An FMI-Based Tool for Robust Design of Dynamical Systems

Maria Henningsson, Johan Åkesson, Hubertus Tummescheit
Modelon AB / Modelon Inc.

Abstract

Concepts from quality sciences, such as robust design, six-sigma, and design-of-experiments have had a large impact on product development in industry. These concepts are increasingly used in a model-based engineering context where data is gathered from simulation models rather than laboratory setups or prototypes.

This paper presents a framework to apply such ideas to analysis of dynamical systems. A set of tools that can be used for uncertainty analysis of dynamical Modelica models is presented. These tools are made available in the FMI Toolbox for MATLAB. The workflow and tools are demonstrated on a cooling loop design problem.

Keywords: Design-of-experiments, Robust design, Controls, Modelica, FMI

1 Introduction

Model-based engineering is a key technology for competitive product development. However, implementing, parameterizing, and validating simulation models of physical systems is time-consuming and costly. To make modeling efforts pay off, it is necessary to systematically consider tools, practices, and workflows to get the most use out of a model portfolio.

In this paper, we present a tool-chain and a methodology to efficiently integrate concepts from robust design and design-of-experiments to design and analysis of dynamical systems. Robust design is a well-established methodology that aims to design products and systems such that they are inherently robust to variations in components and operating conditions. A large amount of research has been directed towards robust design methodologies, that includes concepts such as design of experiments, quality engineering, critical parameter management, and six sigma [8, 5]. In the last years, there has been a growing interest in applying these kinds of techniques to analyze detailed simulation models, see e.g. [1, 2, 6, 7, 10, 4].

The tools presented in the paper are integrated into the FMI toolbox for MATLAB. Functional Mockup Interface (FMI) is a standard that allows importing and exporting dynamical models between different tools. First released in 2010, the standard has quickly been adopted in industry and is currently supported by a large range of tools used for physical systems modeling. FMI is a powerful standard for deploying tool-independent workflows and processes. Together with physical dynamical simulation models from Modelica, it provides a suitable platform for applying robust design concepts to analysis of dynamical systems.

The paper starts by discussing typical applications of simulation models, and providing an overview of design-of-experiments (DoE) concepts. The ideas behind the design of the new DoE tools in the FMI toolbox for MATLAB are then described. The use of the tools is then illustrated for the design of a cooling system, including both static analysis to size the components, and dynamic analysis to design a controller.

2 Applications of simulation models

Simulation model development represents a significant strategical investment in industries such as process, automotive, aerospace, and energy. Simulation models are, however, never an end in themselves but rather a means to answer questions in the engineering design process. Such questions may be:

- Hardware optimization: find dimensions, settings, and operating points for physical components
- Verification: check that performance meets specs in entire operating envelope
- Quality: check probabilistically that system specs are satisfied given tolerances on components
- Controls: design, analyze, and test control algorithms

Simulation models should ideally be developed such that they can provide answers to all of these questions. In the literature that address model-based engineering, it is interesting to note that there is a substantial cultural difference between research publications in the control community compared to the first three items in the list.

Robust design methodologies generally consider coarse large-scale mathematical and statistical models of complex systems. The models may be based on data from physical experiments or supplier spec sheets, and are generally steady-state [10]. The literature is heavily focused on processes, workflows, and tools.

Control engineering literature has a strong mathematical and analytical focus. Dynamics are central, and simplified mathematical model classes are analyzed with rigor. Much research is focused on mathematical analysis but there is surprisingly little written on processes and practices for control design in the product development process. It is commonly noted that there is a significant gap between much of the mathematically-oriented control research community and industrial practice.

In practice, control design is often based on linearization at a single operating point. For the design to be successful, this operating point needs to be representative of the dynamics across the operating range. It is normally difficult to know how the dynamics of the process vary with operating points and parameter uncertainty. Control design methods that go beyond linear models are often cumbersome to use and require a detailed understanding of which nonlinear effects or other process uncertainty that will be relevant. In-depth domain experience or trial-and-error is generally needed to find these dominant effects.

