INTERNATIONAL DESIGN CONFERENCE – DESIGN 2020
https://doi.org/10.1017/dsd.2020.82

ROBUSTNESS IMPROVEMENT USING OPEN SOURCE CODE LIBRARIES

J. Sanchez, Z. Björkman and K. N. Otto ✉
Aalto University, Finland
✉ kevin.otto@aalto.fi

Abstract
Computer tools are commonly used to assess designs. We develop a toolchain using open source code libraries in Python to provide an open source, interactive robust design improvement toolchain. A reference folder contains a script that reads an input parameter value file and runs the simulation. The toolchain executes uncertainty quantification steps by replicating the reference folder. This is repeated for design points, and mean and sigma graphs generated versus each design variable. This fits within a workflow of defining variation modes, design variables, and toolchain execution.

Keywords: robust design, computational design methods, design automation, open source design

1. Introduction

Robust design was introduced by Taguchi as an experimental method to study the effect of different input noise factors on performance variability, and how these can be reduced through design variable selections (Taguchi et al., 2000). Arvidsson and Gremyr (2008) review developments and research into what has become the standard experimental Robust Design Method (RDM), making use of design-of-experiments to reduce the performance variability of a design due to multiple causes. RDM is more than a statistical experiment, it involves multiple steps including identifying possible sources of variability, quantifying their relative contribution with experiments, generating ideas for design changes that may promote variation reduction, and then quantifying the ability of design changes to reduce this variability through a further set of experiments. Executing RDM early in design is needed to reduce the risk of non-compliance when the product goes in production (Thornton, 1999). Executing RDM remains a complex task for many industries (Arvidsson et al., 2003), particularly when used in conjunction with modelling and simulation tools. The inherent complexity has impeded adoption of RDM.
Yet, the need for RDM has increased, where increasingly systems are now design-optimized for higher performance, higher efficiency, and lower cost, see for example Arena et al. (2006) for a discussion on trends in defense system programs. Optimizing a system can result in tighter design margins to achieve higher performance with less mass or volume (Tan et al., 2017). Systems designed with tighter margins are inherently more prone to variability problems (Thornton, 1999). In summary, using modern design methods creates the need for clarifying and understanding how much performance variability there will be in a design, to compare the variability distribution against the targeted design margin, to thereby quantify the future manufacturing quality risks.
We find that one of the reason for impeded adoption of RDM in practice is in part the inaccessibility of tools to complete the necessary RDM calculations. We find that tools readily available to practicing

engineers remain traditional design-of-experiment approaches, which are less effective for computer-simulation-based analyses. Traditional factorial experimental methods make use of only a few levels on each factor which is effective for prototype hardware experimentation, whereas this restriction is not needed for computer-based analyses.

We find RDM making use of uncertainty quantification and improved sampling techniques for computer-based simulation tools are more difficult to access. Therefore, this paper presents an RDM workflow to implement robust design analysis using computer-based simulation tools, coupled to an open source tool. The computational steps of the RDM workflow are coded in the open source scripting language Python to provide a more accessible implementation tool to generate robust design.

This tool uses uncertainty quantification methods to compute variability over a range of considered design changes. Alternative designs are generated using user-selectable methods such as Latin hypercube sampling. The computed results can be visualized graphically to show how variability can be changed through design choices, in the standard RDM workflow approach (Hasenkamp et al., 2009). This enables improved understanding of the design, its inherent robustness, and the impact of design changes on variability. This work provides an easily accessible tool for practicing engineers and the design community to execute an RDM workflow with computer-based simulation tools.

2. Robust design related work

Design of experiments are use in robust design to produce the necessary information to understand the interaction between design variables and system response. Robinson et al. (2004) and Park (2006) provide reviews of works in robust design statistical methods. Hasenkamp et al. (2009) review workflows and practices that facilitate the step by step industrial implementation of RDM, from variable identification through screening through robustness improvement.

Following the RDM practice, we consider a set of noise variables defined completed with probabilistic distribution parameters. The uncertainty in performance can be computed based on the distributions of the noise variables and simulation model. Further following RDM practice, we consider a set of design variables each with a range of possible values that can be chosen, to minimize performance variability. Any combination of design variable values defines a design point.

Computer-based design-of-experiments can make use of non-factorial sampling approaches with associated improved discrepancy. Latin Hypercube formulations are effective (Viana, 2016), as are quasi-Monte-Carlo sequence based sampling methods (Liu and Han, 2017). Chen et al. (2006) pioneered application of uncertainty quantification and sensitivity analysis methods to compute robust designs. Jin et al. (2001) discuss alternative surrogate models for simulation in design. Fang et al. (2005) have described modelling for computer experiments using sampling methods such as Latin hypercube and uniform sampling, and show examples from the automotive industry. These previous research efforts demonstrated the necessary algorithms and methods to implement robust design; we adopt the methods here and make it available using open source code libraries within a scripted RDM workflow.

