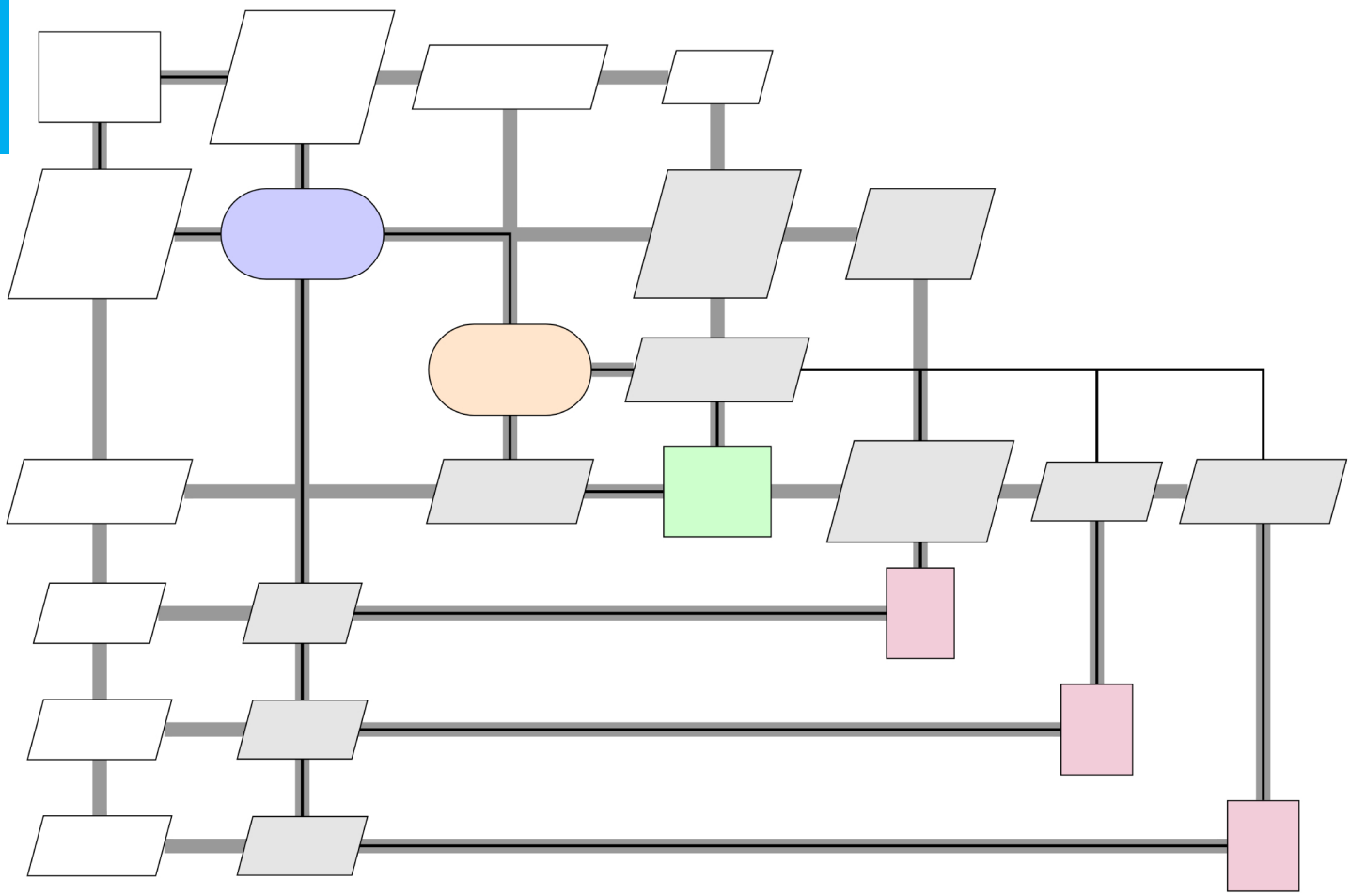
SAS: an advisory framework to support surrogate model based MDAO

C.L.A. de Priester

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

SAS: AN ADVISORY FRAMEWORK TO SUPPORT SURROGATE MODEL BASED MDAO

by

C.L.A. de Priester

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
Flight Performance and Propulsion department,
to be defended publicly on Thursday July 7, 2022 at 13:00.

Thesis committee:	Dr. ir. G. la Rocca	TU Delft, Supervisor
	Ir. M. Pini,	TU Delft, Examiner
	Dr.ir. M. Langelaar,	TU Delft, Examiner
	Dr. N.A.K. Doan,	TU Delft, Examiner

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

PREFACE

Dear reader,

After quite a few years of studying, I am proud to present to you the end result. This thesis marks the end of my time as a student, which has been something I have enjoyed to the fullest. When I arrived in Delft at the start of my studies, I had a hard time believing that I would graduate at a certain point. However, here we are!

For the last few years, I had the honor of studying at the faculty of Aerospace Engineering. The switch from Maritime Engineering to Aerospace Engineering was not always an easy one, but I have never regretted it once. A never-ending enthusiasm from lecturers, but also from fellow students, made the Aerospace journey a pleasure, and I am sad to leave the faculty behind.

First of all, I would like to thank my examination committee, and especially Gianfranco for all his efforts in this thesis. We have had many sessions, sparring about the subject and refining the ideas, and it was a true pleasure of working with you. I know your schedule is filled like no other, but still, you were always available for our productive meetings, which I always enjoyed. Thank you!

With regards to my time at Aerospace Engineering, I have to thank Godert. We both arrived at Aerospace from different backgrounds and managed to overcome the hurdles of adaptation to the physics of these strange, flying, objects. We managed to push each other to go that extra mile, and I think we delivered some great assignments over the years. For these last few weeks, I managed to find a similarly lost soul in the form of Pieter. While many of our friends were doing all kinds of things, but not working, we've spent endless hours in the library, trying to finish our work on time. Your endless supply of coffees was much appreciated... We are almost there!

And of course, this document would not have been laying in front of you without the support of my roommates, parents, brother, and Annelot. They've had to endure countless hours of being bothered with talk about simulations, surrogate models, and other, technical, and most likely, boring, stuff. Especially for my poor Annelot, coming home from a long day in the hospital... It is almost over!

All jokes aside, I am proud to present you this thesis. I hope you enjoy reading it!

C.L.A. de Priester
Delft, June 2022

SUMMARY

This thesis presents the Surrogate Advisory System (SAS), a surrogate-modeling advisory framework to be used in conjunction with Multidisciplinary Design Analysis and Optimization (MDAO) workflows. In MDAO, design procedures are automated by parameterization of a design in design variables, coupling of design competences, and a formulation to exploit these couplings to calculate one or more objective variables. MDAO has the potential to increase the quality of produced designs, due to its automated nature and inherently required synergism between design competences. However, widespread adoption of MDAO in the industry is not yet to be found, due to existing hurdles such as complexity when working with MDAO, or excessive computational costs of MDAO workflow execution. A continuously ongoing effort to reduce these barriers is ongoing, and this work has the goal to be contributing to these efforts.

Surrogate Models (SMs) are a proven method to reduce the computational costs of a workflow. An SM is a model that estimates a certain expensive black-box model, by sampling the black-box model at certain points, storing the produced outputs, and training a predictor on the acquired in- and output samples. A new data point can now be evaluated using the trained SM, which normally is significantly cheaper than the black-box model, reducing the required computational efforts for workflow execution. However, this reduced computational expense comes at the cost of reduced accuracy. This accuracy of the SM can be increased by increasing the number of sample points for which the black-box model is evaluated, leading to a trade-off. More, expensive, sample points require a greater (time) investment, leading to higher accuracy of the generated SM. Fewer sample points reduce the computational costs, but also decrease the models' accuracy. To provide insights into this trade-off, this work presents an advisory strategy, aiming to provide MDAO architects with an extra tool to help in the decision process for this trade-off.

Due to this trade-off, it is not a straightforward task to identify design competences, or a group of design competences, that are suitable to be replaced by an SM. Each different possibility to replace a design competence or group of design competences is called an SM strategy. In SAS, all possible SM strategies are determined, and the logic is implemented to provide advice on whether an SM strategy is a suitable candidate to be replaced with an SM.

To identify these suitable candidates, this work presents a methodology where a provided workflow is analyzed to find the most time-costly design competences. This is achieved by executing the provided workflow for a limited number of samples. The results of the run are analyzed, a timeline of the workflow execution is created, and based on this timeline the bottlenecks, the design competences that contribute the most to the execution time, are identified.

Expensive design competences are inherently a likely candidate to be replaced by an SM, but more information is needed to determine whether they are suited to be replaced. The leading factor is the required time investment to achieve a reasonably accurate SM. To find the relationship between the required investment for SM training and the expected accuracy of a to-be-generated SM, a relationship is used where the required number of samples for an accurate SM is assumed to be dependent on the input dimensions of the considered SM strategy. A larger number of input variables calls for a larger required number of samples, and a smaller number of input variables will require fewer samples to generate an accurate SM. Based on literature, three different estimations for this relationship have been implemented.

Based on the estimation of required samples for an SM strategy, as well as the available information from the workflow analysis, the required time investment needed to implement an SM strategy can now be calculated. Using an estimated required number of iterations for optimization, or estimated expected time for optimization, all potential SM strategies are evaluated and their impact on the total required execution time is determined. From here, the most promising SM strategies can be selected and implemented, ideally leading to a reduction in the computational costs of the workflow.

To further reduce the complexity when working with SMs in an MDAO setting, SAS automates most of the steps required in the SM building process. SAS is capable of converting existing MDAO formulations to a Design of Experiments (DoE) formulation, calculating the sampling plan for the DoE, selecting, building, and validating an SM, and deploying an SM to be used by external tools. The generated executable SMs are based on a Central Data Schema (CDS), and can therefore directly be used for applications using the CPACS standard.

The implemented methodology and developed software are verified using two test cases. An artificially expensive version of the Sellar problem, a standard test problem in MDAO, is analyzed and several SM strategies have been implemented. SAS is shown to perform well for the Sellar problem, confirming the usability of the advisory module and SMs in general.

Secondly, the Maximum Take-Off Mass (MTOM) optimization problem is implemented, and again, the workflow is analyzed and the optimal SM strategies are identified. Three SM strategies are implemented, again showing a potential reduction of computational expense when using SMs. However, an important limitation of the advisory capabilities of SAS is exposed. Due to the difference in nature of a DoE and optimization, and the potentially drastically varying runtimes of design competences for different inputs, the estimations of the required time investment for an SM strategy lack accuracy. The discrepancies are identified to be fueled by 1) different convergence behavior when executing a DoE compared to an optimization and 2) a drastically varying runtime for one of the design competences, based on their inputs.

List of Figures

NOMENCLATURE

ϵ	=	Twist angle
$\mathscr{E}$	=	Error measure
Λ	=	Sweep angle
$\sigma_{t_{exec,i}}$	=	Standard deviation of execution time for discipline i
b	=	Wing span
c	=	Wing chord
C_D	=	Drag coefficient
C_L	=	Lift coefficient
D	=	Drag
L	=	Lift
$\bar{n}_{iter,i}$	=	Average number of iterations for discipline i
N_s	=	Number of samples
$\tilde{t}_{exec,i}$	=	Normalized execution time for discipline i
$\bar{t}_{exec,i}$	=	Mean execution time for discipline i
$\bar{t}_s$ [s]	=	Mean execution time per DoE sample
R	=	Range
W	=	Weight
$\mathbf{x}$	=	design vector

CONTENTS

1

INTRODUCTION

It is well known that Multidisciplinary Design Analysis and Optimization (MDAO) is regarded as a significant opportunity to increase the efficiency of design processes and their produced designs. A changing environment, both in the professional and in the literal sense, forces aircraft manufacturers to set ambitious requirements to reduce emissions, noise and the general environmental footprint of their produced aircraft. As MDAO promises an increase in performance between 8% and 50% for innovative and radically new designs, respectively, MDAO could prove to be very helpful in the process of reaching these goals. [6, 7].

Contrary to the promising perspective, MDAO is currently not being widely adopted by the industry. Both technical and non-technical barriers prevent this adoption [8, 9], but significant efforts are being made to lower these existing hurdles. A major hurdle is the complexity of working with MDAO systems, and ongoing efforts continuously reduce this barrier. The introduction of the eXtended Design Structure Matrix (XDSM) by Lambe and Martins [10] has created a de-facto standard in the visualization of MDAO systems. The XDSM enabled the clear visualization of a problem formulation, solution strategy, process order and couplings. As it can be expected that the MDAO formulation is iterated and improved during a full MDAO lifecycle process [11–13], the XDSM proves to be an essential tool in the efficient communication of MDAO systems between architects.

These expected iterations in the MDAO problem formulation can be cumbersome if needed to be done manually, as it is not uncommon for MDAO systems to exist of thousands of coupling variables, and a large number of design competencies and constraints. Hoogreef [14] proposed InFoRMA, a framework that advises the users of a suitable MDAO architecture and enabled the modeling of MDAO systems using interactive DSM or N^2 charts [15, 16]. Van Gent introduced KADMOS [2, 17], which implements automated MDAO system formulations based on a (directed) graph-theoretic approach and a central data scheme (CDS) such as CPACS [18]. For the interactive visualization of designed MDAO systems, VISTOMS [19] is used, and the workflows are stored in the machine-interpretable format CMDOWS [3]. Where KADMOS is mainly focused on providing automated workflow formulation, the MDAx package proposed by Page Risueño et al. [20] aims to produce MDAO workflows using a graphical user interface (GUI). MDAx also generates workflows based on the CMDOWS format but leaves the burden of formulating a workflow largely to the user.

Another aspect hindering the widespread adoption of MDAO is an excessive computational expense of workflow execution. As workflows tend to be big, and individual design competences can be very expensive to execute, the time required to execute an MDAO workflow can be a prohibitive factor for optimization. Bruggeman [21] implemented several decomposition and sequencing algorithms into KADMOS, leading to a significant decrease in the setup time due to the automated sequencing, and a reduction in the execution times of MDAO systems.

Another feasible approach to reducing the computational costs of a workflow is to implement a surrogate model (SM) strategy. An SM, or *metamodel*, *model of model* or *response surface model*, is used to create a cheap, analytical model that estimates a more expensive model. A typical, non-adaptive generation workflow of a surrogate model is shown in Figure 1.1. Based on a Design of Experiments (DoE), which in recent literature is most often a Latin Hypercube Sampling (LHS) plan, the full model is executed and a certain SM type is fitted to the samples.

Many SM types are available, such as linear regression, Radial Basis Functions (RBF), artificial neural networks (ANN) and (different types of) Kriging [22]. In recent literature especially Kriging, which models

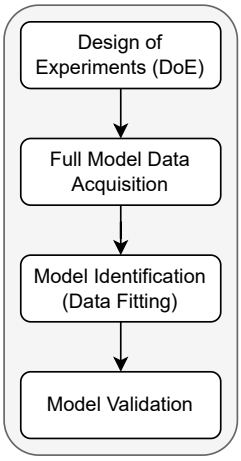


Figure 1.1: The typical lifecycle of a surrogate model [1].

an underlying spatial correlation between design variables, is a popular model choice due to its excellent predictive and inherent variance estimation capabilities. Bouhlel et al. [23] packaged many of the available SMs in the *Surrogate Modeling Toolbox* (SMT), a Python-based package providing the functionality to build DoE's and build SMs.

The usage of SMs in MDAO workflows has been researched extensively, and three main categories emerge from the literature. First, *global surrogate-based MDAO* strategies replace a full MDAO workflow with a Kriging type SM and exploit Kriging's variance predicting capabilities to find an optimum [5, 24]. The strategy shows a significant reduction in computational expense but misses flexibility. As soon as a single element in the workflow changes, the complete SM lifecycle needs to be restarted as the underlying model has changed. Secondly, *adaptive surrogate-based MDAO* is a strategy where every *individual* design competence is replaced with a Kriging-based SM, and variance prediction combined with error propagation formulations are applied to calculate infill points [25, 26].

The third strategy is *non- or semi-adaptive surrogate-based MDAO*. Here, an SM of one or more design competences are generated a-priori and the optimization is executed as normal using the SM-based workflow. Examples are found in the work of Lefebvre et al. [27], who replace clusters of disciplines with SMs and in the work of Wang et al. [28], who use SMs to replace an expensive CFD simulation of a battery thermal management system. Paiva et al. add an exploitation-based infill strategy, where the SM is enhanced around the found SM-based optimum [29]. In their work, a 50% reduction in execution time with respect to a non-SM-based execution, is achieved, while the lost accuracy is minimal.

As such, SM-based MDAO is a proven strategy for the reduction of the computational costs of an MDAO workflow. However, as opposed to the existing automated workflow formulation systems, there is currently no straightforward way to efficiently implement and prototype CDS- and SM-based MDAO strategies. It requires a significant number of manual steps, and often custom software will have to be developed. Furthermore, it is not a trivial task to determine whether a certain design competence, or group of design competences in a workflow is suited to be replaced with an SM.

An opportunity for the removal of another hurdle in the widespread adoption for MDAO systems can therefore be identified. The automated workflow formulation systems provide the opportunity to automatically build and deploy DoE's, the widely adopted CDS paradigm allows for a standardized way of storing data, the CMDOWS format enables the creation of machine-interpretable workflows and SMT presents a very useful toolbox for the generation of sampling schemes and SMs. Therefore, this thesis aims to answer the following research question:

> *How can the current state-of-the-art in MDAO be extended with a framework that automates the surrogate model integration process and that provides insights into the trade-off between accuracy and computational costs for a given surrogate-based MDAO strategy?*

From the research question, the following subquestions are derived:

- How can the lifecycle of an SM-based MDAO system, among which falls the acquisition of samples, training and validation of the model, and deployment of the SM in an MDAO workflow, be automated?

- How can a strategy be formulated that identifies bottlenecks in an MDAO workflow?
- What metrics are suitable to quantify the trade-off between computational efficiency and objective function accuracy when a certain discipline, or group of disciplines, would be replaced by a surrogate model?
- How can the set of metrics and the identification of bottlenecks be summarized into an advise to the end-user on which disciplines are suited to be replaced by a surrogate model?

In this paper, a fully inclusive surrogate modelling framework for MDAO applications, called the Surrogate Advisory System (SAS), is introduced. The presented methodology aims to enable MDAO architects to quickly prototype, validate and integrate SM configurations in their MDAO workflows. The presented software has capabilities 1) to investigate provided MDAO workflows for SM candidates and provide the user with a clear selection of SM strategies and their implications, 2) to provide advise on the number of samples and overall Design of Experiments (DoE) strategy, 3) to execute the selected DoE strategy in the PIDO, and process the samples to a centralized database, 4) to select, validate and deploy a surrogate model, 5) to execute an optimization, assess the accuracy of the final result and apply an exploitation infill strategy if desired.

I

STATE OF THE ART

2

Multidisciplinary Design Analysis and Optimization (MDAO)

Multidisciplinary Design Analysis and Optimization (MDAO) is the governing subject of this thesis. Therefore, a short introduction to the concept of MDAO is provided in section 2.1. In that section, the main principles of MDAO and some common terminology are explained, as well as the concept of the MDAO lifecycle. An automated MDAO workflow formulation tool called KADMOS is extensively used in this work. As its working principles are of importance for later discussed methodology, KADMOS, and its implemented graph-based approach are discussed in section 2.2.

2.1. Introduction to MDAO

Sobieszczanski-Sobieski and Haftka [30] define Multidisciplinary Design Analysis and Optimization (MDAO) as a *methodology for the design of systems in which strong interaction between disciplines motivates designers to simultaneously manipulate variables in several disciplines.* Sobieszczanski-Sobieski [31] states that the MDAO methodology should exploit the state of the art in all the engineering fields which contribute to the engineering system while emphasizing the synergism of the subsystems which comprise the engineering system. He describes these engineering systems as having a single common characteristic, namely that in these designs, *everything influences everything else.* In other words, the performance of a multidisciplinary system is driven by the performance of the interactions between the individual disciplines, instead of by each discipline's standalone performance [32].

Gentle introduction to MDAO

This paragraph contains a gentle introduction to the concept of MDAO for readers new to the subject and can be skipped if the reader has a basic level of knowledge of MDAO. To provide an analogy, these engineering subsystems, or *design competences* can be viewed as departments in a large company that designs certain engineering solutions. For example, an aircraft manufacturer could consist of an aerodynamics department, responsible for the aerodynamic analysis of an aircraft, and a structural department, responsible for the analysis of the structural properties. For a certain design, as decided by another department, both the aerodynamic and structural departments will analyze the design and come up with certain results, such as the lift and drag of a wing, or the required structural weight for a design. It can be expected that the aerodynamic department will need certain data from the structural department and vice versa. In MDAO, the variables containing this data are called *coupling variables.* It can be expected that there will be a certain amount of iterations required to align the structural and aerodynamics department before a design can be considered valid. Hence, the structural and aerodynamics departments are considered to be *coupled*, and their results need to be *converged.*

Now, if we parameterize a certain design, so, for example, we define a methodology where we can define an aircraft based on its length, width, airfoil, etc., we have created a *design vector.* These parameters act as 'knobs' that can be turned, in order to modify a design. The parameters are expected to stay within certain bounds, and all possible combinations of the design variables are called the *design space.* Eventually, the goal is to create an aircraft with certain favorable performance characteristics. For example, we would like to

create an aircraft that uses the least amount of fuel possible for a range of X nm. In this case, the required fuel for such a journey would become the *objective variable*.

MDAO LIFECYCLE

A typical MDAO lifecycle as defined by Van Gent and La Rocca [2] is shown in Figure 2.1. The MDAO lifecycle is separated into two distinct phases: a formulation phase and an execution phase. In the formulation phase, a workflow is formulated starting from a database, or repository, of available design competences. Based on the in- and output variables of these design competences, they can be coupled using *coupling variables* and an automated design routine can be created by formulating an MDAO problem formulation and applying an MDAO solution strategy. In the MDAO problem formulation, the *design variables*, *objective variable(s)*, *constraints* and *Quantitities of Interest* (QOIs) are defined. The MDAO workflow should be defined in such a way, that based on the design variables, the values of the objective variables, constraints, and QOIs can be calculated. For this purpose, an MDAO solution strategy is applied, which is the approach to handling convergence loops, tool orders, and potential conflicts in the generation of the variables.

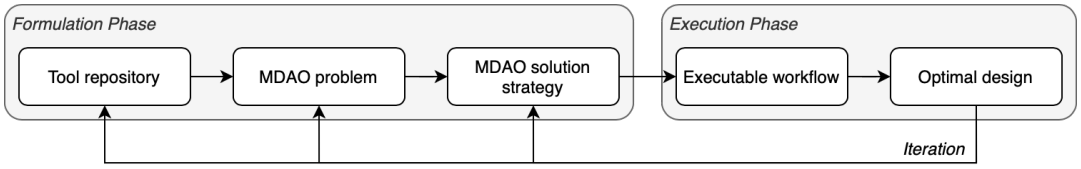


Figure 2.1: Lifecycle of an MDAO development cycle as depicted by Van Gent and La Rocca [2]. The *formulation phase* spans the initial definition of the discipline analysis models or tools to a fully defined MDAO solution strategy. In the *execution phase* the strategy is interpreted as a workflow and executed. This generates an optimal design, which might lead to an iteration of the formulation phase as new insights have been created. The transmission between the formulation phase and the execution phase is either a manual process or a standardized format such as CMDOWS [3] can be used.

Eventually, an MDAO formulation is defined which is ready to be executed. The actual execution of the workflow occurs in the *execution phase* of the workflow, as displayed in Figure 2.1. Depending on the workflow formulation, which can either be an analysis, optimization, or DoE, eventually, the desired result is obtained by the execution of the PIDO. Based on this result, it often occurs that it becomes apparent that changes are needed in the MDAO workflow formulation, making the MDAO lifecycle an iterative process. As MDAO problems can be very large, consisting of many design competences and even more coupling- and state variables, this process can be cumbersome, and is seen as a hurdle to the adoption of MDAO.

2.2. KADMOS

KADMOS is an automated MDAO workflow formulation tool, originally aimed to reduce existing hurdles that existed when working with MDAO systems [2, 17]. It can be expected that MDAO workflows will have to be reconfigured multiple times, and as these workflows can become very large (thousands of coupling variables are not uncommon [12, 13]), manual configurations of workflow can become very time-consuming. KADMOS helps in this process, as it automates most of the normally manual steps, and drastically reduces the time required for workflow (re)-configurations.

KADMOS and its generated MDAO workflow formulations make use of a Central Data Schema (CDS), of which an example is CPACS [18]. A comparison between a CDS and non-CDS is displayed in Figure 2.2. In a CDS, all design competences communicate using a central, pre-defined data schema. All design competences, therefore, work with a single file as input and produce a single file as output. This drastically reduces

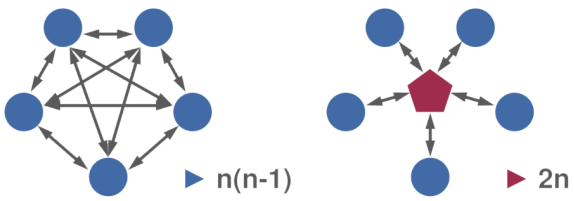
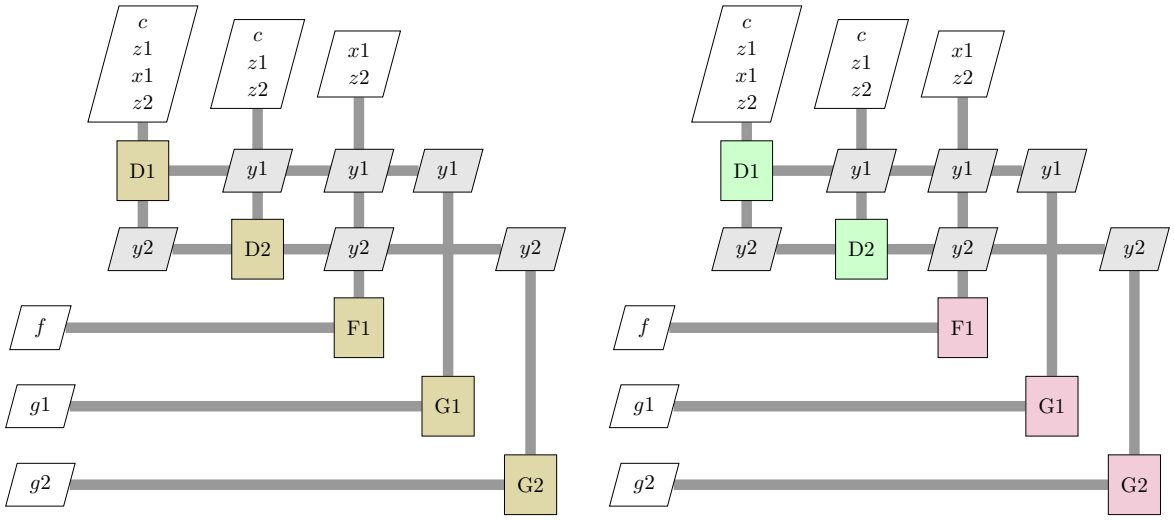


Figure 2.2: Comparison of the connections between tools with and without using a CDS [4].

the number of required interfaces between design competences, but requires the standardization of units and data formats before building the workflow. However, once these conventions have been made, a new design competence can easily be added as no complex interfaces between design competences will have to be remapped.

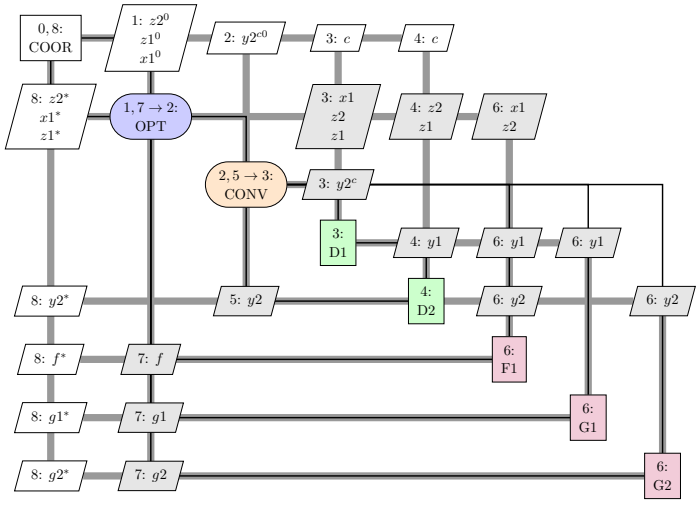
In KADMOS, workflows are represented by directed graphs. Variables and design competences are depicted by nodes, and edges represent the connections between these nodes. Hence, a workflow is represented by the graph $G = (V, E)$, where V are the nodes and E are the edges in the graph.

The first graph that is created in KADMOS, is the Repository Connectivity Graph (RCG). The RCG represents all the design competences available in the tool repository and shows all couplings between design competences due to their common in- and output variables. An example of an RCG is visualized in Figure 2.3a.



(a) Repository Connectivity Graph (RCG)

(b) Fundamental Problem Graph (FPG)



(c) MDAO Data Graph (MDG)

Figure 2.3: XDSM representation of KADMOS generated graphs.

Based on the RCG, the Fundamental Problem Graph (FPG) can be created. In the FPG, the aforementioned problem formulation is defined; variables either become a design variable, a coupling or state variable, an objective variable, or a constraint variable. Furthermore, in the FPG the selection is made on the design competences to use, and non-used design competences are removed from the graph. Based on the selection of design competences and problem formulation, decomposition and sequencing algorithms are applied to find the design competence order where the least number of iterations are required to find a so-

lution to the problem, as researched and implemented by Bruggeman [21]. These are essential to reduce the computational expense, or in other words, execution time, for an MDAO workflow. Lastly, in the FPG generation, problematic variables are resolved. Examples of problematic variables are variables where two design competences produce the same variable, or when a design competence both uses a variable as input and produces the variable as the output. Dealing with these problematic variables entails the manipulation of the graph, and often variables are split up into multiple instances of the same variable. For a reference on the different possible types of problematic variables, and their solution strategy, the reader is recommended to read the paper by Van Gent et al. [2], where an in-depth explanation of the strategies are presented. An example of a finalized FPG is shown in Figure 2.3b.