In practice, PID controllers based on simple experiments or linearized models are widely applied in industry, even though all real-world processes have some amount of nonlinearity or uncertainty. It is not uncommon to find controllers that perform poorly, with slow responses or oscillations at off-design operating conditions.

In this paper, we suggest how approaches from the robust design field can be applied to analysis of dynamical simulation models and control design. In order to use information on process nonlinearities and uncertainty in control design, it is imperative that tools be designed so that the relevant information can be extracted in a convenient and intuitive format.

3 Design-of-experiments

The term design-of-experiments (DoE) denotes methodologies to gather information from a system or process in a systematic way. Originally, it was introduced as a means to collect statistically sound experimental data sets for establishing cause-and-effects relationships.

In recent years, there has been an increasing interest in applying DoE techniques to extract data from simulation models [2, 1].

3.1 Terminology and designs

In the DoE terminology, a *factor* denotes a quantity that is to be varied in the data set. An *experiment* is the procedure of testing the system with a particular choice of factor settings. A *response* is an outcome that is measured in the experiment. A *test matrix* is a list of factor setting combinations to be tested.

Selecting an appropriate test matrix is key to design-of-experiments. Figure 1 shows examples of possible test matrices in two dimensions. The first plot to the left shows a design based on one-factor-at-a-time (OFAT). Here, a nominal operating point is chosen, and off-nominal behavior is checked by varying one factor at a time. This is generally not an effective way to gather information on the system, since no interactions between the variables are considered. Nevertheless, this approach is often applied in an ad-hoc manner to test robustness.

The second plot shows a full-factorial test matrix design. This is equivalent to a multi-dimensional grid, where two or more different levels are chosen for each factor and the test matrix is then assembled from all combinations of the different levels. This method may work well in two dimensions but scales poorly with the number of factors. Other designs that are still based on fixed factor levels but that only select a subset of combinations are often preferred. Examples of such designs are fractional factorial designs, Box-Behnken designs, or central composite designs.

When DoE is applied to physical experiments, it can be advantageous to use a limited set of levels for each factor. For example, each different value of the factor could involve building a separate prototype. In simulation models, however, these constraints are generally not present and the test matrix can be constructed by spreading the test points freely in the ranges of interest. Monte-Carlo simulations spread the test points according to statistical distributions of the factors, which allows to get statistical estimates on distributions of

DOI 10.3384/ECP1409635

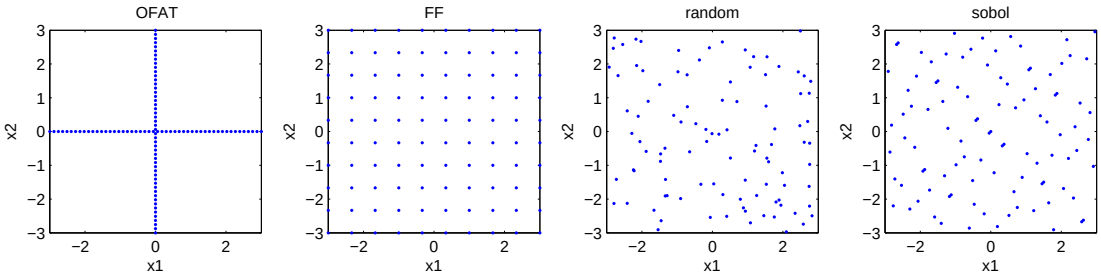

Figure 1: Example of test designs in two dimensions. OFAT = one-factor-at-a-time, FF = full factorial. Each of the four examples contain 100 test points.

the estimates. The third plot in Fig. 1 shows points that have been randomly sampled from uniform independent distributions for x_1 and x_2.

To maximize the information that can be extracted from the process in a limited number of simulations, the points should fill the space spanned by the factors. Quasi-Monte Carlo designs use space-filling algorithms to spread the test points in a way that covers the space better than random sampling. The rightmost plot in Fig. 1 shows a DoE design based on the Sobol space-filling algorithm. The advantages of using space-filling QMC designs instead of random sampling can be substantial in higher dimensions where corner-case are poorly covered by random sampling [3].