This paper builds on previous work to study variability in design. Workflows were developed for implementing model-based uncertainty quantification and sensitivity analysis to understand the variability in a design (Otto et al., 2019) and also to aid in root cause analysis of manufacturing variation problems (Otto and Sanchez, 2019). Here we consider not just quantifying the uncertainty and its contributors, but also to reducing that variation through design optimization. We implement a computer-based design-of-experiments over design variables, to visualize and reduce variability computed over a design-of-experiments over contributing noise variables.

Beyond design-of-experiments methods, other formulations combine uncertainty quantification with numerical optimization methods (Du and Chen, 2001; Akihiro et al., 2007). We choose here to provide studies in the standard RDM approach, showing the impact of design variable changes on performance variability. This provides further clarity on the effects of design changes, rather than only providing the variability optimization solution answer.

Finally, others have considered robustness for early concept phase analysis using methods different from RDM. Sigurdarson et al. (2019) study alternative design concepts for conflicts in requirements, to minimize conflicting requirements. Goetz et al. (2019) consider robust design through function structure modelling. Vogel et al. (2018) considers mechatronic robust design principles. These early concept phase methods complement RDM by seeking better design concepts to which RDM can be applied.

3. Robust design workflow and toolchain

Completing an RDM workflow requires a sequence of methods and procedures (Arvidsson and Gremyr, 2008), including variable identification, variation mode and effect analysis, screening analysis of noise variables, formation and analysis of the robust design experiment, and verification of the computed solution. We develop here an RDM workflow for use with simulation based tools, making use of available open source code libraries.

When a step of the workflow is executed, a set of output artefacts are generated which acts as input to initiate the next step. The sequence of artefacts generation is coded as an executable toolchain. A toolchain is a set of computational tools which compiles a sequence of commands and executes them sequentially. Generally, a toolchain can be built on top of different platforms and coded using a wide variety of programming languages. Here, the Jupyter Notebook platform is chosen to contain the toolchain, coded using Python. The Jupyter Notebook is an interactive open source web-based environment where each Notebook consists of both descriptive hypertext markup language (HTML) blocks and programming code blocks. Jupyter supports over 40 open source-programming languages, such as Python, R, Julia, and Scala. Python was chosen as the programing language mainly due to its large variety of available libraries and supporting ecosystem within the developers' community.

3.1. Robust design workflow

To execute a RDM workflow with simulation tools, a sequence of both manual and computational activities are needed, as outlined in Figure 1. Steps 1 to 4 outline the robust design problem formulation, steps 5 to 11 outline the execution, and the last step is a verification.

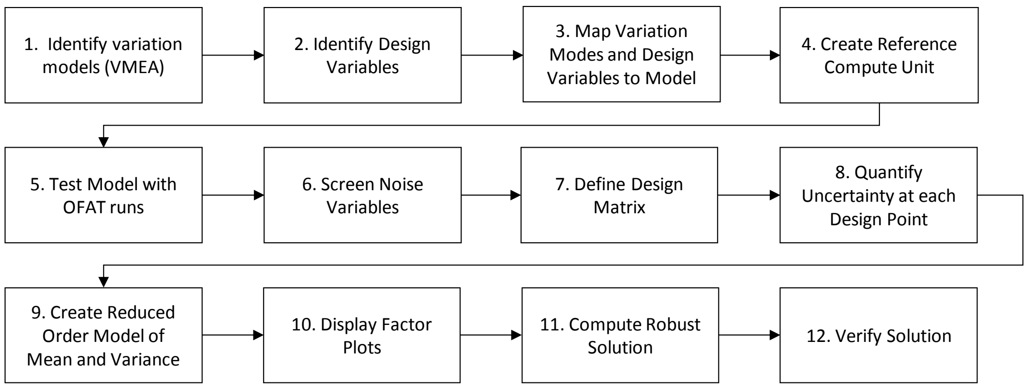

1. Identify variation models (VMEA)
2. Identify Design Variables
3. Map Variation Modes and Design Variables to Model
4. Create Reference Compute Unit
5. Test Model with OFAT runs
6. Screen Noise Variables
7. Define Design Matrix
8. Quantify Uncertainty at each Design Point
9. Create Reduced Order Model of Mean and Variance
10. Display Factor Plots
11. Compute Robust Solution
12. Verify Solution

Figure 1. Model based robust design workflow

The first step in the RDM workflow is a manual task to conceive all possible sources of variation, a variation mode and effect analysis (VMEA) (Johansson et al., 2006). This is a method to identify possible inputs which can contribute to variability in the performance responses and constraints. VMEA is especially useful to clarify what types of models are needed, and how suitable available modeling assets are against these sources of variation. The result of the VMEA analysis is a list of potential sources of variations, listed by components in the design.