To create an executable workflow for the formulated MDAO problem definition, an *MDAO architecture* needs to be applied. Examples of these MDAO architecture are the *Multidisciplinary Design Feasble* (MDF) and *Individual Design Feasable* (IDF) schemes. In an MDF architecture, the optimizer is purely responsible for the design variables, and the solution is internally converged for each optimization iteration using an MDA [33]. For the MDA, a converging strategy can be selected, and typically a fixed-point iteration such as the block Gauss-Seidel or Jacobi [34] is used [32]. This inclusion of an MDA means that for each iteration of the optimizer, a feasible design will be created. Contrary, in an IDF the optimizer extends its responsibility to include the copy variables in the design vector and evaluates the accuracy of the coupling variables using constraints. This means that an MDA is no longer necessary, removing the sequential nature of MDF and enabling full parallel execution of the individual design competences.

Using the selected solution strategy, and corresponding *drivers* such as optimizers, convergers and coordinators, the FPG is transformed to a *MDAO data graph* (MDG) and *MDAO process graph* (MPG). These graphs contain all the required information required for the execution of the MDAO workflow formulation, and an example of a produced MDG and MPG is found in Figure 2.3c.

2.3. CMDOWS

Along with KADMOS, van Gent et al. [3] introduced CMDOWS. CMDOWS is an XML-based schema that aims to enable the description of a computational system, tool repository, and data flow in a standardized way so that any MDAO system can be fully defined in a computer-interpretable format. The schema enables the storage of the tool repository, MDAO problem formulation, MDAO solution strategy, and full workflow formulation. Due to its standardized format, it is platform-independent and new software can be developed for the CMDOWS format in a straightforward manner. It is supported by multiple Process Integration and Design Optimization (PIDO) tools, such as RCE, Optimus, and OpenMDAO in combination with OpenLego. It is considered to be the current state of the art in MDAO workflow storage and is therefore adopted in this work. A workflow is expected to be supplied to SAS in the CMDOWS format, and SAS will produce new workflow formulations in the CMDOWS format as well.

2.4. PROCESS INTEGRATION AND DESIGN OPTIMIZATION (PIDO)

The non-executable, but fully defined, workflow produced in the previous step consists of a specific order of discipline analyses, *drivers* such as optimizers and convergers, constraints, and objective functions. As discussed, in the current state of the art, it can be expected that the workflow formulation is stored in the CMDOWS format.

To execute this non-executable workflow formulation, a program is needed that couples all the variables in the prescribed manner, apply the optimizer and convergence loops, and evaluates the constraints and objective functions. This can be done manually in for example Matlab or Python, but as is discussed earlier, MDAO systems can get very large, very quick.

For this purpose, Process Integration and Design Optimization (PIDO) platforms have been developed. These platforms enable MDAO architects to construct MDAO systems using a user interface (UI), and perform analyses and optimizations using their designed systems. The design competences are 'imported' into the PIDO tools, and often different kinds of optimization algorithms are readily included in the package. For many PIDO toolboxes (RCE, Optimus, OpenMDAO), functionality to load a workflow using the CMDOWS format is also available, which means the 'manual' creation of a workflow using the GUI is no longer necessary, which means a significant amount of manual labor is removed.

The RCE [1] environment is an open-source package developed by DLR [35]. The package tightly integrates

[1] https://rcenvironment.de/

with CPACS and has built-in visualization capabilities for the data schema. RCE is specifically focused on distributed workflow execution, which means that different design competences, or clusters in the workflow, are executed in different locations on a (user-created) server network. RCE includes components such as optimizers, DoE, convergers, and the support to import and execute CMDOWS workflows. Although an extensive graphical user interface is provided in the package, it is possible to execute and run RCE workflows from the command line, which is necessary to support the generation of a DoE and automatic creation of surrogate models in SAS.

Another open-source PIDO platform is found in OpenMDAO [2], which has been developed by NASA [36]. OpenMDAO is fully Python-based and does not provide a GUI to build workflows; the systems have to be built up by code. OpenMDAO provides support for advanced convergers, as well as advanced methods for derivative computations. Support for CMDOWS workflow definition is achieved by using the OpenLEGO plugin, built by De Vries and Van Gent [37].

Gallard et al. [38] have proposed GEMS, a non-commercial Python-based package that focuses on the inclusion of multilevel architectures, as well as providing a standardized way to include disciplines analyses using a Python wrapper or abstract base class for direct integration. No centralized data schema is supported, and support for CMDOWS is missing, as well as detailed information about the implemented algorithms.

Commercial PIDO environments are also widely available, such as Optimus [3] by Noesis Solutions. Optimus provides a comprehensive GUI, as well as a Python API. Furthermore, it provides a package to generate (adaptive) DoE's and build surrogate models. The generation of workflows using CMDOWS files is supported as well. Other commercial PIDO packages include Isight [4], ModelCenter Integrate [5] and HEEDS [6]. All of the commercial packages have been used in MDAO studies.

For this project, it is chosen to implement support for the RCE PIDO. RCE provides all the necessary functionality for the tool, is free to use, and supports the CMDOWS and CDS/CPACS-based design competences.

[2] https://openmdao.org/
[3] https://www.noesissolutions.com/our-products/optimus
[4] https://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/
[5] https://www.phoenix-int.com/product/modelcenter-integrate/
[6] https://www.plm.automation.siemens.com/global/en/products/simcenter/simcenter-heeds.html

3

SURROGATE MODELS

The complexity of the systems and disciplines under analysis within a specific MDAO system is often too high to make use of ready-to-use analytical models, making simulation-based analyses a necessity [39]. These simulations, or high-fidelity models, are typically computationally expensive and time-consuming [40]. The time needed for the execution of these individual analyses can hinder an MDAO system when searching for an optimal design, as the resources needed to explore a large design space are not available.

Surrogate Models (SMs), which are also called *metamodels, models of models* or *response surface models* [41], are used to replace or approximate computationally expensive simulation models [42]. The usage of a surrogate model can be necessary for many situations, most commonly when the amount of necessary evaluations of a high-complexity model makes the execution of a larger system infeasible due to extensive computational costs. As a surrogate model is an analytical approximation of the highly complex 'parent' model, its runtime will be (close to) negligible.

Certainly in MDO systems, where a large number of model executions are needed to either converge a design or find an optimum, the use of surrogate models has been proven to be useful [39, 42]. It enables the optimization of systems that were too complex to converge with only full-complexity disciplinary analyses, as is proven by applications in wing and nacelle design [29, 43, 44], spaceflight [45] and shipbuilding [46].

Generating a surrogate counterpart for a model consists of four steps [1], which are displayed in Figure 3.1. First, a Design of Experiments (DoE) needs to be determined. In the DoE, points within the design space are selected to evaluate the high fidelity model. A DoE aims to capture as much information as possible, with the least amount of points. As these points will be used to evaluate the high-fidelity model, of which in turn the results will be used to train the surrogate model, it is of high influence on the final surrogate model. In section 3.1 different strategies for DoE generation will be discussed.

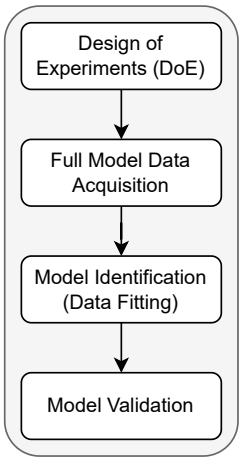


Figure 3.1: The four sequential steps to generate a surrogate model [1].

After the execution of the high-fidelity model on the selected points in the design space, a database of

Label	A	B	C
(1)	-1	-1	-1
a	1	-1	-1
b	-1	1	-1
ab	1	1	-1
c	-1	-1	1
ac	1	-1	1
bc	-1	1	1
abc	1	1	1

Table 3.1: Overview of a two-level or 2^k design. This DoE technique can be used to examine the effects and interactions between design variables. For this example, the design vector x consists of variables A, B, and C. In this table, -1 means the low value of the variable, 1 is the high value. The schematic overview of this design scheme can be found in Figure 3.2. Please note the orthogonality of the above matrix,

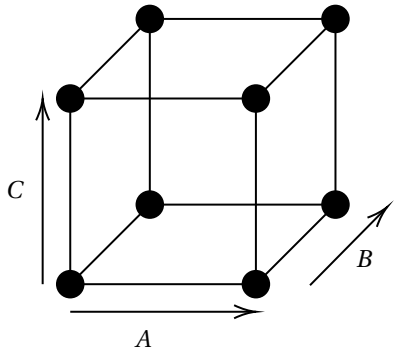


Figure 3.2: Schematic display of a full factorial 2^k experiment design for a three-dimensional design space. The corresponding sampling plan of this figure can be found in Figure 3.1

high-fidelity response data has been obtained. As mentioned before, the goal of the surrogate model is to find an analytical relation that approximates the mapping between the input and output of the high-fidelity model. Many different types of models have been developed for this mapping, for which a review is presented in section 3.2.

3.1. DESIGN OF EXPERIMENTS (DOE)

The first step in the life cycle of an SM is to generate a DoE. In short, the DoE is a set of input points for which the full model will be evaluated. The full model's responses will be used to train the surrogate model. The DoE, also called a sampling plan, has the goal to extract the maximum amount of information from the full model, with an as little number of model evaluations as possible [39]. In the case of a surrogate-based MDAO problem, which will be further discussed in chapter 4, this means that the design space needs to be sampled in such a way, that the surrogate model represents the most areas and features of the design space as possible. This, however, will always be a trade-off between the number of samples and the number of information that can be extracted from these points [39]. In this section, a small selection of available DoE techniques will be discussed. As Yondo et al. [22] recommend, it is preferable to use *adaptive sampling*. In this process, after a surrogate model has been generated using a static DoE, its accuracy is assessed. On the least accurate parts of the surrogate, samples will be added, and the process will be repeated.

3.1.1. FULL OR FRACTIONAL FACTORIAL SAMPLING

Full or fractional factorial sampling is a so-called *classical* DoE technique, which means the random error is expected to be included in the samples as the technique was originally formulated for real-world experiments[39]. In full factorial sampling, all possible combinations of the different discrete design variables are sampled and investigated during each replication of the experiment [22, 39, 47]. In the case of continuous design variables, variables can be discretized which enables the usage of this method.

A strategy that can be used to investigate effects and interactions between design variables, is the two-level factorial design or 2^k design [47]. In this method, each design variable gets a low and high value, and every combination of the design variables is included in the DoE. As the name suggests, a design space with k dimensions needs 2^k samples for this method. An important note to make is that to use the 2^k design, is that the response of the high-fidelity model or experiment should be assumed to be approximately linear over the selected range for the design variables [22]. An example of a full factorial 2^k scheme, $k = 3$, is found in Figure 3.1 and Figure 3.2. This example shows a three-dimensional design space, where each design variable is sampled at both a low (-1) or a high (1) value.

If quadratic effects and interactions between variables need to be mapped, it is necessary to use a three-level design, or 3^k design [47]. Similar to the 2^k method, the variables are divided into three levels. Exceeding three levels is rare, as the generation of such designs is cumbersome and inefficient at retrieving the (complex) underlying mapping from the design variables to the experimental outcome [22].

However, it becomes very clear that for larger design spaces, i.e. Yondo et al. [22] state for more than four dimensions, full factorial designs get very large, very quickly; the aforementioned *curse of dimensionality* start to cause problems. An attempt was made by Finney [48] to solve this problem, as he introduced so-called fractional factorial sampling. The goal is to have a minimum number of experiments (or high-fidelity model executions) but extract a sufficiently well description of the effects of the design variables on the outcome. When the assumption is made that some higher-order interactions are not significant, this goal is achievable when only using a *fraction* of the factorial design, hence the name.

3.1.2. RANDOM SAMPLING

Random sampling [22], or Psuedo-Monte Carlo sampling [49], could be considered to be the most simple form of creating a sampling plan. For a design space with bounds $[x_L, x_U]^j$ for dimension j, one can create a random sampling plan by picking N_s samples. For every sample, and, in turn, for each dimension, a normal distributed random value is picked. This collection of random picks for n dimensions and sample index i then forms $\mathbf{x_i}$.

Random sampling can be considered to be a flexible method, and it can deal efficiently with non-rectangular design spaces [49] and other complex systems [22]. However, care should be taken when using random sampling on a design space that is yet to be explored, as it is well possible that certain areas of the design space are left 'unvisited'.

3.1.3. LATIN HYPERCUBE SAMPLING (LHS)

McKay et al. [50] have published the first work describing Latin Hypercube Sampling (LHS) in 1979. In LHS, each input variable x_k has all portions of its distribution represented are represented in the DoE.

To construct a DoE with a sample size of N_s, each input variable x_k is divided in N_s bins or *strata*. With n design variables, this means that N_s^n bins now exist in the total design space. Now, if every bin is filled with a datapoint exactly once, an LHS representation of the DoE has been generated.

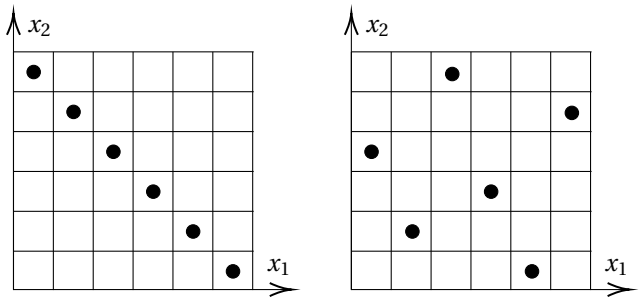


Figure 3.3: Two examples of LHS samplings for a two-dimensional design space with $N_s = 6$. Although both left and right DoEs both fulfill the requirements for an LHS design, the quality of both sampling plans differs greatly. The left sampling has a large area of 'undiscovered' design space, while the right DoE covers a much larger area.

Two examples of LHS sampling plans can be found in Figure 3.3. In this figure, two sampling plans for the same two-dimensional input/design vector $\mathbf{x} = [x_1, x_2]^T$ are shown. Both plans are fully compliant with LHS plans, as the projection of the dimensions x_1 and x_2 are uniformly distributed over all strata. However, the observant reader will notice that the quality of both DoE differs greatly. While one results in a very limited 'view' of the design space, the other covers a much larger portion of the design space. In addition to this, the diagonal arrangement will cause a high spatial correlation, which can lead to ill-conditioned systems of linear equations [49].

This shows a potential pitfall with LHS, as the performance of different LHS schemes can vary largely. This potential lack of uniformity over the design space has been addressed by multiple researchers and has led to multiple improved LHS algorithms. Most of these adjust LHS to include optimization of a certain metric. Examples of such metrics are the minimum distance among design points and correlation among the samples [40].

Hence, the quality of LHS can be improved when it is considered to be an optimization problem itself [39]. An example approach for this has been given by Leary et al., as they introduce the *Optimal orthogonal-array-based LHS*. In their approach, they minimize Equation 3.1, where d_{ij} is the distance between samples i and j [51].

$$d_{LHS} = \sum_{i=1}^{N_s} \sum_{j=i+1}^{N_s} d_{ij}^{-2} \tag{3.1}$$

The algorithm for a randomly generated LHS can be denoted as

$$x_j^{(i)} = \frac{\pi_j^{(i)} + U_j^{(i)}}{N_s} \tag{3.2}$$

for $1 \leq j \leq n$ and $1 \leq i \leq N_s$, where index j denotes the dimension, while i indexes the sample number. Let $U_j^{(i)}$ be a uniform random variable on $[0,1]$ and let π_j be a random permutation of the set $\{0, 1..., N_s - 1\}$[49]. Hence, $\pi_j^{(i)}$ is the item for sample i in the permuted set π_j. The algorithm can be interpreted as follows: for N_s samples a value will be chosen in a certain bin, which is 'selected' by $\pi_j^{(i)}$. Then, within this bin, a location is determined by the value of $U_j^{(i)}$.

3.2. MODEL TYPES

To approximate the black-box model, a surrogate model type needs to be selected. Over the years, Queipo et al. [40], Wang et al. [28], Forrester et al. [52], Koziel et al. [39], Yondo et al. [22] and Alizedah et al. [42] have written review papers on the subject, which shows the livelihood of the research area. As it is not the goal of this review to give an exhaustive review of available surrogate models, this section will highlight a few popular methods to create surrogate models. For a detailed review of surrogate modeling techniques, the reader is referred to one of the aforementioned papers, where especially Yondo et al. [22] have provided a thorough and recent overview of available methods.

3.2.1. POLYNOMIAL REGRESSION

The most simple form of surrogate modeling, and one that has been extensively used over time, is the usage of Linear Regression Models, also called polynomial Response Surface Models [52] or Polynomial Regression Model [39, 40].

In polynomial regression, the high-fidelity model is approached by a set of polynomial basis functions, $\mathbf{z}$, and a set of weights β, which in turn form the possibility to model a underlying polynomial. Eventually, the goal is to create the predictor

$$\hat{y}(\mathbf{x}) = \mathbf{z}(\mathbf{x})^{\mathrm{T}} \beta. \tag{3.3}$$

For example, a simple, second-order polynomial model, using $N_v = 2$, so $\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$, looks like

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2, \tag{3.4}$$

which generalizes to

$$\hat{y}(\mathbf{x}) = \beta_0 + \sum_{j=1}^{N_v} \beta_j x_j + \sum_{i=1}^{N_v} \sum_{j \leq i}^{N_v} \beta_{ij} x_i x_j. \tag{3.5}$$

When this equation is formatted according to Equation 3.3, the set of basis functions becomes

$$\mathbf{z}(\mathbf{x}) = [1, x_1, x_2, x_1 x_2, x_1^2, x_2^2]^{\mathrm{T}}, \tag{3.6}$$

with weight vector

$$\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \beta_{12}, \beta_{11}, \beta_{22}]^{\mathrm{T}}. \tag{3.7}$$

In order to train a linear regression model, one collects all samples points $\mathbf{x}^{(i)}$, $i = 1, ..., N_s$ and their responses $y(\mathbf{x}^{(i)})$ and collect the data in a set of linear equations

$$y(\mathbf{x}^{(i)}) = \sum_{j=1}^{N_{PRG}} \beta_j z_j^{(i)} + \epsilon_i. \tag{3.8}$$

which can be represented in the form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon, \tag{3.9}$$

where $\mathbf{y} = [y(\mathbf{x}^{(1)}), ..., y(\mathbf{x}^{(N_s)})]^T$ and $\mathbf{X}$ is the $N_s \times N_{PRG}$ matrix of evaluations of the basis function on the sampling locations, so $\mathbf{X} = [\mathbf{z}(\mathbf{x}^{(1)}), ..., \mathbf{x}^{(\mathbf{N_s})}]^T$. Using a least-squares estimation, the weights $\boldsymbol{\beta}$ can be determined using

$$\hat{\boldsymbol{\beta}} = \left(X^T X\right)^{-1} X^T \mathbf{y}, \tag{3.10}$$

of which a lengthy derivation is given by [53]. Important to note is that this method depends on the error ϵ, which is assumed to be independently randomly distributed. For deterministic computer simulations, as will be dealt with in this thesis, this is not a valid assumption. Where this method is a regression, an *interpolation* would suit this case better. However, it is possible that numerical noise is introduced in the simulation, which then again provides a valid usecase for the ϵ term in Equation 3.8.

The other important parameter is the number of used basis functions N_{PRG}, which is the order of the underlying polynomial. Usually, the greater the number of basis functions, the higher the accuracy of the model will get. However, one has to watch out for overfitting of the model [52]. A strategy to find the optimal compromise between the bias and variance of the predictor is to minimize the cross-validation error, which is discussed in subsection 7.2.2.

Forrester et al. provide the following characteristics of linear regression in their paper [52]

- Polynomial regressions are unsuitable for non-linear, multi-modal, and multi-dimensional design spaces unless the variable ranges are reduced.
- High-dimensional problems require large amounts of data to estimate the coefficients of high-order polynomials.
- Problems with few dimensions, uni- or low-modality, and large available data sets suit the use of polynomial regression.
- Retrieved polynomial expressions can give useful insights into the effect of design variables on the high-fidelity model.

In the work by Paiva et al. [29], where different surrogate model types are compared, the above characteristics are confirmed. For low-dimensional design spaces, its effectiveness is shown. It requires the least amount of sample points to achieve a reasonably accurate model. However, for higher-dimensional problems, its accuracy starts to drop quickly, and other alternatives produce much better results.

3.2.2. RADIAL BASIS FUNCTIONS (RBF)

The second model type that will be discussed, is the class of Radial Basis Functions (RBF) models. Having been developed for interpolation of scattered multivariate data, it utilizes a weighted sum of radially symmetric functions [22, 39, 40, 52]. Using N_{RBF} of these radially symmetric functions, the model is defined by

$$\hat{y}(\mathbf{x}) = \sum_{j=1}^{N_{RBF}} w_j \phi(\|\mathbf{x} - \mathbf{c}^{(j)}\|), \tag{3.11}$$

where w_i are the model parameters, or weights. $\phi(\|\mathbf{x}-\mathbf{c}^{(i)}\|)$ are the radial basis functions, which are evaluated using the Euclidean distance between prediction $\mathbf{x}$ and $c^{(i)}$, the defined centers for the ith of N_{RBF} basis functions.

The functions used for ϕ have the characteristic that their response is monotonically increasing or decreasing for an increasing distance from a certain central point. Many types of basis functions $\phi(r)$, where $r = \|\mathbf{x} - \mathbf{c}^{(i)}\|$, of which examples [52] include

- Non-parametric basis functions, and
  - linear $\phi(r) = r$,
  - cubic $\phi(r) = r^3$, and
  - thin plate spline $\phi(r) = r^2 \ln r$.
- Parametric basis functions.
  - Gaussian $\phi(r) = e^{-r^2/2\sigma^2}$,
  - multi-quadric $\phi(r) = \sqrt{(r^2 + \sigma^2)}$, and

– inverse multi-quadric $\phi(r) = 1/\sqrt{(r^2+\sigma^2)}$.

In the above summation, the distinction has been made between two types of functions, either parametric or non-parametric functions. For model training, the non-parametric basis functions only need the weights for each basis function w to be determined. When the problem asks for increased freedom to improve the generalization of Equation 3.11, one can use the parametric basis functions, but this yields an extra parameter, in the examples σ, needs to be determined.

Similarly to the method of finding the weights for a polynomial regression, one can determine the weights of the system by collecting the sample points and creating the set of linear equations

$$y(\mathbf{x}^{(i)}) = \sum_{j=1}^{N_{RBF}} w_j \phi(\|\mathbf{x}^{(i)} - \mathbf{c}^{(j)}\|) + \epsilon_i, \tag{3.12}$$

where $y(\mathbf{x}^{(i)})$ are the responses for $\mathbf{x}^{(i)}$, and $i = 1, ..., N_s$. An approach is to make above system square, by taking the sample points as the functions centers, so $\mathbf{c}^{(j)} = \mathbf{x}^{(i)}$ and $N_{RBF} = N_s$. However, this is not a necessary step to make. This system can be denoted as

$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}, \tag{3.13}$$

where $\mathbf{\Phi}$ is the Gram matrix, which is defined by $\mathbf{\Phi}_{ij} = \phi(\|\mathbf{x}^{(i)} - \mathbf{c}^{(j)}\|)$ [39]. This can be solved similarly to the polynomial regression, by making use of

$$\hat{\mathbf{w}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{y}, \tag{3.14}$$

as described by [39].

Alizadeh et al. [42] mention the quick modeling characteristics of RBF and the fact that it is well applicable to high-dimensional problems. Furthermore, if the Gaussian basis function is used, it is also possible to make error predictions with an RBF. However, RBF in its standard form might lack accuracy, but improved formulations are available that reduce this issue [42].

3.2.3. KRIGING

Another commonly used model is Kriging. Originally developed by mining Engineer D. Krige [54], the method was first mathematically formulated by Matheron in 1963 [55]. The first usage of Kriging for the analysis of computer experiments, hence the creation of the surrogate model use case, can be granted to Sacks et al. in 1989 [56].

The main idea of Kriging is that it approximates a high-fidelity model, by modeling the underlying (spatial) correlation between its design variables, and was originally designed for noise free, deterministic high-fidelity models as it is normally an interpolating model [22, 52, 57, 58]. The basic implementation consists of the sum of two parts: a linear regression which models the process and additionally a realization of a stochastic process which models the localized deviations [39, 42]. The most general form is expressed as

$$f_p = \mathbf{g}(\mathbf{x})^{\mathrm{T}}\beta + \epsilon(\mathbf{x}), \tag{3.15}$$

where $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_k(\mathbf{x})]$ is the set of known functions and the error term $\epsilon(\mathbf{x})$ is the realization of a stochastic process. This value is determined by a certain correlation model, for example a *Gaussian* correlation, and has a zero expected value $E[\epsilon] = 0$ and a non-zero variance.. As the dependency of ϵ on $\mathbf{x}$ suggest, the correlation in the error term is dependent on the distance between sample points and training data.

A commonly used interpretation of this is Ordinary Kriging (OK), which is given by

$$f_p = \mu + \epsilon(\mathbf{x}). \tag{3.16}$$

Here, μ denotes the mean of the response of the model at the sample points. This is popular and successful in deterministic simulations, and it is also suggested that it is likely that OK has better predicting properties than UK. Yondo et al. suggest that the fact that the basis functions $\mathbf{g}(\mathbf{x})$ are rarely known a priori, explains this common usage of OK [22].

Let us turn the attention to the main philosophy of Kriging, the stochastic element, and the corresponding correlation model. As mentioned earlier, multiple correlation models can be used [22]. However, multiple researchers such as Queipo et al. [40], Forrester et al. [52], Koziel et al. [39] and Alizadeh et al. [42], only mention the Gaussian correlation model which therefore will be the main focus of this review. However, if one finds the need to experiment with other correlation models, Table 3.2 can serve as a reference. The hereafter mentioned methodology will not be significantly altered when using a different correlation model.

Name	**Correlation Model** $R(\mathbf{x},\mathbf{y})$
Gaussian (Squared Exponential)	$\exp(-\theta_k(\mathbf{x}-\mathbf{y})^2)$
Ornstein-Uhlenbeck (Exponential)	$\exp(-\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert)$
General Exponential	$\exp(-\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert^{\rho_k})$, $0<\rho_k<2$
Spherical	$1-1.5\xi_k+0.5\xi_k^3$, $\xi_k=min\{1,\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\}$
Cubic	$1-3.0\xi_k+2.0\xi_k^3$, $\xi_k=min\{1,\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\}$
Linear	$max\{0,1-\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\}$
Matérn 5/2	$\left(1+\sqrt{5}\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert+\frac{5}{3}\theta_k^2(\mathbf{x}-\mathbf{y})^2\right)\exp\left(-\sqrt{5}\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\right)$
Matérn 3/2	$\left(1+\sqrt{3}\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\right)\exp\left(-\sqrt{3}\theta_k\lvert\mathbf{x}-\mathbf{y}\rvert\right)$

Table 3.2: An overview of different correlation functions to be used in Kriging surrogate models [22, 23]. All functions adhere to $\theta_k>0$.

The error term $\epsilon(\mathbf{x})$ in Equation 3.15 and Equation 3.16 has a covariance matrix defined by

$$Cov[\epsilon(\mathbf{x}^{(i)}),\epsilon(\mathbf{x}^{(j)})]=\sigma^2\mathbf{R}([R(\mathbf{x}^{(i)},\mathbf{x}^{(j)})]), \tag{3.17}$$

where R is the $N_s\times N_s$ correlation matrix. Here $R_{ij}=R(\mathbf{x}^{(i)},\mathbf{x}^{(j)})$ is defined by the selected correlation function $R(\mathbf{x},\mathbf{y})$, which is evaluated for the sampled data points $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$. σ is the standard deviation of response at the sampled points. As discussed, different correlation functions can be found in Table 3.2, but the commonly used Gaussian correlation function is given by

$$R(\mathbf{x},\mathbf{y})=exp\left[-\sum_{k=1}^{n}\theta_k(x_k-y_k)^2\right], \tag{3.18}$$

where θ_k are *hyperparameters* that can be tuned for each of the k dimensions. When the parameters are estimated, the predictor of Kriging takes the form:

$$\hat{\mathbf{y}}(\mathbf{x})=\hat{\mu}+\mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}-\mathbf{1}\hat{\mu}), \tag{3.19}$$

where $\mathbf{r}$ is the correlation vector between the to-predict points and the sample points, which gives that $\mathbf{r}=[R(\mathbf{x},\mathbf{x}^{(1)}),...,R(\mathbf{x},\mathbf{x}^{(N_s)})]$. $\mathbf{1}$ denotes the N_s vector of ones. The estimated mean $\hat{\mu}$ for OK is expressed as

$$\hat{\mu}=\frac{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}}{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}}, \tag{3.20}$$

and, when using UK, can be replaced by

$$\hat{\mu}=\hat{\mu}(\mathbf{x})=\mathbf{g}(\mathbf{x})^{\mathrm{T}}\beta, \tag{3.21}$$

as provided by [52].

The power of Kriging lies in its ability to estimate its uncertainties at unsampled locations. For each prediction point, the Mean Squared Error (MSE), which is equal to the variance of the estimated response as the predictor's bias is zero [59], is known and provided by

$$V(\hat{\mathbf{y}}(\mathbf{x}))=\sigma^2\left[1-\mathbf{r}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}+\frac{\left(1-\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{r}\right)}{\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}}\right]. \tag{3.22}$$

In OK, depending on the correlation model used, either only the hyperparameters θ_k or both θ_k and the order of the exponential in the correlation model ρ_k need to be determined. This is not a straightforward task [22, 52]. However, the in-depth analysis of these methods is considered to be out of scope for this review. If the reader is interested, Kleijnen [58] provides a good description of an algorithm to determine θ_k, based on Maximum Likelihood Estimation.

The computational expense for Kriging mainly comes from many matrix inversions, which can be considered a downside of the model [22]. In 2009, Forrester et al. [52] wrote that amount of dimensions N_s is normally limited to around 20, but Yondo et al. [22] increased this number to a maximum of 50 design variables in 2018. Bouhlel et al. [60, 61] proposed Kriging Partial Least Squares (KPLS) and its improvement KPLSK for this reason, as they enable input spaces of more than 100 variables. Yondo et al. [22] mention another potential pitfall of Kriging, as there is a risk of building an ill-conditioned correlation matrix, especially when using a high dimensional design space or many samples. Spreading the samples is named as the solution.