3.2 Meta-models

The results of a DoE experiment consist of a list of pairs of factor values and response values $\{x_i, y_i\}_{i=1..N}$ where N is the number of experiments. In some cases, it may be sufficient to verify that all y_i fulfill specifications, or to pick one x_i that give the desired outcome of the response. Often, however, we are interested in finding a factor combination x^* in a continuous set $\mathscr{X}$ that optimizes some criteria defined by x and y. That is, it is necessary to interpolate between the points in the test matrix. To do that, some regularity on the map from x to y is assumed.

One way to find x^* is to perform sequential DoE in refined factor ranges close to the expected optimum. To limit the number of required experiments, it is generally better to use the existing data to generate a *meta-model* of the mapping from x to y. A meta-model is a simple empirical model that is obtained by fitting the data to some generic model structure, e.g. using linear regression, splines, neural networks or gaussian processes. Meta-models may also be referred to as surrogate models or emulators.

The role of meta-models is to replace the real system with a simpler representation for analysis such as optimization or verification.

4 FMI tools for dynamical systems DoE

4.1 Modelica and FMI as a platform

Modelica has several attractive features as a platform for robust design of dynamical systems. Modelica component models normally have a sufficient level of detail to study influence of component variability, while still being simple enough to run large batches of experiments. From the nonlinear DAE models, information on the effect of both design parameters, operating conditions, and actuator settings can be investigated.

The FMI standard provides increased flexibility when model development and model analysis can be separated to different tools. The strengths of different tools and the skills engineering teams have developed using specific tools can be put to use more efficiently than if each tool needs to provide features for each potential modeling application. This facilitates cross-team use of the same model portfolio for different purposes.

MATLAB/Simulink is widely used as an environment for control design and implementation, and is thus a natural tool for analysis of model dynamics.

4.2 Tool requirements

The objective of a Modelica-based DoE tool is to remove the hurdles that make engineers resort to ad-hoc one-factor-at-a-time robustness testing.

The user wants to get a general sense of how sensitive the system is to parametric uncertainty. To do this, the user should be in charge of problem formulation, and provide the following input:

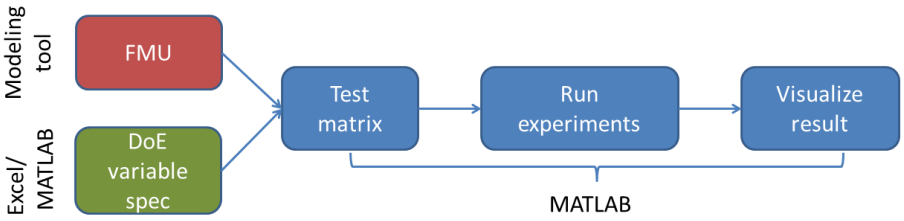

Figure 2: Workflow

- FMU model to be used
- FMU parameters, inputs, and outputs that are factors in the DoE
- Ranges or distributions for FMU parameters
- Number of experiments in the test matrix
- Type of DoE design
- Response variables to analyze or visualize

From this information, the tools should handle the back-end work:

- Construct a test matrix based on the desired DoE design and factor ranges or distributions
- Set FMU parameters
- Simulate the FMU at all points in test matrix, finding steady-state
- If the user specifies values for FMU outputs, find inputs that give these outputs
- Catching errors so that an entire batch of results is not waisted by a single test point that failed to simulate
- Linearizing the system at all points in the test matrix
- Provide convenient methods to visualize the results
- Construct suitable meta-models for analysis

4.3 DoE features in the FMI toolbox for MATLAB

The DoE tools in the FMI toolbox for MATLAB were introduced in release 1.6 with the purpose to make it easy and intuitive to perform DoE batch experiments on FMU models. The workflow is summarized in Fig. 2. The user specifies the names and distributions of DoE factors in either an Excel spreadsheet or a MATLAB file.