An extension of VMEA for robust design is to also consider what design changes can have an impact on reducing the variability. Step 2 is a manual task to identify design variables that might change the variability of the response. These would be design variables that have an interaction effect with the significant noise variables on the response. As with VMEA, typically this list of design variables is the result of a brainstorming workshop with experts of the system under study. The design variables will be analysed in the design-of-experiment to determine the best combination of values which are expected to reduce the variability distribution of the performance response.

https://doi.org/10.1017/dsd.2020.82 Published online by Cambridge University Press

Step 3 in the workflow is to define a simulation model of the design to quantify and predict the performance and constraints behaviour of the system. The variation modes that were previously identified are mapped to the simulation model's input variables, where changing these model inputs will describe the effect of the variation modes. This model mapping step will highlight the adequacy of the model: variation modes which cannot be mapped to a model input indicate the variations not represented in the model. For example, a steady state model cannot represent dynamic variations, and a lumped parameter models cannot represent more refined geometric variations. The completeness of the model can be assessed against the modes identified in the VMEA as the fraction of variation modes represented by model inputs. The result of this step is the summary of variation modes linked to inputs of the simulation model, and a list of model inputs to study.

Step 4 in the workflow is to setup the simulation model for robust design variability study. We create a reference compute unit, which is a computer folder containing all necessary simulation model and calculation elements. For multiple runs of the simulation, this folder is copied and modified for each run. The folder contains an editable comma-separated values (CSV) file which lists the input variables and values to be changed in a single run of the simulation. Similarly within the reference folder, a batch file is created and run as a command line. This batch file contains the commands to initialize the simulation, read the input file, compute the simulation analysis, and finally writes an output CSV file listing the performance responses and values. Having the simulation compute unit modularized within a reference folder allows the workflow code to generate new input variable values for each single run of the simulation, simply by copying the folder and modifying the input CSV file.

Step 5 executes a short set of one-factor-at-a-time (OFAT) runs to ensure that the simulation model converges at any input value different from the nominal. This step is often necessary since the sampling methods used vary all inputs simultaneously, and if one input causes convergence problems, then many of the simulation runs may fail to converge. The toolchain implements a 1% change of each input value, and tests that the simulation solves and converges.

Step 6 then proceeds with the next RDM step, which is to screen the noise factors. Usually the VMEA and simulation models contain a large number of input noise variables, though many turn out to not have much contribution to output variation. As a result, a screening analysis is necessary to identify those noise variables which make a significant contribution to the response. Morris' method provides a fast screening method by rank-ordering the noise variables using a small number of simulation runs (Saltelli et al., 2008). The toolchain generates the Morris samples as reference folder replicates and launches the simulation model in each replicate folder to compute output values. The screening step outcome is a rank ordering of the input noise variables according to output variance contribution. The input variables with little contribution are dropped from consideration in the next step of reducing the uncertainty.

Step 7 of the RDM workflow is to study the impact of design changes to the variability. To specify this an input CSV file is created that define the noise variable distribution type, parameter values and also the range of design variable exploration. The toolchain uses this to define and executes an uncertainty quantification at each design point using the distribution functions defined. Modern methods offer various sampling techniques suitable for computer experimentation, such as quasi-Monte Carlo or Latin hypercube sampling (Giunta et al., 2003). Latin hypercube, Halton and Sobol sampling methods are provided as options in the toolchain, making use of available open source code libraries.

The toolchain first constructs a sampling over the design variables, such as a Latin hypercube sample. Each of these samples is a design point. At each design point, a further sample across the noise variables is made, according to the distribution parameter values. With this the complete set of runs, defined as design and noise experimental configurations, have been defined as an input matrix.

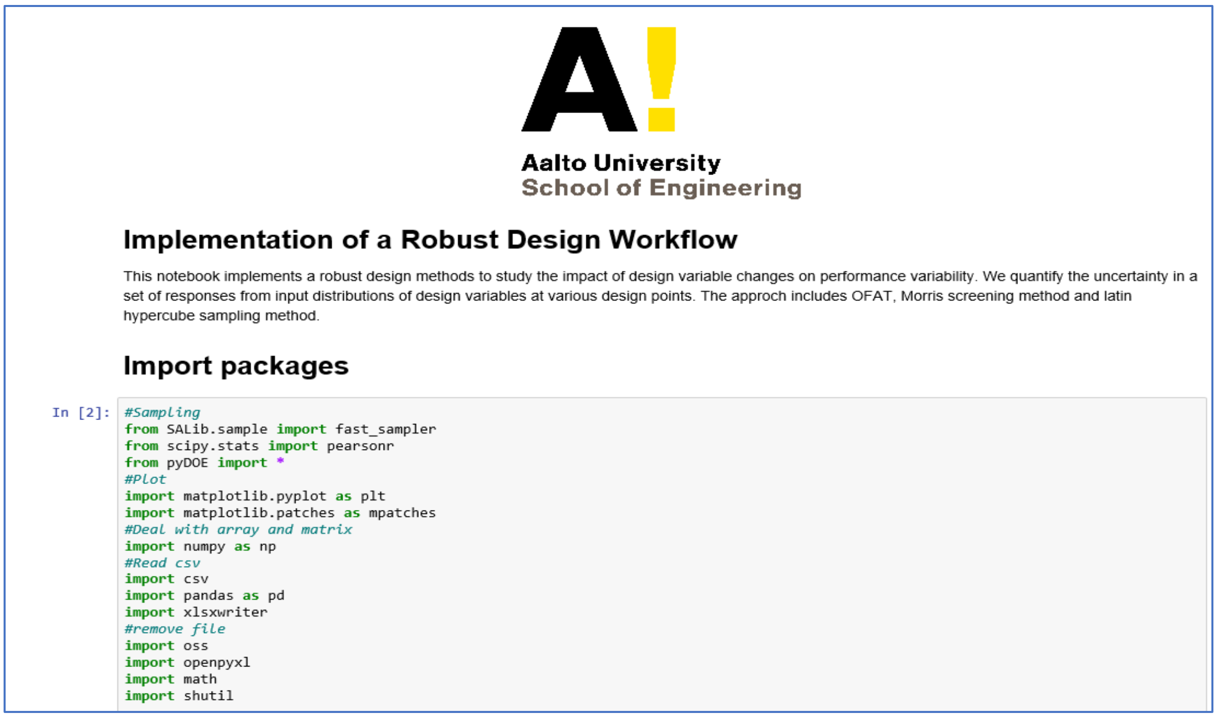
Step 8 executes the batch of calculations. The toolchain generates an output folder containing a set of sub-folders, one for each of the simulation runs. Each of these folders are copies of the reference folder with the input CSV file modified per the experiment matrix. Within each folder the simulation is executed by the batch file to compute the performance response. The uncertainty is quantified at each design point by considering the set of runs with the design point fixed but with the noise variables changing. At each design point, a histogram of the run results is formed and the mean and variance computed. This is repeated at each design point. These results are collected and summarized into a matrix which also lists the design variable values.