3.3. SURROGATE MODELING TOOLBOXES

As can be read in the previous sections, many SM types and DoE strategies are available. Multiple toolboxes have been developed that package these model types and DoE strategies. In this section, a small overview of available toolboxes is presented, and a selection is made on which toolbox to select for implementation in the to-be-designed system.

Many packages supporting machine-learning techniques have been developed for Python, for example, the well-known modules Scikit-learn [62] and TensorFlow [63]. These packages support a wide range of model types but often miss the bundled functionalities that specific surrogate modeling toolboxes contain. Machine learning is also slightly different from the notion of surrogate modeling, as typically samples are less expensive in typical machine learning practices. Often, many samples are available which favors the use of machine learning models such as artificial neural networks (ANN). When creating SMs, samples are considered to be more expensive and therefore require a subtly different approach, and mainly other model types such as Kriging, as it possesses powerful prediction capabilities for limited available data.

One of these toolboxes specifically developed for surrogate modeling, is the Surrogate Modeling Toolbox (SMT)[1] [23]. This toolbox contains a wide range of DoE strategies, as well as an extensive collection of surrogate types. A key feature of the toolbox is the focus on derivatives of the surrogates, which is useful for gradient-based optimization. The package can be seen as truly a toolbox, where the capabilities of generating a DoE and generating a surrogate model are available. Functionality such as automated surrogate model selection, execution of the DoE, and automated model validation are unavailable in the tool. Another commonly used tool, the Matlab-based SUMO[2] [64] also provides these functionalities. This toolbox has to goal to function in a highly autonomous manner, by automatically varying sample sizes and automatically selecting surrogate models.

For the to-be-designed system, the less-autonomous approach of SMT seems to be the most useful. SMT provides all the required features and is fully Python-based. It is very likely that the system could also be developed with the use of SUMO, but SMT's simplicity and native availability in Python, the language of choice for this Thesis, SMT is selected. However, extensions to SMT will be needed to support a CDS-based approach, as well for the creation of executable SMs outside of SAS.

[1] https://smt.readthedocs.io/en/latest/
[2] http://sumo.intec.ugent.be/SUMO_toolbox

4

SURROGATE-BASED MDAO

In this section, the current state-of-the-art of the inclusion of surrogate models within MDAO systems will be discussed. From literature, three main strategies can be derived. In the first approach, the complete MDAO system is being replaced by a surrogate model, and that surrogate is optimized. The second is adaptive surrogate-based MDAO, where the model is continuously updated with new sample points, based on a certain *infill* strategy. In the third strategy, the non- or semi-adaptive surrogate-based MDAO, a certain expensive discipline is replaced by a surrogate model, that has been trained using a certain DoE and is not updated during the optimization process, or only updated for the found optimum and nearby.

4.1. GLOBAL SURROGATE-BASED MDAO

The first approach is researched by Xu et al. [24], where the application of the Efficient Global Optimization (EGO) algorithm, by Jones et al.[65], for MDO is investigated. In the EGO approach, an objective function is replaced by a surrogate model (Kriging) and optimized using an infill criterion such as EI. Hence, in the application of this algorithm for an MDAO system, the complete MDA is replaced by the surrogate model. Based on an initial DoE, the surrogate model is generated, and an optimization is executed. For each infill point, the MDA is executed again and the surrogate model is extended to include the new point. In their study, Xu et al. [24] show to drastically reduce the number of discipline calls compared to a normal gradient-based algorithm, with an achieved reduction up to a factor of 6.6 depending on the MDAO architecture used.

Bartoli et al. [5, 66] proposed an evolution of EGO, called SEGOMOE, which in turn is an evolution of Super Efficient Global Optimization (SEGO) [67]. Its overall flow is displayed in Figure 4.1. SEGO is an adapted version of EGO to handle constrained optimization problems. SEGOMOE enhances (S)EGO further by applying a WB2 infill-criterion by Watson and Barnes [68], as well as by applying a so-called Mixture Of Experts (MOE) methodology. MOE is a method of building a surrogate model, where multiple models, called *local experts*, are used to approximate certain subspaces of the input space, which are eventually combined. In their work, Bartoli et al. [5, 66] propose to use the algorithm in combination with an initial sampling using an optimized version of LHS, and using the Kriging variants KPLS and KPLS+K. Both of these Kriging variants are available for use in SAS.

4.2. ADAPTIVE SURROGATE-BASED MDAO

Examples of adaptive surrogate-based MDAO strategies are sparse, but a proposal is found in the work by Dubreuil et al. [25, 69], where they present the Efficient Global Multidisciplinary Design Optimization (EGMDO) strategy. In this work, each discipline is replaced by a Kriging model, apart from the objective function. The induced uncertainties by replacing the disciplines' high-fidelity models with the surrogates are 'traced' and quantified in the objective function. This means that an uncertainty prediction is available for the objective function, similarly to a Kriging model. Using this uncertainty prediction it is capable of implementing an EI-inspired algorithm that calculates the infill points, and efficiently locates an optimum. An impressive reduction in discipline evaluations is achieved. For a test case, the 638 evaluations needed when using an MDF-COBYLA-based optimization are reduced to 13 evaluations for the EGMDO algorithm, while also improving the robustness of the optimization. From their research, an IDF-based EGO algorithm using the earlier mentioned WB2 had the highest robustness and needed 44 evaluations of the disciplines to find an

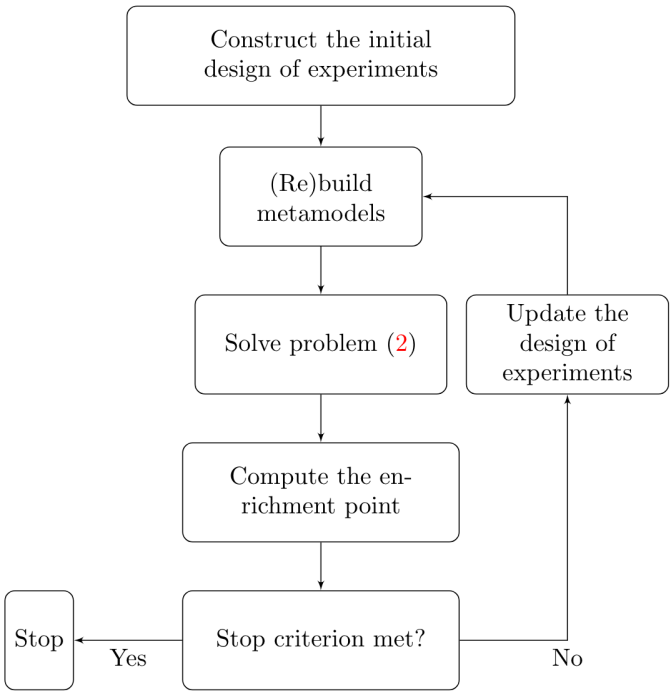


Figure 4.1: Overview of the SEGOMOE algorithm [5].

optimum. Hu and Mahadevan [26] have proposed a similar approach, applied to time-dependent multidisciplinary reliability analysis, although a different mathematical formulation for the error propagation through the disciplines is used.

4.3. NON- OR SEMI-ADAPTIVE SURROGATE BASED MDAO

Examples of non-adaptive strategies are found in the work of Wang et al. [70] use surrogate models to replace an expensive CFD simulation of a battery thermal management system in an MDF system and make use of the COSMOS [71] surrogate model selection algorithm. Chen et al. [72] perform a surrogate-based MDAO on an autonomous underwater vehicle, where again a surrogate model is generated for a CFD model of a hull. Lefebvre et al. [27] replace clusters of disciplines with Kriging models, which they use to optimize an aircraft. Paiva et al. [29] propose a semi-adaptive surrogate-based MDAO system, where an expensive wing-analysis tool is replaced by a surrogate model, and the surrogate model is refined at a previously found optimum.

As is mentioned by Dubreuil et al. [25], the non-adaptive strategy poses a significant demand on the initial accuracy of the surrogate model, which is necessary to converge to a global optimum. The following strategies are derived from the available literature. Paiva et al. [29] propose a slightly altered version of LHS. Each newly generated sample is being tested for the (nonlinear) constraints so that the DoE only consists of 'feasible points'. After an optimum is found, it refines the surrogate model in a trust region around the located optimum. Chen et al. [72] and Lefebvre et al. [27] both use an (optimized) LHS, but Lefebvre et al. [27] initially recommends to create the sampling plan using an adaptive DoE strategy, such as the one proposed by Da Ronch et al. [73]. A more extensive review of adaptive DoE methods, used in combination with Kriging, is to be found in the review by Fuhg et al. [74].

This impact of the necessity for a highly accurate initial surrogate model on the computational expense is not well documented in the literature. Chen et al. [72], Wang et al. [70] and Lefebvre et al. [27] do not provide a comparison between the costs of optimization using a high-fidelity or surrogate model. Paiva et al. [29] do provide such a discussion. It is clear that in certain situations, the reduction in the required amount of function calls is significant (about 50% reduction), and the lost accuracy is minimal. However, situations occur where there is no significant reduction in discipline evaluations, in this specific case it occurs when a specific amount of input variables are used. The authors recommend a further tuning of the models' parameters to improve results for this test case. it has to be noted that in this work, the complete MDAO system is replaced by a surrogate model, except for the objective function and constraints. For one of the test cases, the latter

elements have been replaced by surrogates as well.

II

METHODOLOGY AND IMPLEMENTATION

5

WORKING WITH MDAO WORKFLOWS

This chapter discusses the implemented methodology in SAS with respect to MDAO workflows. In SAS, an MDAO workflow is provided, and using this workflow, several operations will need to be performed. First of all, a provided MDAO workflow contains all the possible SM strategies. In this work, an SM strategy is defined as a design competence or group of design competences that would be replaced with a surrogate model. The process of the extraction of all design competences from an MDAO workflow is discussed in section 5.1. In several stages of SAS, the provided workflow will have to be converted to a DoE formulation. This implemented methodology for this process is covered in section 5.3.

5.1. IDENTIFICATION OF SM STRATEGIES

In this work, an SM strategy is defined as a design competence or group of design competences that would be replaced with a surrogate model. To assess which strategies would be the most effective in reducing the computational complexity of a workflow, first a selection of all possible SM strategies needs to be created. These SM strategies can later be investigated for their performance, and eventually, an SM strategy can be selected and implemented in a workflow.

The starting point of the software will be an initial optimization workflow, from where all the design competences are extracted. This leads to an initial list of SM strategies, as each of the design competences can theoretically be replaced with an SM. Groups of design competences have to abide by a certain set of rules, to ensure the feasibility of the generated set of SM candidates. So, a group of design competences

1. Can contain a driver, only if the complete nested loop is included in the SM strategy
2. Cannot "break out" of a nested loop without including the complete nested loop
3. Cannot contain a DoE driver or the most-outer-level optimization driver

As an example, the workflow for the Sellar optimization problem is shown in Figure 5.1. For this workflow, five individual design competences are found: D1, D2, F1, G1, and G2. All these design competences are valid SM strategies. Due to the presence of a convergence loop, not all possible surrogate groups are valid strategies. For example, an SM strategy consisting of the converger, D1, and D2 is valid as it entails the complete nested loop and therefore meets rule 1. However, the strategy consisting of the converger and D1, is not valid, as it breaks rule 1 because D2 is not included. Taking D1 and D2 is a valid strategy as it adheres to all the rules, but a strategy where D1, D2, and F1 would be replaced with an SM, is not as it breaks rule 2. For this strategy to be valid, the converger would have to be included. Based on the rules, the possible SM strategies for the Sellar workflow are found in Table 5.1. Although these SM strategies are valid, it cannot yet be stated whether they are strategies that would be beneficial to implement.

5.2. INTERFACE FOR AN SM STRATEGY

For a selected SM strategy, a design competence or group of design competences that are replaced with an SM, its in- and output variables need to be determined. As the goal is to replace the design competences with the SM, the in- and output variables of the SM strategy need to be determined.

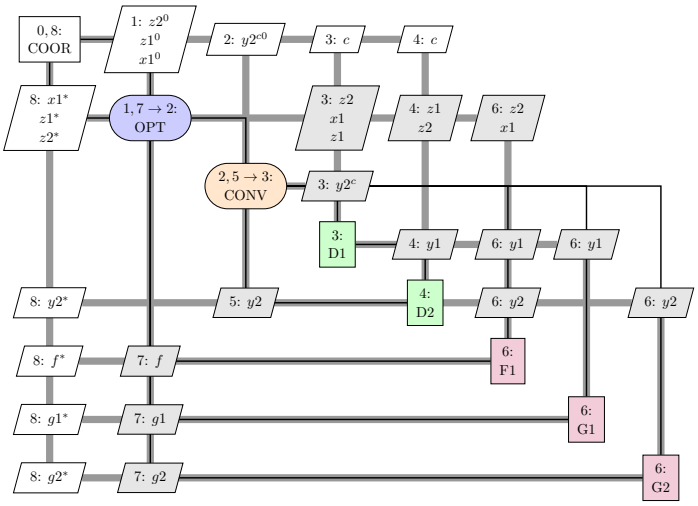


Figure 5.1: XDSM of the Sellar problem

Strategy	Design Competences
1 to 5	Either D1, D2, F1, G1 or G2
6	Converger, D1, D2
7	D1, D2
8	Converger, D1, D2, F1
9	Converger, D1, D2, F1, G1
10	Converger, D1, D2, F1, G1, G2
11	F1, G1
12	F1, G1, G2
13	G1, G2

Table 5.1: Valid SM strategies for the Sellar workflow

The in-and outputs of the SM strategy can be deduced from the original MDAO workflow formulation. For an SM strategy consisting of a single design competence, the process is straightforward: all the in- and outputs of the original design competences are maintained and will be included in the SM. For an SM consisting of multiple design competences, the process is more complicated. The in-and outputs of the SM should be able to produce the required variables for other design competences in the workflow and should be able to model the behavior of each of its encapsulated design competences.

Two concepts, namely a variable's *source* and *targets*, from the graph-theoretic approach from KADMOS are introduced here, necessary for the determination of an SM strategy's in- and output variables [2]. As discussed earlier in section 2.2, a variable is modeled as a node in the graph theoretic approach and a variable node is denoted by v_v. On the other hand, a function node, which represents a design competence, is denoted by v_f. A relation between a variable node and a function node is defined by a data edge e_d. For an input variable to a design competence, such a data edge is denoted as $e_d = (v_v, v_f)$. For an output variable, the relationship is denoted by $e_d = (v_f, v_v)$ For a node, the set of all incoming edges are denoted by $E^-(v)$, and the number of incoming edges is called the *indegree* $\delta^-(v)$. For a correctly formulated FPG and MDG, $\delta^-(v) = 1$. So, for a certain variable node v_v, $E^-(v_v) = \{(v_f, v_v)\}$. The function node v_f is called the *source* of variable node v_v. Similarly to the set of all incoming edges, the set of all outgoing edges is denoted by $E^+(v)$ and the size of this set indicates the *outdegree* $d^+(v)$ of the node. For a correctly formulated FPG and MDG, $\delta^+(v) \geq 1$ For a variable node v_v, the function nodes v_{fi} in $E^-(v_v) = \{(v_v, v_{f1}), \ldots, (v_v, v_{fi})\}$ are called the variable's *targets*.

The starting point in the determination of the in- and output variables of an SM strategy is the combined (unique) set of all in- and output variables of its design competences. However, if an output variable of one of the design competences is solely an input variable of another design competence in the SM strategy, this variable is not used anywhere else and can be omitted from the SM. So, for a certain output variable to become an output variable of the SM, the following rule applies:

> The targets of an output variable node must not only be design competences later in the design competence order of the SM strategy, than the source of the output variable

If an SM strategy is converged, meaning a complete convergence loop is replaced by an SM and hence a *Converger* is present in the SM strategy, the rule is extended by:

> For a converged SM, the targets of an output variable can additionally be not only design competences in the group, even if the variable's source comes later than the variable's target

This is because the output variables that are now omitted from the SM strategy, are variables that are converged but are only used within the group of design competences contained in the SM strategy. Hence, the variable will not be required in the workflow anymore when the SM strategy is implemented. Similarly to the rule for an output variable, an input variable for the SM strategy must adhere to

> The source of an input variable must not be in preceding design competences in the SM strategy

For the case of a converged SM, this rule is extended by:

> For a converged SM, the source of an input variable must not be in the complete group of design competences

Based on these rules, the algorithm for the determination of the in- and output variables for an SM strategy can be defined. The methodology for the determination of the in- and outputs of the SM is displayed in the activity diagram in Figure 5.2.

5.3. MANIPULATING MDAO WORKFLOWS

For the inclusion of SMs in MDAO workflows, changes will have to be made to an original MDAO workflow structure. The following changes can be expected to be required when only an original optimization workflow is provided as the starting point in the software.

1. The full optimization needs to be changed to a DoE
2. A partition of the workflow needs to be converted to a DoE for the generation of a set of samples for a specific SM strategy
3. A workflow needs to be modified to include an SM, while maintaining an identical workflow architecture as the non-SM-based workflow

The change of a full optimization workflow to a full DoE is a straightforward manipulation. The selected problem formulation and MDAO architecture can be directly transferred to the DoE, where objective variables and constraints variables become Quantities of Interest (QOIs). The DoE for a specific subset of design competences introduces some logic, due to a change in the in- and output variables of the workflow. For a certain SM strategy, where a group of design competences would be replaced with an SM, a DoE needs to be generated. The design variables of the DoE equal the input variables of the SM, and the output variables of the SM equal the QOIs of the DoE. The process for the determination of these in- and output variables is discussed in section 5.2.

After the determination of the required in-and-output variables, the available workflow, which is a KADMOS-based MDG/MPG graph, is returned to the FPG formulation. All architectural elements, such as copy variables and drivers, are removed, and the problem formulation is build from the ground up. The aforementioned MDAO architecture is applied, and a new MDG/MPG for the desired formulation is automatically generated.

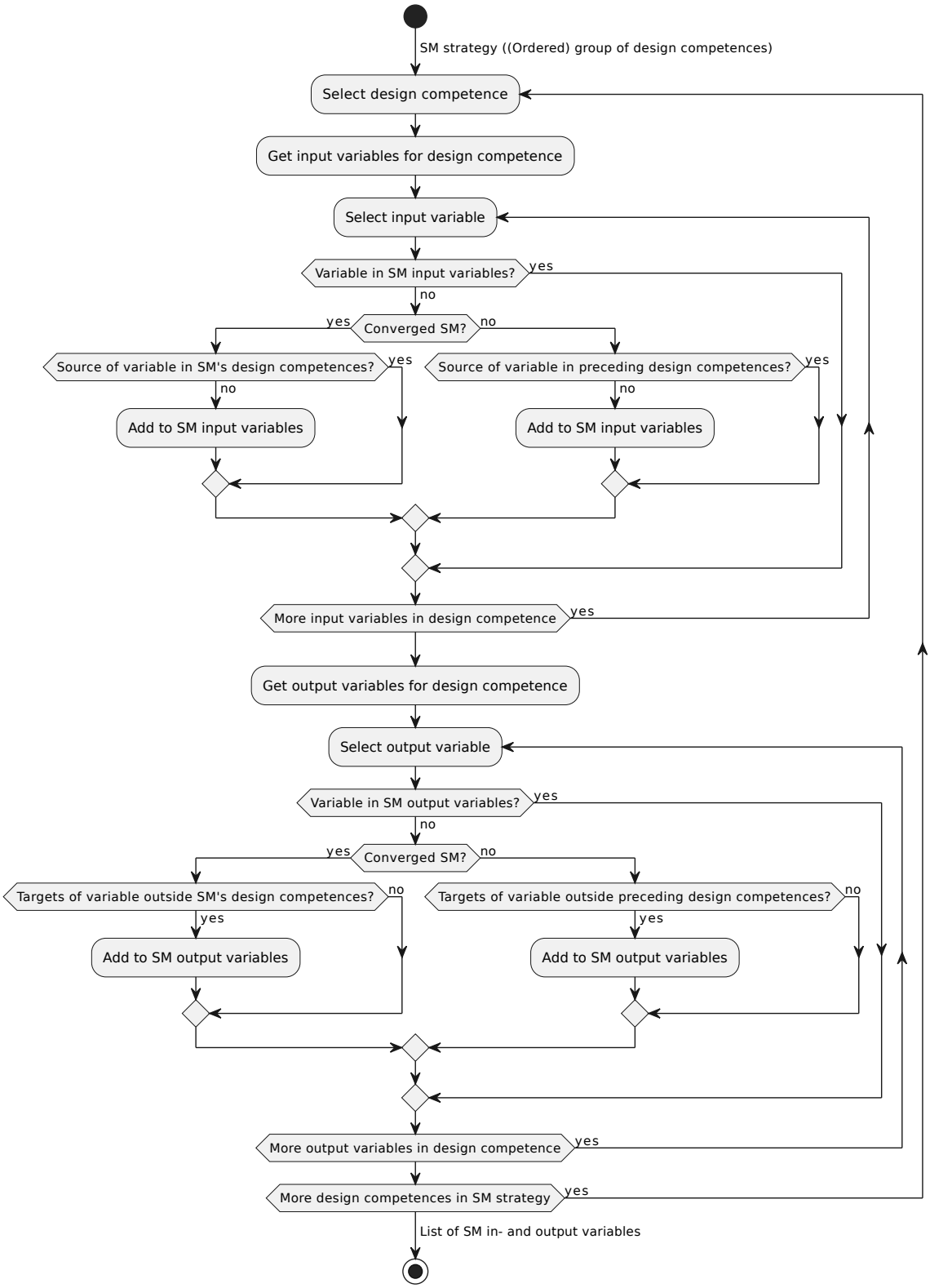


Figure 5.2: Activity diagram of the methodology used for the selection of the in- and output variables for an SM strategy.

6

PROVIDING ADVISE

A major part of the work is to provide an MDAO architect with extended insights into the behavior of their MDAO workflow and indicate opportunities to increase a workflow's computational efficiency by implementing an SM. This chapter discusses two main steps in this process, where first of all a strategy is discussed to analyze a certainly provided workflow. In section 6.1 the methodology for the identification of bottlenecks, as well as the extraction of other workflow describing metrics, is discussed. In section 6.2 the process of identifying the actual best SM strategies is explained.

6.1. ANALYZING A WORKFLOW

An MDAO workflow can be expected to consist of multiple, and often a large number of, design competences, and each of these disciplines will show different behavior during workflow execution. Often, individual design competences can prove to impose a bottleneck on the workflow execution, leading to excessive computational costs of a workflow.

The behavior of a design competence is not always constant for each execution. Depending on the inputs, or other difficult-to-control factors, it can occur that drastically varying execution times are observed for the same design competence. Obtaining insights into this behavior can be useful for MDAO architects, as it can help understand the behavior of the workflow and reduce the 'black box' feeling that has been reported when working with MDAO systems. Certainly when all inputs and outputs for each execution of a design competence are available, having an insight into execution behavior can prove very useful. Where for the calculation of the implications of an SM strategy on the workflow's runtime this information is a necessity, it can also be helpful for the aforementioned reasons.

To obtain the required data for the determination of bottlenecks and overall analysis of the workflow, a methodology is presented where the original optimization workflow is converted to a DoE, and this DoE is then executed for a limited set of samples. This execution run is called the *profiling run*. By extracting the execution *timeline* of the workflow, a set of metrics can be calculated that helps to provide these insights.

For a design competence i_d, the extracted execution times for execution number j are

$$\mathbf{t}_{exec,i_d} = \left\{ t_{exec,i_d,1}, \ldots, t_{exec,i_d,j} \right\}, \tag{6.1}$$

where $j = \{1, \ldots, N_{exec,i}\}$ and N_{exec,i_d} is the total number of execution for the design competence. Based on this set of execution times, multiple metrics for design competence i_d can be calculated, such as the total execution time t_{exec,tot,i_d}, which is defined as

$$t_{exec,tot,i_d} = \sum_{j=1}^{N_{exec,i_d}} t_{exec,i_d,j} = \sum \mathbf{t}_{exec,i_d}, \tag{6.2}$$

the mean execution time

$$\bar{t}_{exec,i_d} = \frac{t_{exec,tot,i_d}}{N_{exec,i_d}}, \tag{6.3}$$

and similarly the minimum, maximum and standard deviations of $\mathbf{t}_{exec,i_d}$ can be calculated. For the completed profiling run, the total runtime $t_{wf,tot}$ and number of completes samples N_s are also extracted. For

design competence i the normalized runtime $\tilde{t}_{exec,i_d}$ is calculated by

$$\tilde{t}_{exec,i_d} = \frac{t_{exec,tot,i_d}}{t_{wf,tot}}. \tag{6.4}$$

The average number of iterations for design competence i is defined by

$$\bar{n}_{iter,i_d} = \frac{N_{exec,i_d}}{N_s}. \tag{6.5}$$

Based on these metrics, and the assumption that the workflow will behave reasonably similar runtime and iterations behavior in the optimization compared to the profiling run, expected bottlenecks during optimization are identified. Furthermore, as will be discussed in the next section, most of the required data to estimate the consequences of an SM strategy is now available.

6.2. IDENTIFYING OPTIMAL SURROGATE MODEL STRATEGIES

The identification of surrogate candidates is not a trivial task. Where in the profiling of the software clear bottlenecks can arise for workflow execution, it is not a given that bottlenecks in a workflow are also feasible SM strategies. It is possible that it is excessively expensive to train a sufficiently accurate SM. As will be discussed in this section, this determination is assumed to be based on two major characteristics:

1. The number of input variables of the SM strategy
2. The execution time for the design competences contained in the SM strategy

6.2.1. ESTIMATING REQUIRED NUMBER OF SAMPLES

The number of samples needed to train an accurate SM is difficult to select a-priori [22]. First of all, the notion of a 'sufficiently accurate SM' is already difficult to determine, and highly dependent on external circumstances. Very early in the design process, it is possible that a less accurate SM is necessary in order to quickly assess designs and get a more general idea of a design's performance. Later in the design stage, more accurate calculations might be needed, and hence a more accurate SM is required.

For a specific SM strategy, the in- and output variables of the to-be-generated SM can be calculated using the methodology as described in **??**. As the input variables are now know, the input dimension N_{v,i_s} of an SM strategy i_s is known as well. In literature, three main rules of thumb emerge that indicate an estimation of the required number of samples for 'reasonable' accuracy based on the dimension of the input dimension. Kaufman et al. [75] propose to use

$$N_s = \frac{3}{4}(N_v + 1)(N_v + 2), \tag{6.6}$$

where N_v is the number of input variables for an SM strategy. Jia et al. [41] simplified the formulation to

$$N_s = (N_v + 1)(N_v + 2). \tag{6.7}$$

Jones et al. [65] suggest to use

$$N_s = 10N_v, \tag{6.8}$$

which has been confirmed to be a valid estimation by Loeppky et al. [76], which have tested the formulation for the LHS sampling scheme and Kriging model type. In SAS, these rules of thumb are summarized in the concept of Implied Accuracy IA, which is defined as

$$IA = \frac{N_{s,realized}}{N_{s,advised}} \cdot 100\%. \tag{6.9}$$

The IA is an estimated measure of how well an SM is trained for a selected computational *investment*, which is directly correlated to the available number of samples $N_{s,realized}$ and the input dimensionality of the SM strategy. The goal of the metric is to directly provide an insight into the required investment for an expected accurate SM while removing the differences in N_v for different SM strategies so that multiple SM strategies can be compared and optimal strategies can be selected.

6.2.2. ESTIMATING CONSEQUENCES OF AN SM STRATEGY

Replacing a design competence, or group of design competences with an SM is called an SM strategy and will lead to a change in workflow characteristics, in both aspects of accuracy and execution time. Based on the metrics from section 6.1, an estimation of the effects of an SM strategy on the runtimes can be made.

The training of an SM will require a certain time investment, as samples for the training of the SM will need to be acquired. These samples are obtained by executing a DoE for a number of samples N_s. When a DoE is executed for an SM strategy consisting of a single design competence, the required time for SM training as a function of N_s for a SM strategy i_s can be estimated by

$$t_{training,i_s}(N_s) = \bar{t}_{exec,i_d} \cdot N_s. \tag{6.10}$$

For a SM strategy consisting of a group of design competences, this equation needs to be modified to take account for both the combined runtimes, as well as possible required iterations when a converged SM is used. As discussed in section 5.2, a converged SM strategy includes a converger and replaces a complete convergence loop with an SM. Hence, only converged samples are used for the generation of such an SM. For a non-converged SM strategy $i_s = \{i_{d,1}, \ldots, i_{d,n_d}\}$ where n_d is the number of design competences included in the SM, the required time investment for the generation of an SM strategy i_s using N_s samples becomes

$$t_{training,i_s}(N_s, i_s) = \frac{\sum_{j=1}^{n_d} \left(\bar{t}_{exec,i_{d,j}} \cdot \bar{n}_{iter,i_{d,j}} \right) \cdot N_s}{\min \left\{ \bar{n}_{iter,i_{d,j}}, \ldots, \bar{n}_{iter,i_{d,n_d}} \right\}}. \tag{6.11}$$

In the denominator, the minimum value of the average number of iterations for each of the design competences is found. For a non-converged workflow, this number will be equal for all included design competences and will therefore the iterations are canceled out. For an SM strategy where design competences are included from both outside and inside of a convergence loop, the iterations are accounted for. For the edge case where only a complete convergence loop is replaced by an SM, the denominator needs to be set to 1, because the converger is not regarded as a design competence in the software and therefore not present in the set $\left\{ \bar{n}_{iter,i_{d,j}}, \ldots, \bar{n}_{iter,i_{d,n_d}} \right\}$. Failing to adjust the denominator for this leads to an underestimation of the training time, as it fails to account for the required iterations when training a converged SM.

However, this information is not very useful by itself. The goal is to find the *gained* time in the final optimization by replacing a design competence, or set of design competences, with an SM. In other words, *what is the effect on the optimization budget when we implement an SM strategy $i_s = \{i_{d,1}, \ldots, i_{d,n_d}\}$ that is trained using a sample set of size N_s?* To provide an estimation of this metric, the user is requested to either provide a *desired number of objective function calls*, or *expected/maximum time for optimization*. It is assumed that a prediction using a surrogate model costs one second, so $t_{pred,sm} = 1$s.

