# Towards the Industrialization of MDAO

D. De Vries


Technische Universiteit Delft

TU Delft

# Towards the Industrialization of MDAO

by

D. De Vries

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday December 11, 2017 at 09:30 (CET).


| | | |
|---|---|---|
| Student number: | 4132033 | |
| Thesis registration number: | 174#17#MT#FPP | |
| Project duration: | February 2, 2017 – November 8, 2017 | |
| Thesis committee: | Prof.dr.ir. L.L.M. Veldhuis, | TU Delft, FPP, Chair Flight Performance |
| | Dr.ir. G. La Rocca, | TU Delft, FPP, Supervisor |
| | Dipl.-Ing. S. Binder, | Airbus, Aeromechanic Systems, Supervisor |
| | Dr.ir. M.B. Zaayer, | TU Delft, Wind Energy |




TU Delft

# Preface

Multidisciplinary Design Analysis and Optimization (MDAO) promises major advances in aircraft design. However, judging from todays aircraft, MDAO is clearly not fully embraced by the industry yet. This is due to technical difficulties (coupling incompatible tools) and non-technical barriers (intellectual property).

Recent research, such as the AGILE project, aims at enabling 3rd generation MDO, where collaboration of distributed teams is key. Maturing MDAO is important to eventually make Overall Aircraft Design possible. However, the industry is still in the early days of 1st generation MDO application.

This thesis proposes a new 1st generation MDO 'pipeline' benefiting from the latest tools developed for 3rd generation MDO. This open-source tool-chain makes it easy to connect analyses and include gradient information. It is shown that gradient inclusion yields dramatically reduced computational costs. Rapid (re)configuration and inclusion of gradient information using this pipeline is demonstrated by considering the Sellar problem and a wing optimization.

This report represents the culmination of all the work done over the course of the nine months between February and November 2017. During this time, I was supported by experts, colleagues, friends, family members, and loved ones. I would like to express my thanks to my academic supervisor, Gianfranco La Rocca, supervisor in the industry, Simon Binder, and expert advisor, Imco van Gent. Their advice and knowledge was invaluable to my work. A special thanks goes out to my colleague and friend, Jasper Bussemaker, who supported me not only with many stimulating discussions about the work, but also by getting into trouble with my on many Friday nights out. Next, two people without whom I would never have been able to even consider ever writing a thesis deserve my deepest gratitude: my mom and dad, Maarten and Yvonne de Vries. Finally, I want to thank my fiancée and the love of my life, Sarah Rucker, who is always there for me when I need it the most.

*Daniël de Vries*
*Delft, December 2017*

# Contents

# List of Figures

# List of Tables

# List of Code Fragments

# 1

# Introduction

*How can MDAO be made more applicable for conceptual design in the industry?* This is the question this thesis attempts to answer. This section expands on this question and provides the reader with a non-technical overview of the thesis.

Taking aeronautical engineering as an example, it is obvious from the product that MDAO is not fully embraced yet in the conceptual design phase: the tube-and-wing configuration is still the standard, fig. 1.1. This design makes it relatively easy to decouple the many disciplines that govern the design of


(a) American Airlines fleet [5] (b) Lufthansa fleet [57]

Figure 1.1: Fleet of of the largest North American and European airlines: American Airlines and Lufthansa

an aircraft into smaller chunks which are easier to analyze and design in isolation [34]. This approach still relies heavily on scientific breakthroughs and incremental technological improvements on the level of the individual disciplines [34]. However, as the science and technology of the disciplines mature, the tube-and-wing configuration reaches the summit of its potential. The only way forward from there with true potential improvement of the product and its lifecycle is to focus on interdisciplinary synergies. In the long run, this means abandoning the relative safety of the tube-and-wing configuration in favor of unconventional aircraft configurations. In fact, is expected that unconventional configurations arise

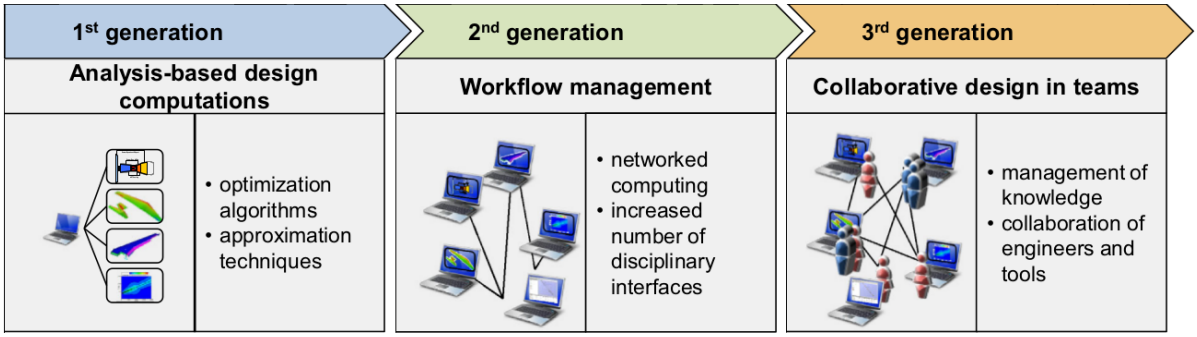



Figure 1.2: First, second and third generation MDAO frameworks. Edited from [62]

naturally when the aircraft is analyzed optimized as a truly multidisciplinary system; that is, when Overall Aircraft Design (OAD) can be achieved using an MDAO system.

Recent work has been focused on developing tools and procedures for third generation MDAO frameworks [62], fig. 1.2. In these 3rd generation frameworks collaboration of scientifically - but also physically - separated teams of experts is key [14]. When a system being designed is considered as a whole and in more detail, the complexity reaches a point where it makes less and less sense to attempt to create a framework with tightly coupled analysis tools. Instead it becomes important that knowledge is managed by those who are expert in their respective field, and for them to disseminate their knowledge to the rest of the team. The analysis tools which are part of a design study stay within the respective domains of each team's expertise. They are connected to one another with a networking strategy which allows for the transfer of data needed as input for, and resulting as output from each team's tools [9, 16]. As such the knowledge of each expert team can be shared with the others to achieve a common goal without sacrificing control over their analysis tools. Hence, this approach helps to protect intellectual property and respects the borders between corporate (network) domains [9, 16]. It is a possible solution to overcome these important non-technical barriers encountered when attempting to apply MDAO in the industry [11].

An example of a project taking this approach is AGILE: Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts [1, 14]. AGILE's main target is the reduction of the time needed to setup and solve an MDO problem by 40% [1]. The envisioned solution lies in the development of a framework and frame of mind to support the move to 3rd generation MDAO [14]. The frame of mind proposed by the project is known as the AGILE Paradigm [16]. Three concepts are formalized by the paradigm: Design Competence (DC), Knowledge Architecture (KA), and Collaborative Architecture (CA). DCs represent analysis tools; KAs use the knowledge encapsulated by DCs to create an architecture of multidisciplinary analyses; and CAs, in turn, make it possible to use knowledge from distributed teams of experts without having to tightly couple their DCs. AGILE tries to enable the move from 2nd to 3rd generation MDAO frameworks. If a system can be developed which allows for this loose coupling whilst still achieving fast iteration turnaround times, MDAO can truly take center stage in the engineering industry.

During conceptual design, it is of particular importance that an MDAO system can be configured and reconfigured rapidly. At this stage of the design process the design freedom is maximal. This means the design space is large, and the analysis tools are of relatively low fidelity. This phase of the design process involves smaller teams of design engineers, which have to identify major options and trade them off to one another. An MDAO system suitable for this phase of the design process looks very differently than one targeting the detail design or even preliminary design phases. The engineers using such a system during this early phase of the design need to have the freedom to extend and modify the framework and architecture on the fly in order to enable them to quickly tradeoff different approaches and find the best strategy. From this it can be deduced that extensibility and ease of use are key requirements of an MDAO system suitable for conceptual design.

At the same time, high performance and fast turnaround times with respect to performing an MDA or MDO run are required, as this will allow engineers to investigate more design options within a shorter amount of time. Thus, greatly increasing the coverage of the full design space during a campaign. However, when a sufficiently large design space needs to be covered by an optimization problem, and

hence the number of design variables increases, computational cost of function evaluations quickly increases as well. In order to optimize a system using a gradient-based algorithm, gradients of inputs to outputs are required. If tools are considered so called 'black boxes' – that is, simple in-/output blocks – the only way to obtain the gradients is through finite differencing. This makes an already expensive system even more expensive to optimize, since the number of function evaluations needed at each design point scales directly with the number of inputs of each block in the system. To avoid the computational cost from blowing up as the number of analysis tools and design variables increases it is essential to utilize analytical gradients whenever they are available.

Hence, three key points can be identified. An MDAO system targeting the conceptual design phase needs to:

1. be easily extensible;
2. allow for rapid (re)configuration of analysis tools and architecture;
3. make use of gradient information whenever it is available.

These three points answer the main question of how MDAO can be made more applicable to the conceptual design phase in an industrial setting. They are in themselves also three sub-questions that need to be answered:

1. How can an MDAO system be developed to be extensible?
2. How can a system allow for rapid (re)configuration of analysis tools and architecture?
3. How can gradient information be incorporated into such a system without sacrificing extensibility and the ability of rapid (re)configuration?

Within AGILE, tools are considered to be black boxes. These black boxes are then coupled to one another within a Process Integration and Design Optimization (PIDO) system. Two such systems are targeted, namely the Remote Component Environment (RCE), which DLR develops, and Optimus, developed by Noesis [96]. These PIDO systems are especially suitable for 3rd generation MDAO frameworks, because they fully support remote analysis tools. Through a special protocol developed within AGILE, known as BRICS, these tools can be used within large, integrated workflows in PIDO environment, whilst remaining under full control of the team of experts that owns them [9]. BRICS is an implementation of the networking strategy that was mentioned before, where data needed as input for or resulting as output from analysis tools can be shared, without having to sacrifice intellectual property and respecting the borders between domains. Furthermore, these systems have the advantage of allowing the user to interact with their models intuitively through a Graphical User Interface (GUI). In it, tools are represented by blocks, which can be dragged-and-dropped easily to rearrange the system. However, the inputs and outputs of these blocks have to be connected to one another manually. This makes reordering them or changing the MDO architecture actually very work intensive.

Recent work within the AGILE project, by Gent et al., has provided a solution to these shortcomings in the form of KADMOS [95, 97]. This software allows the user to formulate and manipulate an MDO problem from a higher, abstract level. In order to analyze how tools can be coupled to one another, and how they can be part of various MDAO strategies, KADMOS uses graph theory [cite]. The software uses a neutral data format as in- and output. This data format takes the form of the XML schema known as CMDOWS [94, 96]. Within a CMDOWS XML file information about a repository of tools (DCs), the Knowledge Base (KB), can be stored. Using KADMOS this information can then be used to construct an overview of the possible connections between the DCs within the KB, known as the Repository Connectivity Graph (RCG). Tools can then be easily removed and rearranged, the user can mark design variables, constraints, and objective, and the user can specify the MDAO strategy using KADMOS' API. The result is an overview of the fundamental problem, known as the Fundamental Problem Graph (FPG). Finally, KADMOS can automatically add optimizer blocks, MDA converger blocks, meta-model blocks, a coordinator block, and all connections between them and the DCs. The final overview of the problem, cast in a specific solution strategy, is represented by a MDAO Problem Graph (MDG), describing the tools and special blocks, their connections, and which variables are design variables, constraints, and objectives, and a MDAO Process Graph (MPG), which describes in which order they are to be executed.