A class, `FMUDoESetup` holds the information on the FMU model to be simulated, the factor distributions, and simulation options. A batch of experiments is run by calling methods of `FMUDoESetup` that correspond to different DoE designs: full factorial, Sobol-sequence based, or Monte Carlo random simulation. There is also a possibility to use a custom test matrix. The DoE factors can be both FMU parameters, FMU inputs, or FMU outputs. If outputs are defined as factors, an equivalent number of inputs must be defined as 'free' inputs with min- max- and nominal values. These inputs are then optimized iteratively to obtain the specified outputs at each test point.

These methods return an object of the class `FMUDoEResult` that stores experiment results, including input-, output-, and state data, simulation status, and linearization at all test points. Data can be visualized by calling methods of the `FMUDoEResult` class to plot the main effects between the factors and reponses, or to generate Bode diagrams or step responses for the ensemble of linearized systems at all test points.

In the future, support for meta-models may be added. An interesting application of meta-models generated from the DoE is to provide simplified subsystem models that can be used as a substitute for the full Modelica subsystem model in simulations of larger systems.

DOI 10.3384/ECP1409635

5 An example

This section demonstrates a detailed example on how the tools are used in the design of an engine cooling system with feedback control. The design process consists of the following steps:

- Verifying feasibility of the suggested cooling loop architecture
- Sizing the components
- Verifying the design for worst-case heat load
- Designing a robust controller for the system
- Verifying the closed-loop dynamics

Note that the example is chosen to illustrate the tools and methodology; the architecture and parameter values do not directly correspond to any existing system.

5.1 Model and specs

We consider a cooling loop architecture where the coolant temperature is actively regulated through an electronically controlled coolant pump. Active control of coolant temperature is achieved through modulating the coolant pump speed, as opposed to the conventional cooling system architecture where a passive thermostat valve adjusts the coolant flow to maintain safe coolant temperature. Better control of coolant temperature can help the engine management system achieve better over-all fuel economy [9].

A Dymola model of the cooling loop is shown in Fig. 3. The model is an example model provided in the Modelon Liquid Cooling Modelica library. Inputs and outputs were added to the model (inputs: heat flow from engine, pump speed, and air mass flow through radiator, outputs: pump mass flow, liquid temperature drop over radiator, temperature of liquid after the engine, and temperature of the liquid at the pump entrance).

The following specifications are set for the system

- The engine-out coolant temperature $T_{coolant}$ must not exceed 100 °C (373.15 K).
- The system should handle a heat load of Q_{flow}= 100 kW
- The ambient temperature operating range is $T_{ambient} \in$ [-20°C, 45°C]

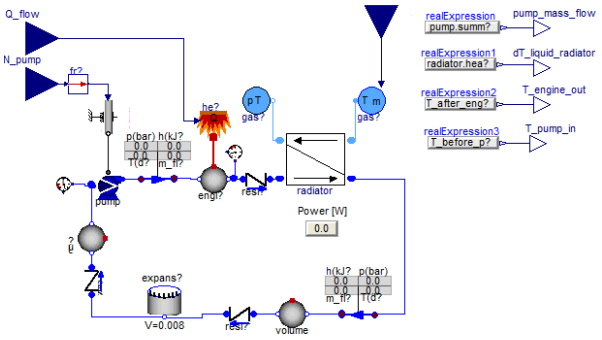

Figure 3: The cooling system model in Dymola

Three design parameters are considered: the maximum pump speed N_{pump}, the heat exchange efficiency η of the radiator, and the capacity of a fan that circulates air through the radiator when the ram air flow is not sufficient to provide coolin.

5.2 Sizing the system

The first step in the design process is to screen the design space to see if there exist some combination of parameters that meet the specifications. For this task, we look at the worst-case heat-load: maximum engine heat load and maximum ambient temperature. We let the three design parameters be factors in the DoE setup, and choose wide ranges for these factors to get a sense of feasibility of the design. The setup is summarized in Table 1.