https://doi.org/10.1017/dsd.2020.82 Published online by Cambridge University Press

Next, step 9 is to fit a reduced order model to the design variables of both the computed mean and variance. To assess the adequacy of the fits, the surrogate model error is computed and graphed.
Step 10 in the RDM workflow is to generate factor plots, to provide a means of understanding the impact of design changes on the response variability. The factor plots graph the effect of the design changes on the performance response and the performance response standard deviation (alternatively, the variance or an S/N ratio could be graphed). Using the factor plots, selections of design variable values can be made by selecting those values which minimize the standard deviation or variance.
Next, step 11 in the RDM workflow is a manual selection of optimum values for the design variables. This can be done in the usual way of scrutinizing the factor plots for design variable values with lower performance variance. It can also be done in the toolchain as an optimization formulation, using the surrogate models of the mean and variance. One can minimize the variance subject to a constraint on the mean, for example. The results will be the values of the design variables that minimize the performance response variation while maintaining a reasonable response value.
Finally, the last step in the RDM workflow is to verify the chosen design configuration. This should be done through the fabrication and testing of the design. Preceding this fabrication axctivity, an uncertainty quantification can be done over the noise variables at the new optimum design variable values. The toolchain executes this analysis as above, creating another set of noise variable samples and evaluating the performance response. From these values, the mean and variance of the performance response can be computed to show the reduction in variability over the original nominal value results.

3.2. Python toolchain

The toolchain is coded in Python, presented within the Jupiter Notebook environment. The Notebook consists of a series of hypertext HTML blocks and computer code blocks, which can be easily edited and executed. Figure 2 shows the Jupyter Notebook in where the toolchain is implemented. It offers a variety of text elements, such as figure plots, equations, and narrative text which make the document readable as well as executable. Jupyter Notebooks are designed to implement scripted workflows, with a set of code blocks that are executed step by step. The Jupyter environment also allows the user to easily document instructions or comments between code blocks as HTML blocks. Notebooks can be easily shared.


Aalto University
School of Engineering

Implementation of a Robust Design Workflow

This notebook implements a robust design methods to study the impact of design variable changes on performance variability. We quantify the uncertainty in a set of responses from input distributions of design variables at various design points. The approch includes OFAT, Morris screening method and latin hypercube sampling method.

Import packages

```
In [2]: #Sampling
        from SALib.sample import fast_sampler
        from scipy.stats import pearsonr
        from pyDOE import *
        #Plot
        import matplotlib.pyplot as plt
        import matplotlib.patches as mpatches
        #Deal with array and matrix
        import numpy as np
        #Read csv
        import csv
        import pandas as pd
        import xlsxwriter
        #remove file
        import oss
        import openpyxl
        import math
        import shutil
```

Figure 2. Toolchain developed in a Jupyter Notebook

https://doi.org/10.1017/dsd.2020.82 Published online by Cambridge University Press

One important characteristic of the Python code blocks is the support of standard and open sources library packages such as CSV (Lundh, 2001), pyDOE (Baudin, 2015), and SALib (Herman and Usher, 2017). We used CSV for manipulating data (i.e reading and writing). SALib is a free open library for Python, it offers packages for sampling and sensitivity analysis. PyDOE is an experimental design package that can generate designs such as Latin hypercube matrices and full factorial experiments. Code libraries are easily imported and executed. As Jupyter Notebooks and Python are open source tools, they are in constant improvement by the global developer community. As such, there is inherently a wealth of support information on the calculations including statistical modelling, numerical simulation, etc., and the formulations are well documented online.