The true power of KADMOS/CMDOWS comes into the light after these graphs have been created and stored in a CMDOWS file. The PIDO systems RCE and Optimus have been extended to allow the user to parse a CMDOWS file, yielding a runnable workflow representative of the problem exactly as it was formulated. In this thesis, this toolchain and workflow, taking a repository of tools through KADMOS to an executable workflow, will be referred to as 'the AGILE pipeline' or simply as 'the pipeline'. Regardless of the choice between RCE and Optimus, the pipeline clearly satisfies the requirement of rapid (re)configurability set before. However, the other two requirements are not (fully) satisfied. This will be described next. Regardless, the notion of the Knowledge Base together with the abilities of KADMOS and its neutral data format, CMDOWS, are an answer to the question of how to rapid (re)configuration of MDAO problems can be achieved.

The pipeline as it exists presently does not, however, adequately answer the question of how to achieve extensibility in such a system on all accounts. This is mainly due to Optimus being a proprietary, closed source package. Because its code is not publically available, it is not possible to add to or change functionality of the core of Optimus. The user is also not free to use parts of Optimus in any derivative work, since the proprietary license agreement will not permit this. In contrast, KADMOS, CMDOWS, and RCE enable extensibility by default, because they are made available as open-source software. Open-source software naturally lends itself to extensibility, because anyone can obtain a copy of the code and make changes/additions where they please. The concept of the Knowledge Base by itself, being a concept, is easily extensible too, because anyone is free to add new definitions to it or derive ideas from it.

Thus, being open-source is an important enabler of extensibility. However, it does not automatically ensure that a software framework actually is extensible. For any software to be extensible its architecture and codebase need to be developed with this requirement at its roots. If done properly, Object-Oriented Programming (OOP) makes it easy to achieve reusable, extensible code [30]. KADMOS follows this approach as well [95]. Therefore, it is proposed in this thesis that an open-source, OOP framework would be the most suitable solution to the question of how an MDAO system can be developed to be extensible.

Despite its disadvantage of being proprietary/closed source, Optimus does have an important advantage over the open-source RCE framework: it allows for analysis tools to specify gradients, whereas RCE does not. Hence, the pipeline satisfies the requirement of utilizing analytical gradients if Optimus is used. This comes at the cost of sacrificing the first requirement, however. Furthermore, there are no provisions for dealing with analytical gradients in the current concept of the Knowledge Base and within KADMOS/CMDOWS. Therefore, an extension to the concept of the KB would be required.

From this discussion, it is clear that the AGILE pipeline as it exists presently does not fully satisfy the requirements listed before. Therefore, in order to truly convince the industry that MDAO can be a valuable and useful tool in the conceptual design phase, a third end to the AGILE pipeline, next to RCE and Optimus, is needed. This new end needs to be able to parse CMDOWS files containing the definition of an MDO problem, automatically turn it into a representative, executable problem, and make it easy to use gradient information whenever it is available. All the while, this system should be open-source, object-oriented, and should be developed with extensibility and reusability at its core.

An optimization framework that fits the requirements is OpenMDAO, developed by the NASA Glenn Research Center [31, 41, 63, 69, 70]. It is an open-source, object-oriented framework, written in Python, which allows the user to decompose models into a set of smaller components (analogous to DCs). The main advantage of OpenMDAO over RCE and Optimus is its powerful mathematical architecture, Modular Analysis and Unified Derivatives

(MAUD), which enables efficient automatic multidisciplinary gradient evaluation [45]. OpenMDAO has a clear and easy to use API allowing the user expose gradient information of any or all components to the framework. Instead of performing the finite differences for these components to estimate the gradients, the information provided directly by the tools is then used. The system level gradients are computed automatically by OpenMDAO, by carrying out the chain rule from the individual components to the top level. All the while, it remains possible to use black boxes as well. That is; supplying gradients is optional on the discipline level, and not an all or nothing situation.

Furthermore, OpenMDAO allows a problem to be run on High Performance Computing (HPC) environments easily using the Message Passing Interface (MPI). No extra code is needed to achieve this; using OpenMDAO a problem can be set-up to run monolithically, parallel, and distributed with the same codebase. This makes the framework perfectly suited to handle problems on an industrial scale, while

at the same time remaining extensible and easy to use.

This thesis proposes a solution to make MDAO more applicable for the conceptual design phase in an industrial setting in the form of a new open-source software framework, bringing together the strengths of the AGILE pipeline and OpenMDAO. An extension to the notion of the Knowledge Base as proposed within the AGILE project is proposed, which allows for Discipline Components to specify and communicate gradient information. Despite this extension, KADMOS will still be used to formulate and manipulate an MDO problem in this enhanced pipeline, yielding valid CMDOWS files. Then, the extended definition of the KB, along with the definition of the CMDOWS schema, are used to automatically construct a fully functioning OpenMDAO representation of the problem by the proposed framework. What sets the framework apart from the abilities of RCE and Optimus is that is uses the extended notion of the KB to automatically supply analytical gradients to the OpenMDAO framework whenever a DC provides them. The proposed system will benefit from the powers of the tools developed for 3rd generation MDAO, such as KADMOS, and uses them to improve the state-of-the-art of 1st generation frameworks in order to warm the industry to the idea of widespread application of MDAO. This solution is therefore the proposed answer to the main research question of this thesis.

The rest of this report is structured as follows. First, a detailed background of relevant research and technology will be given. This overview is contained in chapter 2. Next, the technical work of the thesis will be presented, in chapter 3. This chapter will describe the strategy and implementation of the proposed software framework meant to answer the main question of the thesis. The capabilities of the framework will be demonstrated by means of two use cases: the Sellar problem and an aerostructural wing optimization problem. These will be described and their results presented in chapter 4. The report will finally be concluded in chapter 5. Attached to this report is a list of references, **??**, and an appendix, appendix A. Parts of these will be referred to throughout the text.

# 2

# Background

## 2.1. Aircraft Design

Aircraft design, as is true for any design process, involves trading off different characteristics against one another to arrive at a final package that is able to perform all desired features within given limits and with a satisfactory performance. The main focus of this thesis is the development of an MDO architecture that should make this trading off more efficient than it is in a traditional design process. More so, the goal is to develop and architecture that is more effective than the current state of the art MDO techniques. It should be applicable to any design process involving multiple disciplines. However, before discussing these abstract concepts, it is important to realize that it was problems related to aircraft design that gave birth to what is now known as MDO [36–38, 40, 60, 78–81]. Furthermore, the test case to asses the performance and abilities of the techniques developed in this thesis will be an aircraft design problem. Therefore it is important to have a solid base overview of what aircraft design is, when and where MDO arises from it, and how this test case fits in the overall design of the aircraft.

This chapter will start by giving a brief overview of the birth of aviation and aircraft design in section 2.1.1. Then an overview is given of the steps that are taken to get from the conception of a new design to the final aircraft, identifying the different phases of aircraft design, and what is taken into account during these, in section 2.1.2. A high-level overview of the different disciplines required to design an aircraft will be given thereafter, in section 2.1.4, along with how these are connected to one another. This will directly show where MDO can be used for aircraft design. Then the driving requirements and objectives of aircraft design are discussed in section 2.1.5. This section will explain that wing design has a large impact on the overall aircraft design, and will therefore be considered as benchmark during this thesis. Finally, section 2.1.6 discusses the current approach of wing design and its shortcomings, the design requirements of wings, as well as recent developments. This section will show that it is important to appreciate the multidisciplinary nature of the design process, and that benefits can be expected if this is done early in the design.

### 2.1.1. History and Current Work

The first flying contraptions created by man were kites, invented by the ancient Chinese around 1000 B.C. [19]. During the ages that followed the invention of the Kite, visionaries did not stop dreaming about achieving human flight. However, it was not until the late 1700s and 1800s that the aerodynamics of flight was starting to be understood and investigated within the modern scientific method.

In the 1890s Otto Lilienthal, a German engineer and aviation pioneer, became known as ”the father of flight” after he became the first man to have flown a heavier than air aircraft in sustained flight. What is more significant about Lilienthal, however, is his rigorous scientific approach. He studied the flight of birds, and documented their wings’ aerodynamics. His observations lead him to write his book ”Der Vogelflug als Grundlage der Fliegekunst”, which translates to ”Birdflight as the Basis of Aviation” [55]. His book describes the fundamentals of flight, the design and aerodynamics of wings, and even discusses the pressure distributions along sections and surfaces. Along with his earlier scientific publications, books, and experiments, Lilienthaler’s work is considered instrumental in the advent of aviation.

The Wright Brothers were well aware of Lilienthal's work, and recognized the importance of his contribution to aviation [104]. However, the Wrights developed their theory of flight based on their own wind tunnel experiments when they found the aeronautical data of Lilienthal's did not give them correct results. Their determination and scientific rigor let to their world famous first powered fights in 1903 [103].

An revolutionary aerodynamic discovery was made in 1904 by Ludwig Prandtl [6]. He presented his concept of the viscous boundary layer at a mathematical congress in Heidelberg, Germany. A publication followed describing the mathematics and implications of this new theory soon thereafter [71]. Prandtl's theory was incredibly important for our understanding of aerodynamics and is still applied today in wing design.

Before World War I aircraft were constructed with wood and fabric [19]. Towards the end of the war, however, metal started to make its way into aircraft construction. In 1915 the first all-metal aircraft took flight for the first time: the Junkers J 1. Adopting metals as the main construction material for aircraft brought a revolution to aircraft design. It vastly increased the possibilities of the designs.

The period between the first and second world wars became known as the ”Golden Age of Aviation” [19]. During World War II flight speeds were pushed to the limit by both sides. However, the wing designs of the time, being straight wings, produced excessive amounts of drag at high speeds. The Germans were onto a solution to the problem, though, as early as 1930. Under the lead of Adolph Bussemann, wing sweep was first investigated in 1935. This concept would enable higher speeds than ever before and remains a standard in modern wing designs.

The next big advance in aviation can be ascribed to another German scientist: Dietrich Küchemann [19]. He played an important part in the development of supersonic flight, helped develop the concept of the delta wing, and is known for his work on the Concorde. His 1978 book ”The Aerodynamic Design of Aircraft” [50] is still considered a standard textbook on aerodynamic design. Küchemann is also known for his work on the transonic area rule and his ”Coke bottle” area distribution to minimize wave drag. However, the transonic area rule is usually attributed to Whitcomb [99]. Furthermore, something even more relevant to this work, both Küchemann and Whitcomb also did important work on novel wingtip designs to reduce drag. Küchemann gives name to his invention, the Küchemann tip, and Whitcomb is the man behind the winglet.

There are many discoveries, revolutions, and advances that were not covered in this short, selected overview of the history of aviation and aircraft design. As mentioned before, the main focus of this work is on novel techniques that use existing knowledge and analysis methods to design complex systems such as aircraft. The interested reader is referred to works such as [19] that focus specifically on the history of the aeronautical industry. However, standard textbooks on aircraft design, such as [46, 50, 72, 74, 92, 93], usually contain a more engineering overview of this history as well. Before concluding this section a selection of state of the art areas of research will be discussed. Once more, only a few topics are highlighted, because the focus is not on the development of these analysis methods.

Active boundary layer control, active flutter control, active gust load alleviation, natural laminar wings, and advanced automated optimization are such areas that are actively being researched in the industry. Active boundary layer control is actually a topic that was already being investigated in the 1920s [42] in the form of suction on airfoils to prevent flow separation. This concept is now actively being investigated in the industry [13, 22, 51, 54, 58, 77].