If the user submits the desired number of objective function calls, N_{obj}, the impact of a surrogate model affects the required time to execute these function calls. For the original, non-SM-based workflow, the estimated execution time is given by

$$t_{wf}(N_{obj}) = \bar{t}_s \cdot N_{obj}, \tag{6.12}$$

where $\bar{t}_s$ is the mean execution time for one DoE sample, as found in the analysis module. Now, if an SM strategy $i_s = \{i_{d,1}, \ldots, i_{d,n_d}\}$ is implemented, the required time for both the data acquisition and SM-based optimization for an SM strategy i_s, t_{wf,i_s}, is defined by:

$$t_{wf,i_s} = t_{wf}(N_{obj}) \cdot (1 - \sum_{j=1}^{n_d} \tilde{t}_{exec,i}) + t_{training,i_s}(N_s, i_s) + N_{obj} \cdot t_{pred,sm} \cdot \bar{n}_{iter,i_s}. \tag{6.13}$$

In the other case, if the user submits an expected or maximum time for optimization, t_{wf}, the introduction of a surrogate model in the workflow impacts the number of possible objective function evaluations within the time budget. For the original workflow, the estimated number of objective function evaluations as a function of an execution time t_{wf} is given by

$$N_{obj}(t_{wf}) = \left\lfloor \frac{1}{\bar{t}_s} \cdot t_{wf} \right\rfloor. \tag{6.14}$$

The new expected number of objective function calls can now be calculated using

$$N_{obj,i_s} = \left\lfloor \frac{t_{wf} - t_{training,i_s}(N_s, i_s)}{\bar{t}_s \cdot (1 - \sum_{j=1}^{n_d} \tilde{t}_{exec,i}) + t_{pred,sm} \cdot \bar{n}_{iter,i_s}} \right\rfloor. \tag{6.15}$$

In this equation, the numerator represents the newly available time for workflow execution *after* the surrogate model has been trained, and the denominator equals the execution time per objective function evaluation for the surrogate model-based workflow. The above calculations are formulated for the replacement of a *single* discipline with an SM and need to be lightly adjusted when a surrogate candidate is presented that encapsulates multiple design competences. Assuming that the architect is capable of providing an initial estimate for N_{obj} or t_{wf}, all the information is now available to provide the MDAO architect with a trade-off between computational efficiency and (implied) accuracy,

7

SURROGATE MODELS

One of the goals of this work is to automate the surrogate model lifecycle. In this chapter, a part of the methodology is discussed that aims to reach this goal. The details of the implementation of SM training, data acquisition, and SM deployment into a CDS/CPACS compatible format are discussed in detail in section 8.2, as these subjects are specific implementations in the software. More abstract concepts such as the SM type selection, model validation techniques, and the notion of *hidden constraints*, are discussed here.

7.1. MODEL SELECTION

A selection of commonly used SM types have been discussed in section 3.2. Already in that section, many possible model configurations can be found. For the discussed model types, different sub-types are available and each will have a certain effect on the accuracy of the SM. This makes model selection not a straightforward task and has traditionally been a manual and time-consuming task [41]. However, for the accuracy of the model, it is of high importance that the correct SM type and subtype are selected, as drastic variances in accuracy can occur between these different (sub)types, as will be discussed later in **??**.

SAS implements an automated model selection algorithm, based on an exhaustive search. The automated selection algorithm finds all possible surrogate model configurations available in SAS for a provided subselection of model types and trains and evaluates each of the provided model configurations. This evaluation is done using one of the implemented methodologies from section 7.2, and comes up with the most accurate SM for a provided sample set. The implemented algorithm is shown in algorithm Algorithm 1.

Algorithm 1 Automated SM type and configuration selection algorithm. A list of available error metrics can be found in Table 7.1.

```
Generate candidate list of all possible surrogate model types and configurations
Filter candidate list for user-provided selection
Start parallel pool
while Model candidate available do
    Assign model configuration to free worker
    if Continuous hyperparameters available then
        Find optimum continuous hyperparameters using optimization
        Store error for optimized hyperparameters
        Store optimum continuous hyperparameter configuration
    else
        Evaluate model
        Store error
    end if
end while
Select candidate model and model configuration with minimum error
Return optimum model configuration
```

The algorithm could introduce performance issues for large sample sets and high input dimensionalities,

Metric	Equation
Root Mean Square Error (RMSE)	$RMSE = \sqrt{\frac{1}{N_{val}} \sum_{i=1}^{N_{val}} (y_i - \hat{y}_i)^2}$
Root Mean Square Percentage Error (RMSPE)	$RMSPE = \sqrt{\frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \frac{y_i - \hat{y}_i}{\hat{y}_i}^2}$
Normalized Root Mean Square Error (NRMSE)	$NRMSE = \frac{\sqrt{\frac{1}{N_{val}} \sum_{i=1}^{N_{val}} (y_i - \hat{y}_i)^2}}{\max(\mathbf{y}) - \min(\mathbf{y})}$

Table 7.1: Overview of metrics used to quantify generalization error [77]. Here, N_{val} is the number of validation points, y_i an observed value, $\hat{y}_i$ a predicted value and $\overline{y}$ the average of the observed values. σ denotes the standard deviation of the observed data set.

due to its exhaustive nature. However, for smaller sample sets, the performance is more than acceptable, and often insignificant compared to the cost of data acquisition. The algorithm utilizes parallelization in the evaluation of the candidate model type and configurations, which drastically increases the performance, especially for modern, high-core-number CPUs.

7.2. MODEL VALIDATION

For a generated SM it is of high importance to know its predictive capabilities, and therefore its accuracy needs to be assessed. In this section two implemented validation schemes are discussed.

Some SM types have intrinsic approximation error estimating capabilities, such as Kriging, which is capable of estimating the expected variance for a prediction point. But, as the underlying correlation function of Kriging is making use of spatial information, this error estimation is large distance-based [74] and might not be representative of the actual, true error. Furthermore, for other models, such as linear regression and (most types of) Radial Basis Functions, a standalone procedure needs to be in place to assess the predictors' accuracy.

SAS utilizes two validation schemes: split-sample and cross-validation. A short discussion on the used methodology for the validation schemes are discussed in subsection 7.2.1 and subsection 7.2.2 for split-sample and k-fold, respectively. An overview of metrics used to assess the errors is shown in Table 7.1.

7.2.1. SPLIT-SAMPLE

In split-sample validation the available samples are split in two parts: a training set and a testing set [40, 78]. As the name suggests, the SM is trained on the training set of the data, where the testing set will be used to estimate the SM's predictive capabilities by calculation of error metrics such as displayed in Table 7.1.

Split-sample validation comes with a few drawbacks. This approach comes with a few drawbacks. First of all, the data is not efficiently used. The available samples are often acquired by execution of an expensive model, and a significant part of the data will now not be used to either train or validate the model. Secondly, the split-sample validation scheme has a high variance, meaning the error can change significantly for changes (or a different distribution of samples) in the training and testing set. In other words, it is possible that either the training or testing set is not representative of the complete domain, leading to an erroneous error estimation.

7.2.2. CROSS-VALIDATION

Another methodology used for the validation of an SM, is the *cross-validation (CV)* methodology, also called k-fold cross-validation or N-fold cross-validation [40, 78]. The goal of the method is to use as much of the available data to train the model, while also maintaining the possibility to assess the generalization error using a testing data set.

In cross-validation, the complete data set is split up into k subsets, all of a similar size. Initially, $k-1$ data sets are taken to train the surrogate model, while the remaining data set is used to calculate the generalization error. In the next iteration, a different data set is omitted and the other $k-1$ sets are used to train the model. This is repeated k times, hence k surrogate models will need to be generated. After this, the overall generalization error can be computed by averaging the k obtained error measures. Literature shows that in practice $k = 5$ or $k = 10$ are commonly used and show good performance [52, 79]. If $k = N_s$, so the amount of subsets equals the number of samples, the method is called *Leave-on-Out CV*.

One of the advantages of using cross-validation is that the bias of the generalization error estimation is nearly non-existent, as all points will be contained in the testing set once, and $k-1$ times in the training

set. The variance of the error estimation is also greatly reduced compared to split-sample, but can still be unacceptably high [40]. A major drawback to the method is the need to generate multiple surrogate models, which can become a burden if model training is expensive.

7.3. HIDDEN CONSTRAINTS

The disciplinary analyses in an MDAO workflow can be prone to failure at certain points in the design space. These areas are called *hidden*, *unknown*, *unspecified*, or *forgotten* constraints, and their failed input points are called *nonevaluable* [80]. These failures can be caused by multiple factors, such as when physically non-realistic parameters are evaluated and if a disciplinary analysis fails to converge. These situations occur often (up to 60% of the function evaluations [80]), and failure to account for the hidden constraints in an SM can lead to drastically different optimization results.

In SAS, a methodology has been implemented that handles these hidden constraints. The assumption is made that when a hidden constraint is encountered in the evaluation of a discipline, a certain indicator, or *flag*, can be found in the calculated output. These flags are expected to be a certain variable and value combination that indicates unexpected behavior in the disciplinary analysis. For example, a discipline could return an empty, negative, or NaN value, if something goes wrong during the evaluation.

Hidden constraints can be added to a discipline in SAS, by adding a *HiddenConstraint* object to a discipline. A HiddenConstraint object consists of:

- *Flag(s)*: One or more variable and value combinations that indicate whether a hidden constraint has been violated.
- *Action(s)*: One or more actions that happen when a hidden constraint is violated. For example: If a hidden constraint is violated, set the output variable *varA* = *NaN*.
- *Switch to ignore in training*: If this is set to true, samples violating the hidden constraints will be ignored in the training of the design competence's SM, but will be used for the training of the hidden constraints predictor.

Following the proposed methodology by Lee et al. [81], a second, separate, predictor will be employed to predict a violation of hidden constraints. This classifier will map the input variables to a label, that indicates when a hidden constraint is violated. The training data of the predictor is generated by evaluating all samples against the formulated hidden constraints and assigning a label to the sample if the hidden constraints are violated. Lee et al. [81] propose to use the *random forest* classifier [82], which is therefore selected for the hidden constraints predictor in SAS.

7.4. ADAPTIVE SAMPLING

Static sampling is not the most efficient way of generating an accurate surrogate model over the complete range of input variables [22]. A widely considered better approach is to utilize an adaptive sampling technique [74]. A full review and implementation of several adaptive sampling schemes are considered to be out of scope for this work, but one of the goals of the developed software is to enable quick integration of these kinds of algorithms. As a proof of concept, two adaptive sampling schemes have been implemented in SAS. First, the Expected Improvement for Global Fit (EIGF) [83] has been selected. The EIGF algorithm is an adjustment of the Expected Improvement (EI) algorithm and aims to improve the global accuracy of the model rather than to find the global minimum, as is the case for EI. The Refinement Criterion RC_{EIGF} for EIGF is formulated by

$$RC_{\text{EIGF}}(\boldsymbol{x}) = \left(\widehat{\mathcal{M}}(\boldsymbol{x}) - y\left(\boldsymbol{x}^{\star}\right)\right)^2 + \sigma^2_{\hat{Y}}(\boldsymbol{x}), \tag{7.1}$$

where for a certain point of interest $\mathbf{x}$, the difference between the predicted value $\widehat{\mathcal{M}}(\boldsymbol{x})$ and closest sample $y\left(\boldsymbol{x}^{\star}\right)$ indicates a large expected variation in the model, and the variance $\sigma^2_{\hat{Y}}(\boldsymbol{x})$ indicates the largest intrinsic uncertainty. The suggested new sample point is found by the evaluation of

$$\boldsymbol{x}^{m+1}_{\text{EIGF}} = \underset{\boldsymbol{x}^{\star}\in\mathbb{X}}{\arg\max}\left(RC_{\text{EIGF}}\left(\boldsymbol{x}^{\star}\right)\right), \tag{7.2}$$

where $\mathbb{X}$ denotes the design space.

Secondly, a simpler adaptive sampling method is implemented that solely makes use of the variance prediction when using a Kriging-like SM type. For this adaptive sampling algorithm, the Refinement Criterion RC_σ is defined by

$$RC_\sigma(\mathbf{x}) = \sigma^2_{\hat{Y}}(\boldsymbol{x}). \tag{7.3}$$

A new sample point is now found by the evaluation of

$$\boldsymbol{x}_\sigma^{m+1} = \underset{\boldsymbol{x}^\star \in \mathbb{X}}{\arg\max} \left(RC_\sigma \left(\boldsymbol{x}^\star \right) \right). \tag{7.4}$$

Due to the potential presence of a hidden constraint in the data, and therefore in the SM, a proposed new sample point is checked using the hidden constraints predictor. All the proposed sample points which are expected to violate a hidden constraint are rejected, and therefore only expected valid sample locations are proposed.

8

SOFTWARE IMPLEMENTATIONS

In this chapter, the details of the implementations of the earlier discussed methodologies are discussed. The detailed methodology of most of the implemented logic is discussed in the previous chapters, where this chapter provides an overview of the software architecture. The overall structure and implementations of specific modules are discussed in section 8.1. A more in-depth analysis is presented for the Surrogate Modeling module and is presented in section 8.2.

8.1. OVERALL STRUCTURE

An abstract class diagram of the developed SAS package is found in Figure 8.1. The center object, SAS, is the main interface for the end-user to interact with. It acts as the controller of the complete software package, directing all the interactions with the other objects. When SAS is initiated for the first time on a machine, it creates a *workspace* on the local drive, where produced data, such as the database structure, run outputs, and created SMs, are stored.

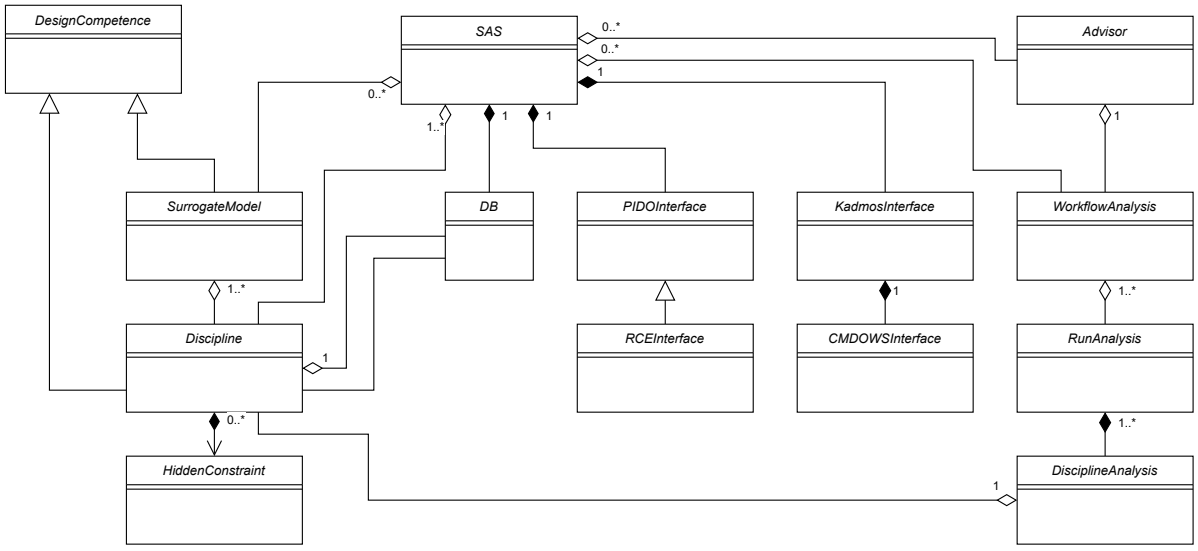


Figure 8.1: Abstracted class diagram of SAS.

SAS can be used in several different settings. First of all, it provides a useful toolbox for MDAO architects to quickly prototype SMs using the built-in functionality, and deploy the CDS-based SMs into an existing MDAO workflow formulation. Additionally, SAS can be used to find the bottlenecks in workflows and to find the optimal SM strategies. The lifecycle of a typical SAS use case, where the profiling and advisory module can be used, is shown in an activity diagram in Figure 8.2. The figure also illustrates interaction with the user, KADMOS, and the PIDO. The following subsections highlight specific parts of the developed software and certain design decisions are outlined.

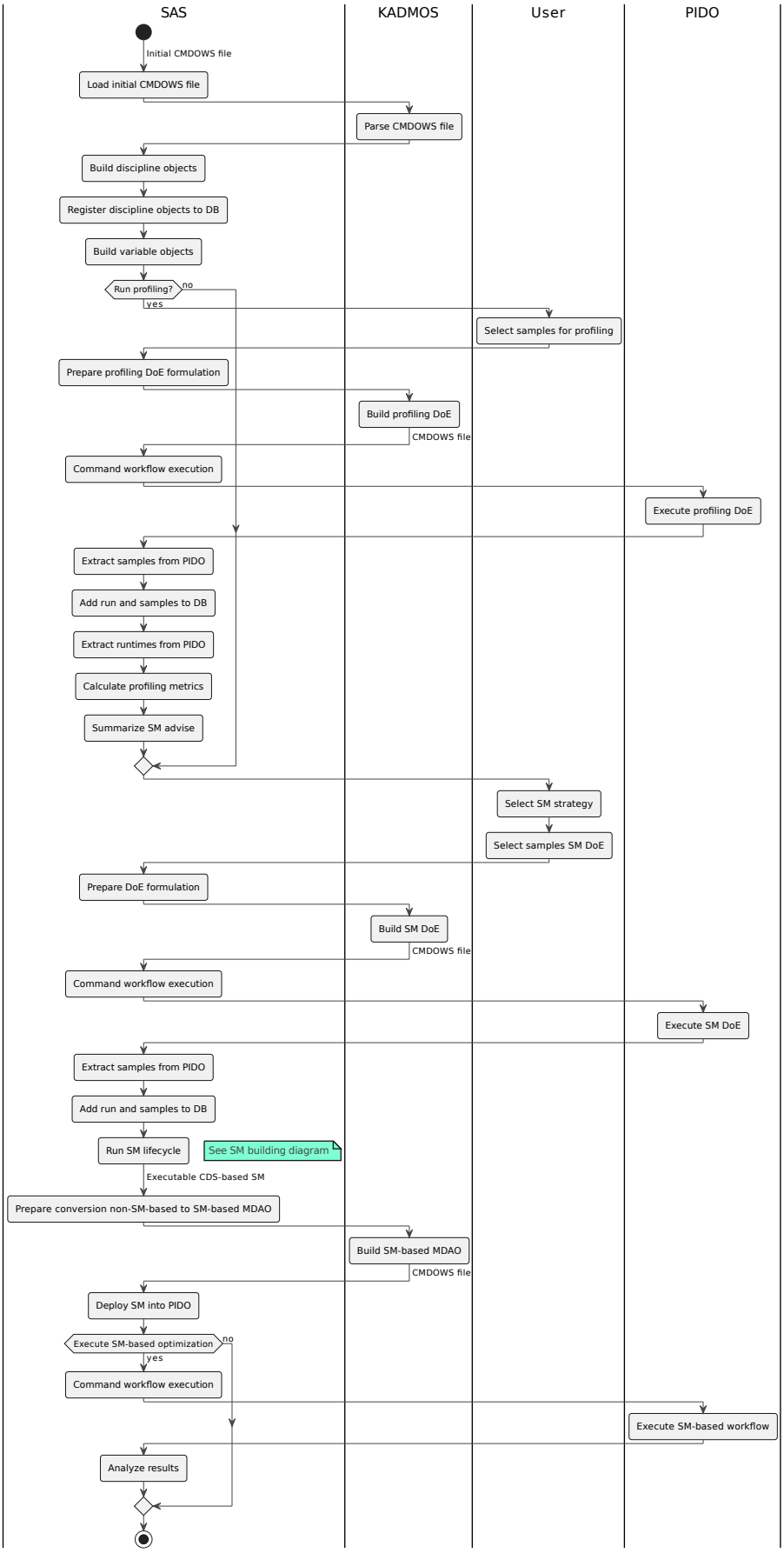


Figure 8.2: Activity diagram of the a typical SAS lifecycle.

8.1.1. DESIGN COMPETENCES

In SAS, design competences are considered to be one of two distinguished types, a *Discipline* or a *SurrogateModel*, and both are implementations from the *DesignCompetence* class. The parent DesignCompetence class implements the more general characteristics of both a *Discipline* and *SurrogateModel*. The DesignCompetence class contains the logic applicable to both children classes, such as the registration of the design competence to the centralized database, as well as the management of the information derived from, and required information for the storage of a design competence in the CMDOWS format.

All available design competences from the original workflow formulation are stored in individual *Discipline* objects. Each discipline consists of a set of in- and output variables, and the object keeps track of all its available samples in the design database. Furthermore, the earlier discussed *hidden constraints*, which are implemented in their own *HiddenConstraints* objects, are added by the user to the corresponding discipline. The SurrogateModel class is further discussed in section 8.2.

8.1.2. KADMOS AND CMDOWS INTERFACE

SAS implements an interface to the KADMOS and CMDOWS software packages and extends the packages' functionalities for SAS' requirements. The *KadmosInterface* class is tasked with the tasks as discussed in chapter 5. It implements all the functionality to interact with the originally provided workflow, such as the determination of the in- and output variables for an SM strategy, as well as the modifications of the workflow to generate the required DoE workflows and SM-based optimizations.

To reduce the complexity of working with the workflows, the original CMDOWS software module is extended in SAS. It contains the logic to quickly extract all the design variables and their nominal values and bounds, the constraints and their conditions, and the objective function from a workflow.

8.1.3. DATABASE STRUCTURE

SAS implements a centralized database, with the goal to store all usable information on the available design competences, executed runs, and all generated data. The implemented database structure is shown in Figure 8.3.

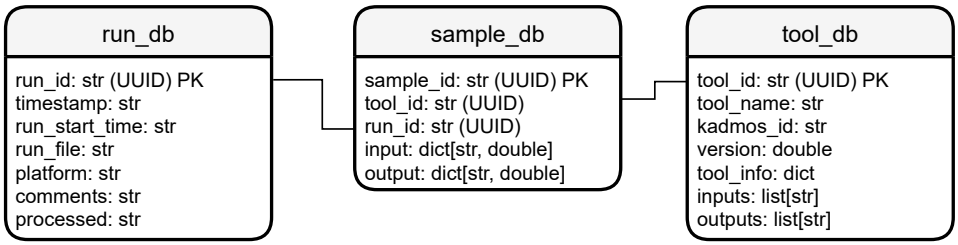


Figure 8.3: Database diagram of the currently implemented database structure.

As can be seen in the database diagram, three separate databases are deployed. First of all, the *run_db* database is employed to keep track of all executed runs. Keeping track of all executed runs ensures the tractability of all samples. Secondly, the *tool_db* database is implemented. In the tool database, all available design competences that have been used in SAS are stored, including different versions and their in- and outputs. Lastly, the sample database stores all acquired samples, and each sample can be traced to a specific run and tool or design competence. Especially the tractability of a sample to a tool/design competence is of importance, as it can be expected that design competences are updated regularly. For the generation of an SM, it is essential to know with samples belong to which version of the design competence, as failing to do so could result in an SM that is trained on faulty data.

The database has been developed to be as independent of the rest of the system as possible, to facilitate a more centralized approach to storing data in the database. The interface to the database is implemented in the *DB* class, which provides a clear interface for accessing data in the database, as well as providing data to the database to store. The reason for the suggested centralized approach for data storage is broader than the usage in SAS alone. It could prove very useful for design departments to develop a compact adapter that processes already available data and inserts it into the database. If this is done continuously during earlier stages of the workflow formulation or a design process, such as during the testing of design competences, it is possible to already build up a significant number of samples, leading to a reduced required time investment for the generation of an SM.

Currently, the databases are created locally using the JSON format. However, it is expected that for future versions of SAS a more centralized, possibly hosted database, could be beneficial, as it could provide a central space for distributed collaborators to store their data.

8.1.4. WORKFLOW ANALYSIS MODULE

As discussed in section 6.1, one of the goals is to identify the bottlenecks in a provided MDAO workflow. For this purpose, the workflow is converted to a DoE using the earlier mentioned methodologies, and executed for a given number of samples N_s.

The results of the executed DoE are processed, and a timeline of the run is calculated. To calculate the required metrics for workflow analysis, the *WorkflowAnalysis* class, combined with the *RunAnalysis* and the *DisciplineAnalysis* classes, are implemented. The more detailed class diagram of the combined Analysis modules is shown in Figure 8.4. The WorkflowAnalysis object receives one or more *run_id*'s from SAS and populates the RunAnalysis and DisciplineAnalysis objects with the information from the provided run(s).

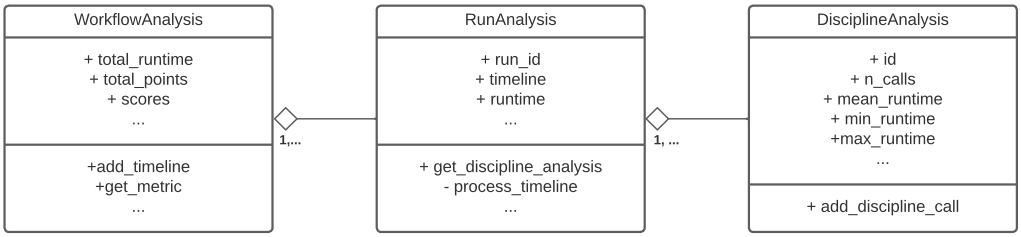


Figure 8.4: Class diagram of the analysis module of SAS.

Each of the objects encapsulates the analysis of a different *level*. In the WorkflowAnalysis objects, all the available information is combined and the most general information can be extracted from here. Each run, so each execution of a workflow in the PIDO environment, that is provided to the WorkflowAnalysis module, is analyzed in the RunAnalysis object. Lastly, the information of each call of a discipline contained in the run is stored in the DisciplineAnalysis class. By using this class structure, all the available information is tractable and a full analysis of workflow behavior can be made. In these modules, the metrics as discussed in section 6.1 are calculated and are available for interpretation in a later stage.

8.1.5. PIDO INTERFACE

To be able to replace design competences with an SM, it is required to execute the design competence for a DoE to obtain the samples needed to train the model. For MDAO-based applications, this means that there needs to exist an interface to a PIDO tool, necessary to execute the DoEs. In this work, an additional need for such an interface emerges as the goal is to provide information on the bottlenecks in a workflow, as well as to provide advice to the user on which SM strategies are the most beneficial. For this purpose, the interface with the PIDO is needed to obtain the execution times of a workflow, as well as the individual execution times of each design competence. For these purposes, SAS implements an interface that adheres to the following requirements. The PIDO interface should be able to

- Execute optimizations from SAS
- Execute DoEs from SAS, possibly for a specific sample or set of samples
- Extract the results of workflow execution in SAS
- Extract a timeline of the workflow execution
- Deploy new design competences (e.g. an SM) into the PIDO

In SAS, the requirements for a PIDO interface are described in the abstract *PIDOInterface* class. To add support for additional PIDOs, an implementation for this class can be written and added to the software. Currently, only the interface for RCE is implemented in SAS.

For the implementation of the RCE interface, several necessary workarounds have been implemented to ensure the required functionality. Due to the implementation of these solutions, the developed interface for

SAS can be considered a development in itself, as it enables close interaction with RCE directly from Python. A few of the main difficulties when interfacing with RCE are:

- Limitations in the automated importing of CMDOWS files:
  - It is impossible to automatically start the execution of a workflow from a CMDOWS file, meaning manual interaction is needed to convert a CMDOWS file to an RCE executable workflow. The number of times that this is necessary, is minimized as much as possible.
  - Requested sample points in a DoE CMDOWS file are not imported correctly in RCE. For this purpose, a workaround has been developed that manually 'injects' design points into the native RCE workflow files.
- Design competences are supposed to be manually imported into RCE. This process has been automated in SAS, enabling the deployment of surrogate models within workflows without additional steps.

The potential use-cases of the developed interface are broader than the usage in SAS alone. The new interface presents an opportunity for MDAO architects to efficiently integrate and automate the formulation- and execution phase of an MDAO lifecycle. For future research, the interface could therefore be used as a useful tool to reduce manual and repetitive steps.

8.2. SURROGATEMODEL CLASS

The *SurrogateModel* class is one of the key developments for this work. The *SurrogateModel* class consists of one or more *Discipline* objects, which correspond to selected the SM strategy. The in-and output variables of the SurrogateModel object are calculated using the methodology from section 5.2, and the applicable hidden constraints in the underlying data are derived from its Discipline objects. The SurrogateModel object implements all of the methodology as described in chapter 7, where the building of the SMs, as well as predictions, are done using the Surrogate Modeling Toolbox (SMT) [23].

8.2.1. SM LIFECYCLE

The SurrogateModel class implements the automation of non-data-acquisition parts of the SM lifecycle from Figure 3.1. The building of the SMs and the prediction of points using the SMs is implemented using the Surrogate Modeling Toolbox (SMT) as developed by Bouhlel et al. [23]. The other logic, such as the retrieval of data, filtering of the hidden constraints as well as the training of the hidden constraints predictor, is all managed internally. The activity diagram describing the implemented software is shown in Figure 8.5.

8.2.2. DATA PROCESSING

In SAS, a method of data acquisition and processing compared to the "traditional" way is implemented. Normally, one would build a DoE, execute the DoE in a PIDO, extract the results from the PIDO and build a DoE based on the acquired data. However, this approach has a few limitations:

- Data for SM training is only available when it comes directly from the DoE. It only acquires the in- and output data for each sample, disregarding the in- and outputs of each individual execution of the design competences within the DoE. Therefore, for DoEs containing multiple design competences, a significant amount of data is left unused
- For each SM strategy, a new DoE will have to be formulated. Existing data from other, earlier executed DoEs, MDAs, or optimizations, cannot be used as originally the specifically required in- and output samples of design competence, were originally not stored

To conclude, the traditional approach lacks flexibility. One would like all data that is generated to be available and stored, as potentially this data can be useful in later stages. For this reason, SAS implements an approach where instead of only storing the in- and outputs of a DoE, all in- and outputs of each design competence within a run is stored. This requires some data processing to select the correct data when building an SM, but ensures the highest flexibility, and enables all previously generated data to be used.