3.3. Example: Stirling engine analysis

In previous work (Otto et al., 2019; Otto and Sanchez, 2019) workflows were developed applying uncertainty quantification and sensitivity analysis methods to identify root causes of manufacturing quality problems. Here we use the same Stirling engine project described earlier, now as an example of a system requiring robustness improvement. As a university project, it is complete with a data set available for study. The Stirling engine has attracted attention in recent times as an object of optimization for its many attractive characteristics over the combustion engine, such as minimal pollution generation, high thermal efficiency and multi-heat-source capability. Hence, we implemented the robust design workflow on the Stirling engine.
At Aalto University students fabricated, assembled and tested Stirling engines as part of the senior level machine design course. A Stirling engine manufactured during the course can be seen in Figure 3. Students measured the speed at which the crank shaft rotates when there is no load attached to it, which is known as the no-load speed. The no-load speed tests demonstrated 25% variation in speed across the fabricated engines, due to variations in fabrication. This outcome exposed the high sensitivity of the Stirling engine to manufacturing and assembly variations. To determine if the variability of the design could be reduced, a robust design analysis was undertaken by the teaching team.

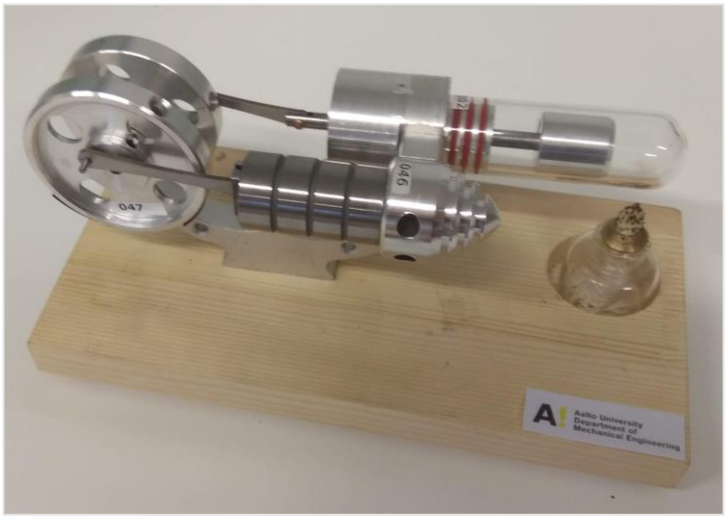


Figure 3. Miniature Stirling engine manufactured in the course

First, to implement the workflow outlined in Figure 1, we compiled a list of potential sources of variation in engine performance as a VMEA. From this, 42 variation modes were identified, including modes such as part tolerance, heat flux variability, air leakage, friction between components and misalignments. The majority of the variation modes were due to manufacturing component variation.
Following Step 3, these variation modes were then mapped to inputs of a Stirling engine simulation model. A Matlab simulation model was used to compute the power generated based on a Schmidt power formulation (Ureili, 2010). The model input variables were typically a compound of several variation modes. The 42 original variation modes were thereby represented by 11 model input variables in a created compute unit (Steps 4-5). Using the toolchain, we first performed a Morris

screening analysis to determine which of these noise variables significantly contribute to the engine power variability (Step 6). We varied all inputs over a ±1% range, and the screening results from the Morris analysis are shown in Figure 4. From this, it can be seen that the largest contributing noise variables are the clearance and swept cold side volumes, as well as the heat exchanger diameter, and the other input variations don't significantly contribute.