Active flutter control and gust load alleviation are concepts that promises reduced wing structural weights by actively using the control surfaces to prevent flutter and reduce the excess loads introduced into the structure by gusts, turbulence, and as an effect of maneuvering. Once more, these concepts are not new, but are becoming increasingly important because aircraft designs move to increasingly slender, high aspect ratio wings or reduced sweep angles. The industry is investigating whether these concepts are feasible and if they show enough promise to apply in commercial aviation [100, 101].

Natural laminar wings offer the promise of dramatically reduced drag by increasing the fraction of the wing that experiences laminar flow; hence, moving the transition line as far aft as possible. This concept is not new and has been investigated for decades. The most notable commercial aircraft manufacturer actually looking into NLF in practice is HondaJet [28, 29]. However, it is still investigated today [4].

Last, but certainly not least, advanced automated optimization techniques are being investigated, benchmarked and applied widely. Of course this topic deserves some attention, because it is especially relevant for this thesis. The key idea is to consider many of a designs parameters as design variables

and to define a function of these that ranks designs. Then advanced computer algorithms are applied to change the design variables automatically and repeatedly to seek an improved design point. The crux of automated, computer controlled aircraft design optimization, however, is the excessive computational cost of adequately high fidelity analysis software, as well as the large amount of highly coupled disciplines that need to be taken into account if a somewhat detailed study is to be performed. This is the topic of the next chapter. A notable research group on this topic is the MDO lab at the University of Michigan [59] lead by Joaquim Martins. Martins and his team perform high fidelity aerostructural optimizations of wings using full blown CFD and FEM analyses [48].

### 2.1.2. The Design Process

Most well established aircraft design textbooks divide the traditional design process in three phases [46, 72, 74, 92, 93]

1. Conceptual design phase;
2. Preliminary design phase;
3. Detailed design phase.

The following description of the three design phases is based on [93]. In the conceptual phase largely creative decisions are made w.r.t the configuration and overall layout of a new aircraft. Usually multiple possibilities are explored and compared to one another according to some figures of merit. Crude, empirical relations are employed to get rough estimates of a concepts' viability and performance. Once a promising concept is selected, the design progresses to the preliminary design phase. There is some overlap at the start of the preliminary design phase with the conceptual design phase, where the overall layout can still be adjusted based on early findings from the preliminary phase. In the preliminary phase the configuration is designed and evaluated using predominantly analytical, medium fidelity methods. In this phase, a large amount of iterations are usually required to arrive at a final configuration. Therefore, high fidelity methods are generally avoided in the traditional process. At the end of the preliminary design phase, the configuration is frozen and the detailed design phase is started. This last design phase uses high fidelity methods to determine the exact performance of all the parts of the aircraft. All parts of the aircraft, down to the bolts and rivets, have to be sized and analyzed here. At some point during the detailed design phase, parts of the aircraft as it is at that point are build and subjected to physical tests. This can be seen as the maximum fidelity analysis method one could employ to evaluate the performance of a (part of the) design - it is also the most costly.

The classical schematic that is often shown to summarize this subdivision and the design process as a whole is shown in fig. 2.1, with the three main design phases highlighted.

Indeed, there is some overlap between the different phases of the process. This allows for some back-propagation of findings in more detailed analyses to earlier phases, but overall it is an inflexible and unidirectional process. The main downfall is the fact that the configuration is frozen at the end of the preliminary phase. This means the detailed design phase cannot make adjustments to the configuration if it is found this would be beneficial from more detailed analysis. This can lead to increased detailed design cycle times, dramatically increased cost of manufacturing and, ultimately, unit cost [34].

Grose describes that the traditional design process is functionally oriented [34]. He explains fig. 2.1 as follows. Research and development is a constant, ongoing effort. The different disciplines that affect the performance of aircraft are researched by teams both in-house within an aircraft manufacturer, and independently from one another. The goal of this ongoing R&D is to discover and develop game changing technology. These new technologies are pushed downward by the researches and incorporated into new aircraft designs. By operating as such, aircraft manufacturers tried to always have the latest cutting edge technology and stay ahead of the competition. Grose explains that this approach becomes less and less suitable as the industry matures, and most there is to know about the different disciplines has already been mastered. Solely relying on new technological breakthroughs to remain competitive is then no longer a viable business model. Instead, improvements can be made by attacking the design process itself. As explained before, the traditional process is inflexible and unidirectional. If, on the other hand, the complete design, including the configuration, were still being modified during the detail design phase of the traditional process, or even during manufacturing, the process would of course be extremely hard to control. However, if the vast majority or even the entirety of the process

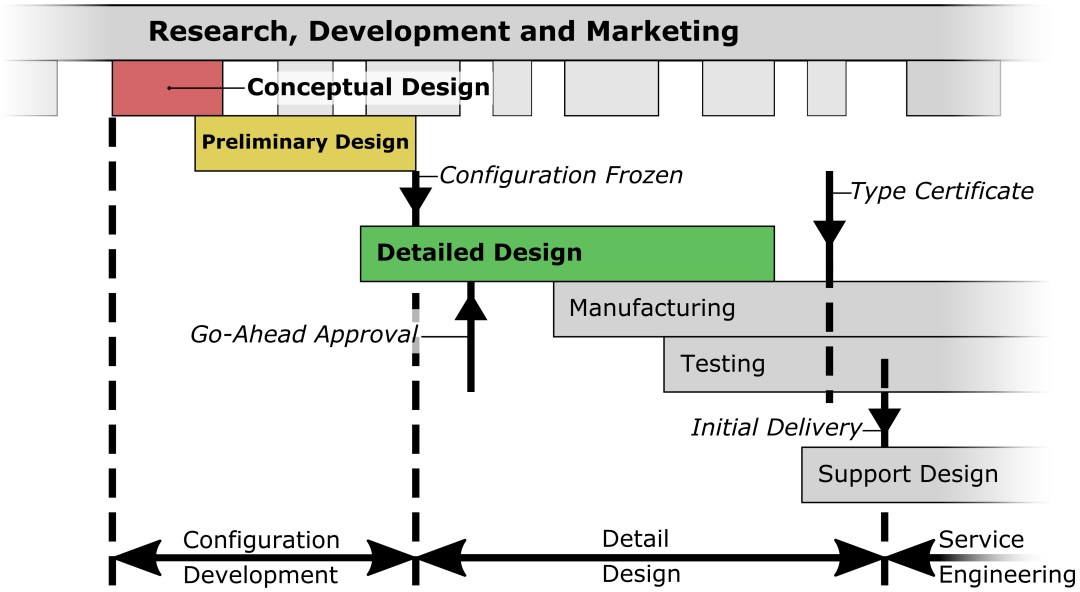



Figure 2.1: Traditional design process schematic. Based on [92].

were digital and automated, it would in theory be possible to take into account high fidelity details in the overall design of new aircraft.

In this 1994 paper, Grose proposes a new model for the process of aircraft design. He calls it a *system oriented approach* [34]. The main idea is to fuse the different disciplines as early on as possible. Sharing knowledge and techniques amongst them. Furthermore, a dedicated team is put in charge of combining the knowledge from all disciplines into a single, best design. This team is now aware of all the disciplines and can take all of them into account, including their interactions. The expertise of the individual disciplines is seen as a set of competences, rather than a set of pipelines, in this new model. This enables the design integration team to direct the different competence teams in a much more collaborative, reactive, and flexible manner. Hence, it is possible to (back-)propagate findings from one discipline to the others in order to account for the interdisciplinary influences. In other words: the multidisciplinary nature of the design process is embraced rather than compartmentalized. Interestingly, this interdisciplinary corporation was not new; scientists like Schmit [78–81] and Haftka [36–38, 40] already realized that better performance could be attained by fusing disciplines starting in 1960. This will be elaborated on in greater detail in **??**, however. What is important to take away from this notion is the fact that MDO automatically arises here from this improved design process model.

### 2.1.3. Design Process Management

In 1981 Steward introduced the *Design Structure Matrix* (DSM) to the engineering community [89]. The purpose of this concept was to visualize and manage the design process of complex systems like aircraft. Figure 2.2 shows the final example Steward gives of a DSM for the design of an electric car concept.

This is a tool still used by engineers today to organize, plan, and manage the design process of complex systems involving multiple, interacting disciplines. Disciplines are listed by name below one another on the left side of the matrix. They inhabit the position in the matrix that is in their respective row, and on the diagonal. In fig. 2.2 these locations are marked with a circle and a cross. The off diagonal crosses and numbers represent variables passed between the disciplines. In Steward's original DSM concept, variables below the diagonal are passed forward, whereas those above the diagonal are passed backward. This is opposite from the modern implementation, where feed-forward variables are above, and feed-back variables are below the diagonal. In the ideal situation only feed-forward variables would be present. If this were the case, no loops would be required to obtain a converged, consistent set of variables. All disciplines would only have to be executed once and in the oder in which they appear in the DSM. However, often there will be disciplines that need input from disciplines that are executed after them, and hence loops are necessary. By partitioning the DSM it can be made clear

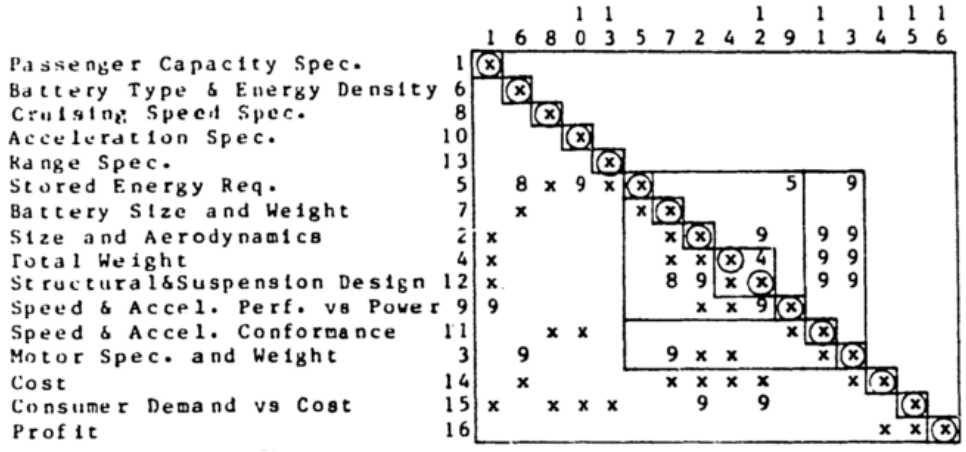



Figure 2.2: Example of a DSM for the design process of an electric car. Taken from [89].

where these loops exist. This is shown in fig. 2.2 with the rectangles. Feed-back variables and loops are encapsulated by these partitions.

The name ”Design Structure Matrix” is often interchanged arbitrarily with the name *N2-chart* (from $N \times N$). The original description of the N2 diagram was first described by Lano in 1977 [53]. The main difference is that it included descriptions of variables passed between the different disciplines.

Recently, Lambe and Martins proposed an extension to the DSM/N2-chart called *Extended Design Structure Matrix* (XDSM) [52]. An example is shown in fig. 2.3.