The aforementioned database structure from subsection 8.1.3 is specifically designed with this in mind. All samples are fully tractable, as it is known which specific workflow execution, or run, they belong to, as well as which specific, ordered, sample they are in that run. Based on this information, samples from different

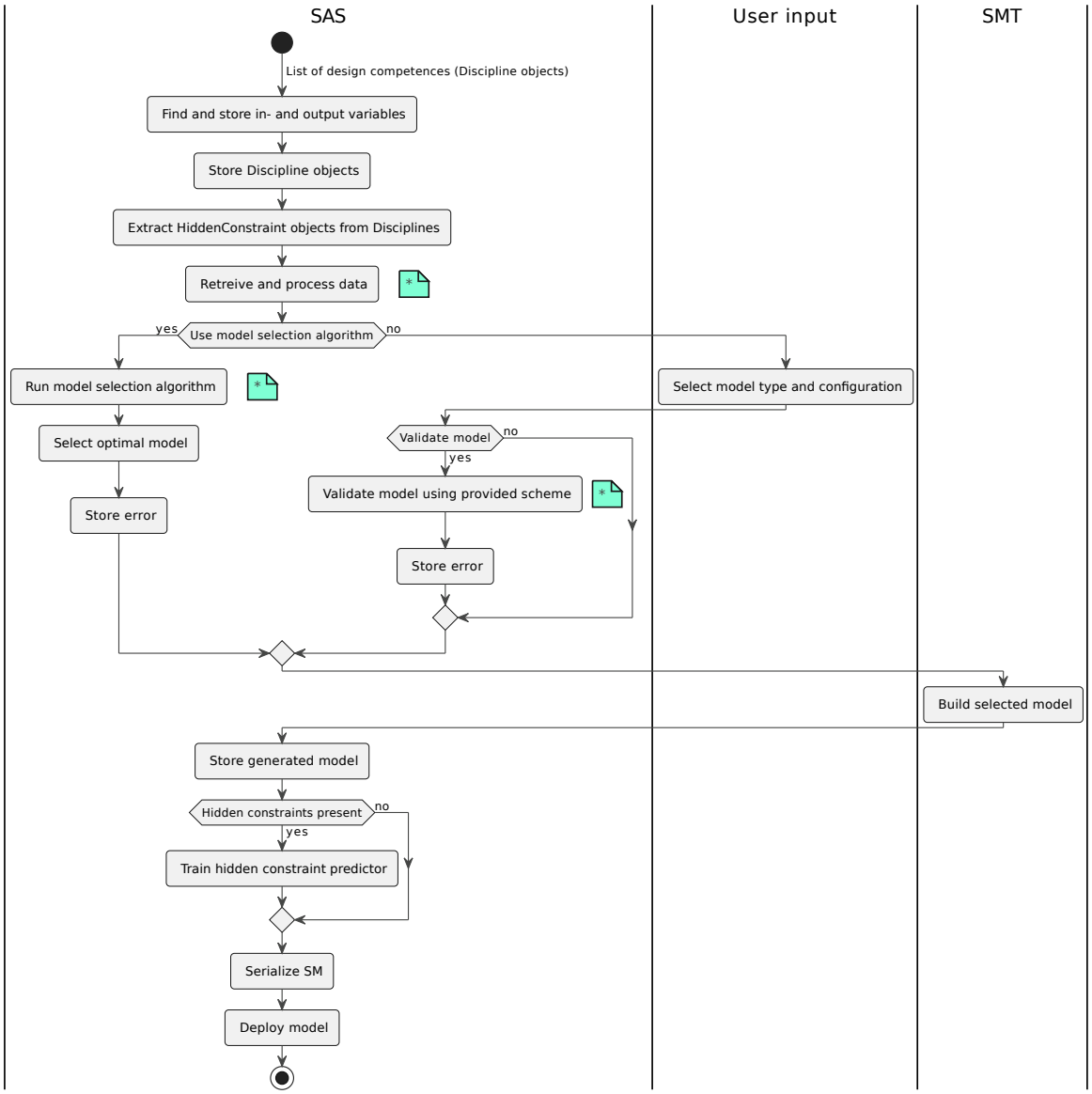


Figure 8.5: Acitivity diagram of the normal SM building procedure as implemented in SAS. The * indicate an abstraction, of which the algorithms can be found in this Thesis.

design competences can be matched afterwards, necessary for the generation of SMs that contain multiple design competences.

Especially for SMs containing a converger, this approach introduces some challenges. As all in- and outputs of each of the design competences are stored, it is unknown which specific samples are converged. In the traditional approach, only the converged samples would be returned as a result, in SAS' approach, this requires the post-processing of the samples. The solution is rather straightforward: for each sample, the run and ordered index in the run are known. In an SM with multiple design competences, for the first design competence, concurring samples containing identical values for their input variables, except for the to-be-converged variables, the sample with the highest ordered index will contain the converged values. This index is stored and used to extract the data of the following design competences. The process is described in Algorithm 2.

Algorithm 2 Algorithm for data processing in SM object

```
Get ordered design competences contained in SM
Request batched data from design competences (data grouped on run)
Determine supplying design competence of each in- and output variable
if Non-converged SM then
    Match runs and samples within runs between design competences
    For each match, retrieve the in- and output variables at the supplying design competence
    Store samples
else
    Select first design competence in SM
    Select non-coupling variables from design competence input
    for Run in available runs do
        For each set of identical inputs, select the last index
        Retrieve in- and output variables of supplying design competence at index
    end for
    Store samples
end if
if Hidden constraints present then
    Find constrained samples using hidden constraint flags
    Remove from valid sample set
else
    Add all samples to valid sample set
end if
```

8.2.3. DEPLOYMENT OF AN SM

Predictions using the SM happen in one or two steps, depending on whether hidden constraints are active in the SM. If a hidden constraint is active, the requested points are evaluated using the hidden constraints predictor, and if the hidden constraint is predicted to be active, its predefined actions are executed. If no hidden constraint is present or the defined point is predicted to be valid, the output is predicted using the SM.

Once an SM has been generated, it has to be separated from the Python session in order to be usable in other applications such as the PIDO. Furthermore, as discussed earlier, the goal is to create SM packages that are usable with a CDS. For this goal, the SurrogateModel object is serialized using Python's *pickle* package, and a new Python file is built that requires three inputs provided by the command line: 1) a *pickle* file containing the serialized SurrogateModel object, 2) an input file containing the CDS for the to-be-evaluated datapoint and 3) an output file that will contain the SMs output in the CDS format. The serialized SurrogateModel object and the new Python script are saved at a specified location, making the SM readily available for other applications.

III

VERIFICATION

9

TEST CASE I: SELLAR PROBLEM

In this chapter, the implemented methodology and developed algorithms will be applied to a modified version of the Sellar optimization problem. The Sellar optimization problem has become a standard test case over the years and is a simple mathematical problem consisting of two 'design competences', an objective function, and two constraints. The XDSM of the Sellar problem is displayed in Figure 9.1. The formulations of the design competences, objective function, and constraints are:

$$\begin{aligned}
\text{D1} &\Rightarrow y_1 = c \cdot \left(z_1^2 + x_1 + z_2 - 0.2 \cdot y_2\right) \\
\text{D2} &\Rightarrow y_2 = c \cdot \left(\sqrt{y_1} + z_1 + z_2\right) \\
\text{F} &\Rightarrow f = x_1^2 + z_2 + y_1 + e^{-y_2} \\
\text{G1} &\Rightarrow g_1 = \frac{y_1}{3.16} - 1 \\
\text{G2} &\Rightarrow g_2 = 1 - \frac{y_2}{24.0}
\end{aligned} \tag{9.1}$$

The design vector $\mathbf{x}$ is defined as

$$\mathbf{x} = \{z_1, z_2, x_1\}. \tag{9.2}$$

The optimization problem is formulated as

$$\begin{aligned}
\min \quad & f(\mathbf{x}) \\
\text{with respect to} \quad & \mathbf{x} \\
\text{subject to} \quad & g_1 \geq 0 \\
& g_2 \geq 0 \\
& -10 \leq z_1 \leq 10 \\
& 0 \leq z_2 \leq 10 \\
& 0 \leq x \leq 10 \\
& c = 1
\end{aligned} \tag{9.3}$$

For the verification of the work done in this Thesis, the computational cost of the workflow has been artificially increased by implementing a delay in the design competences. The first design competence, D1 will be delayed with 8 s, D2 will be paused for 6 s. The goal of the addition of the delay is to make the Sellar problem more representative of a possible encountered real-world problem by the software so that SAS' advisory capabilities can be verified. Furthermore, it is a good example of a workflow where significant gains can be achieved by using an SM, as will be shown in section 9.1. Here, SAS will be utilized to provide advice on possible SM strategies and SM accuracy levels. These strategies will be deployed using the functionality of the developed software, and the results will be analyzed.

9.1. IMPROVING COMPUTATIONAL EFFICIENCY

In this section, the Sellar problem will be analyzed and an attempt is made to reduce its execution time by implementing SM strategies. In the first steps, a provided workflow containing the Sellar problem formulation

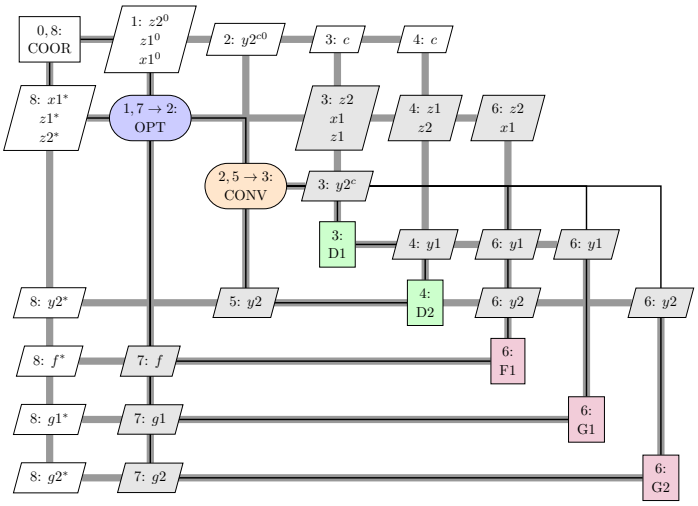


Figure 9.1: XDSM of the Sellar problem, as used in the first test case.

is analyzed using the profiling algorithm and the bottlenecks will be identified in subsection 9.1.1. Based on this workflow profile, an analysis will be made and three advised SM strategies emerge, as is discussed in subsection 9.1.2. These strategies will be implemented for different levels of accuracy, and their results are compared in subsection 9.1.3. An activity diagram of the implemented experiment is shown in Figure A.2.

9.1.1. IDENTIFICATION OF BOTTLENECKS

SAS is initiated by providing a CMDOWS file of the Sellar optimization problem. The design variables, design competences, and corresponding in- and output variables are extracted from the workflow and the connection to the PIDO is initiated. To find the bottlenecks in the workflow, the profiling algorithm is executed for $N_s = 3$ samples. This number is selected arbitrarily and is a decision of the MDAO architect. More samples will most likely lead to a better prediction but require a larger time investment. Running the profiling for fewer samples is more time-efficient, but can lead to wrong estimations. It is recommended to run the profiling for at least three points, to evaluate different areas of the design space. Within the algorithm, the workflow is converted to a DoE, as shown in Figure 9.2, and executed in the PIDO for the selected number of samples.

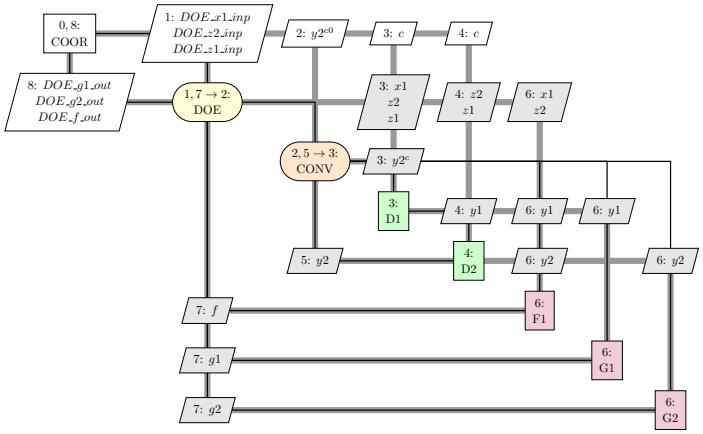


Figure 9.2: XDSM of the DoE for the Sellar problem, as will be used for the profiling of the workflow.

After the workflow execution, the produced data is processed: a timeline of the execution is extracted and all in- and output samples of the included design competences are stored in the design database. Based on the timeline, the analysis on bottlenecks can be made and is displayed in Figure 9.3, where the normalized runtimes are shown. As expected, it can be seen that both D1 and D2 are the most expensive design competences in the workflow by a significant margin.

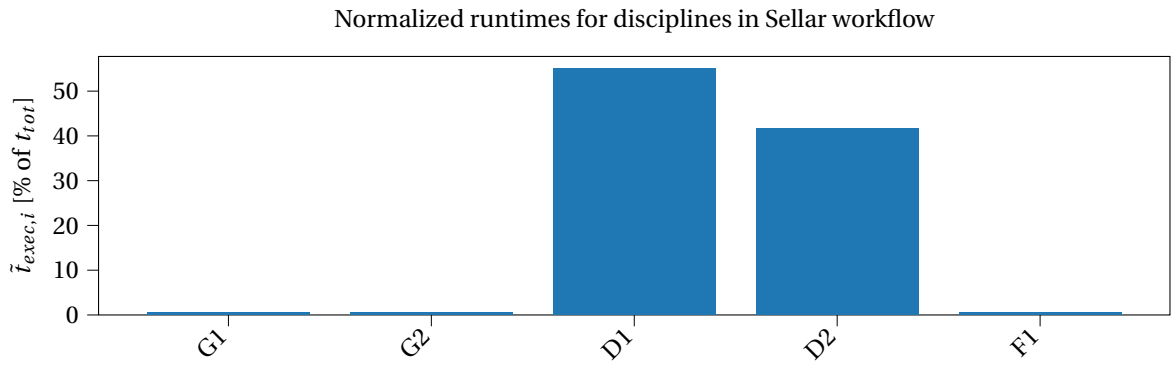


Figure 9.3: Results of the profiling run for the Sellar workflow.

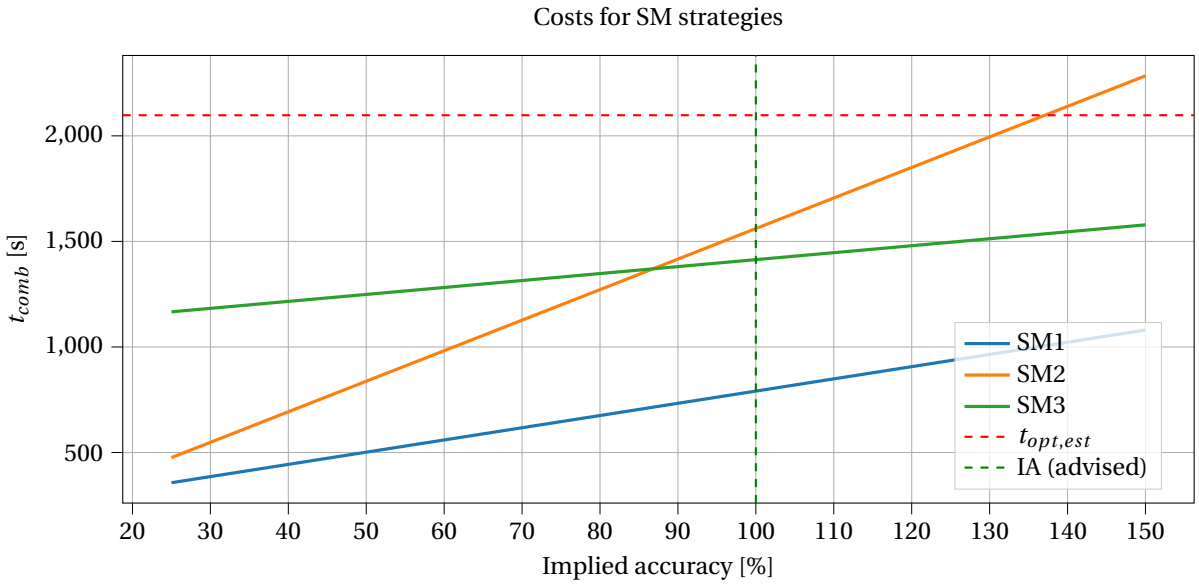


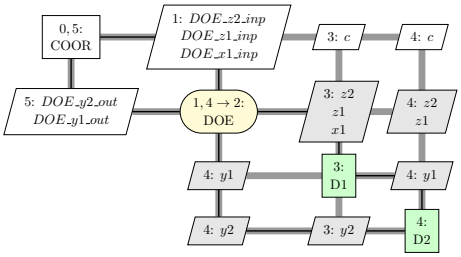
Figure 9.4: Estimated total combined times for data acquisition and SM-based optimization for the top three SM strategies, compared to the estimated non-SM-based optimization

9.1.2. ADVISING ON SURROGATE MODEL STRATEGIES

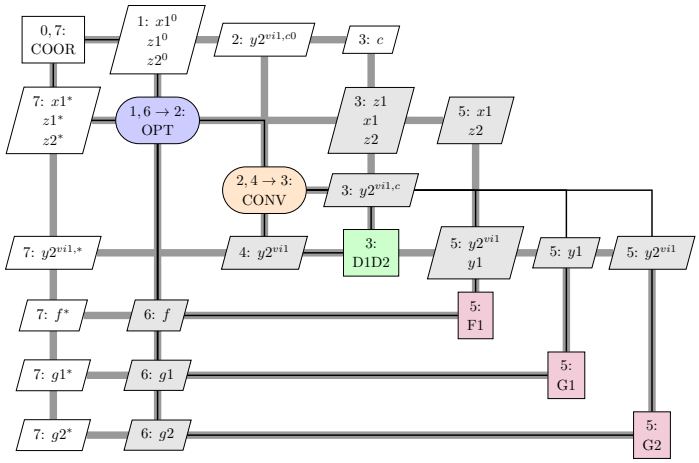
Based on the profiling run advice on SM strategies (a design competence, or group of design competences that is replaced by an SM) can now be formulated. Using the implemented methodology for the analysis module, as discussed in section 6.2, an estimation can be made of the consequences of a specific strategy.

For the analysis module, an estimation of the number of objective function calls for the optimization is required. Alternatively, an estimation of the required time for optimization can also be provided. In this example, an initial estimation of $N_{obj} = 42$ is used. This value comes from the execution of the full optimization, which is still feasible to execute for the weighted Sellar problem. Normally, the estimation of these parameters will require expertise. The resulting advice is shown in Figure 9.4. The underlying data, containing all possible strategies for the Jones advisor, is added in section A.3.

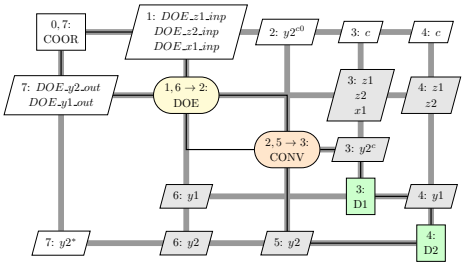
The top three calculated strategies are shown in the figure and in Table 9.1. The order is determined by sorting the strategies for the *Jones*-advisor, hence $N_{s,advised} = 10N_v$, and for the 100% sample advice. As could be expected for the artificially weighted D1 and D2 design competences, there are all included in the optimal SM strategies. For the provided workflow, the strategy where D1 and D2 are replaced by a single SM appears to be the most efficient. Secondly, the replacement of the complete convergence loop including D1 and D2 is deemed to also be an effective strategy to reduce the computational expense of the workflow. Lastly, the calculations show that only replacing $D1$ with an SM also shows potential benefits for the overall workflow execution.



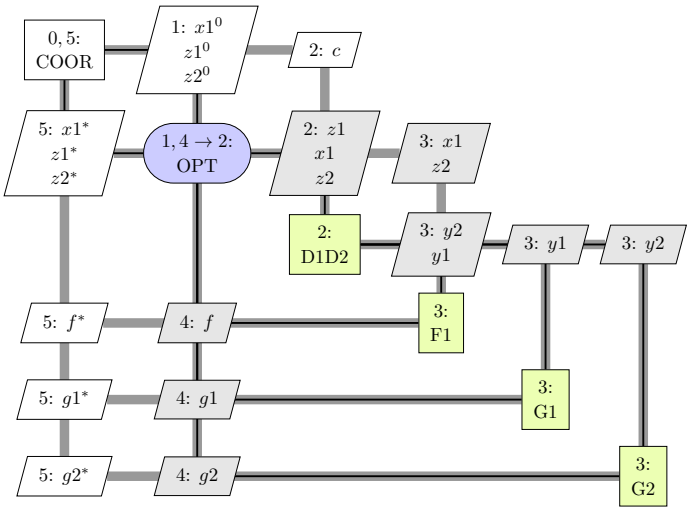
(a) DoE: D1 and D2 are replaced by a non-converged single SM



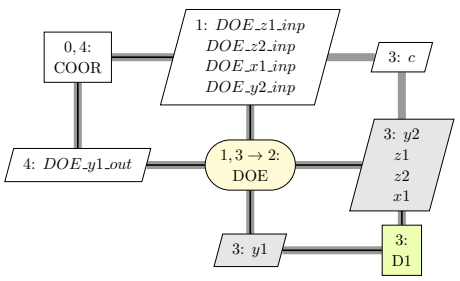
(b) SM-based optimization: D1 and D2 are replaced by a non-converged single SM



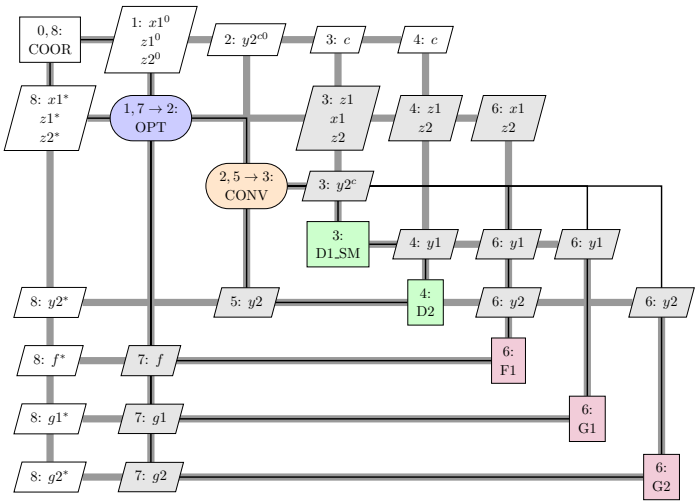
(c) DoE: D1 and D2 are replaced by a converged, single SM



(d) SM-based optimization: D1 and D2 are replaced by a converged, single SM



(e) DoE: D1 is replaced by an SM



(f) SM-based optimization: D1 is replaced by an SM

Figure 9.5: Overview of the generated workflows for the generation of three different SM strategies. The left column shows the DoEs for the strategies, the right the new, SM-based optimization workflows.

Identifier	SM strategy	Input variables	Output variables
SM1	D1, D2	z_1, z_2, x_1, y_2	y_1, y_2
SM2	Converger, D1, D2	z_1, z_2, x_1	y_1, y_2
SM3	D1	z_1, z_2, x_1, y_2	y_1

Table 9.1: The top three advised SM strategies for the Sellar problem.

9.1.3. IMPLEMENTATION AND RESULTS

METHODOLOGY OF VERIFICATION

To verify the advisor's results, all top three strategies are implemented and their results are compared. Each of the strategies will be executed for a different number of samples, as to investigate the accuracy of the discussed 'rules of thumb' for the SM accuracy with respect to the number of samples. For each of the selected SM strategies, a DoE is generated using the software and are displayed in Figure 9.5a, 9.5c and 9.5e.

The DoEs are executed for a varying number of samples, corresponding to an IA,

$$IA = \frac{N_{s,realized}}{N_{s,advised}} \cdot 100\% \tag{9.4}$$

of 50%, 75%, 100%, 125% and 150% as calculated using the Jones sample advise $N_s = 10 \cdot N_v$. Each of the sample sets will be generated as an LHS, without the addition of an adaptive sampling scheme. Furthermore, the standard Kriging model will be used as the SM type for all the generated SMs, so the automated surrogate selection algorithm will not be used. Each generated SM will be validated using a k-fold validation scheme using $k = 5$ and the RMSPE metric. Additionally, the SMs are validated using a Leave-One-Out validation scheme, which is a k-fold validation scheme where $k = N_s$, and again the RMSPE metric is calculated.

For each generated SM, the SM is deployed in the workflow and the SM-based Sellar optimization is executed. For this purpose, the original workflow is automatically converted to the SM-based workflow, and for the different SM strategies, the resulting XDSM diagrams are displayed in Figure 9.5b, 9.5d and 9.5e. Additionally, to be represented in the XDSM format, the workflows are also produced in the CMDOWS format and can be imported into RCE. The CMDOWS files are (manually) imported into RCE, and executed directly from SAS.

The SM-based optimization produces an optimized objective value, as well as the optimal design vector. As to find the accuracy of the SM at the found optimum, the found optimal design vector is tested in the full, non-SM-based workflow. This results in a non-SM-based objective value, which can be compared to the SM-based objective value. Ideally, these values are almost similar: this means the SM-based optimization produces a valid design. The difference between the SM-based optimal objective value and the non-SM-based objective value at the SM-based optimal design vector $\mathscr{E}_{obj}$, is denoted as

$$\mathscr{E}_{obj} = \frac{f_{SM}(\mathbf{x_{opt,SM}}) - f(\mathbf{x_{opt,SM}})}{f(\mathbf{x_{opt,SM}})}, \tag{9.5}$$

where $f_{SM}()$ denotes the SM-based workflow, and $f()$ the non-SM-based workflow.

Another point of comparison for the found objective value using the SM-based optimization is its difference to the optimal objective value found using the full, non-SM-based optimization. In other words, it is a measure of how well the SM-based optimization is able to find the optimum as calculated in the non-SM-based optimization. To find the non-SM-based objective value, the full non-SM-based workflow is executed once and the optimal design vector $\mathbf{x_{opt}}$ and corresponding objective function value $f(\mathbf{x_{opt}})$ are stored. The difference between the objective value found using the SM-based workflow and the non-SM-based workflow $\mathscr{E}_{opt}$, is now defined by

$$\mathscr{E}_{opt} = \frac{f_{SM}(\mathbf{x_{opt,SM}}) - f(\mathbf{x_{opt}})}{f(\mathbf{x_{opt}})}. \tag{9.6}$$

RESULTS

The results of the implementation of the different SM strategies are discussed in this subsection. For the first strategy, where D1 and D2 are replaced with a non-converged SM, the results are displayed in Figure 9.6. The second strategy, where D1, D2, and the converger are replaced by an SM, is shown in Figure 9.7. Lastly, the results of the strategy where only D1 is replaced are shown in Figure 9.8.

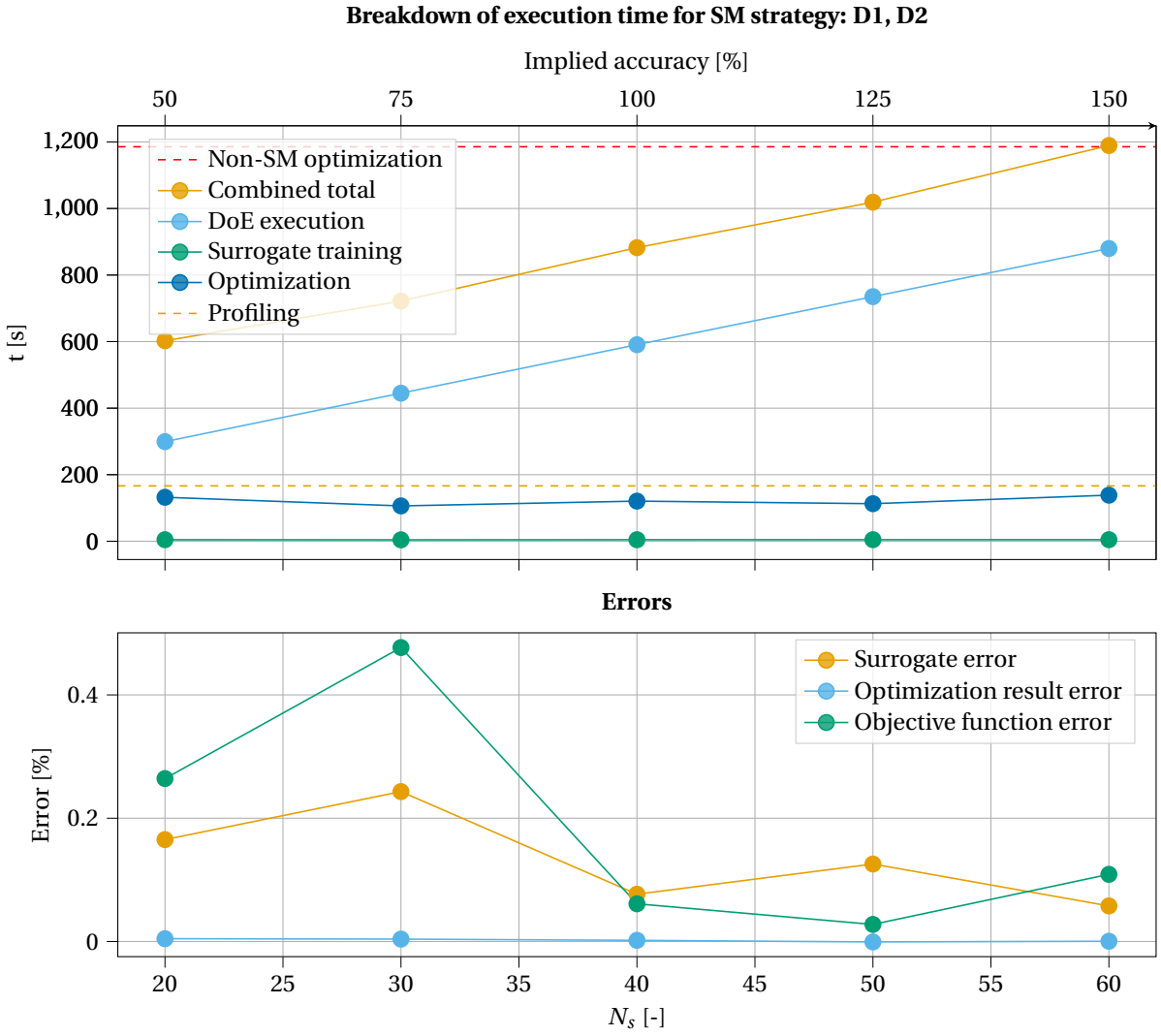


Figure 9.6: Results of the replacement of D1 and D2 with an SM.