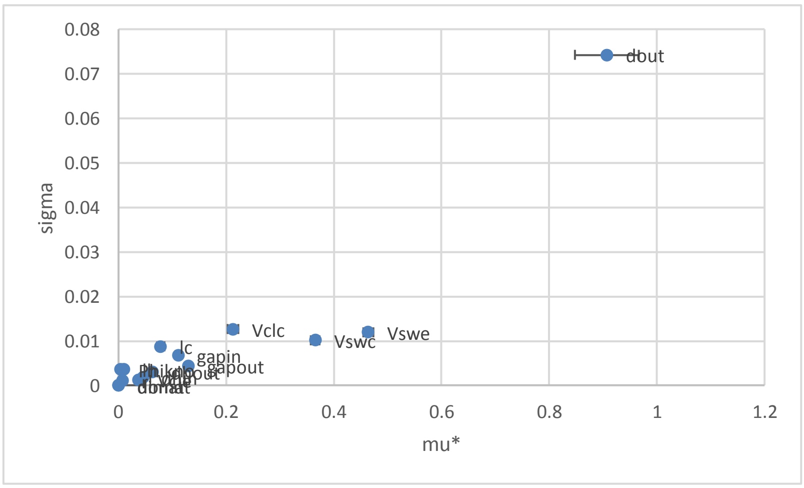


Figure 4. Morris analysis of the computed power variability

Having identified the significant contributors to the engine power variability, we proceeded to implement the main experiment. We had chosen design variables of the hot and cold side volumes, which could be easily changed and also impact the power generated (Step 7). We chose a range of ±20% of the nominal design, in order to explore how such design changes impact the variability due to the tolerances, yet not so large as to require changes to the design concept itself. We implemented Latin hypercube sampling for the design points, with 4 design variables and 40 design points.
Next (Step 8), we applied the toolchain to generate samples across the noise variables according to their distribution parameters again using Latin hypercube sampling. This is repeated at each design point. Folders are created per each simulation run and the batch file in each folder is executed. The simulation computed response (engine power) at each sample point is collected, and a histogram formed depicting the power uncertainty at each design point. Further, the response (power) mean and standard deviation at each design point is collected into a main experiment matrix CSV file.
Next we fit a surrogate model for the experimental matrix results (Step 9). It this case, we implemented an ordinary least squares surrogate, which fit to an r^2 of 0.99 for the average power and 0.94 for the power standard deviation. The surrogate model error plot for the mean and standard deviation generated can be seen in Figure 5.

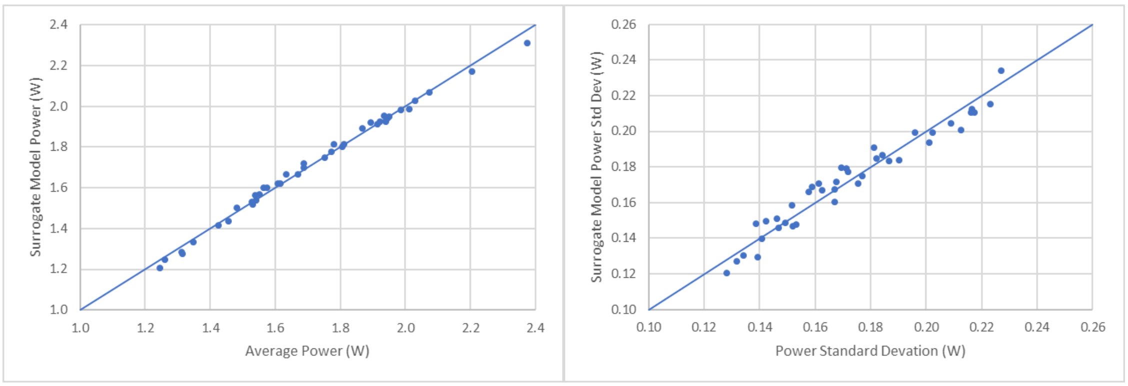


Figure 5. Surrogate model errors

To visualize the effect of variations of each design variables over the performance response and performance response deviation, factor plots are generated (Step 10). Figure 6 shows the generated factor plots for power mean and power sigma against each design variable. We can appreciate easily from Figure 6 that the cold side swept volume affects positively the generated engine power. This can be used to shift the mean of the generated power, a bigger value will increase the power without significantly increasing the standard deviation. Meanwhile the hot side swept volume is useful to reduce power variability, a small value will decrease the standard deviation. The hot side dead volume can be ignored; changing it does not improve change on the mean or variance of the power. These results suggest a more robust configuration, they show how design variables affect the power mean and power variance.

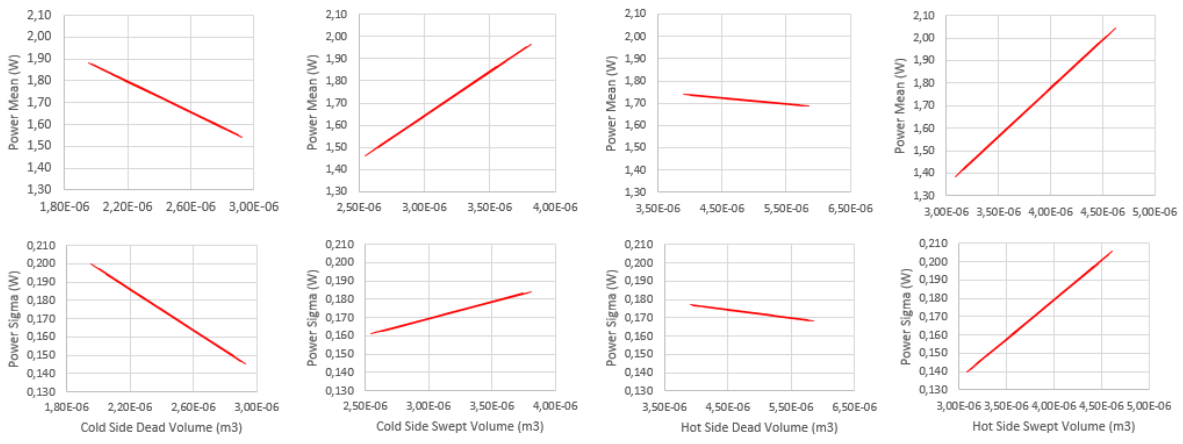
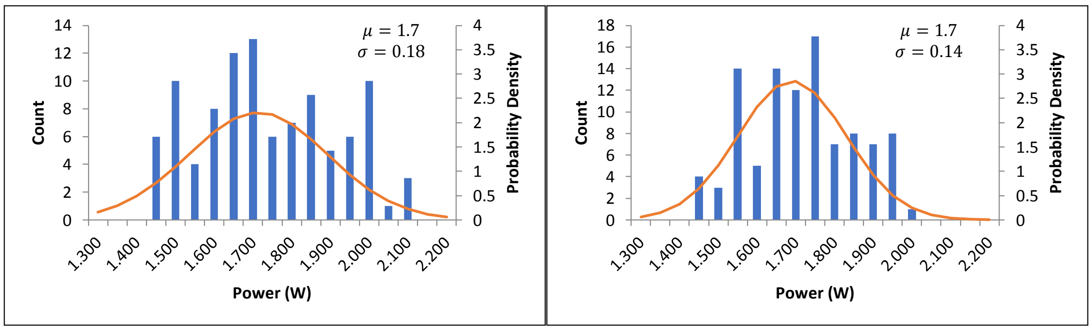