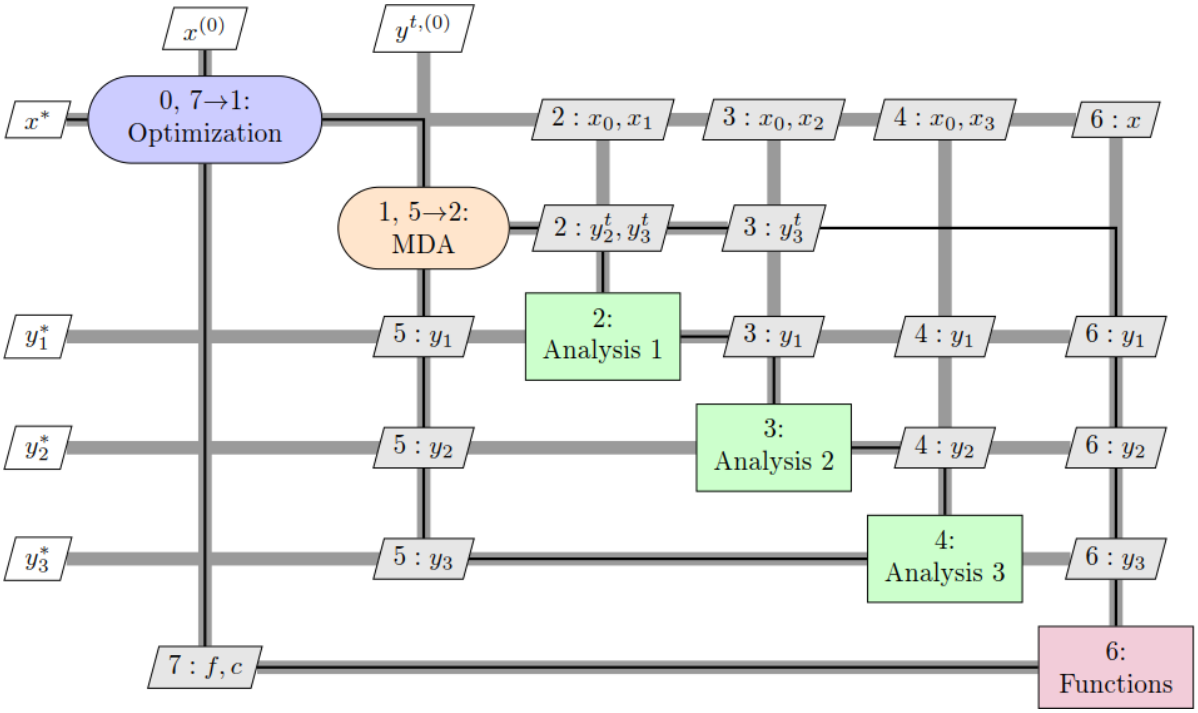



Figure 2.3: Example of an XDSM for a MDF MDO architecture. Taken from [52].

Like the DSM’s/N2-charts, analysis/disciplines are shown on the diagonal and connecting variables are shown on the off diagonal positions. Note that feed-forward variables are always above, and feed-back variables always below the diagonal in XDSM’s, as with modern DSM’s/N2-charts. The definition of the XDSM scheme adds the use of colors and block shapes, and the formalization thereof, to make clear how they differ in function. Variables are represented by parallelograms. White variable blocks represent system in- and output variables (*design variables*), whereas gray blocks represent internal

variables passed between disciplines (*coupling variables*). Green rectangles represent analysis modules; for example, an aerodynamics tool that calculates the drag based on the wing shape. Generally, these modules are considered black boxes, the internal workings of which are not known and/or not of interest to the system integrator. Red rectangles represent function blocks. These do not perform analyses of their own, but instead use output from one or more analysis modules in some form of numerical formula; e.g., to compute objective and/or constraint function values. The orange rounded rectangles represent so called multi-discipline analyses (MDA). In a sense, they combine multiple different analyses modules to yield one, consistent set of variables. Hence, they do not execute any analyses internally, but call their child analysis modules and gather their output until some predefined condition is satisfied; for example, convergence criteria. Effectively, an MDA block with one or more child analysis modules could be replaced by a single analysis block performing the work of the MDA and all the containing analyses. This can be seen as the equivalent of a partition in the DSM. Finally, blue rounded rectangles represent optimizer blocks. They are similar to MDA's: they do not perform any analyses of their own, but instead call all child blocks repetitively until an optimum set of design variables is obtained.

Another important addition that the XDSM scheme describes are the thick, gray data channels and thin, black data flow lines. The data channels make clear how data flows through the system, whereas the data flow lines show the order of execution. Along with these two definitions, the numbers inside the variable and execution blocks are formalized. These numbers indicate the order of execution/use of the respective blocks, starting from 0. Two numbers separated by a comma, followed by a rightward arrow towards a third, signify a loop. The first two numbers are the first and last time the block is executed during the loop, and the number following the arrow is the block to be executed first interior of the loop. The XDSM of fig. 2.3 has two such loops: the optimizer loops between indices 0 and 7, and calls 1 within the interior of its loop first; the MDA block loops between 1 and 5 and starts 2 in its interior loop.

Finally, it is also possible to stack multiple blocks to summarize the XDSM if they are executed in parallel. An example of an XDSM featuring this is shown in fig. 2.4. Here an arbitrary number

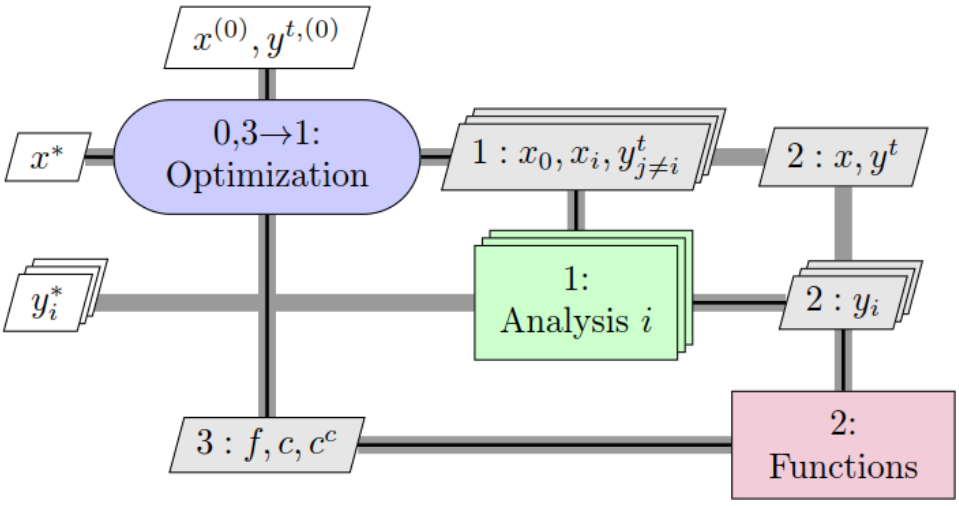



Figure 2.4: Example of an XDSM involving stacked blocks. Taken from [52].

of analysis modules are executed in parallel within an optimization loop. This is an example of an Individual Discipline Feasible, or IDF MDO architecture [60], but more will be said about this in **??**.

### 2.1.4. The Multidisciplinary Nature of Aircraft Design

A general overview of systems design processes has been given (section 2.1.2) and a means to manage them has been discussed (section 2.1.3). Now it is time to discuss aircraft design specifically within those frameworks.

Figure 2.5 shows the typical disciplines involved in the design of an aircraft and the connections between them as presented by Sobieszczanski [88]. He also shows a corresponding N2-chart. However, it is not representative of all the connections in fig. 2.5. Instead a XDSM representative of all the

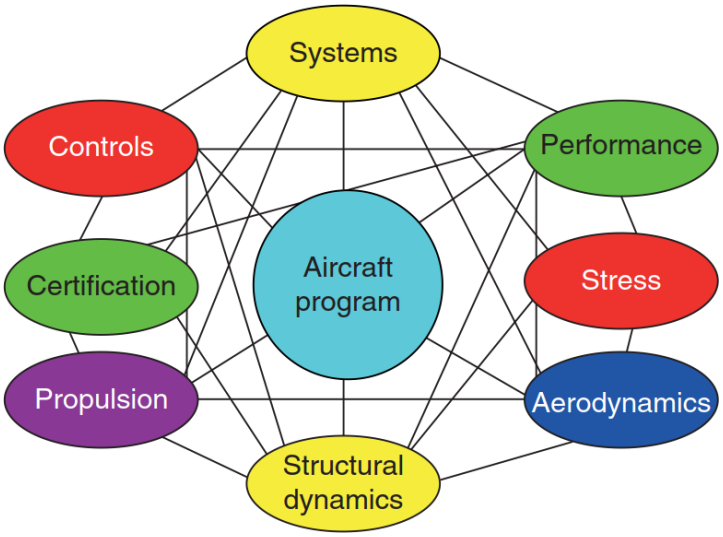



Figure 2.5: Typical aircraft design disciplines and their connections. Taken from [88].

connections is shown in fig. 2.6. The central ”Aircraft program” block from fig. 2.5 is represented by

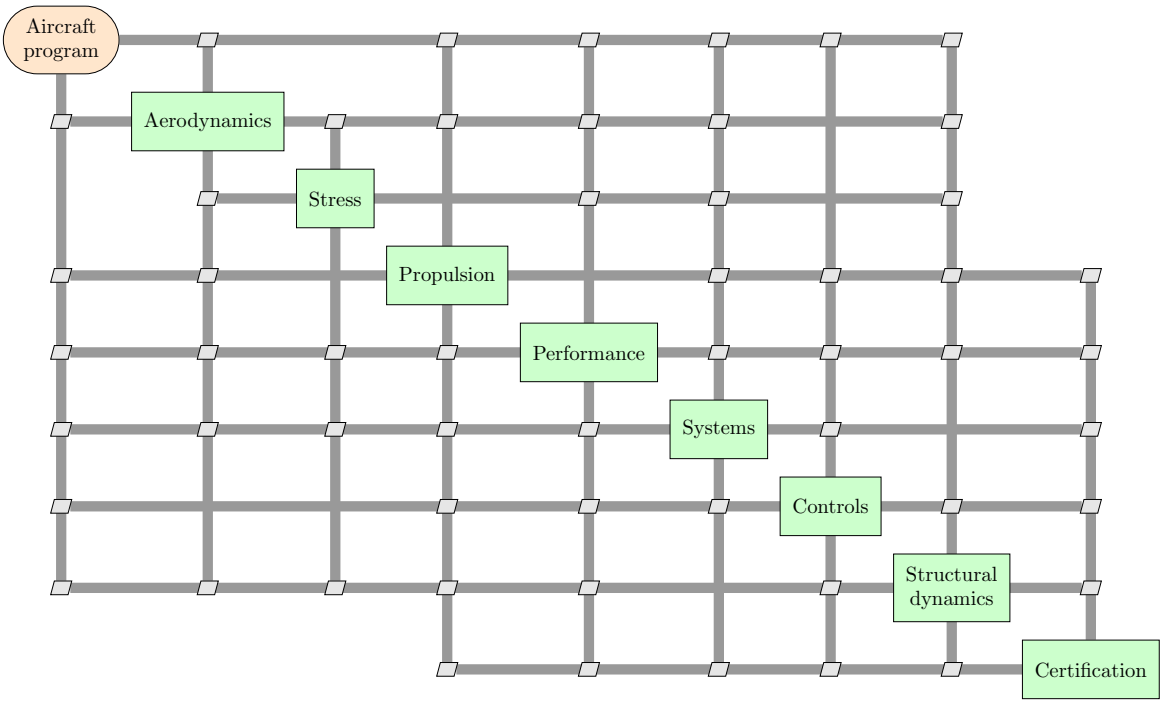



Figure 2.6: XDSM representation of fig. 2.5.

an MDA block here. This block can be seen as a common database containing both geometrical and performance characteristics of the aircraft that can be used and fed by the different disciplines. When used in an optimization, this block could be in charge of delivering a consistent design to the optimizer, for example. All the other disciplines are represented by analysis blocks. The connections between them are represented by empty data channels, since the exact variables that are passed between the disciplines and the aircraft program were not specified in [88]. However, from these figures it is directly apparent that aircraft design is indeed very multidisciplinary and that there are a lot of interdisciplinary interactions.