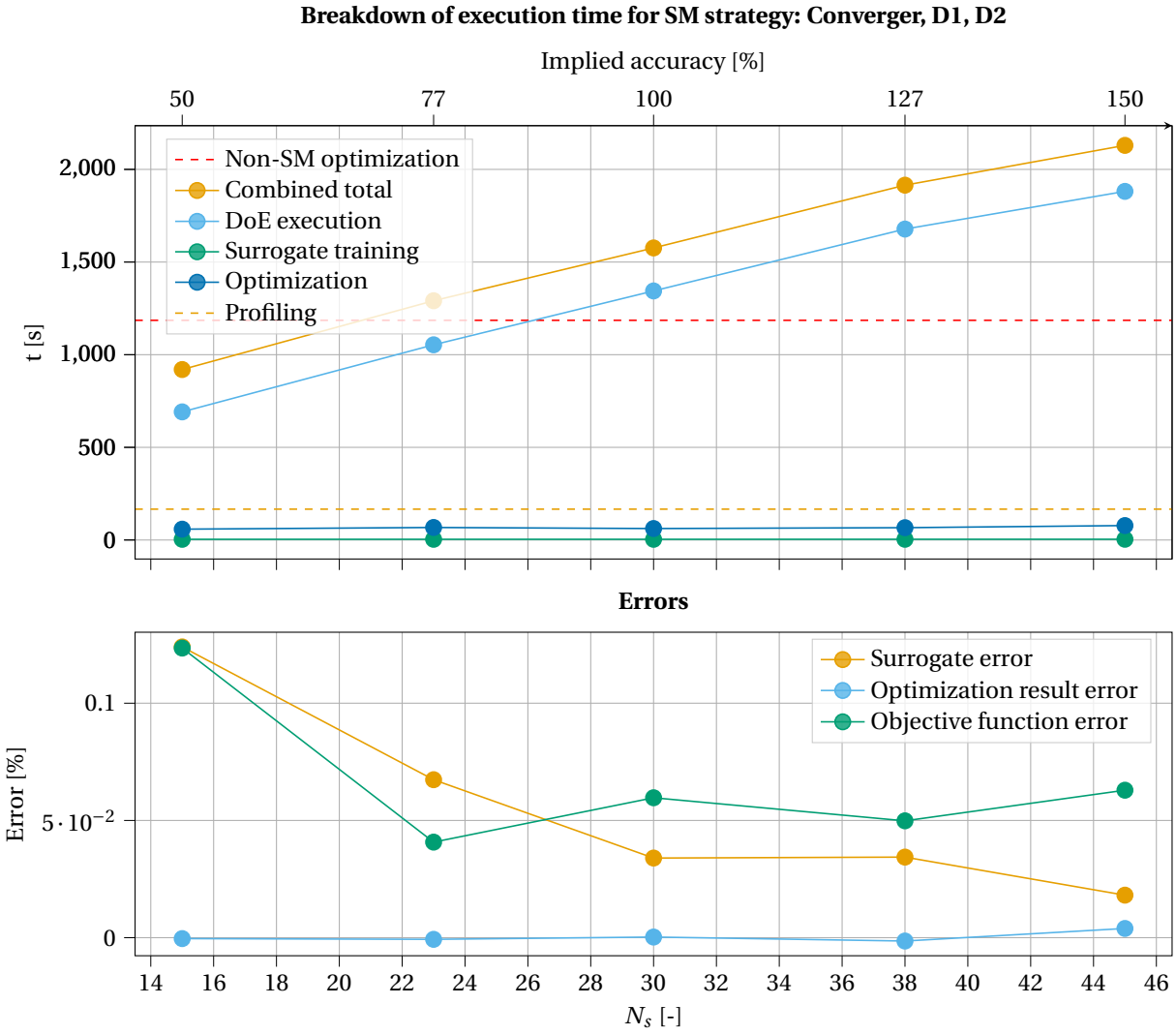


Figure 9.7: Results of the replacement of the converger, D1 and D2 with an SM.

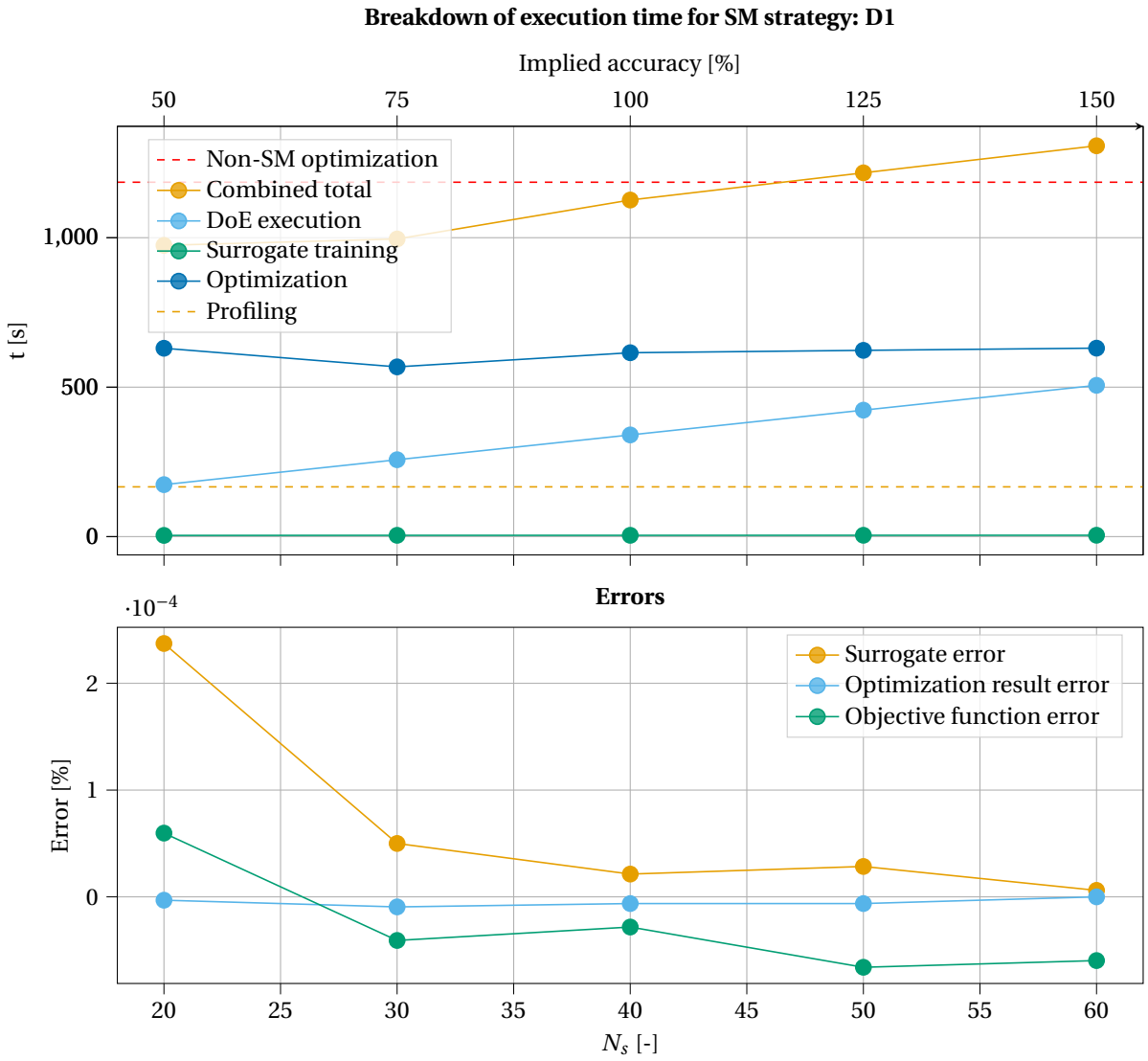


Figure 9.8: Results of the replacement of D1 with an SM.

In all figures, it can be seen that a reduction of execution time is achieved until a certain number of samples N_s. Furthermore, as expected, the surrogate error reduces as the sample size increases.

Another observation is that the optimization error $\mathscr{E}_{opt}$ does not reduce with the number of samples, independent of the surrogate strategies. This can be explained by the fact that the optimization is limited by the $G1$ constraint in combination with the objective function. From the objective function, it can be seen that the goal would be to minimize the values for x_1, z_2, and y_1, while y_2 should be maximized, as it minimizes the e^{-1} component. Where x_1 and z_1 are design variables, y_1 is calculated by the SM and therefore prone to errors in the model. However, as the value is limited by the constraint $G1$ to $y_1 < 3.16$, this value is still reached and therefore the optimization still produces the value near its non-SM-based optimum.

There are large differences in the performance of the SM strategies, in both the surrogate error $\mathscr{E}_{SM}$ and the objective function error $\mathscr{E}_{obj}$. Although it is not the most computationally efficient option, it can be seen that the strategy where $D1$ is replaced with an SM achieves the highest accuracy. Even for the smallest N_s at $IA = 50\%$ or $N_s = 20$, the small error RMSPE of $2.37 * 10^{-}4$ is achieved. As D1 is a simple function to model, as it is almost a linear combination of its parameters, this behavior is to be expected.

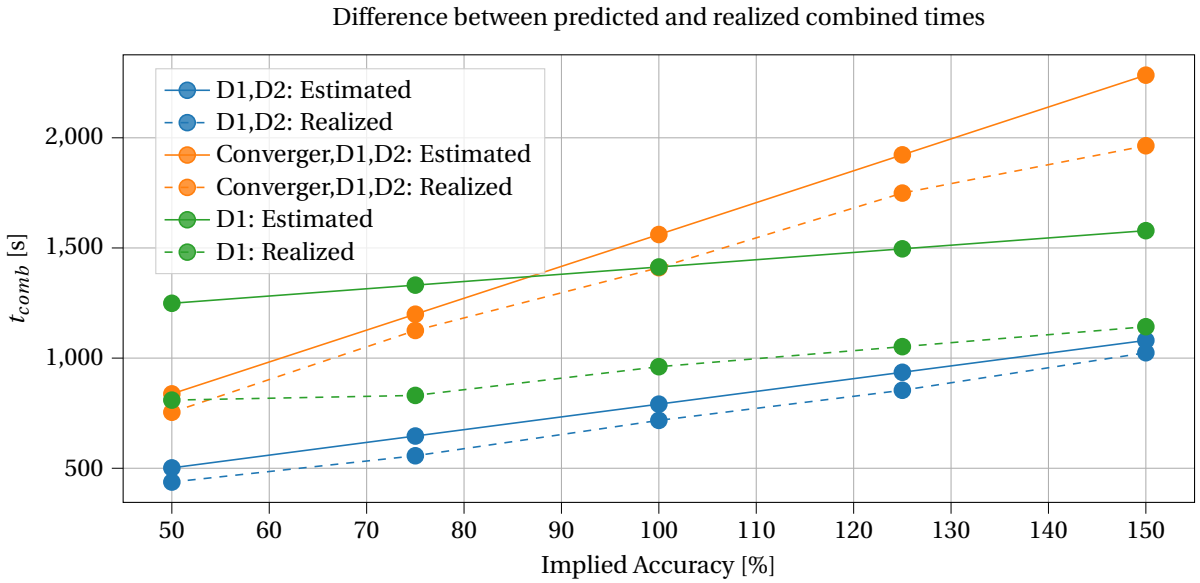
The results also show that it is more difficult to model D1 and D2 with an SM. The main factor contributing to this is a large input space. Due to the nature of the design variables, it can be expected that, $0 \leq y_1 \leq 120$. As the sample advice indicates $N_s = 40$, this will lead to a significant distance between samples. As the goal is to minimize y_1, as discussed earlier, it is likely the behavior of the combination of D1 and D2 is not modeled accurately enough in that area. Most likely, an infill strategy is needed.

PERFORMANCE OF THE PROVIDED ADVISE

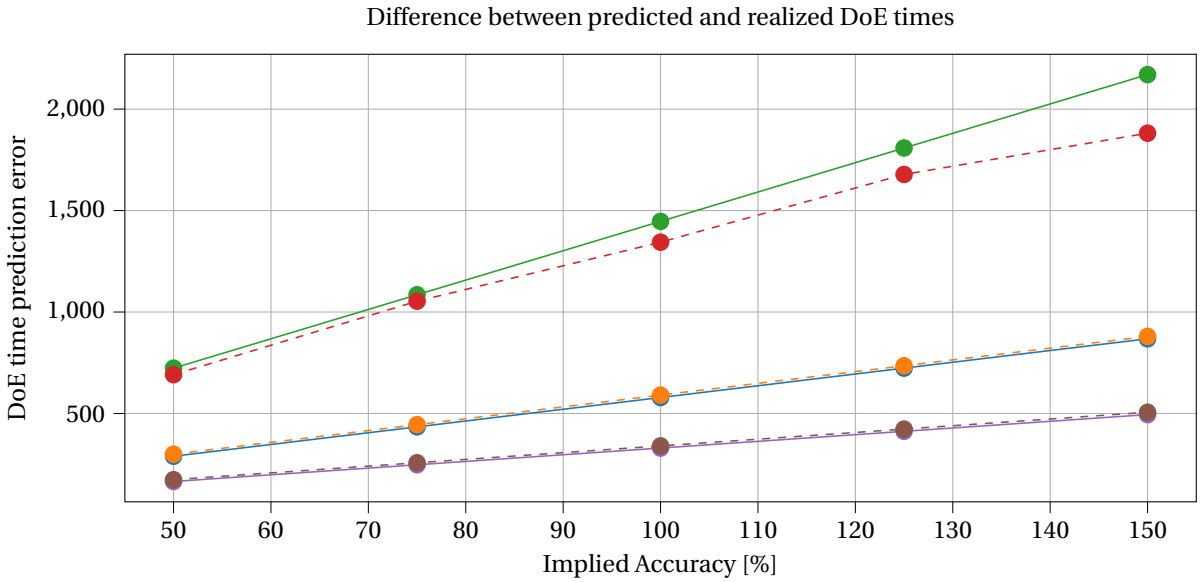
In Figure 9.9a, and comparison is made between the estimated and realized combined runtimes for the Sellar problem. It can be seen that the SM strategy containing both D1 and D1, and the SM strategy containing the complete convergence loop, are reasonably accurately predicted. The SM strategy only containing D1, is not.

To get a deeper understanding, Figure 9.9b, shows the difference between the expected and realized times needed for the execution of the DoE. In this figure, it is clear that a good prediction is made on the runtimes for the DoE execution. This inherently means that the time required for optimization is significantly too large for the D1 SM strategy. This proves that the estimation of optimization performance, certainly for workflows where optimizations stay the mayor contributor to the overall runtimes such as for the D1 SM strategy, is difficult.

This is also acknowledged by the fact that the full, non-SM-based optimization is much shorter than initially expected. This is caused by a large deviation of the required number of samples during optimization, compared to the number of iterations for a DoE. In a DoE, larger steps are taken in the design space, often leading to more required iterations in the converger. In the profiling run, these iterations for optimization are overestimated and therefore the estimated optimization runtime is significantly overestimated as well.



(a) Combined runtimes



(b) DoE runtimes

Figure 9.9: Difference between estimated and realized runtimes for the Sellar SM strategies.

10

TEST CASE II: MAXIMUM TAKE-OFF MASS

In this chapter, the developed methodology is applied to the Maximum Take-Off Mass (MTOM) optimization problem. In the MTOM problem, the maximum take-off weight $W_{to,max}$ (and corresponding maximum take-off mass $m_{to,max}$, $W_{to,max} = m_{to,max} \cdot 9.81$) of an airplane is minimized by the optimization of a wing. A proposed design is analyzed using a viscous and inviscid aerodynamic calculation, and structural and performance analysis. The optimization problem is formulated using an MDF Gauss-Seidel architecture, of which the resulting XDSM diagram is displayed in Figure 10.1. The constrained optimization problem is formulated as:

$$\begin{aligned} \min \quad & W_{to,max}(\mathbf{x}) \\ \text{with respect to} \quad & \mathbf{x} \\ \text{subject to} \quad & \mathbf{x}_{LB} \leq 0 \leq \mathbf{x}_{UB} \\ & \left(\frac{W_{TO,max}}{S}\right) - \left(\frac{W_{TO,max}}{S}\right)_{ref} \leq 0 \\ & V_t - V_f \leq 0 \end{aligned} \tag{10.1}$$

Here, x denotes the design vector, and $\mathbf{x}_{LB}$ and $\mathbf{x}_{UB}$ are the lower and upper bounds of the design vector, respectively. The used design variables, initial values, and defined bounds are displayed in Table 10.1. S denotes the surface of the wing, and the second constraint ensures the wing loading for a calculated design cannot exceed the wing loading of the original design. Lastly, V_t and V_f are the parameters for the tank volume, and the fuel volume respectively, and the constraint ensures the required calculated fuel load can be fitted in the wing.

Design variable	**Lower bound**	**Nominal value**	**Upper bound**
$\Lambda_{inner,LE}$ [°]	15.435	22.05	28.665
c_{root} [m]	6.58	9.4	12.22
ϵ_{mid} [°]	-3.25	-2.5	-1.75
ϵ_{tip} [°]	-3.25	-2.5	-1.75
b_{outer} [m]	9.352	13.36	17.368
c_{tip} [m]	1.925	2.75	3.575
$\Lambda_{outer,LE}$ [°]	21.35	30.5	39.65

Table 10.1: Design variables for the MTOM optimiation problem. All values are bounded within + – 30% from their nominal value.

All design competences in the MTOM problem make use of a CDS. The objective function is calculated in the *WeightAnalysis* discipline, that calculates the maximum take-off weight $W_{TO,max}$ as

$$W_{TO,max} = W_{AW} + W_{str} + W_F. \tag{10.2}$$

Here, W_{AW} is the (constant) weight contribution of the no-wing-attached airplane. The structural weight W_{str} is calculated in the *StructuralAnalysis* design competence using the EMWET tool, a quasi-analytical tool

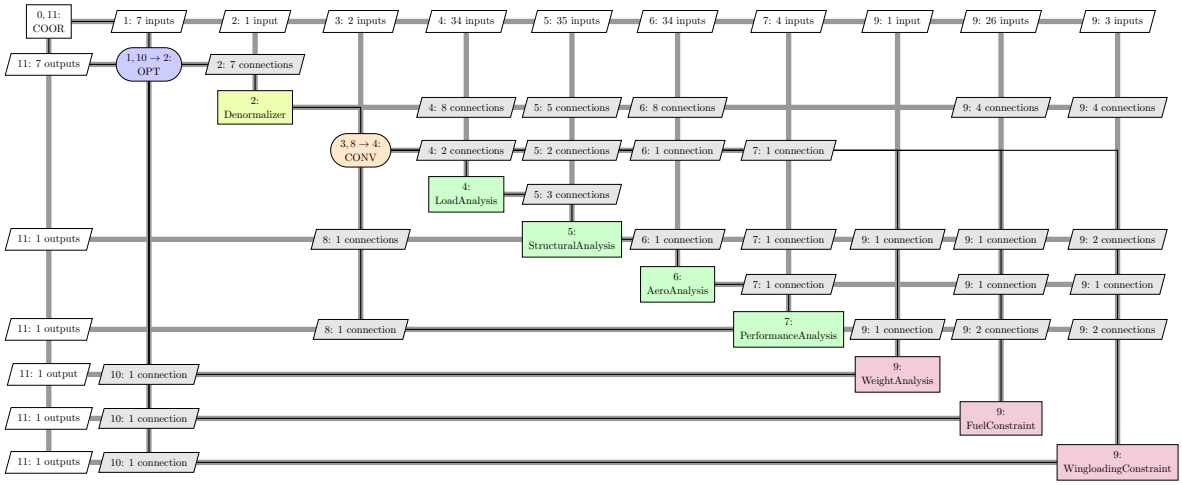


Figure 10.1: XDSM of the MTOM problem. A full, non-summarized version of the XDSM including all variable connections, can be found in Figure A.1.

developed by Elham et al. [84]. The tool is capable of accurately calculating the required structural mass for a wing design and corresponding wing loading. The loads on a designed wing are calculated in the *LoadAnalysis* discipline, which makes use of the Q3D aerodynamic solver [85]. Q3D makes use of a vortex lattice method code and is capable of calculating the viscous drag and wing loads for a wing. For the LoadAnalysis, Q3D is executed in the non-viscous mode and produces the lift distribution over the wing planform. To calculate the lift over drag ratio L/D, needed for the performance calculations, Q3D is later executed in viscous mode in the *AeroAnalysis* design competence.

The fuel weight W_F is calculated in the *PerformanceAnalysis* discipline. In this design competence, the Breguet equations for the estimation of range and fuel weights have been implemented, following the proposed methodology of Roskam [86]. The range of the complete design is given by

$$R = \frac{V_{cr}}{C_T} * \frac{L}{D} * ln\left(\frac{W_{start-cr}}{W_{end-cr}}\right), \tag{10.3}$$

where R is the range in nautical miles, which is considered to be fixed for this design problem, as are the velocity V and specific fuel consumption C_T. The drag D component in the L/D, consists of both the wing drag and the drag of the no-wing-attached airplane $C_{D,aw}$, which is calculated before the workflow is started. The drag coefficient for the full airplane is now calculated found by

$$C_D = C_{D,w} + C_{D,aw} * \frac{S_{ref}}{S}, \tag{10.4}$$

where $C_{D,w}$ is the calculated value from the AeroAnalysis discipline, S the designed wing area, $C_{D,aw}$ the no-wing-attached drag of the reference airplane, and S_{ref} the wing area of the reference airplane.

For a pre-determined range R, the ratio between the fuel weight at the start and end of the cruise phase, $\frac{W_{start-cr}}{W_{end-sr}}$, is now calculated by rewriting Equation 10.3, hence

$$\frac{W_{start-cr}}{W_{end-sr}} = exp\left(R\frac{C_T}{V_{cr}}\frac{C_D}{C_L}\right). \tag{10.5}$$

Using this ratio, the required fuel weight W_f is now calculated using

$$W_F = \left(1 - 0.938\frac{W_{end-sr}}{W_{start-cr}}\right) * W_{TO,max}, \tag{10.6}$$

where 0.938 is the factor that accounts for the fuel usage in the non-cruise phases of the flight. In the *FuelConstraint* the required volume for W_F is compared to the volume of the fuel tank, ensuring that the required fuel can be carried. Lastly, the *WingLoading* discipline ensures the wing loading does not exceed the wing loading of the original design.

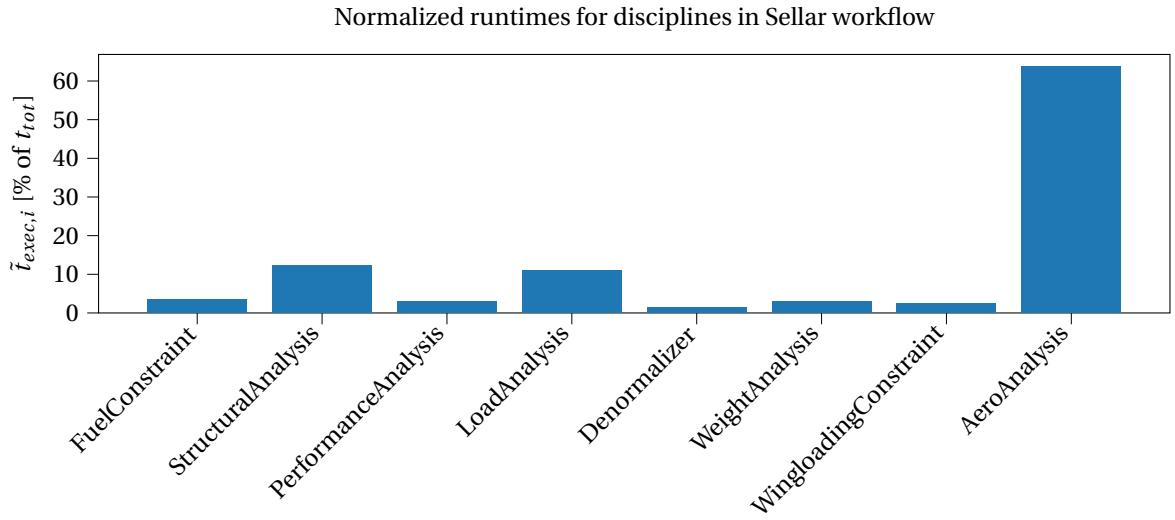


Figure 10.2: Normalized runtimes for the MTOM optimization problem, generated using a DoE and $N_s = 4$.

10.1. SCENARIO I: IMPROVING COMPUTATIONAL EFFICIENCY

In the first scenario using the MTOM problem, the same experimental setup is used as in the assessment of the Sellar problem from chapter 9. The MTOM optimization workflow is analyzed for its bottlenecks and based on this analysis, an advice is formulated for SM strategies to be implemented in the workflow. However, as a slight deviation from the Sellar problem, it is decided to not only implement the most promising SMs but to implement three different categories. First of all, all the disciplines within the convergence loop are replaced with an SM. Secondly, the complete convergence loop, so including the converger, is replaced. Lastly, only the AeroAnalysis discipline is replaced with an SM.

10.1.1. IDENTIFICATION OF BOTTLENECKS AND ADVISING ON SM STRATEGIES

As for the Sellar test case, the originally provided MTOM optimization formulation is converted to a DoE, and executed for $N_s = 4$. This number equals half of the input space for the MTOM problem, where $N_v = 8$, and is selected as a trade-off to efficiently analyze the workflow, while still having a reasonable number of samples to analyze the workflow behavior. The results of the analysis, in the form of the normalized runtimes of the design competences, are shown in Figure 10.2.

In the figure, it can be seen that a clear bottleneck is found in the AeroAnalysis design competence. It contributes to more than 60% of the overall runtime in the profiling run, indicating a clear candidate for a potential SM. Using the generated workflow analysis, and by making the estimation that $N_{obj} = 49$ for the eventual optimization, an advice can now be formulated. The raw data for this advise is shown in section A.4, and is visualized in Figure 10.3. Five different strategies are shown, of which the details are presented in Table 10.2. The first four, are the four calculated best strategies when using the Jones advisor ($N_v = 10 \cdot N_S$), and sorting for $IA = 100\%$. For validation purposes, additionally, a fifth SM strategy is added, where the complete convergence loop would be replaced with an SM.

It can be seen that SM1 and SM2 are very similar in their expected performance, which is to be expected when looking at the difference between the two strategies. In SM2, the PerformanceAnalysis design competence is omitted from the strategy compared to SM1, which is a very small contributor to the overall computational expense and therefore has little impact on both the required training time for an SM strategy, as well as for the eventual optimization performance. Similarly, SM3 and SM4 are also almost equal in their performance, for the same reason.

For validation purposes, three distinctly different strategies are selected for implementation. The SM1, SM4, and SM5 strategies are further investigated.

Identifier	SM strategy	Input variables	Output variables
SM1	LoadAnalysis StructuralAnalysis AeroAnalysis PerformanceAnalysis	$\mathbf{x}$, W_{str}^c, W_F^c	W_{str}, W_F
SM2	LoadAnalysis StructuralAnalysis AeroAnalysis	$\mathbf{x}$, W_{str}^c, W_F^c	L/D
SM3	AeroAnalysis PerformanceAnalysis	$\mathbf{x}$, W_{str}^c, W_F^c	W_F
SM4	AeroAnalysis PerformanceAnalysis	$\mathbf{x}$, W_{str}^c, W_F^c	L/D
SM5	Converger LoadAnalysis StructuralAnalysis AeroAnalysis PerformanceAnalysis	$\mathbf{x}$	W_{str}, W_F

Table 10.2: Five advised SM strategies for the MTOM problem. SM1 to SM4 are the calculated most optimal SM strategies, SM5 is manually added to investigate the behaviour of a converged SM strategy.

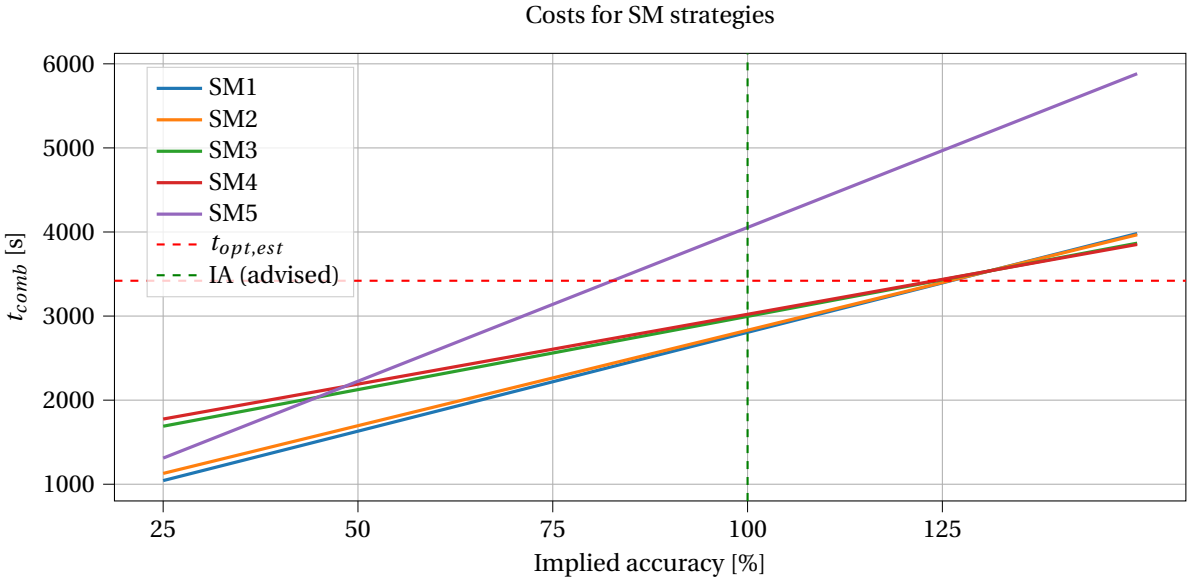


Figure 10.3: SM strategy advise for the MTOM optimization problem.

10.1.2. RESULTS

In this section, the results of the implemented SM strategies are discussed. For SM1, the results are shown in Figure 10.4, for SM4 in Figure 10.5 and for SM5 in Figure 10.6. The error measures used are identical to the ones in the Sellar test case, where the surrogate error is the RMSPE calculated using a k-fold validation scheme using $k = 5$, the objective function error by Equation 9.5 and the optimization error by Equation 9.6. The detailed data of the optimization performance of the SM strategies is shown in Table 10.3.