Figure 6. Factor plots of mean and sigma per design variable

In order to make a proper selection of optimum design variable values, we also formulated and solved an optimization (Step 11). The objective function was set to the power standard deviation, and a constraint was defined that the average engine power be no less than the power generated when using nominal values for the design variables.
To verify this, the last step in the workflow is to recompute the uncertainty using the new design point, using the same level of input variations. Figure 7 shows the histogram of the power response when using nominal values for the design variables versus optimum values for the same design variables. The sigma power of the response decrease its value from 0.18 to 0.14, demonstrating a 23% variation reduction.



(a) (b)

Figure 7. Uncertainty of engine power a) at nominal design values b) at robust configuration values

4. Conclusions

Scripted toolchains offer a means to disseminate design methods as easily portable and executed code. Here, we developed an RDM workflow for simulation-based tools, to study a design for sources of variation and to study the variation reduction impact that design changes can have. Doing this makes the design research that has gone into creating computer-based algorithms and methods for robust design more readily available.

https://doi.org/10.1017/dsd.2020.82 Published online by Cambridge University Press

The approach described makes use of quasi-Monte Carlo simulation to represent the variability, and computer based designed experiments to represent the design alternatives. This is useful for performance variations of up to 4 sigma variations. Problems with margins of interest at 5 or more standard deviations need other formulations such as optimization methods (Du and Chen, 2001; Akahiro et al., 2007).

The next steps in this work are to test these deployment assertions by further providing the toolchain to industrial partners for their own internal uses. Preliminary indications from a small sample of industrial users are positive, but this remains future work. Nonetheless, we feel implementation of design methods into standard workflows implemented in scripted toolchains provides a mechanism for more broad deployment.

An example was shown from a university teaching course, where the instructors and teaching assistants sought improvements to a Stirling engine design, to enable students to fabricate and build an engine with higher predictability. The example demonstrates how the toolchain is capable of helping the team to understand sources of variation and to identify design changes that can make the design more robust.

Acknowledgements

This work was made possible with support from an Academy of Finland, Project Number 310252.

References

Akihiro, T. et al. (2007), "A Conceptual Design Support Methodology Based on Structural Optimization Techniques Using Function-Oriented Elements", *Proceedings of the International Conference on Engineering Design, ICED'07*, Paris, France, August 28-31, 2007.

Arena, M.V. et al. (2006), *Historical Cost Growth of Completed Weapon System Programs*, RAND Corporation, Santa Monica, CA.

Arvidsson, M., Gremyr, I. and Johansson, P. (2003), "Use and Knowledge of Robust Design Methodology: A Survey of Swedish Industry", *Journal of Engineering Design*, Vol. 14 No. 2, pp. 129-143. https://doi.org/10.1080/0954482031000138192

Arvidsson, M. and Gremyr, I. (2008), "Principles of Robust Design Methodology", *Quality and Reliability Engineering International*, Vol. 24 No. 1, pp. 23-35. https://doi.org/10.1002/qre.864

Baudin, M. (2015), pyDOE. https://github.com/tisimst/pyDOE

Chen, W., Jin, R. and Sudjianto, A. (2006), "Analytical Global Sensitivity Analysis and Uncertainty Propagation for Robust Design", *Journal of Quality Technology*, Vol. 38 No. 4, pp. 333-348. https://doi.org/10.1080/00224065.2006.11918622

Du, X. and Chen, W. (2001), "A most Probable Point-Based Method for Efficient Uuncertainty Analysis", *Journal of Design and Manufacturing Automation*, Vol. 4 No. 1, pp. 47-66. https://doi.org/10.1080/15320370108500218

Fang, K., Li, R. and Sudjianto, A. (2005), *Design and Modelling for Computer Experiments*, Chapman and Hall/CRC.