### 2.1.5. Design Requirements and Objectives

Like any design, aircraft design is driven by design requirements. These requirements are imposed by government regulations, such as the Federal Aviation Regulations (FAR) [24] and the European airworthiness requirements (CS25) [23], as well as by the market/customer. Certification requirements put lower limits on performance and safety aspects. New aircraft have to fulfill all of these to be allowed entry into service. Market/customer requirements are generally not directly imposed on the design by customers. Instead the manufacturers make an assessment of what what they predict the market desires in a new aircraft. The market requirements can be put into the following categories [49]:

- Community acceptance;
- Airport compatibility;
- Economic efficiency;
- Reliability and robustness.

Community acceptance mainly has to do with how well passengers evaluate the aircraft, as well as how noisy the aircraft is. Airport compatibility determines which airports an aircraft can service, putting requirements and restrictions on, for example, landing/takeoff distance, airport gate space, etc. Economic efficiency mainly refers to how cost effective operating an aircraft is. This depends on factors like fuel usage, turnaround time and maintenance costs. However, the latter is also captured by the reliability and robustness. For this thesis the main driver for the economic efficiency category is considered to be fuel efficiency, which is valid especially during conceptual/early preliminary design. Finally, reliability and robustness refer to maintainability. A reliable aircraft will require less maintenance, and a robust design makes maintenance simpler and cheaper.

After the hard requirements imposed by governmental regulations, economic efficiency is usually considered the most important driver of aircraft design and optimization. As was just mentioned, in this thesis the main economic factor for aircraft operators is considered to be fuel efficiency. A good measure for the fuel efficiency of aircraft is the specific range (SR). This is a measure used by Airbus internally as well when determining and optimizing the Direct Operating Costs (DOC) [7]. The specific range is defined as the ratio of the distance covered and the fuel burned. According to [7] the specific (ground) range can be written as

$$\mathrm{SR} = \frac{a_0 \left( M \frac{L}{D} \right)}{\frac{\mathrm{SFC}}{\sqrt{T/T_0}} W}, \tag{2.1}$$

where SR is the specific range considering the ground distance covered in $\mathrm{m\,N^{-1}}$, $a_0$ is the speed of sound at sea level in $\mathrm{m\,s^{-1}}$, $M$ is the Mach number, $L/D$ is the aerodynamic efficiency, SFC is the specific fuel consumption in $\mathrm{N\,s^{-1}\,N^{-1}}$, $T/T_0$ is the temperature fraction at the flight altitude, and $W$ is the weight of the aircraft in N. However, a measure that is more useful when comparing the fuel efficiency of different aircraft with potentially different missions, can be obtained by multiplying both sides of eq. (2.1) with the payload weight, $W_{\mathrm{PL}}$. Let this parameter be known as the payload specific range (PSR). It can be written as

$$\mathrm{PSR} = \frac{a_0 \left( M \frac{L}{D} \right)}{\frac{\mathrm{SFC}}{\sqrt{T/T_0}} \frac{W}{W_{\mathrm{PL}}}}. \tag{2.2}$$

This new quantity expresses how far a given payload weight can be flown per unit fuel burned, its units are $\mathrm{N\,m\,N^{-1}}$. It depends on three factors that can be influenced to increase efficiency:

- The aerodynamic characteristics of the aircraft, $M \frac{L}{D}$;
- The engine characteristics, $\frac{\mathrm{SFC}}{\sqrt{T/T_0}}$;
- The weight to payload weight fraction, $\frac{W}{W_{\mathrm{PL}}}$.

The engine characteristics are fully captured by the fraction of the SFC and the temperature fraction, and not by the SFC alone, because the operation of the engine depends heavily on the temperature and hence the external environment. Installing a more efficient engine means this fraction becomes smaller, which leads to an increased PSR. The term $M\frac{L}{D}$ fully captures the aerodynamic characteristics of the aircraft. The fraction $L/D$ by itself is the aerodynamic efficiency, but it depends on the Mach number, $M$. Therefore the product of the two is a more accurate measure of the aircraft's true aerodynamic efficiency taking into account this Mach number dependence. An aerodynamically more efficient aircraft has a higher value for this term, and therefore a higher PSR. Finally, the weight fraction captures how efficient the aircraft has been designed in terms of weight. Reducing the structural, systems, and fuel weight reduces this fraction and therefore increases the PSR.

In this thesis engine characteristics are not taken into account. Thus the aerodynamic and weight characteristics of the aircraft are considered the driving factors in the design of an economically efficient aircraft. As will be discussed in the next section these two should not be considered separately of one another, because they are highly coupled.

### 2.1.6. Wing Design

Traditionally the wing design process is as follows, as outlined by Roskam in [75]. First a number of different configurations are thought up by a small group of engineers. Then, in the conceptual design phase, the general planform parameters such as the surface area, aspect ratio, span, etc. are determined through an iterative process. The design is based on regression analysis of data from reference aircraft that perform similar missions as the new design is intended and low fidelity analysis tools, as was explained for general aircraft design in section 2.1.2. The most promising configuration is then selected from the available options.

Next, the aerodynamics team designs the wing in more detail, based on higher fidelity analyses methods. The end results is a refined set of wing design parameters along with a description of the so called "cruise flight shape" [102]. The cruise flight shape shape is the shape the wing assumes under design cruise loads. It is the shape for which the wing has been optimized by the aerodynamicists; it is what they intend the wing to look like during cruise.

Then the design is handed over to the other design teams. The structures team is put in charge of designing a wing structure able to handle the loads imposed on the wing during on- and off-design conditions, as specified by FAR/CS25 [23, 24], and transfer those into the rest of the airframe. Load conditions such as a $2.5g$ pull-up maneuver, high speed dive, and ground loads are taken into account, for example. Regardless of these load cases and the structural requirements introduced onto the structural design by these, the structures team has to design the structure in such a way that the wing assumes the intended cruise flight shape specified by the aerodynamics team during cruise conditions. Obviously this greatly limits the design freedom and agility. However, the process is linear and easy to manage with limited technology and resources, which is why it made its way into the standard commercial design process.

This static, unidirectional process not only has the potential to lead to suboptimal designs, it can also lead to a number of dangerous instabilities caused by the interaction of the structural deformations and aerodynamics during off-design operation if these are not taken into account. This interaction between structural deformations and aerodynamics is what is known as aeroelasticity. Airworthiness requirements put stringent limits and margins on the aircraft's performance during these off-design conditions. An important example of this are the specified gust loads that the aircraft has to be able to sustain. During gusts the loads on the airframe can fluctuate violently. High loads can be imposed on the structure that need to be absorbed without failure. In the traditional design process mathematical methods are employed to predict the effect of these loads. Based on the predicted stresses and torque resulting from these loads the structure is stiffened to avoid failure, often with a relatively large safety factor.

Luckily there are also success stories of endeavors to attack the problem in a more fruitful manner. Passive load alleviation, also referred to as aeroelastic tailoring, is a concept that uses a clever wing structure design that neutralizes the feedback between the aero loads and structural deformations causing excessive stresses and torque in the structure. More effort is required to be put into the design, and intensive, two-way communication and feedback between the aerodynamics and structures teams is required to achieve this. However, in return the final weight of the wing structure can be reduced.

Gust load alleviation can be taken a step further still. Active aeroelstic wings use control surfaces

to purposefully twist and deflect the wing at will. By doing so, not only gust load alleviation can be achieved, but maneuver loads can be reduced, and dynamic phenomena such as flutter can be controlled as well. The latter is known as active flutter control. Furthermore, active aeroelastic control can be used to increase the maneuverability of the aircraft beyond classical limits. Successful flight tests were performed by a joint team of the US Airforce, Boeing's research division Phantom Works, and NASA with a modified F/A-18, the X-53, displaying the advantage of even more cooperation between structural and aerodynamic design teams.

Besides gust and maneuver loads several aeroelastic instabilities need to be considered as well. What follows is a summary of the most common instabilities. They can be put into two categories: static and dynamic.

Static instabilities are caused by a positive feedback between aerodynamic loads and the resulting structural deformations. Two such instabilities are observed in aircraft: divergence and control reversal. Divergence occurs when positive wing bending induced by aerodynamics loads increases the wing twist (positive, nose up). A larger wing twist causes the load to move further outboard and also further increases the lift. This in effect leads to an increased bending moment which bends the wing further, and increases the twist even more, etc. This leads to a theoretically infinite bending moment, which of course leads to structural failure of the wing. Control reversal is in a way the opposite effect. It occurs when a trailing edge control surface deflection causes the wing to twist in the opposite direction. This can happen when a downward deflected trailing edge locally moves the application point of the lift vector aft and behind the elastic axis of the section. This causes a negative bending moment about the y-axis, twisting the affected part of the wing with its leading edge downward. The local angle of attack is then reduced, which leads to the opposite effect that was expected and intended when the control surface was deflected.

Dynamic instabilities are caused by a delayed interaction between structural deformations and aerodynamic loads, leading to oscillatory motion. Flutter and buffeting are two examples of such instabilities that can occur during flight. Flutter is a direct coupling between loads and deformations of the wing. Buffeting happens when shocks on the top and bottom surface of the wing oscillate back-and-forth, which in turn leads to an oscillation in the magnitude and action point of the local aero loads. Both lead to an up-and-down shaking or vibrating and/or back-and-forth twisting of the wing, making control difficult or impossible and lead damage to the aircraft and its payload. Besides the dangers of these effects, they are to be avoided when designing a commercial jet in particular as well, because they cause dismal ride quality.

The airworthiness requirements stipulate limits within which an aircraft should never experience these instabilities. For example, CS25 stipulates that transport aircraft should not experience buffet at speeds of up to 15% above the dive speed [23]. In the traditional wing design process divergence was originally solved by simple trial and error. Control reversal was a phenomenon that was often encountered by test pilots, leading to accidents more than once. The same goes for flutter and buffet. American test pilots of experimental aircraft would often report that the aircraft would start shaking and trembling uncontrollably when flying it at or above certain speeds. The shaking would stop once the speed was reduced. Obviously this all revealed a lack of knowledge and appreciation for the intricate interaction of aerodynamics and structures in the traditional design process. As a remedy models were deviced to predict when these instabilities would occur. Aircraft were designed to stay well within these limits within the full anticipated range of the flight envelope, often with heavy safety factors. The net effect of this approach is a heavier structure and/or limited wing designs.

Modern aircraft designs use new materials and techniques that are much lighter and stronger, such as composites and composite-hybrid materials. If not properly designed, these materials hold the potential to make an aircraft more sensitive to gust and maneuver loads, and more susceptible to aeroelastic instabilities. The unidirectional design process has already proven to be problematic in this context. For example, during data review the FAA concluded that Boeing's new version of the 747, the 747-8, had a flutter problem at certain load cases [25]. If there had been more interaction and feedback between the aerodynamics and structures departments, this could have been foreseen and avoided. To address the issue without having to redesign the entire aircraft, Boeing modified the control system to actively combat the flutter at the affected flight/load conditions. They successfully convinced the authorities that the system was sufficiently capable of removing the flutter and sufficiently reliable to justify a controller approach to solving the problem [56]. In effect, the safety margin on the requirements of a the structure alone to handle flutter had been pushed back.

This trend has been continued since, allowing manufacturers to save weight on the structural design by implementing active load alleviation controllers. An interesting question can be posed at this, though: would it be possible to realize even more savings by fully embracing the aeroelastic nature of wings early in the design process and tailoring the wing, both passively and actively, given the possibility of pushing back the margins? This is the question being addressed by, for example, the Flexop project, funded by the EU [27].