A few observations can be made. First of all, it is observed that the expected computational gains from Figure 10.3, are not achieved. A further discussion on the reasons for this will later be discussed, in the next section.

Another observation that is made, is that the predictive performance of the SMs is sufficient. For all SMs, at $IA = 100\%$, the errors are within 0.05 or 5%. But, for non of the SMs, this goal is reached within the time that would be needed for the non-SM-based optimization. However, for lower IAs, a reduction in the required computational times is achieved for SM1 and SM4. For SM1, when including the time required for

IA	$W_{TO,max,SM}(\mathbf{x}_{\mathbf{opt,SM}})$			$W_{TO,max}(\mathbf{x}_{\mathbf{opt}})$			Non-SM-based Objective
	SM1	SM2	SM3	SM1	SM2	SM3	
25%	0.928	0.929	0.898	0.952	0.929	0.938	
50%	0.930	0.909	0.913	0.920	0.925	0.964	
75%	0.913	0.919	0.921	0.942	0.916	0.942	0.900
100%	0.901	0.926	0.892	0.930	0.926	0.935	
125%	0.889	0.884	0.905	0.938	0.924	0.910	
150%	0.908	0.925	0.890	0.924	0.924	0.909	

Table 10.3: Optimization results by implementing the SM strategies.

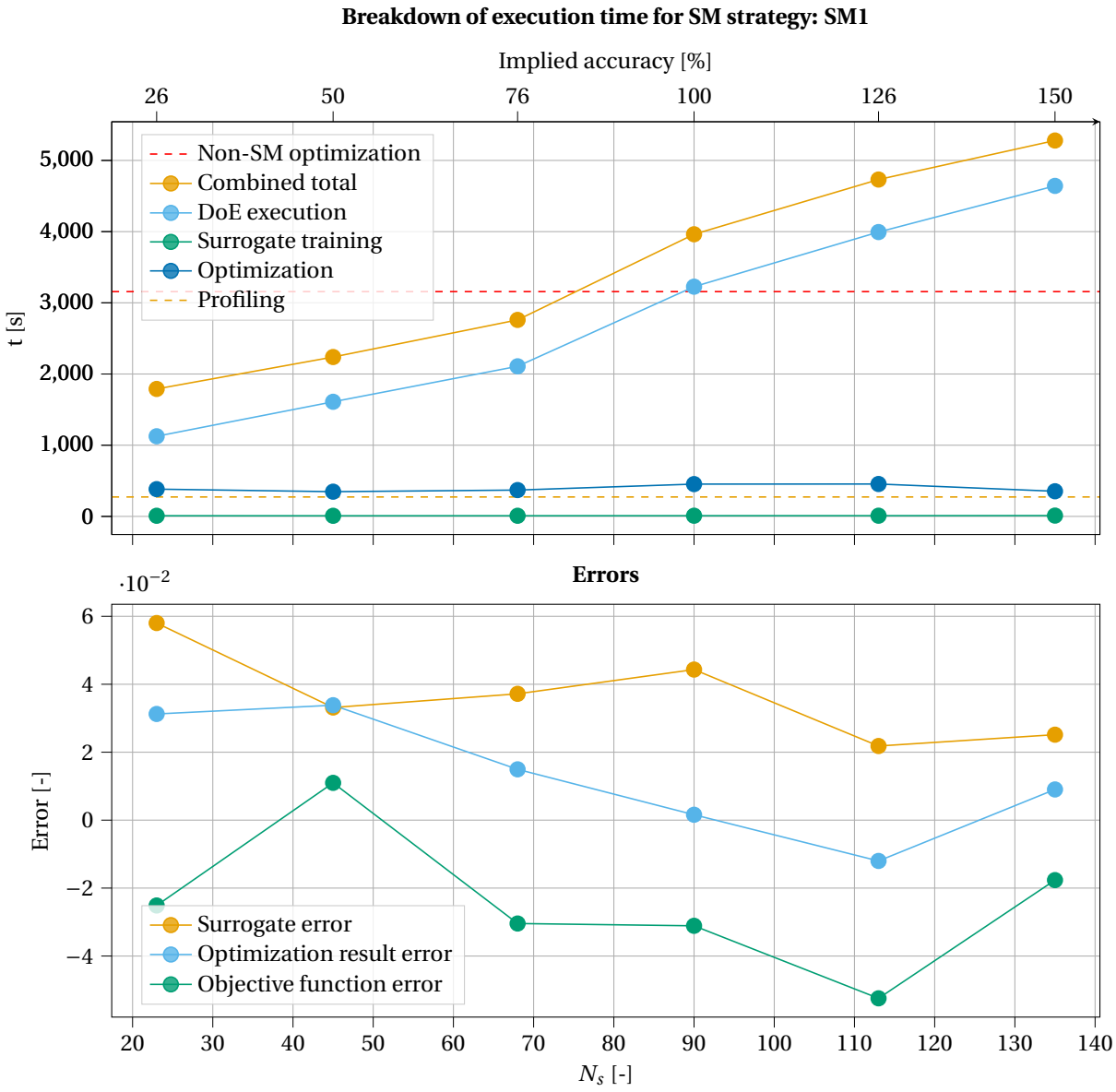


Figure 10.4: Results of implementation of SM1.

the profiling of the workflow, $\Delta t_{opt} = -398s$, a 12.6% reduction in execution time over the non-SM-based optimization.

A comparison between the optimal design vectors of the SM-based optimization for the SM strategies, and the full, non-SM-based optimization, is shown in Figure 10.7. It can be seen that there is significant

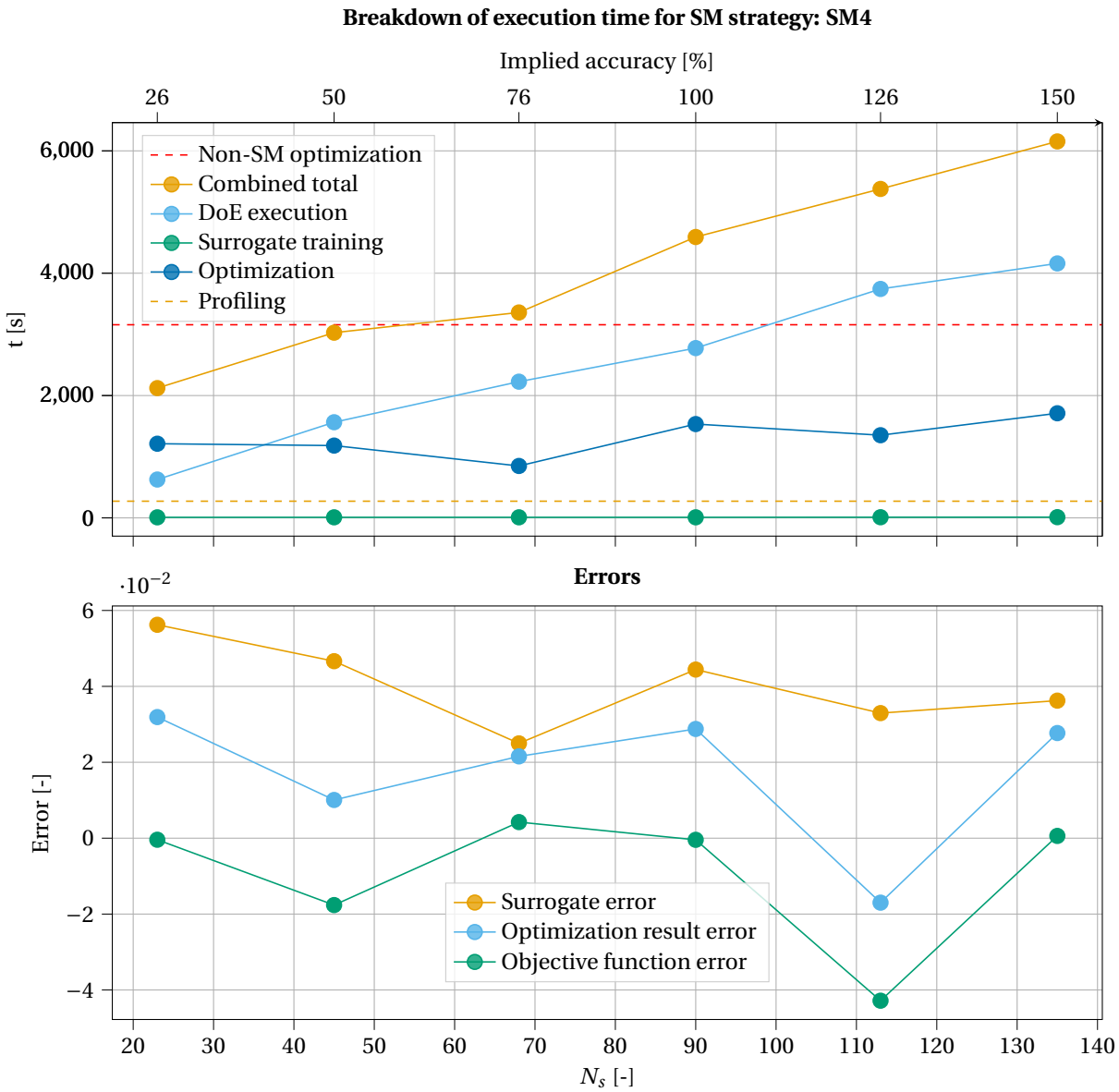


Figure 10.5: Results of implementation of SM4.

variation between the values for the variables in the optimal design vectors. Some driving variables, such as the root chord length c_{root}, are generally modeled well, while other variables such as the tip chord c_{tip} show a greater variation.

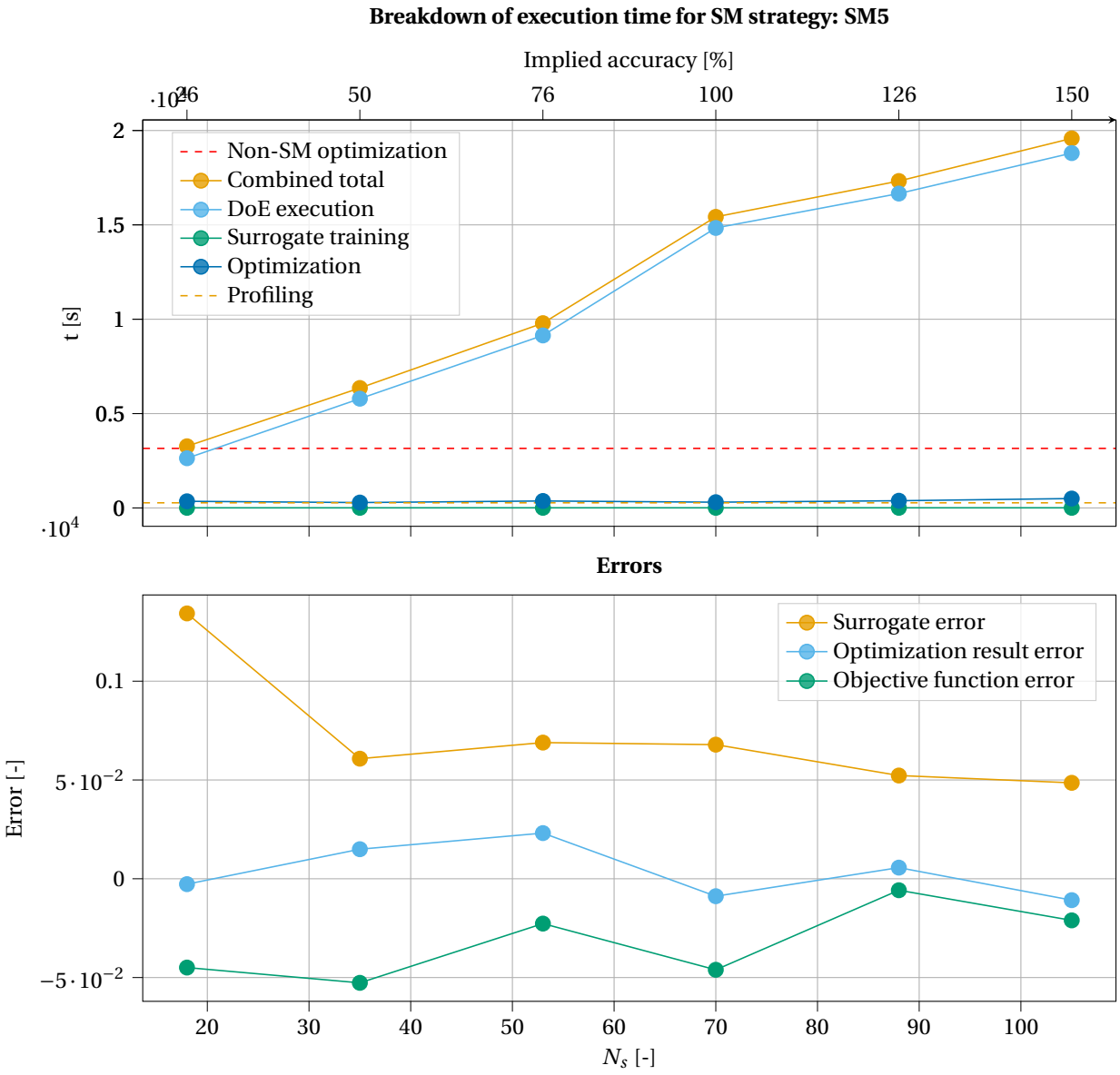
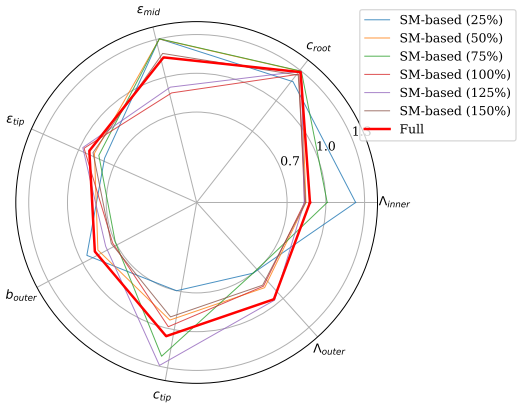
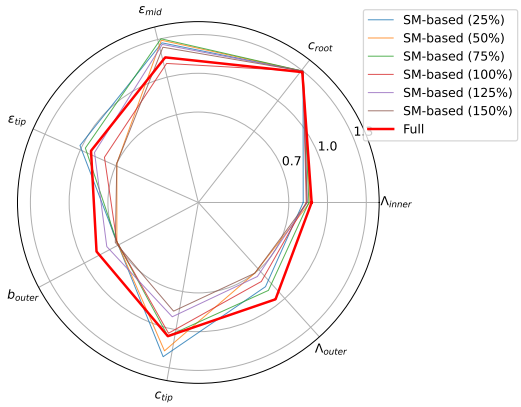


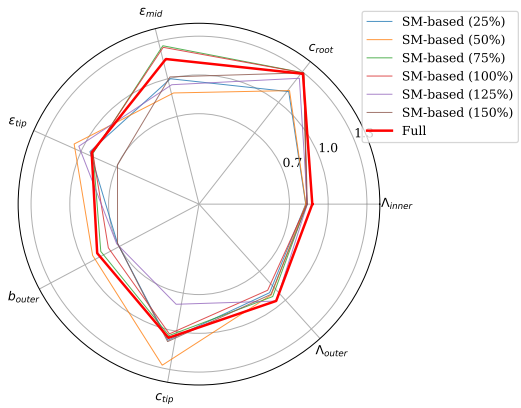
Figure 10.6: Results of implementation of SM5.



(a) SM1



(b) SM4



(c) SM5

Figure 10.7: Resulting design vectors for the different SM strategies and *IAs*.

10.1.3. PERFORMANCE OF ESTIMATIONS

Compared to the assessment of the provided advice in the Sellar test case, a different image is sketched in the MTOM problem. A comparison between the estimated combined runtimes, as well as the estimated runtimes for the executions of the DoEs, is found in Figure 10.9. The estimated required time for the non-SM-based optimization was $t_{opt,est} = 3420s$, and the realized optimization time was $t_{opt} = 3157s$. Compared to the Sellar problem, this difference is much smaller. However, in the Sellar problem, the estimations of the required runtimes for DoE execution were much more accurate, especially for the converged SM strategy SM5.

The more accurate optimization time estimation and the less accurate DoE runtime estimation, are directly correlated. In the MTOM problem, the profiling analysis required an average number of 2 iterations to converge to a valid result. During the execution of the DoE, this number has risen to 3.9. This is a drastic increase, and especially for such a convergence loop containing an expensive design competence such as the AeroAnalysis discipline, large errors occur in the estimations. However, as can be seen in the figure, it is not the only cause for the mismatch in the data, as also the non-converged SM strategies suffer from inaccuracies in the estimation.

These latter inaccuracies are introduced by semi-random, long evaluations of the AeroAnalysis design competence. For a large set of samples, the runtimes of the AeroAnalysis discipline are shown in **??**. These variations do not appear too often and do not appear in the profiling run. However, during the executions of the DoEs for the SM strategies, these excessive runtimes do occur and therefore introduce an error in the runtime estimations.

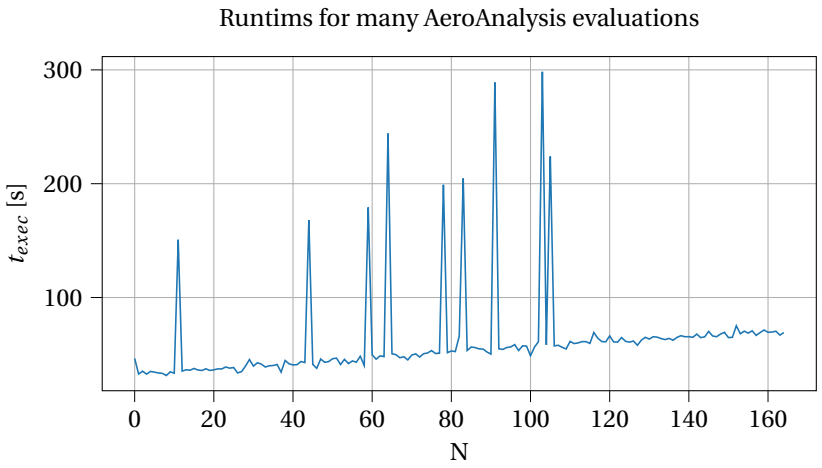
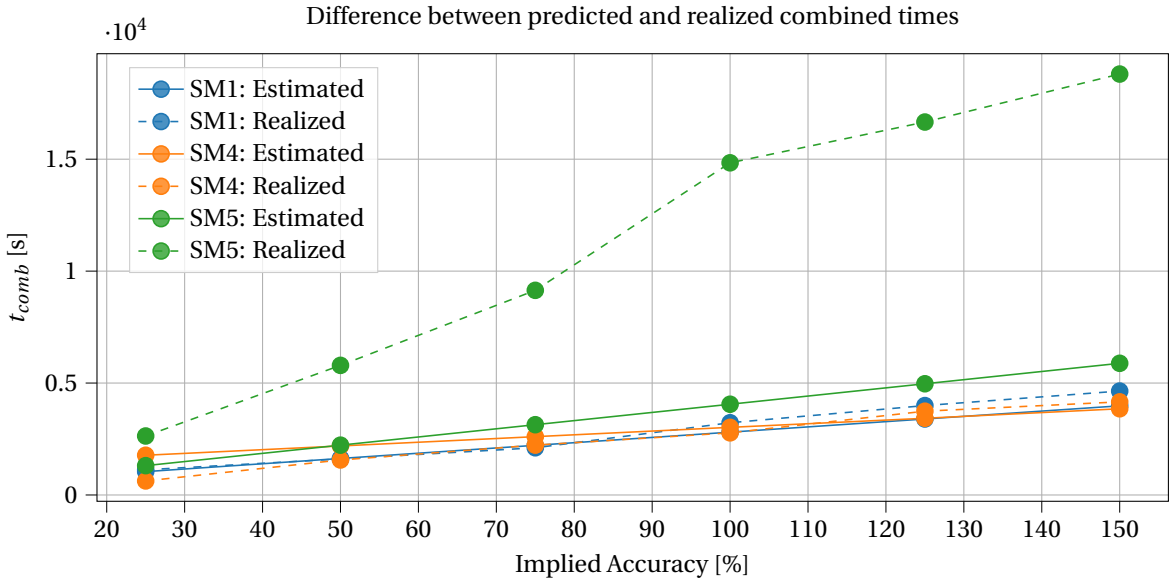
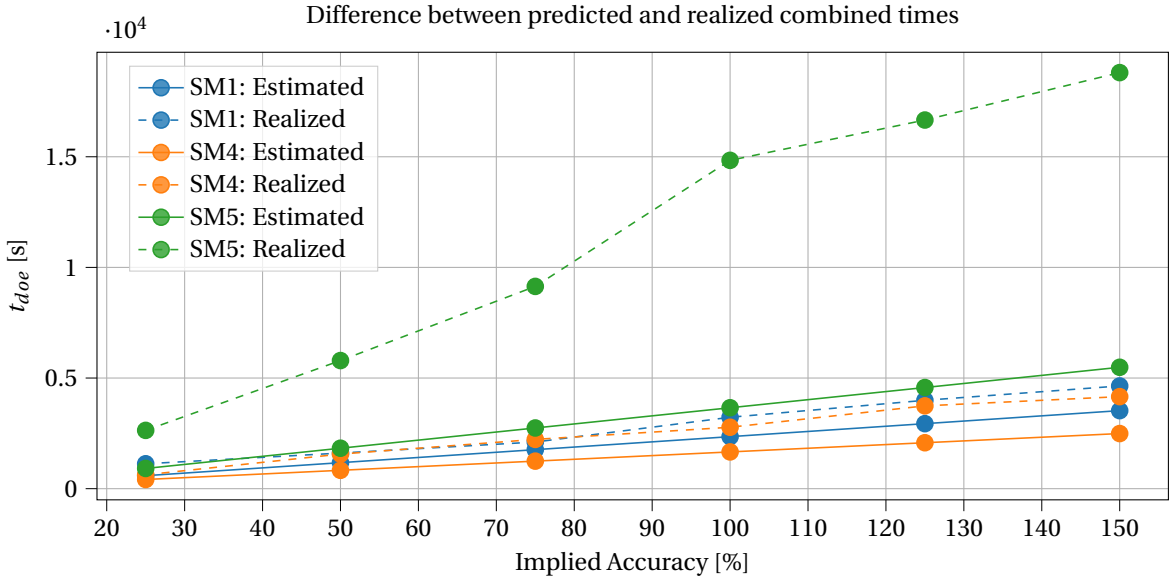


Figure 10.8: Runtimes for the AeroAnalysis design competences.



(a) Combined runtimes



(b) DoE runtimes

Figure 10.9: Difference between estimated and realized runtimes for the MTOM SM strategies.

10.2. SCENARIO II: GENERATING AN ACCURATE SM FOR THE AEROANALYSIS DISCIPLINE

The second scenario aims to generate an accurate and re-usable SM for the computationally expensive AeroAnalysis discipline. Where in the previous scenario care was taken to reduce the computational costs of the complete process as much as possible, the focus is now to create a highly accurate SM. Due to its re-usability, it can reduce the computational costs of many following MDAO workflow executions, and increases the *front loading* of a design process.

To increase the re-usability for the to-be-generated SM, the design variables for the AeroAnalysis are extended. Where for the full optimization a fixed airfoil geometry was used, the SM will be trained for a varying airfoil geometry at the wing root and tip. The airfoil geometry is parameterized using Bernstein coefficients, using a total of twelve coefficients for the definition of an airfoil. As two airfoils are modeled, 24 additional variables are added to the design space of the SM. Furthermore, as the AeroAnalysis design competence is dependent on the aircraft weight, the SM also needs to be trained for a varying fuel weight W_f and structural weight W_{str}.

10.2.1. SAMPLING STRATEGY

For this scenario, the SM will be generated using samples as proposed by the implemented adaptive sampling strategies. As a starting point, the *Jones* sample advisor is used, which is defined by:

$$N_s = 10N_v, \tag{10.7}$$

for an implied accuracy of 75%. As

$$IA = \frac{N_{s,realized}}{N_{s,advised}} \cdot 100 \tag{10.8}$$

and $N_v = 33$ for the AeroAnalysis discipline, $IA = 75\%$ corresponds to

$$75 = \frac{N_s}{10 \cdot 33}$$
$$N_s = 247.5 = 248,$$

samples. Using the implemented adaptive sampling algorithms, N_s will be increased to $IA = 125\%$, which corresponds to $N_s = 413$. The adaptive sampling is executed for both implemented adaptive sampling algorithms, as described in section 7.4.

10.2.2. RESULTS

By executing the above implementation, the results as seen in Figure 10.10 are generated. The initial error is 4.6%, and both strategies manage to reduce the error when the samples are increased. However, there is still a lot of noise to be seen in the generated error. This can be contributed to the fact that a k-fold validation scheme is used, which is known to have a relatively high variance, as it depends on the selected folds. Furthermore, it can also mean that the model encounters new, difficult-to-model points, for its new function evaluation. The Expected Improvement for Global Fit (EIGF) algorithm appears to be superior in its sample finding compared to the simple maximization of Kriging's variance predictor.

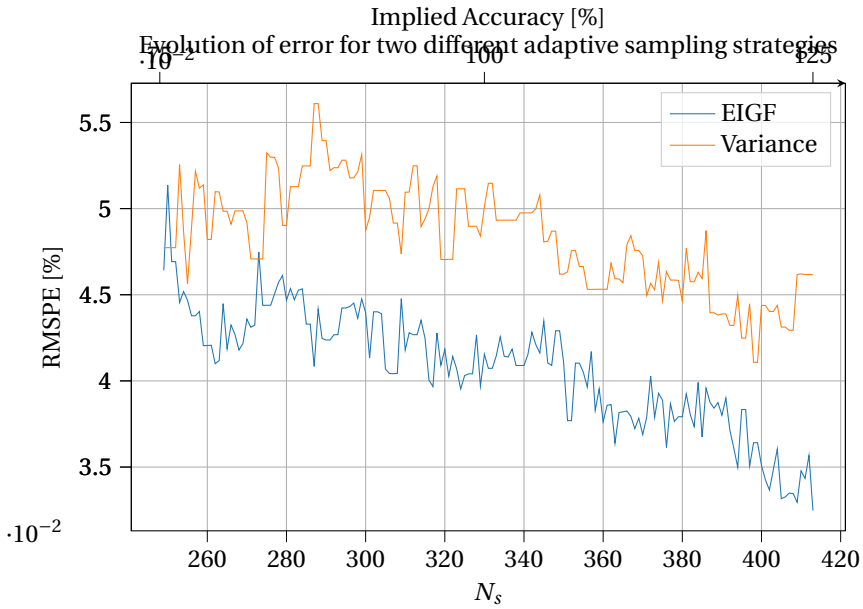


Figure 10.10: Behaviour of the error for two adaptive sampling strategies for the generation of an SM for the AeroAnalysis design competence.

11

CONCLUSIONS & RECOMMENDATIONS

In this work, the Surrogate Advisory System (SAS) is presented. It is shown that where the computational costs of MDAO workflows are often a limiting factor, surrogate models (SMs) can be a viable approach to reduce this existing hurdle for widespread MDAO adoption. The developed software drastically reduces the complexity of working with SMs in MDAO systems, as a comprehensive toolbox is available to quickly include SMs into MDAO workflows, to create SM-based MDAO systems.

Concerning the research subquestions, the following conclusions can be drawn:

- *How can the lifecycle of an SM-based MDAO system, among which falls the acquisition of samples, training and validation of the model, and deployment of the SM in an MDAO workflow, be automated?*
  This thesis shows that the surrogate model lifecycle can be automated by creating a system that comprises both the formulation and execution phases of the original MDAO lifecycle. By utilizing automated workflow formulation systems for the automated generation of required DoE formulations, as well as for the implementation of generated surrogate models into workflows, the formulation step of surrogate models is automated. By implementing an interface between a PIDO tool and the developed software, also the execution of these DoE's has become highly automated, solely hindered by a current lack of full support of the CMDOWS standard in RCE. The different phases in the surrogate model lifecycle are coordinated in the developed software, which creates a powerful toolbox for MDAO architects.

- *How can a strategy be formulated that identifies bottlenecks in an MDAO workflow?*
  SAS implements a workflow profiling algorithm that identifies bottlenecks in workflows by executing a small DoE and analyzing the execution timeline. From this workflow execution, an analysis is made which calculates, among other metrics, normalized runtimes for design competences. It is shown that these normalized runtimes indicate bottlenecks in provided workflows.

- *What metrics are suitable to quantify the trade-off between computational efficiency and objective function accuracy when a certain discipline, or group of disciplines, would be replaced by a surrogate model?*
  It is found that an initial estimation of the optimization time for the full, non-SM-based workflow is required to identify a potential performance gain when an SM is implemented. If this estimation is available, a calculation can be made that estimates this performance gain by comparing the required training time for an SM, and the estimated number of samples needed to train a sufficiently accurate SM. The estimation for the number of samples is made on existing rules of thumb in literature, but the Sellar test case shows that the actual performance of an SM can vary drastically. The implemented relationships from literature do not account for the size of the design space, meaning the size between the bounds of design variables, which can lead to inaccuracies in the data.

- *How can the set of metrics and the identification of bottlenecks be summarized into an advice to the end-user on which disciplines are suited to be replaced by a surrogate model?* SAS implements a methodology

where the implications of the inclusion of an SM in a workflow are calculated for all possible SM strategies. A comparison between different strategies can be made by the introduction of the concept of *implied accuracy* of a to-be-generated SM. This implied accuracy is the ratio between the number of samples in a DoE and the advised number of samples by the rules of thumb. For each of the implied accuracy, the combined impact on the total optimization time is displayed.

The conclusions of the subquestions lead to an answer to the main research question, which was formulated as:

> *How can the current state-of-the-art in MDAO be extended with a framework that automates the surrogate model integration process and that provides insights into the trade-off between accuracy and computational costs for a given surrogate-based MDAO strategy?*

This work shows that it is possible to both automate the surrogate model building process, as well as to provide insights into the trade-off between accuracy and computational costs for a given surrogate-based MDAO strategy. Where the necessary investments for the generation of an SM can be calculated quite accurately, the required number of samples to reach a certain accuracy is deemed to be difficult. Furthermore, due to the highly unpredictable behavior of workflows during optimization, it is also shown to be difficult to provide an accurate indication of when an SM is feasible to replace a (group of) discipline(s) for a single optimization run.

