity methods for aerodynamic optimization. *Aerospace Science and Technology*, *97*, 105592. https://doi.org/https://doi.org/10.1016/j.ast.2019.105592

Martins, J. R. R. A., & Lambe, A. B. (2013). Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, *51*(9), 2049–2075. https://doi.org/10.2514/1.J051895

McKenney, T. A., Kemink, L. F., & Singer, D. J. (2011). Adapting to changes in design requirements using set-based design. *Naval Engineers Journal*, *123*(3), 67–77. https://doi.org/10.1111/j.1559-3584.2011.00331.x

Montgomery, D. C. (2019). *Design and analysis of experiments*. John Wiley & Sons. ISBN: 978-1-119-49244-3

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Perez, R. E., Jansen, P. W., & Martins, J. R. R. A. (2012). pyOpt: A python-based object-oriented framework for nonlinear constrained optimization. *Structures and Multidisciplinary Optimization*, *45*(1), 101–118. https://doi.org/10.1007/s00158-011-0666-3

Saves, P., Lafage, R., Bartoli, N., Diouane, Y., Bussemaker, J., Lefebvre, T., Hwang, J. T., Morlier, J., & Martins, J. R. R. A. (2024). SMT 2.0: A surrogate modeling toolbox with a focus on hierarchical and mixed variables gaussian processes. *Advances in Engineering Sofware*, *188*, 103571. https://doi.org/https://doi.org/10.1016/j.advengsoft.2023.103571

Seada, H., & Deb, K. (2016). A unified evolutionary optimization procedure for single, multiple, and many objectives. *IEEE Transactions on Evolutionary Computation*, *20*(3), 358–369.

Singer, D. J., Doerry, N., & Buckley, M. E. (2009). What is set-based design? *Naval Engineers Journal*, *121*(4), 31–43. https://doi.org/10.1111/j.1559-3584.2009.00226.x

Sobek, D. K., & Ward, A. C. (1996). Principles from toyota's set-based concurrent engineering process. *International Design Engineering Technical Conferences and Computers and*

*Information in Engineering Conference*, *97607*, V004T04A035. https://doi.org/10.1115/96-DETC/DTM-1510

Spinelli, A. (2023). PDOPT. In *GitHub Repository*. GitHub. https://github.com/spinjet/pdopt-code

Spinelli, A., Anderson, L., Enalou, H. B., Zaghari, B., Kipouros, T., & Laskaridis, P. (2022). Application of probabilistic principles to set-based design for the optimisation of a hybrid-electric propulsion system. *IOP Conference Series: Materials Science and Engineering*, *1226*(1), 012064. https://doi.org/10.1088/1757-899X/1226/1/012064

Spinelli, A., Enalou, H. B., Zaghari, B., Kipouros, T., & Laskaridis, P. (2022). Application of probabilistic set-based design exploration on the energy management of a hybrid-electric aircraft. *Aerospace*, *9*(3). https://doi.org/10.3390/aerospace9030147

Spinelli, A., Krupa, G. P., & Kipouros, T. (2023). Set-based design space exploration to investigate the effect of energy storage durability on the energy management strategy of a hybrid-electric aircraft. *AIAA SCITECH 2023 Forum*. https://doi.org/10.2514/6.2023-0837

Spinelli, A., Krupa, G. P., Kipouros, T., Berseneff, B., & Fiette, S. (2023). Investigation of the operational flexibility of a regional hybrid-electric aircraft. *Journal of Physics: Conference Series*, *2526*(1), 012021. https://doi.org/10.1088/1742-6596/2526/1/012021

Ward, A. C., & Seering, W. P. (1993). Quantitative inference in a mechanical design 'compiler'. *Journal of Mechanical Design*, *115*(1), 29–35. https://doi.org/10.1115/1.2919320

Yondo, R., Andrés, E., & Valero, E. (2018). A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses. *Progress in Aerospace Sciences*, *96*, 23–61. https://doi.org/https://doi.org/10.1016/j.paerosci.2017.11.003