## 2.2. Multidisciplinary Design Optimization

Designing a new, complex system almost always involves more than one discipline - usually a wide range of them. This was highlighted in section 2.1. This multidisciplinary nature has led to the birth of Multidisciplinary Design Optimization (MDO), which, according to Martins and Lambe [60], can be attributed to Schmitt [78–81] and Haftka [36–38, 40], who involved more than one discipline in structural optimization studies. Since then it has evolved into a mature, separate field of research. Aircraft design in particular is an excellent example of a highly coupled, multi-objective and multidisciplinary practice. These three markers, high level of coupling between disciplines, multiple objectives, and large number of disciplines, are the main reasons for using MDO [60].

One of the most extensive reviews of multidisciplinary design optimization approaches is the work of Joaquim R. R. A. Martins and Andrew B. Lambde [60]. This work could very well be used as a textbook for scientists and engineers seeking an introduction into the field of MDO, and as a reference for those experienced with it in the field. It is cited by nearly all authors in the field of MDO that published after its publication, that were surveyed for this literature study. This makes it one of the most relevant references within the entire field of MDO, even though it was published four years ago as of writing this thesis - in 2013.

The paper starts with an overview multidisciplinary design optimization. It describes what MDO is in the most generic sense:

> ”[...] a field of engineering that focuses on the use of numerical optimization for the design of systems that involve a number of disciplines or subsystems.” [60]

The paper explains that the need to use MDO arises from the fact that the performance of complex systems has a strong dependency on the interactions of the disciplines that govern them. Hence, solely considering each governing discipline separately leads to suboptimal designs. This was also discussed in section 2.1. Furthermore, the design process can be sped up by employing automated design tools and applying MDO from the beginning.

Next, a brief history of MDO is given. Matins and Lambe attribute the birth of MDO to eight papers coauthored by Schmit [78–81] between 1960 and 1984, and by Haftka [36–38, 40] between 1973 and 1979. By doing this, Martins and Lambe recognize the importance of Schmit’s and Haftka’s early work for what would evolve into one of the most important engineering fields in modern engineering design.

What follows is an account of this history, by review of the papers by Schmit and Haftka that Martins and Lambe mention. These works are the first contributions to creating the field of multidisciplinary design optimization. Therefore they bear important historical significance to MDO and therefore to this thesis. Although the actual optimization usecases described in these papers are outdated, these papers introduce and define concepts and terminology still used in the current state-of-the art research. In this respect they are relevant to this thesis because of more than their historical value, because they define the common point of departure still employed by scientists and engineers today.

### 2.2.1. The Birth of MDO

Lucien A. Schmit researched aeroelasticity and structural analysis. He approached designing and analyzing structural systems by what he called ”systematic synthesis”, as he describes in [80]. This is significant, because it is a first description of numerical optimization for engineering problems. Schmit was also interested in aerodynamic analysis and design. He published a book on supersonic airfoil design in 1965 [79]. In this book Schmit describes design methods that also account for aeroelastic effects in the design of hypersonic airfoils. It is the first work that describes engineering design techniques combining multiple disciplines. Hence, it is a very important, first reference for the advent of multidisciplinary design optimization. In his 1981 and 1984 works [78, 81] he details his concept of structural synthesis further. He defines the concepts of design variables, objective and constraint

functions. These definitions form the cornerstone of all optimization approaches now. However, it is significant to realize that this was one of the first works to list these definitions so that readers of the paper and researchers in the field would have, as Schmit writes himself, ”[...] a common point of departure” [78].

Raphael T. Haftka also contributed greatly to the development of MDO. Like Schmit, Haftka was interested in coupling advanced analysis software, like Finite Element Methods (FEM), to optimization codes. In 1973 Haftka published a paper titled ”Automated Procedure for Design of Wing Structures to Satisfy Strength and Flutter Requirements” [36]. The indications of this paper are pristine, as it describes a true multidisciplinary design optimization of wings, taking into account aerodynamics, structures, and even control. Then in 1975 he published another paper in which he compares two different optimization procedures taking into account flutter requirements [40]. This is a move towards treating MDO in a more abstract manner, discussing possible ways to approach an optimization problem, as is done in the modern literature about MDO architectures such as this thesis. In 1977 Haftka coauthored yet another paper on wing optimization [37]. In this paper the wing planform shape is also taken into account, and aerodynamic characteristics are constrained and optimized. This problem is still attacked today by scientists and engineers. Haftka’s paper is therefore profoundly important historically, since it is the first of its kind in this large research field. Later, in 1979, thermal analysis is added as another discipline to his optimization studies [38]. Haftka authored and coauthored countless other papers and books on the matter. In fact, in 2016 a paper was published by the title ”Parallel surrogate-assisted global optimization with expensive functions – a survey” [39], which is precisely one of the techniques this thesis focuses on.

### 2.2.2. Definitions and Terminology

As was mentioned in section 2.2.1, Schmit’s 1981 paper, [78], is one of the first papers to list the definitions and concepts still in use today when describing optimization problems. It is fitting to use this paper as reference for listing the definitions and terminology of MDO.

Two types of parameters are defined by [78]:

- Preassigned parameters;
- Design variables.

The first denote constant parameters of a design that are either not allowed to be changed, or parameters which cannot be controlled. Design variables, on the other hand, are parameters that can be changed, and should be changed, in order to find an improved or optimal design.

Next, Schmit describes the role of load conditions and failure modes. In the context of his work, which is in the field of structural engineering, load conditions refer to a combination of mechanical and/or thermal loads to which the design will be subjected. Failure modes are described as ”[...] structural behavior characteristic subject to limitation” [78]. The combination of load conditions and failure mode limitations enter into an optimization problem in the form of (in)equality constraints. These constraints depend on the design variables, and determine whether a set of design variables is feasible or not.

Finally, an optimization algorithm needs a means to evaluate and compare designs. This is done by defining an objective function, which is dependent on the design variables, and attributes a score to each combination of design variables.

A slightly different/extended terminology and classifications of parameters will be used here, following the example of Martins and Lambe [60]. Table 2.1 summarizes both the different named parameters and the corresponding mathematical notation. This table has been adopted directly from [60]. The specific terminological terms have been printed in bold.

Design variables have the same definition as described by Schmit [78]. These could be parameters such as the wingspan and aspect ratio. Schmit’s preassigned parameters are not explicitly mentioned in this terminology. They are considered non-present in the optimization problem formulation. They could, however, consistently be put under the header of state variables within this terminology.

Coupling variables are considered part of the design variables by Schmit. Here, however, they are considered separately because they play a different role in the MDO architectures than regular design variables do. Furthermore, the ratio of the amount of coupling variables and the number free design variables can be an indication of how highly coupled an MDO problem is. This indication can aid the selection of a suitable MDO architecture, since some architectures are better suited for highly

Table 2.1: Terminology and corresponding mathematical notation for MDO problems. Adopted from [60].

| Symbol | Definition |
|---|---|
| $x$ | Vector of **design variables** |
| $y$ | Vector of **coupling variables** (outputs from discipline analyses) |
| $\bar{y}$ | Vector of **state variables** (variables used inside only one discipline analysis) |
| $f$ | **Objective function** |
| $c$ | Vector of **design constraints** |
| $c^c$ | Vector of **consistency constraints** |
| $\mathcal{R}$ | Governing equations of a discipline analysis in residual from (**discipline analysis constraints**) |
| $N$ | Number of disciplines |
| $n_{()}$ | Length of a given variable vector |
| $m_{()}$ | Length of a given constraint vector |
| $()_0$ | Functions or variables that are shared by more than one discipline |
| $()_i$ | Functions or variables that apply only to discipline $i$ |
| $()^*$ | Functions or variables at their optimal value |
| $\tilde{()}$ | Approximations of a given function or vector of functions |
| $\hat{()}$ | Independent copies of variables distributed to other disciplines |

coupled problems than others [60]. An example of a set of coupling variables when performing an aeroelastic optimization of a wing are deformations of the structure. These are calculated by a structural analysis module given a loading distribution, and are needed by an aerodynamic module to calculate the aerodynamic forces and moments for the deflected wing shape.

The state variables were already mentioned and require little attention. They are contained within a single discipline analysis. These values are generally not returned as outputs by analyses and are not of interest to the rest of the MDO architecture. Especially when a discipline analysis module is a black box, these variables are even unaccessible from the outside. For example, if a vortex lattice method (VLM) is used to calculate the lift and induced drag of a wing, influence coefficient matrices are calculated by the tool. These are generally only used inside the VLM and can thus be considered state variables.

Objective functions take design variables and coupling variables as input and return a scalar value. These values can then be used by optimizers to find an optimal design point. This could be a function to calculate the amount of fuel required to perform a mission given the aerodynamic, structural, and engine characteristics, for example. Design constraints are similar to objective functions. They also process design and coupling variables into scalar values. Generally an MDO problem has multiple constraints. These are then grouped in a vector of the constraint values. In [60] only inequality constraints are considered. This convention is adopted in this thesis as well. An example of a design constraint in aircraft design is a minimum cruise velocity or the requirement that stresses in the structure always be lower than the yield stress of the material.

When disciplines are connected by one or more coupling variables, they have to be run in sequence. However, they can be decoupled by making a copy of the involved coupling variables such that they can be run in parallel. The result is that the coupling variables calculated by one module and the copies from the previous iteration used as input by another may be different, or inconsistent. To drive the optimizer towards convergence a set of equality constraints are put in place that are satisfied when the copies of the coupling variables match their actual, updated values. These constraints are referred to as consistency constraints.

Finally, sometimes it is possible to exploit knowledge of the internal structure of an analysis discipline. In this case the previous statements about state variables are not necessarily true and they can be controlled by the optimizer directly. To describe how the modules function internally the discipline analysis constraints are defined. These constraints pertain to the link between the coupling variables state variables inside the disciplines. That is $\mathcal{R}_i = 0$ if the state variables $\bar{y}_i$ are the solution of the governing equations of a discipline analysis for a given set of input variables $y_i$. If this construction is possible, then the optimizer is fully in control of convergence, even within the discipline analyses.

### 2.2.3. MDO Architectures

Martins and Lambe define an MDO architecture as the ”[...] combination of problem formulation and organizational strategy [...]” [60]. They describe that an architecture both describes how different models are coupled to one another, as well as how the optimization problem as a whole is solved. MDO architectures are put into two different categories:

- Monolithic architectures;
- Distributed architectures.

Monolithic architectures solve only one optimization problem, which is in fact the main problem to be solved. Distributed architectures, on the other hand, split the overall problem into smaller subproblems and delegate optimizing those to dedicated optimizers. A system optimizer directs these sub-optimizers and is in charge of controlling the overall system optimization.

Selecting an MDO architecture and optimization algorithm(s) may seem arbitrary and an issue of semantics, however it can have a large impact on the successful execution and optimality of the final design. Depending on the type of problem to be solved not all MDO architectures and algorithms are suitable, whereas some may be especially inclined towards a certain problem [60]. Martins and Lambe explain that, for example, the choice between a gradient-based and gradient-free optimization algorithm should be driven by the possibility to execute the analysis module(s) in parallel within the framework of a given MDO architecture. Generally, a gradient-free algorithm may be capable of finding a global optimum, whereas a gradient-based algorithm does not guarantee that, but at the price of an increased number of function evaluations. If gradient information can be calculated efficiently, then gradient-based algorithms should be preferred if parallelization is not feasible. However, if parallelization is supported then gradient-free algorithms are of interest. This could be the case, for example, for distributed architectures, as explained by Martins and Lambe.