Therefore, the following major conclusion can be drawn. SMs have continuously proven to be very usable in MDAO applications, and the presented work provides a very powerful new tool in the hands of MDAO architects. Especially the automated capabilities lead to a significant reduction in the required time to set up and modify SM strategies. However, care needs to be taken when utilizing the advise module. It is very capable of predicting bottlenecks and the training costs of an SM but lacks accuracy in the a-priori prediction of SM accuracy for a given number of samples. That said, it is recommended to use the tools mainly to increase the front loading in the design process, by creating SM of design competences early on in the process. Having an SM available, leads to a drastic reduction of all following workflow executions, while still maintaining reasonable accuracy.

Based on the work, the following recommendations for future research can be made:

- *Improvement of existing relationships between the dimension of an input space, number of samples, and estimated accuracy.*
  In this work, it is shown that the current rules of thumb, where the required number of samples for an expected accurate SM is related to the number of variables of its input space, holds reasonably well, but edge cases are present. A lot of potentially useful information, such as the non-linearity of the sample set and the size of the input space, is left unused. It can be expected that the formulation can be made significantly more accurate, and an investigation is needed on how available information can be used for this purpose.

- *Improvement of SM strategy advise using more data from a workflow analysis run.*
  The implemented strategy for the determination of time bottlenecks in a workflow, works well. It is able to clearly isolate the most expensive design competences, which leads to a sound basis for advising on SM strategies. However, currently, it only uses the average values of runtimes in the advise calculations, where data such as the standard deviation of runtimes, is available. It is shown in the work that the behavior of workflows can be difficult to predict due to variations in runtimes of individual design competences, as well as variations in the convergence behavior of workflows. Further research would be recommended to increase the estimations on workflow behavior.

- *Implementation of global- and adaptive SM-based MDAO formulations*
  As discussed in the literature review, three main categories for SM-based MDAO exist. This work currently falls under the umbrella of Non- or semi-adaptive surrogate-based MDAO, but the other categories, global- and adaptive surrogate-based MDAO, have also proven to be powerful. Due to the highly automated nature of SAS, an extension of the application to these categories should not be difficult and is likely to lead to more efficient MDAO workflow formulations.

A

APPENDIX

A.1. FULL XDSM FOR THE MTOM PROBLEM

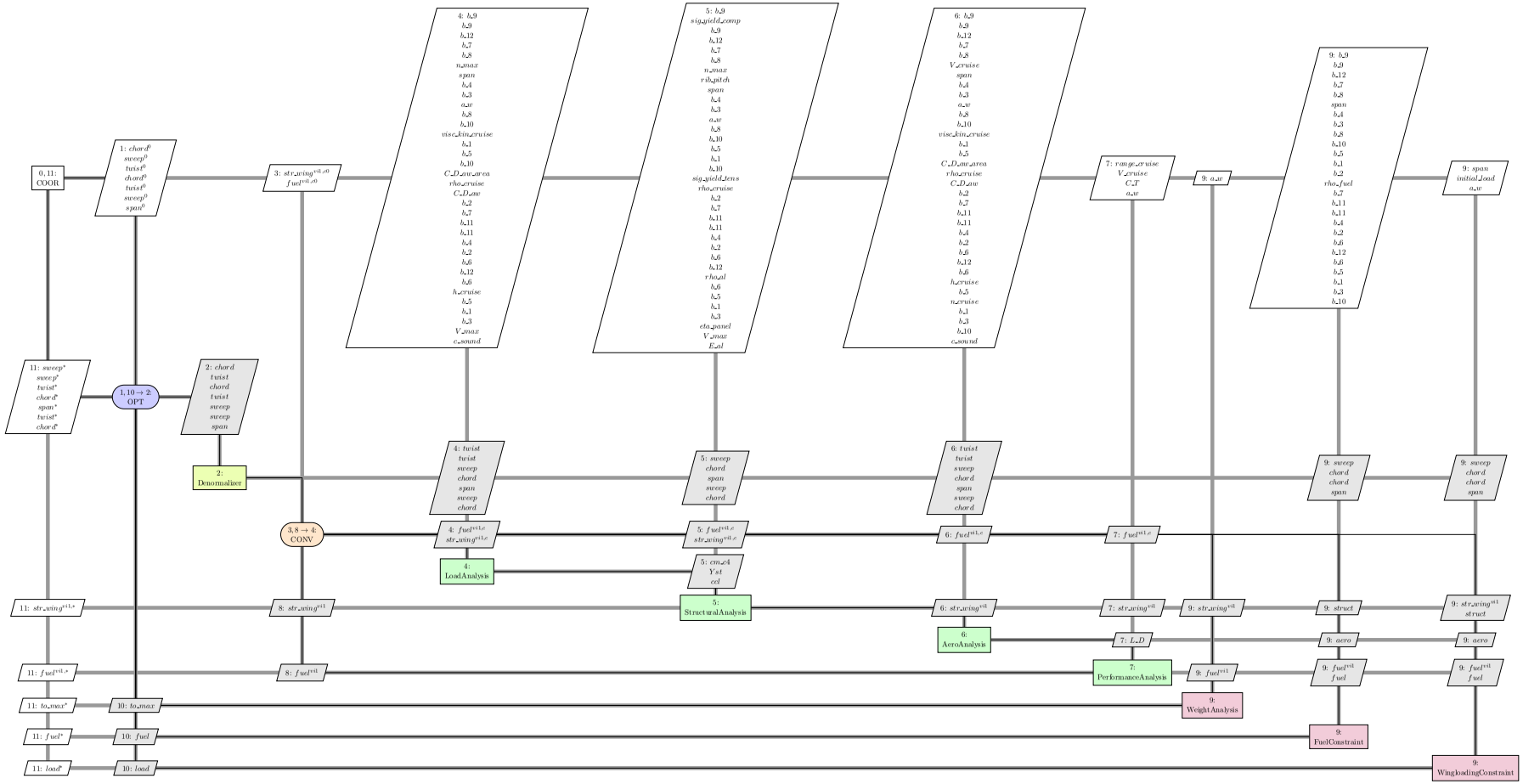


Figure A.1: Full XDSM of the MTOM problem. Some variable names are doubled, due to the usage in a CDS scheme where there identification (e.g. inner_wing, outer_wing) comes earlier in the xml path.

A.2. EXPERIMENTAL DESIGN FOR ADVISE VERIFICATION

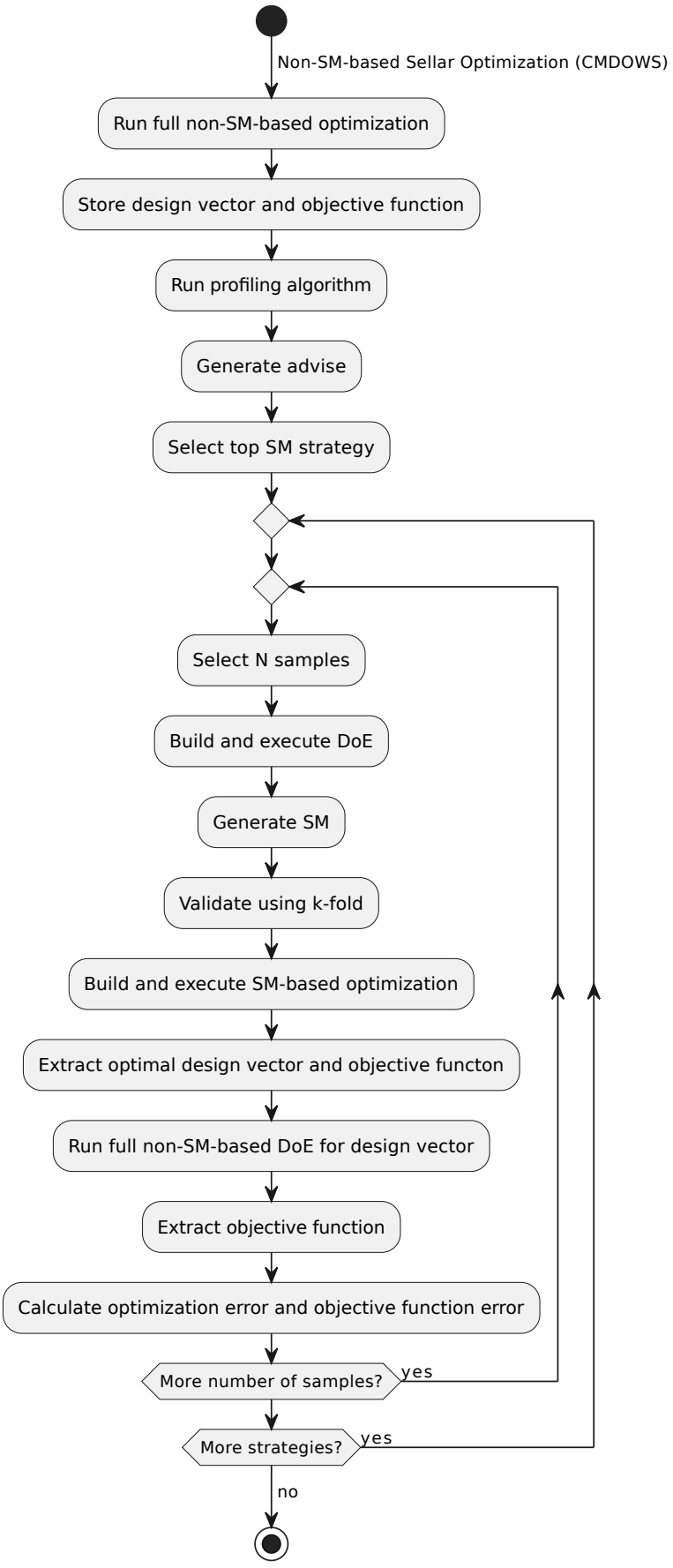


Figure A.2: Experimental design used for both test cases. It aims to compare different advised strategies, as well as the accuracy for a given number of samples. All steps are timed, as such a detailed profile of a full SM optimization lifecycle can be extracted.

A.3. RAW DATA FOR SELLAR SM STRATEGY ADVISE

	candidate	25	50	75	100	125	150
0	D1	-931.252	-848.848	-766.444	-684.04	-601.636	-519.232
0	ConvergerD1D2	-1621.6935	-1260.021	-898.3485	-536.676	-175.0035	186.669
0	D1D2	-1740.697	-1596.028	-1451.359	-1306.69	-1162.021	-1017.352
0	D2	-685.01125	-638.3125	-591.61375	-544.915	-498.21625	-451.5175
0	F1	32.848	35.708	38.568	41.428	44.288	47.148
0	F1G1	24.912	30.252	35.592	40.932	46.272	51.612
0	F1G1G2	15.045333	23.468667	31.892	40.315333	48.738667	57.162
0	G1	32.204	32.824	33.444	34.064	34.684	35.304
0	G1G2	21.415667	24.197333	26.979	29.760667	32.542333	35.324
0	G2	29.820833	30.591667	31.3625	32.133333	32.904167	33.675

Table A.1: Raw data for the determination of the optimal MTOM SM strategies. Numbers indicate the calculated difference in time for full optimization, including data generation. Generated using Jones ($N_s = 10 \cdot N_v$).

A.4. RAW DATA FOR MTOM SM STRATEGY ADVISE

	candidate	25	50	75	100	125	150
0	Denormalizer	27.68825	40.892	54.09575	67.2995	80.50325	93.707
0	LoadAnalysis	-185.176688	-113.747625	-42.318563	29.1105	100.539563	171.968625
0	ConvergerLoadAnalysisStructuralAnalysisAeroAna...	-2109.292	-1194.952	-280.612	633.728	1548.068	2462.408
0	LoadAnalysisStructuralAnalysis	-528.702125	-376.33775	-223.973375	-71.609	80.755375	233.11975
0	LoadAnalysisStructuralAnalysisAeroAnalysis	-2291.500813	-1723.817375	-1156.133937	-588.4505	-20.767063	546.916375
0	LoadAnalysisStructuralAnalysisAeroAnalysisPerf...	-2376.842	-1789.052	-1201.262	-613.472	-25.682	562.108
0	StructuralAnalysis	-216.532625	-126.6045	-36.676375	53.25175	143.179875	233.108
0	StructuralAnalysisAeroAnalysis	-1822.906	-1161.2335	-499.561	162.1115	823.784	1485.4565
0	StructuralAnalysisAeroAnalysisPerformanceAnalysis	-1901.545	-1213.06375	-524.5825	163.89875	852.38	1540.86125
0	AeroAnalysis	-1644.798688	-1229.479625	-814.160563	-398.8415	16.477563	431.796625
0	AeroAnalysisPerformanceAnalysis	-1730.139875	-1294.71425	-859.288625	-423.863	11.562625	446.98825
0	PerformanceAnalysis	19.254438	25.956625	32.658813	39.361	46.063188	52.765375
0	WeightAnalysis	-31.3825	-23.01375	-14.645	-6.27625	2.0925	10.46125
0	WeightAnalysisWingloadingConstraint	-51.658875	16.5555	84.769875	152.98425	221.198625	289.413
0	FuelConstraint	-22.666	19.214	61.094	102.974	144.854	186.734
0	FuelConstraintWeightAnalysis	-78.522875	6.2515	91.025875	175.80025	260.574625	345.349
0	FuelConstraintWeightAnalysisWingloadingConstraint	-128.089875	-12.7605	102.568875	217.89825	333.227625	448.557
0	WingloadingConstraint	9.433	39.988	70.543	101.098	131.653	162.208

Table A.2: Raw data for the determination of the optimal MTOM SM strategies. Numbers indicate the calculated difference in time for full optimization, including data generation. Generated using Jones ($N_s = 10 \cdot N_v$).

BIBLIOGRAPHY

[1] J. R. R. A. Martins and S. A. Ning, *Engineering design optimization* (Cambridge University Press, Cambridge ; New York, NY, 2021).

[2] I. van Gent and G. La Rocca, *Formulation and integration of MDAO systems for collaborative design: A graph-based methodological approach,* Aerospace Science and Technology **90**, 410 (2019), publisher: Elsevier Masson SAS.

[3] I. van Gent, G. La Rocca, and M. F. Hoogreef, *CMDOWS: a proposed new standard to store and exchange MDO systems,* CEAS Aeronautical Journal **9**, 607 (2018), publisher: Springer Vienna.

[4] I. van Gent, G. La Rocca, and L. L. Veldhuis, *Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems,* in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (American Institute of Aeronautics and Astronautics, Denver, Colorado, 2017).

[5] N. Bartoli, T. Lefebvre, S. Dubreuil, R. Olivanti, N. Bons, J. R. R. A. Martins, M.-A. Bouhlel, and J. Morlier, *An adaptive optimization strategy based on mixture of experts for wing aerodynamic design optimization,* in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (American Institute of Aeronautics and Astronautics, Denver, Colorado, 2017).

[6] K. Bowcutt, *A Perspective on the Future of Aerospace Vehicle Design,* in *12th AIAA International Space Planes and Hypersonic Systems and Technologies* (American Institute of Aeronautics and Astronautics, Norfolk, Virginia, 2003).

[7] J. Vandenbrande, T. Grandine, and T. Hogan, *The search for the perfect body: Shape Control for multidisciplinary design optimization,* in *44th AIAA Aerospace Sciences Meeting and Exhibit* (American Institute of Aeronautics and Astronautics, Reno, Nevada, 2006).

[8] G. Belie, *Non-Technical Barriers to Multidisciplinary Optimization in the Aerospace Industry,* in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* (American Institute of Aeronautics and Astronautics, Atlanta, Georgia, 2002).

[9] S. Shahpar, *Challenges to overcome for routine usage of automatic optimisation in the propulsion industry,* The Aeronautical Journal **115**, 615 (2011).

[10] A. B. Lambe and J. R. Martins, *Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes,* Structural and Multidisciplinary Optimization **46**, 273 (2012).

[11] S. S. Chauhan, J. T. Hwang, and J. R. R. A. Martins, *An automated selection algorithm for nonlinear solvers in MDO,* Structural and Multidisciplinary Optimization **58**, 349 (2018).

[12] N. M. Alexandrov and R. M. Lewis, *Reconfigurability in MDO problem synthesis, Part 1,* Collection of Technical Papers - 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference **1**, 73 (2004).

[13] S. Tosserams, A. T. Hofkamp, L. F. Etman, and J. E. Rooda, *A specification language for problem partitioning in decomposition-based design optimization,* Structural and Multidisciplinary Optimization **42**, 707 (2010).

[14] M. Hoogreef, *Advise, Formalize and Integrate MDO Architectures: A Methodology and Implementation,* Ph.D. thesis, Delft University of Technology (2017).

[15] D. V. Steward, *Design Structure System: a Method for Managing the Design of Complex Systems.* IEEE Transactions on Engineering Management **EM-28**, 71 (1981).

[16] R. Lano, *The N2 chart,* TRW Software Series (1977).

[17] I. van Gent, *Agile MDAO Systems: A Graph-based Methodology to Enhance Collaborative Multidisciplinary Design*, Ph.D. thesis, Delft University of Technology (2019).

[18] A. Rizzi, M. Zhang, B. Nagel, D. Boehnke, and P. Saquet, *Towards a unified framework using CPACS for geometry management in aircraft design,* 50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition , 1 (2012).

[19] B. Aigner, I. van Gent, G. La Rocca, E. Stumpf, and L. L. M. Veldhuis, *Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems,* CEAS Aeronautical Journal **9**, 695 (2018).

[20] A. Page Risueño, J. Bussemaker, P. D. Ciampa, and B. Nagel, *MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems,* in *AIAA AVIATION 2020 FORUM* (American Institute of Aeronautics and Astronautics, VIRTUAL EVENT, 2020).

[21] A.-L. Bruggeman, *Automated Execution Process Formulation using Sequencing and Decomposition Algorithms for Collaborative MDAO,* Master's thesis, Delft University of Technology, Delft (2019).

[22] R. Yondo, E. Andrés, and E. Valero, *A review on design of experiments and surrogate models in aircraft real-time and many-query aerodynamic analyses,* Progress in Aerospace Sciences **96**, 23 (2018).

[23] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins, *A Python surrogate modeling framework with derivatives,* Advances in Engineering Software , 102662 (2019).

[24] C.-Z. Xu, Z.-H. Han, K.-S. Zhang, and W. Song, *Surrogate-based optimization method applied to multidisciplinary design optimization architectures,* in *31st congress of the International Council Of The Aeronautical Sciences (ICAS 2018)* (2018).

[25] S. Dubreuil, N. Bartoli, C. Gogu, and T. Lefebvre, *Towards an efficient global multidisciplinary design optimization algorithm,* Structural and Multidisciplinary Optimization **62**, 1739 (2020).

[26] Z. Hu and S. Mahadevan, *Adaptive Surrogate Modeling for Time-Dependent Multidisciplinary Reliability Analysis,* Journal of Mechanical Design **140**, 021401 (2018).

[27] T. Lefebvre, N. Bartoli, S. Dubreuil, M. Panzeri, R. Lombardi, W. Lammen, M. Zhang, I. van Gent, and P. D. Ciampa, *A clustered and surrogate-based MDA use case for MDO scenarios in AGILE project,* 2018 Multidisciplinary Analysis and Optimization Conference , 1 (2018).

[28] G. G. Wang and S. Shan, *Review of Metamodeling Techniques in Support of Engineering Design Optimization,* Journal of Mechanical Design **129**, 370 (2007).

[29] R. M. Paiva, A. R. D. Carvalho, C. Crawford, and A. Suleman, *Comparison of Surrogate Models in a Multidisciplinary Optimization Framework for Wing Design,* AIAA Journal **48**, 995 (2010).

[30] J. Sobieszczanski-Sobieski and R. T. Haftka, *Multidisciplinary aerospace design optimization: Survey of recent developments,* Structural Optimization **14**, 1 (1997).

[31] J. Sobieszczanski-Sobieski, *Multidisciplinary design optimization: An emerging new engineering discipline,* (Rio De Janeiro, 1993).

[32] J. R. Martins and A. B. Lambe, *Multidisciplinary design optimization: A survey of architectures,* AIAA Journal **51**, 2049 (2013).

[33] E. J. Cramer, J. E. Dennis, Jr., P. D. Frank, R. M. Lewis, and G. R. Shubin, *Problem Formulation for Multidisciplinary Optimization,* SIAM Journal on Optimization **4**, 754 (1994).

[34] J. F. Epperson, *An introduction to numerical methods and analysis,* second edition ed. (John Wiley & Sons, Inc, Hoboken, New Jersey, 2013).

[35] B. Boden, J. Flink, N. Först, R. Mischke, K. Schaffert, A. Weinert, A. Wohlan, and A. Schreiber, *RCE: An Integration Environment for Engineering and Science,* SoftwareX **15**, 100759 (2021).

[36] J. S. Gray, J. T. Hwang, J. R. Martins, K. T. Moore, and B. A. Naylor, *OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization,* Structural and Multidisciplinary Optimization **59**, 1075 (2019), publisher: Structural and Multidisciplinary Optimization.

[37] D. de Vries, I. van Gent, G. La Rocca, and S. Binder, *OpenLEGO demonstration: A link between AGILE and OpenMDAO,* in *First European OpenMDAO workshop* (2017).

[38] F. Gallard, C. Vanaret, D. Guenot, V. Gachelin, R. Lafage, B. Pauwels, P.-J. Barjhoux, and A. Gazaix, *GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation,* in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (American Institute of Aeronautics and Astronautics, Kissimmee, Florida, 2018).

[39] S. Koziel, D. E. Ciaurri, and L. Leifsson, *Comput. Optimization, Methods and Algorithms,* Springer-Verlag Berlin Heidelberg 2011 **356**, 33 (2011).

[40] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, *Surrogate-based analysis and optimization,* Progress in Aerospace Sciences **41**, 1 (2005).

[41] L. Jia, R. Alizadeh, J. Hao, G. Wang, J. K. Allen, and F. Mistree, *A rule-based method for automated surrogate model selection,* Advanced Engineering Informatics **45**, 101123 (2020), publisher: Elsevier.

[42] R. Alizadeh, J. K. Allen, and F. Mistree, *Managing computational complexity using surrogate models: a critical review,* Research in Engineering Design **31**, 275 (2020), publisher: Springer London.

[43] T. Long, L. Liu, S. Zhou, J. Wang, and L. Meng, *Multi-Objective Multidisciplinary Optimization of Long-Endurance UAV Wing Using Surrogate Model in ModelCenter,* in *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (American Institute of Aeronautics and Astronautics, Victoria, British Columbia, Canada, 2008).

[44] T. Lefebvre, N. Bartoli, S. Dubreuil, M. Panzeri, R. Lombardi, P. Della Vecchia, L. Stingo, F. Nicolosi, A. De Marco, P. Ciampa, K. Anisimov, A. Savelyev, A. Mirzoyan, and A. Isyanov, *Enhancing optimization capabilities using the AGILE collaborative MDO framework with application to wing and nacelle design,* Progress in Aerospace Sciences **119**, 100649 (2020).

[45] R. Shi, L. Liu, T. Long, Y. Wu, and G. G. Wang, *Multidisciplinary modeling and surrogate assisted optimization for satellite constellation systems,* Structural and Multidisciplinary Optimization **58**, 2173 (2018).

[46] Y. Feng, Z. Chen, Y. Dai, F. Wang, J. Cai, and Z. Shen, *Multidisciplinary optimization of an offshore aquaculture vessel hull form based on the support vector regression surrogate model,* Ocean Engineering **166**, 145 (2018).

[47] D. C. Montgomery, *Design and analysis of experiments,* ninth edition ed. (John Wiley & Sons, Inc, Hoboken, NJ, 2017).

[48] D. J. Finney, *The Fractional Replication of Factorial Arrangements,* Annals of Eugenics **12**, 291 (1943), _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1943.tb02333.x.

[49] A. A. Giunta, S. F. Wojtkiewicz, and M. S. Eldred, *Overview of modern design of experiments methods for computational simulations,* 41st Aerospace Sciences Meeting and Exhibit , 1 (2003).

[50] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,* Technometrics **42**, 55 (2000).

[51] S. Leary, A. Bhaskar, and A. Keane, *Optimal orthogonal-array-based latin hypercubes,* Journal of Applied Statistics **30**, 585 (2003).

[52] A. I. Forrester and A. J. Keane, *Recent advances in surrogate-based optimization,* Progress in Aerospace Sciences **45**, 50 (2009).

[53] E. B. Saff and A. D. Snider, *Fundamentals of matrix analysis with applications* (John Wiley & Sons, Inc, Hoboken, New Jersey, 2015).

[54] D. G. Krige, *A statistical approach to some basic mine valuation problems on the Witwatersrand,* Journal of the Southern African Institute of Mining and Metallurgy **52**, 119 (1951), _eprint: https://journals.co.za/doi/pdf/10.10520/AJA0038223X_4792.

[55] G. Matheron, *Principles of geostatistics,* Economic Geology **58**, 1246 (1963).

[56] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and Analysis of Computer Experiments,* Statistical Science **4** (1989), 10.1214/ss/1177012413.

[57] J. P. Kleijnen, *Kriging metamodeling in simulation: A review,* European Journal of Operational Research **192**, 707 (2009).

[58] J. P. Kleijnen, *Design and Analysis of Simulation Experiments*, International Series in Operations Research & Management Science, Vol. 230 (Springer International Publishing, Cham, 2015).

[59] J. P. Kleijnen, *Regression and Kriging metamodels with their experimental designs in simulation: A review,* European Journal of Operational Research **256**, 1 (2017).

[60] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, *Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction,* Structural and Multidisciplinary Optimization **53**, 935 (2016).

[61] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier, *An Improved Approach for Estimating the Hyperparameters of the Kriging Model for High-Dimensional Problems through the Partial Least Squares Method,* Mathematical Problems in Engineering **2016**, 1 (2016).

[62] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python,* Journal of Machine Learning Research **12**, 2825 (2011).

[63] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: A System for Large-Scale Machine Learning,* in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, Savannah, GA, 2016) pp. 265–283.

[64] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene, and K. Crombecq, *A surrogate modeling and adaptive sampling toolbox for computer based design,* The Journal of Machine Learning Research **11**, 2051 (2010), publisher: JMLR. org.

[65] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient Global Optimization of Expensive Black-Box Functions,* Journal of Global Optimization **13**, 455 (1998).

[66] N. Bartoli, I. Kurek, R. Lafage, T. Lefebvre, R. Priem, M. Bouhlel, J. Morlier, V. Stilz, and R. Regis, *Improvement of efficient global optimization with mixture of experts: methodology developments and preliminary results in aircraft wing design,* in *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (2016).

[67] M. J. Sasena, *Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations,* Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations (2002).

[68] A. G. Watson and R. J. Barnes, *Infill sampling criteria to locate extremes,* Mathematical Geology **27**, 589 (1995).

[69] S. Dubreuil, N. Bartoli, T. Lefebvre, and C. Gogu, *Efficient global multidisciplinary optimization based on surrogate models,* in *2018 Multidisciplinary Analysis and Optimization Conference* (American Institute of Aeronautics and Astronautics, Atlanta, Georgia, 2018).

[70] X. Wang, M. Li, Y. Liu, W. Sun, X. Song, and J. Zhang, *Surrogate based multidisciplinary design optimization of lithium-ion battery thermal management system in electric vehicles,* Structural and Multidisciplinary Optimization **56**, 1555 (2017).

[71] A. Mehmani, S. Chowdhury, C. Meinrenken, and A. Messac, *Concurrent surrogate model selection (COSMOS): optimizing model type, kernel function, and hyper-parameters,* Structural and Multidisciplinary Optimization **57**, 1093 (2018).

[72] X. Chen, P. Wang, and D. Zhang, *Surrogate-Based Multidisciplinary Design Optimization of an Autonomous Underwater Vehicle Hull,* in *2017 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)* (IEEE, Anyang, 2017) pp. 191–194.

[73] A. Da Ronch, M. Panzeri, M. A. Abd Bari, R. d'Ippolito, and M. Franciolini, *Adaptive design of experiments for efficient and accurate estimation of aerodynamic loads,* Aircraft Engineering and Aerospace Technology **89**, 558 (2017).

[74] J. N. Fuhg, A. Fau, and U. Nackenhorst, *State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging,* Archives of Computational Methods in Engineering **28**, 2689 (2021).

[75] M. Kaufman, V. Balabanov, A. A. Giunta, B. Grossman, W. H. Mason, S. L. Burgee, R. T. Haftka, and L. T. Watson, *Variable-complexity response surface approximations for wing structural weight in HSCT design,* Computational Mechanics **18**, 112 (1996).

[76] J. L. Loeppky, J. Sacks, and W. J. Welch, *Choosing the Sample Size of a Computer Experiment: A Practical Guide,* Technometrics **51**, 366 (2009).

[77] S. Shan and G. G. Wang, *Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions,* Structural and Multidisciplinary Optimization **41**, 219 (2010).

[78] M. Kubat, *An Introduction to Machine Learning* (Springer International Publishing, Cham, 2017).

[79] B. Clarke, E. Fokoue, and H. H. Zhang, *Principles and Theory for Data Mining and Machine Learning*, Springer Series in Statistics (Springer New York, New York, NY, 2009).

[80] J. Müller and M. Day, *Surrogate Optimization of Computationally Expensive Black-Box Problems with Hidden Constraints,* INFORMS Journal on Computing **31**, 689 (2019).

[81] H. Lee, R. Gramacy, C. Linkletter, and G. Gray, *Optimization subject to hidden constraints via statistical emulation,* Pacific Journal of Optimization **7**, 467 (2011).

[82] L. Breiman, *Random forests,* Machine learning **45**, 5 (2001), publisher: Springer.

[83] C. Q. Lam, *Sequential adaptive designs in computer experiments for response surface model fit*, PhD Thesis, The Ohio State University (2008).

[84] A. Elham, G. La Rocca, and M. van Tooren, *Development and implementation of an advanced, design-sensitive method for wing weight estimation,* Aerospace Science and Technology **29**, 100 (2013).

[85] J. Mariens, A. Elham, and M. J. L. van Tooren, *Quasi-Three-Dimensional Aerodynamic Solver for Multidisciplinary Design Optimization of Lifting Surfaces,* Journal of Aircraft **51**, 547 (2014).

[86] J. Roskam, *Airplane Design: Preliminary sizing of airplanes* (DARcorporation, 1985).