Table 1: Experiment setup for screening the design space

variable	dist	min	max	value
Q_flow [W]	constant			1e5
T_ambient [K]	constant			318.15
N_pump [rpm]	uniform	50	2000	
mflow_gas [kg/s]	uniform	0.5	5	
efficiency	uniform	0.4	0.9	

We run a batch of 100 experiments with a Sobol QMC-design where the three factors are uniformly distributed between their maximum and minimum values in a cube. Figure 4 shows the resulting steady-state temperature plotted against each of the DoE factors. The green dots represent test points where the specification on engine-out coolant temperature is fulfilled, and the red dots represent test points that do not meet the specification. Not surprisingly, the test points that

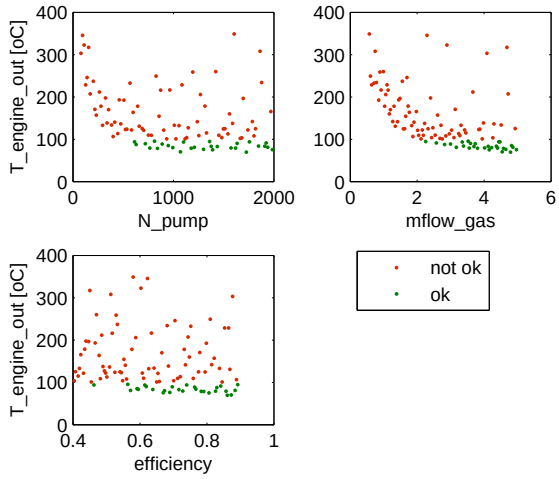

Figure 4: Result of screening design

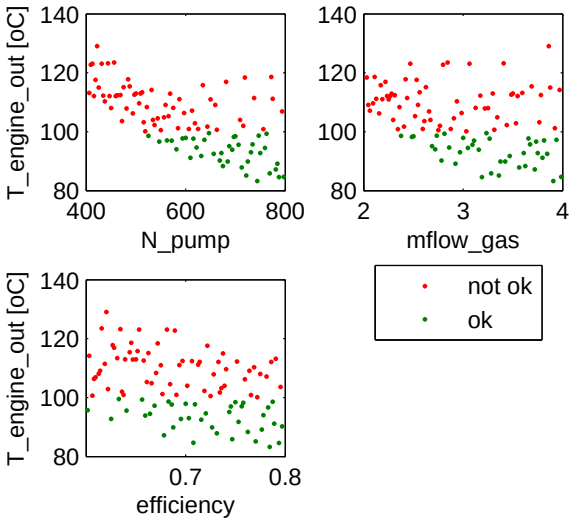

Figure 5: Result of sizing experiments

fulfill the specs have a high pump speed, a high radiator gas flow, and a high heat exchanger efficiency.

The next step is to zoom in the design variables to a narrower range to find appropriate component specifications that meet the subsystem specification on engine-out coolant temperature. The DoE setup for this sizing task is given in Table 2. A new set of uniformly distributed test points in the new factor ranges was generated, and the effects on coolant temperature are shown in Fig. 5.

Table 2: Experiment setup for sizing the system

variable	dist	min	max	value
Q_flow [W]	constant			1e5
T_ambient [K]	constant			318.15
N_pump [rpm]	uniform	800	1200	
mflow_gas [kg/s]	uniform	2	4	
efficiency	uniform	0.6	0.8	

From Fig. 5, we can now decide on a set of component specs on the pump and radiator that would meet subsystem specification. This would normally involve a trade-off between cost and availability of components. We will here choose the design

$$\begin{aligned} N_{pump}^{max} &= 650rpm \\ \eta &\geq 0.65 \\ m_{flow} &\geq 3kg/s \end{aligned} \tag{1}$$

At worst-case conditions in terms of heat load and ambient temperature, this design gives $T_{coolant} = 96.8°C$.

5.3 Dynamics

In the next step, we examine the dynamic response from pump speed command to engine-out coolant temperature. The DoE factor setup is given in Table 3. The factors were chosen to represent a range of operating conditions for the system.

Figure 6 shows the Bode diagram of the linearized systems at each point in the test matrix. It can be seen that the system is nonlinear: there is a large difference in the frequency response at different operating points.