The main contribution of Martins and Lambe’s paper is the uniform terminology they use to described vastly different MDO architectures and how they use this to define an overarching classification of architectures. As has undoubtedly become apparent to the reader by this point, this paper is used as the backbone of this chapter and, in fact, of the thesis. As has been mentioned before, this work has been cited time and time again by publications about MDO related topics. This strongly underlines the importance and scientific relevance of this work - not just for this thesis, but for the field of MDO in general. Martins and Lambe’s MDO architecture classification will therefore be summarized here. It will be used as a road map onto which the architecture to be developed during this thesis will be overlain. By doing so a clear picture will be given of the place of this thesis work within the field of MDO in a well established framework.

Martin and Lambe’s classification of MDO architectures is based on the difference between two well known monolithic architectures known as the Individual Discipline Feasible (IDF) and Multiple Discipline Feasible (MDF) architectures. These can themselves be linked to the All At Once (AAO) and Simultaneous Analysis and Design (SAND) architectures respectively, as described in [60]. All four of these are monolithic architectures. Since these four architectures from the departure point of Martins and Lambe’s classification they will be described in some detail first. This should give the reader new to the field of MDO a good stating point and will therefore ensure a common point of departure to go forward from.

#### Monolithic Architectures

Example XDSM’s of the four monolithic architectures, AAO, SAND, IDF, and MDF are shown in figs. 2.7a to 2.7d respectively. These have been taken directly from [60]. Indeed, as was described to be the property that all monolithic architectures share before, in all of these figures there is only one optimizer present. Besides this similarity, however, they are otherwise quite different.

The AAO architecture can be described as the most basic, straightforward approach to solving a multidisciplinary design optimization problem. It could be said to be the ”brute force” way of translating the mathematical formulation of an optimization problem directly into MDO. As can be seen from its XDSM, the optimizer is given control of all design, coupling, and state variables. Consistency of coupling variables between different disciplines is not directly guaranteed by the sequence of operations of the architecture and are under the control of the optimizer. Likewise, consistency between coupling and state variables is directly controlled by the optimizer as well. Overall system consistency and

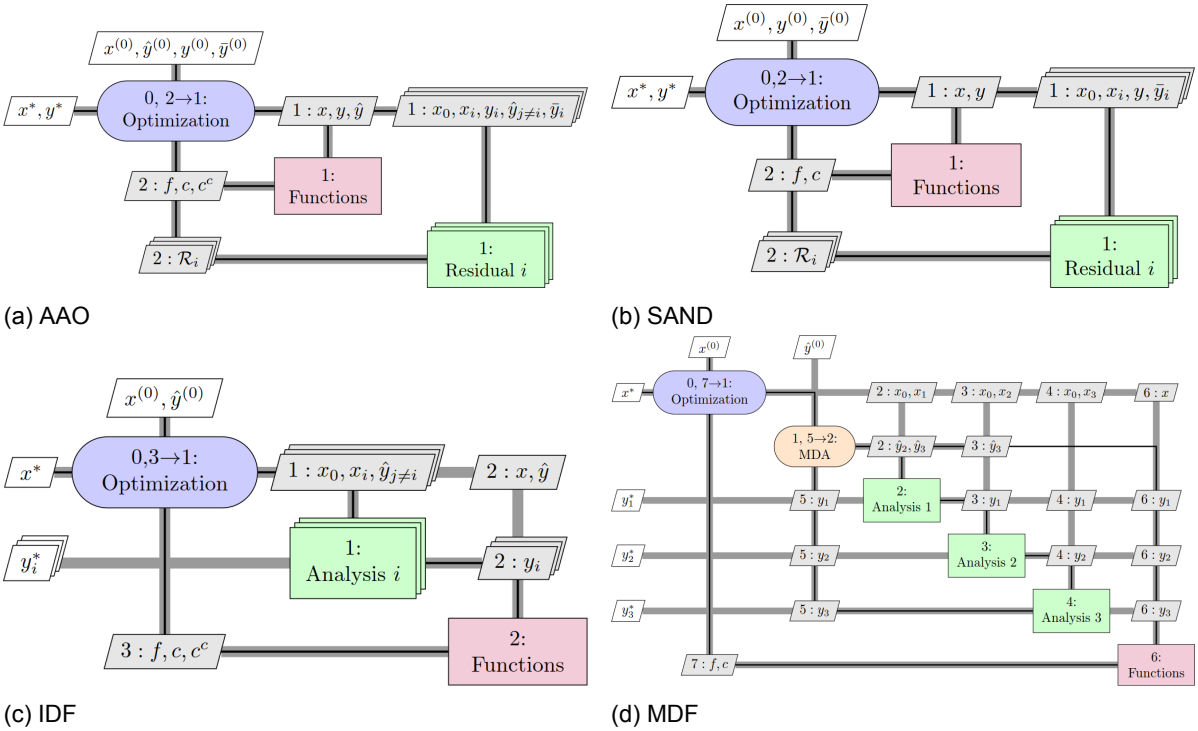



Figure 2.7: The four monolithic MDO architectures. Taken from [60].

feasibility are likewise controlled by the optimizer directly. According to Martins and Lambe this architecture is never implemented in reality. Their explanation for this is the fact that the (linear) consistency constraints can easily be eliminated [60].

In fact, if the consistency constraints are eliminated from the AAO architecture an equivalent SAND architecture is obtained. In fig. 2.7b this is obvious: the parameters $c^c$ and $\hat{y}_i$ have been removed w.r.t. fig. 2.7a. What this means mathematically is also quite straightforward. Instead of making copies of the coupling variables, one consistent set is used. Thus the copies are removed and the consistency constraints become redundant and obsolete. Like the AAO architecture, the SAND architecture gives control of the state variables to the optimizer. This allows for the optimizer to explore infeasible regions of the design space that would otherwise be inaccessible. As Martins and Lambe describe, this has the potential to lead to quick solutions.

The problem with both of these architectures is twofold. First of all, it needs to be possible to directly control the values of the analyses' state variables. This is a major restriction, because this is certainly almost never possible. As mentioned before, most analysis tools are presented as black boxes. The internal process is obscured and cannot be influenced. If such modules are present in the MDO problem it is not possible to apply AAO or SAND architectures. Secondly, there is a risk of obtaining an infeasible and inconsistent design when the optimizer stops prematurely. This is due to the fact that the optimizer is directly responsible for the internal consistency of the analysis modules. If the optimizer stops before their residuals have been brought (close) to zero, the design point is in fact nonexistent. The computational expenses have then been in vain, because the design cannot be used in any way [60]. For these reasons these architectures are very seldomly applied in practice. As Martins and Lambde describe, however, they form the basis from which the IDF and MDF architectures depart, and subsequently form the basis of their classification of distributed architectures. Therefore it is worthwhile to discuss them.

Like the SAND architecture was obtained from the AAO architecture by eliminating the consistency constraints, so the IDF architecture is obtained from it by removing the discipline analysis constraints. Again this is clearly visible by comparing fig. 2.7c to fig. 2.7a and concluding that, indeed, the residuals, $\mathcal{R}_i$, and state variables, $\bar{y}_i$, have been removed. Furthermore, the functions block has been put in sequence and behind the analyses blocks. This reflects what has been done to achieve the elimination of the residuals and state variables: the state variables, $\bar{y}_i$, and coupling variables, $y_i$, of an analysis

module are considered functions of the design variables, $x_0$ and $x_i$, and copies of the coupling variables not pertaining to that analysis module, $\hat{y}_{j\neq i}$. The analysis modules can now be considered black boxes that respond to an input with an output. Like in the AAO architecture, consistency on the system level is kept under the control of the optimizer. This still allows the optimizer to explore infeasible regions of the design space, which can potentially aid convergence. However, the problem remains that an infeasible design can be obtained when the optimizer stops prematurely. Nevertheless, unlike with the SAND architecture, such a situation is not completely worthless, because the individual disciplines are at least consistent and feasible individually. Note that this is precisely where this architecture gets its name: Individual Discipline Feasible.

Finally, the MDF architecture is obtained by eliminating both the consistency constraints and the discipline analysis constraints. Once more it can by observed that the corresponding parameters are indeed all absent in fig. 2.7d. However, it is less obvious how the XDSM of the MDF architecture follows from that of the AAO under these eliminations. The way to understand this transition, is to consider the three example analyses along with the new Multi Discipline Analysis (MDA) block as one, combined analysis module. In that case the shape of the XDSM is exactly equivalent that of the IDF architecture. This is not just a trick to comprehend the changes: it is exactly the point of the MDF architecture. By eliminating both the consistency constraints and the discipline analysis constraints from the system problem formulation, the overall system being handled by the optimizer is always fully consistent. This is an important advantage that the MDF architecture holds over the other three. Unlike the other three, a premature stop of the optimizer still returns a consistent design.

At discipline level consistency is achieved by switching from the residual formulation to the implicit functional, or black box formulation. At the interdisciplinary level, however, something external needs to be done to ensure overall consistency. This task is delegated from the optimizer to the MDA block in the MDF architecture. The MDA block internally works similarly to what the optimizer would do to control system consistency. However, it lacks the task to do this in the direction of an improved design. For example, the MDA block could perform a simple Gauss-Seidel iteration to converge the analyses. The obvious disadvantage is that this convergence iteration has to be performed for every single optimization increment, leading to an increased number of function evaluations. However, by properly arranging the analysis modules and selecting an appropriate convergence algorithm this process can be much improved [60]. Martins and Lambe note that another obvious advantage of the MDF architecture over the other three is the fact that it is the smallest problem formulation possible. The optimizer only controls the design variables, objective function, and design constraints.

As described, the SAND, IDF, and MDF architectures can be obtained from the AAO architecture by removing certain combinations of constraints and variables from its formulation. To clarify and summarize these relationships a commutative diagram can be drawn with the four monolithic architectures at its four corners. Such a diagram is shown here in fig. 2.8, which has been modified from [60].

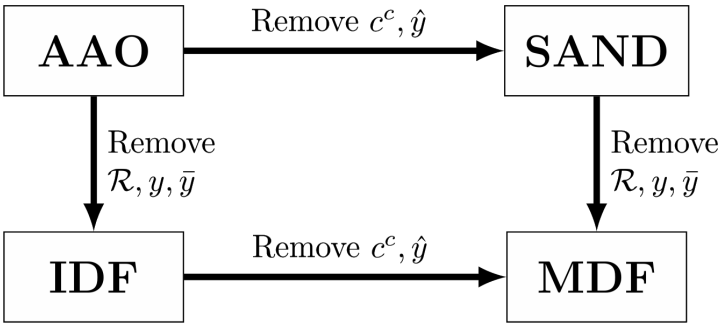



Figure 2.8: Commutative diagram of the four monolithic architectures. Edited from [60].

Distributed Architectures

Now that an overview has been given of the four different monolithic architectures the distributed architectures can be discussed. Once more this section follows the flow of Martins and Lambe's paper [60].

As was mentioned before, one of the most relevant contributions that Martins and Lambe made with their paper is the classification of distributed MDO architectures. Paragraph 2.2.3.2.2 describes this.

Before describing this classification, however, it is important to explain what the motivations are for turning away from the relative simplicity of monolithic architectures in favor of distributed architectures. This will be discussed in paragraph 2.2.3.2.1.

**2.2.3.2.1 Motivations** Martins and Lambe start by explaining that the original motivation to switch from monolithic to distributed architectures is to decrease solution time by making smart use of the structure of the problem [60]. Problems that are especially geared towards this exploitation of their structure are network flow problems and resource allocation problems, according to Martins and Lambe.