Giunta, A., Wojtkiewicz, S. and Eldred, M. (2003), "Overview of Modern Design of Experiments Methods for Computational Simulations", *41st Aerospace Sciences Meeting and Exhibit*, Reno, NV, January 6-9, 2003. https://doi.org/10.2514/6.2003-649

Goetz, S. et al. (2019), "Robustness Evaluation of Product Concepts based on Function Structures", *Proceedings of the Design Society: International Conference on Engineering Design*. Cambridge University Press, Vol. 1 No. 1, pp. 3521-3530. https://doi.org/10.1017/dsi.2019.359

Hasenkamp, T., Arvidsson, M. and Gremyr, I. (2009), "A Review of Practices for Robust Design Methodology", *Journal of Engineering Design*, Vol. 20 No. 6, pp. 645-657. https://doi.org/10.1080/09544820802275557

Herman, J. and Usher, W. (2017), "SALib: An Open-source Python Library for Sensitivity Analysis", *The Journal of Open Source Software*, Vol. 2 No. 9. https://doi.org/10.21105/joss.00097

Jin, R., Chen, W. and Simpson, T. (2001), "Comparative Studies of Metamodelling Techniques Under Multiple Modelling Criteria", *Structural and Multidisciplinary Optimization*, Vol. 23 No. 1, pp. 1-13.

Johansson, P. et al. (2006), "Variation Modes and Effect Analysis: A Practical Tool for Quality Improvement", *Quality and Reliability Engineering International*, Vol. 22 No. 8, pp. 865-876. https://doi.org/10.1002/qre.773

Kluyver, T. et al. (2016), "Jupyter Notebooks-a publishing format for reproducible computational workflows", *Proceedings of the 20th International Conference on Electronic Publishing. Amsterdam*, pp. 87-90. https://doi.org/10.3233/978-1-61499-649-1-87

Liu, S. and Han, J. (2017), “Energy efficient stochastic computing with Sobol sequences”, *Proceedings of the Conference on Design, Automation & Test in Europe, European Design and Automation Association*, Lausanne, Switzerland, March 27-31, pp. 650-653. https://doi.org/10.23919/DATE.2017.7927069

Lundh, F. (2001), *Python standard library*, O’Reilly Media, Inc.

Matlab (2019), *2018. 9.7.0.1190202*, The MathWorks Inc, Natick, Massachusetts.

Otto, K., Wang, J. and Uyan, T. (2019), “Using Open Source Code Libraries for Robust Design Analysis”, *Proceedings of the International Conference on Engineering Design*, pp. 1733-1742. https://doi.org/10.1017/dsi.2019.179

Otto, K. and Sanchez, J. (2019), “Model Based Root Cause Analysis of Manufacturing Quality Problems Using Uncertainty Quantification and Sensitivity Analysis”, *Proceedings of the Computers and Information in Engineering Conference*, Anaheim, USA, August 18-21, Volume 1, pp. 18-21. https://doi.org/10.1115/DETC2019-97766

Park, G. (2006), “Robust design: An overview”, *AIAA Journal*, Vol. 44 No. 1, pp. 181-191. https://doi.org/10.2514/1

Robinson, T.J., Borror, C.M. and Myers, R.H. (2004), “Robust parameter design: A review”, *Quality and Reliability Engineering International*, Vol. 20, pp. 81-101. https://doi.org/10.1002/qre.602

Saltelli, A. et al. (2008), *Global Sensitivity Analysis: The Primer*, John Wiley & Sons.

Sigurdarson, N., Eifler, T. and Ebro, M. (2019), “Functional Trade-offs in the Mechanical Design of Integrated Products-Impact on Robustness and Optimisability”, *Proceedings of the 22nd International Conference on Engineering Design (ICED19)*, Delft, The Netherlands, 5-8 August 2019, Cambridge University Press, 1(1), pp. 3491-3500. https://doi.org/10.1017/dsi.2019.35

Tan, J., Otto, K. and Wood, K. (2017), “Relative impact of early versus late design decisions in systems development”, *Design Science*, Vol. 3

Taguchi, G., Chowdhury, S. and Taguchi, S. (2000), *Robust Engineering*, McGraw-Hill, New York. https://doi.org/10.1007/s13398-014-0173-7.2

Thornton, A.C. (1999), “Variation Risk Management using Modeling and Simulation”, *ASME. Journal of Mechanical Design*, Vol. 121 No. 2, pp. 297-304. https://doi.org/10.1115/1.2829457

Ureili, I. (2010), Stirling Cycle Machine Analysis. www.ohio.edu/mechanical/stirling/

Viana, F.A. (2016), “A tutorial on Latin hypercube design of experiments”, *Quality and Reliability Engineering International*, Vol. 32 No. 5, pp. 1975-1985. https://doi.org/10.1002/qre.1924

Vogel, S. et al. (2018), “Robust Design for Mechatronic Machine Elements-How Robust Design Enables the Application of Mechatronic Shaft-Hub Connection”, *DS 92: Proceedings of the 15th International Design Conference*, pp. 3033-3040. https://doi.org/10.21278/idc.2018.0203