We can now investigate the influence of the DoE factors on the dynamic response. Figure 7 shows the influence of the DoE factors on the steady-state gain of the linearized systems. The pump speed is the factor that has the largest influence on the steady-state gain.

5.4 Control design

The Bode diagram for the ensemble of linearizations in Fig. 6 can be used to design linear controllers that will have sufficient phase margins at all linearization points.

For nonlinear systems, there are no general guarantees that a controller that stabilizes each possible linearization will globally stabilize the closed-loop nonlinear system. In many cases, however, this is more of an academic concern. The alternative to looking at the ensemble of linearization may be to look at linearization in a single point. The ensemble of linearizations provides significantly more information.

A PI-controller with $K = -50$ and $T_i = 100$ was de-

DOI 10.3384/ECP1409635

Table 3: Experiment setup for evaluating the design

variable	dist	min	max	nominal
Q_flow [W]	free	0	1e5	1e4
T_ambient [K]	uniform	253.15	318.15	
N_pump [rpm]	uniform	20	650	
mflow_gas [kg/s]	uniform	3	10	
efficiency	uniform	0.65	0.8	
T_engine_out [K]	uniform	350	370	

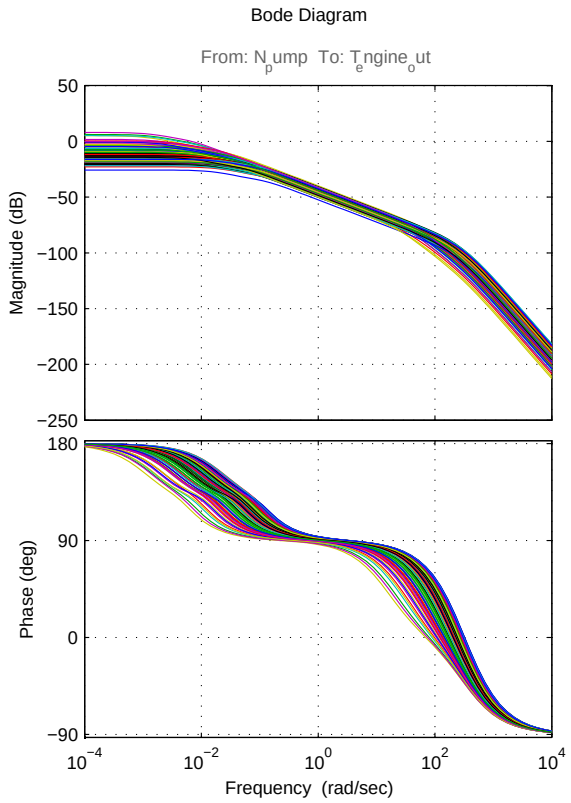

Figure 6: Ensemble Bode diagram showing the magnitude and phase for linearizations at all points in the test matrix.

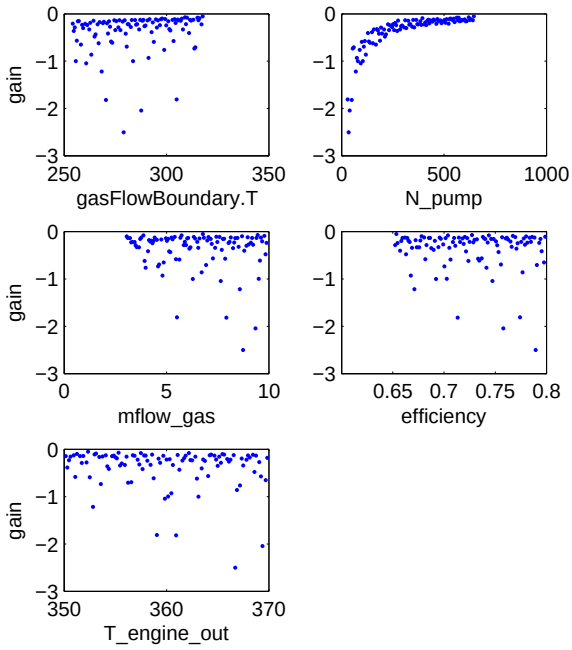

Figure 7: Steady-state gain from N_{pump} to $T_{coolant}$ of linearized system at all test points

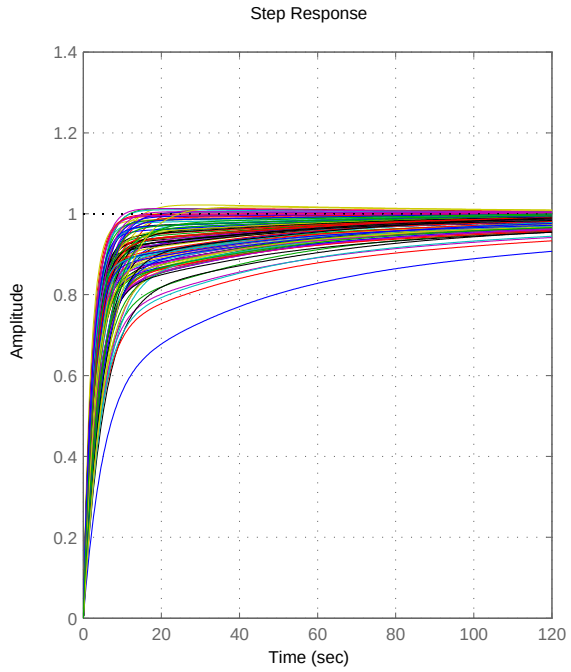

Figure 8: Step response of the closed-loop system

signed based on the ensemble Bode diagram.

Figure 8 shows the ensemble step response of the closed-loop systems corresponding to the linearizations at each point in the test matrix.

6 Conclusions

We have presented tools and workflows to apply robust design methods to dynamical models. The tools are available in the FMI Toolbox for MATLAB. The aim in developing these tools has been to provide methods to work with parametric uncertainty in dynamical models in a lightweight and intuitive manner.

Each of the analysis and design steps in the cooling loop example involve less than ten MATLAB commands. By taking care of back-end work, the tools allow engineers to get answers from models in a systematic fashion rather than ad-hoc testing.

It would be interesting to see an increased academic focus on design process, tools, and workflows for control design. In this context, Modelica and the FMI standard provide a powerful platform.

References

[1] D. W. Apley, J. Liu, and W. Chen. Understanding the effects of model uncertainty in robust design with computer experiments. *Journal of Mechanical Design*, 128:945–957, 2006.

[2] R. A. Bates, R. S. Kenett, D. M. Steinberg, and H. P. Wynn. Achieving robust design from computer simulations. *Quality Technology & Quantitative Management*, 3(2):161–177, 2006.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] D. Bouskela, A. Jardin, Z. Benjelloun-Touimi, P. Aronsson, and P. Fritzson. Modelling of uncertainties with Modelica. In *Proceedings of the 8th International Modelica Conference*, pages 673–685, 2011.

[5] C.M. Creveling, J.L. Slutsky, and D. Antis. *Design for Six Sigma*. Prentice Hall, Upper Saddle River, New Jersey, USA, 2003.

[6] B. Johansson and P. Krus. Probabilistic analysis and design optimization of Modelica models. In G. Schmitz, editor, *Proceedings of the 4th International Modelica Conference*, pages 247–254, 2005.

[7] P. N. Koch, R.-J. Yang, and L. Gu. Design for six sigma through robust optimization. *Structural and Multidisciplinary Optimization*, 26:235–248, 2004.

[8] K. Otto and K. Wood. *Product Design*. Prentice Hall, Upper Saddle River, New Jersey, USA, 2001.

[9] M. H. Salah, T. H. Mitchell, J. R. Wagner, and D. M. Dawson. A smart multiple-loop automotive cooling system—model, control , and experimental study. *IEEE/ASME Transactions on Mechatronics*, 15(1):117–124, 2010.

[10] C. Zang, M. I. Friswell, and J. E. Mottershead. A review of robust optimal design and its application in dynamics. *Computers & Structures*, 83:315–326, 2005.