To be able to decompose an optimization problem the structure of the problem has to be understood first. Martins and Lambe explain that problems that are suitable for decomposition usually fall within one of two categories:

- Problems with complicating constraints;
- Problems with complication variables.

In the ideal situation a problem exhibits neither one of these complications. In that case the problem can directly be decomposed into $N$ separate subproblems, one for each discipline. However, this means the true nature of the problem was most likely misunderstood or mis-explained at first. Because if a problem can simply be decomposed into one optimization problem per discipline it would not be an MDO problem anymore. Equations (2.3) and (2.4) show generic optimization problems that fall into the first and second category respectively. These equations have been adopted from [60] and follow the mathematical notation introduced in table 2.1.

$$\begin{aligned} \text{min.} \quad & \sum_{i=1}^{N} f_i\left(x_i\right) \\ \text{w.r.t.} \quad & x_1, \ldots, x_N \\ \text{s.t.} \quad & c_0\left(x_1, \ldots, x_N\right) \geq 0 \\ & c_1\left(x_1\right) \geq 0, \ldots c_N\left(x_N\right) \geq 0. \end{aligned} \tag{2.3}$$

$$\begin{aligned} \text{min.} \quad & \sum_{i=1}^{N} f_i\left(x_0, x_i\right) \\ \text{w.r.t.} \quad & x_0, x_1, \ldots, x_N \\ \text{s.t.} \quad & c_1\left(x_0, x_1\right) \geq 0, \ldots c_N\left(x_0, x_N\right) \geq 0. \end{aligned} \tag{2.4}$$

By observing these equations the names complicating constraint and variables become obvious. Equation (2.3) has multidisciplinary constraints, $c_0$, and eq. (2.4) has multidisciplinary design variables. Both of these features complicate the decomposition of the problem. The first allows for a separation of the objective function into $N$ independent factors, but the presence of $c_0$, which is a function of all discipline design variables, does not allow the complete problem to be separated into $N$ separate subproblems. The second has global design variables, $x_0$, which means the objective function and constraint functions cannot simply be separated.

In engineering settings problems rarely exhibit obvious means to exploit their structure to efficiently decompose them. This is mainly due to the fact that nonlinear problems, which real world engineering problems almost exclusively are, are hard to decompose in an advantageous manner. Despite this limitation, decomposing design (optimization) problems is standard practice in the engineering industry. The main motivation for doing so lies in the organizational structure of the industry. Different teams are traditionally given some degree of autonomy on executing their part of the design, as has been discussed in section 2.1.6 for wing design. This is precisely an example of a decomposition of an overall design (optimization) problem. Hence, distributed MDO architectures conform better to the current industrial environment.

Finally, another motivation to decompose MDO problems is to balance computational costs. The example that Martins and Lambe give to explain this is apt and relevant to this thesis: aerostructural optimization. Often a fairly sophisticated aerodynamic analysis is performed, but only a linear structural solver is employed. In this case the aerodynamic analysis can take several orders of magnitude more time to run than the structural solver. If they were executed in sequence and executed the same number of times, the computational load would be poorly distributed between them. Instead, the problem could be decomposed such that the computationally cheap structural solver performs its own, internal optimization for each computationally expensive aerodynamic analysis. A system optimizer then commands the aerodynamic analysis and the internal sub-optimization loop of the structure to bring the overall design to an optimum. In this case the architecture can be tweaked to achieve a fair balance of the computational load taken by the aerodynamic and structural analyses.

**2.2.3.2.2 Classification** As was mentioned before, in section 2.2.3.1, the classification of distributed MDO architectures presented by Martins and Lambe is based on the monolithic architectures. More specifically, the classification is based on which constraints are removed from the originally AAO problem, as was depicted schematically in fig. 2.8.

Specifically the main difference between IDF and MDF is the use of the MDA block. The first level of Martin and Lambe's classification of distributed architectures is based on this principle. That is, distributed architectures that use some form of an MDA loop inside the architecture to control system consistency are labeled *distributed MDF* architectures, whereas those without it, that use coupling variables in the system formulation, are labeled *distributed IDF* architectures. Distributed IDF architectures are further divided into two categories: those using multilevel optimization, and those using penalty functions. Martins and Lambe show the connection between the different classes of architectures with arrows that related them to the IDF and MDF architectures. This is a very clear, geometric way to keep them apart. Figure 2.9 is a compact version of the diagram shown in [60]. To compactify this diagram,

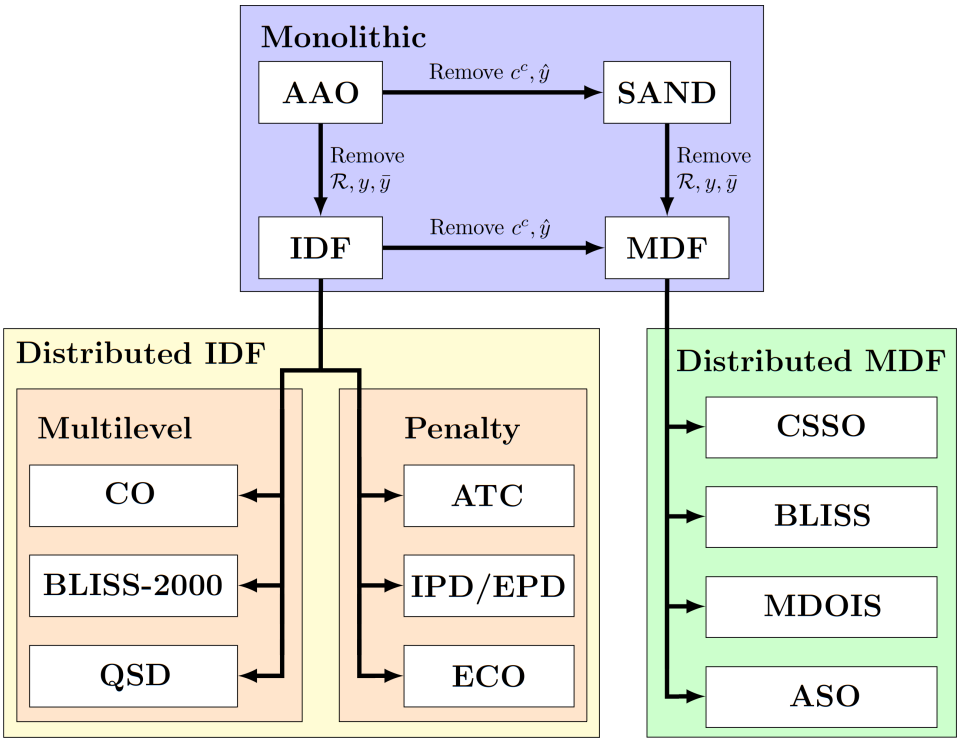



Figure 2.9: MDO architecture classification diagram. Edited from [60].

the names and descriptions of the different example architectures have been removed. Therefore, the meaning of the acronyms of the different architecture are listed in table 2.2. The interested reader is referred to [60] for the full descriptions.

## 2.2.4. Difficulties of MDO

To achieve the most optimal design and increase the iteration frequency it would be beneficial to implement a competent MDO framework from the very start of the design process. However, in practice MDO is not widely applied [3, 96, 97]. The reasons for this are technical difficulties/limitations [2] (e.g. computer power, availability of analysis tools) and non-technical, organizational aspects [11, 85, 87].

Gent et al. [3] refer to the work of Flager and Haymaker [26]. They made an account of how long different activities of the design process took for an MDO process as compared to a legacy design process, applied to the design of a hypersonic vehicle. Gent et al. show a figure from [26] displaying these metrics side by side. It is shown here in fig. 2.10. As can be seen, the specification time needed to set-up an MDO project is almost triple that of the legacy process. In return, however, both the execution time and management time have been dramatically reduced. This leaves more time to interpret the results and making design decisions. What is also striking is the iteration duration and the number of iterations that could be performed during the design process. For the legacy process iterations were

Table 2.2: Meaning of the MDO architecture acronyms

| Acronym | Meaning |
|---|---|
| CO | Collaborative Optimization |
| BLISS(-2000) | Bilevel Integrated Systems Synthesis |
| QSD | Quasiseparable Decomposition |
| ATC | Analytical Target Cascading |
| IPD | Inexact Penalty Decomposition |
| EPD | Exact Penalty Decomposition |
| ECO | Enhanced Collaborative Optimization |
| CSSO | Concurrent Subspace Optimization |
| MDOIS | MDO of Independent Subspaces |
| ASO | Asymmetric Subspace Optimization |

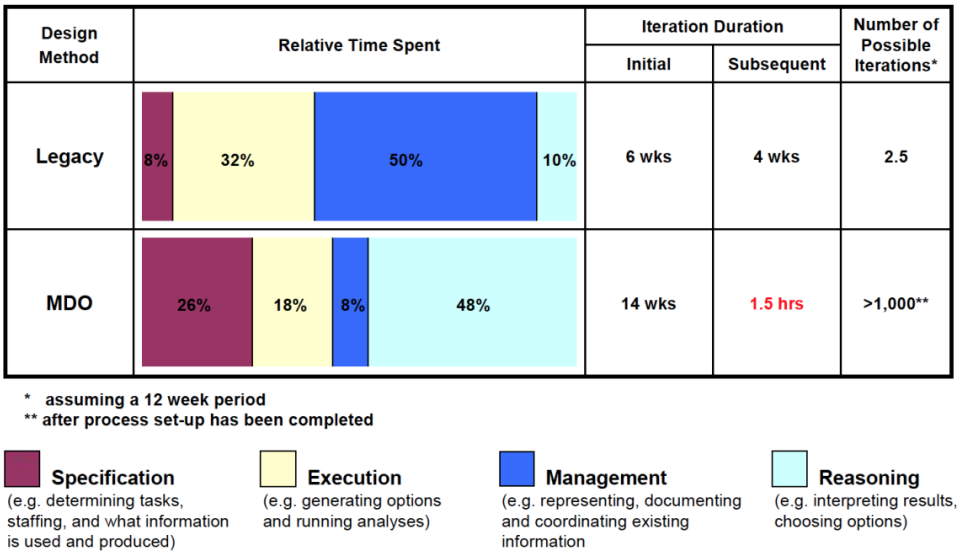



Figure 2.10: Time of different activities performed during the design process using MDO or legacy project management. Copied from [3].

slow, in the order of multiple weeks to more than a month, and so only a few iterations were possible. The MDO process, on the other hand, took more than twice as long to perform the first iteration as compared to the legacy process, but then obliterated the time it took to perform subsequent iterations, bringing it down from weeks to the order of hours. This allowed for over 400 times more iterations than the legacy process.

The reduction in management costs is also impressive; it was reduced by 84% by applying MDO. This shows what automation can do for any process in general. If managing the project becomes too expensive, automation can cut it down to the minimal effort possible. Of course an initial investment has to be made to set this automation up, which will increase costs in the short term first.

### 2.2.5. The AGILE Project

A number of workshops have recently identified and specified what is necessary for MDO to become more readily applicable in the industry [87, 105]. This led to the conception of the EU funded AGILE project (Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts) [1, 14]. The project is described by Ciampa and Nagel in [15]. The scope of the development within AGILE is three fold, as listed on the project website [1]:

- Advanced optimization techniques and strategies;
- Techniques for collaboration;
- Knowledge-enabled information technologies.

Recent efforts within and related to the project [3, 14, 16, 64, 96, 97] as well as in different projects [8, 17, 35, 91] have put a lot of thought and work into the last two points, progressing them beyond the previous state of the art. Of course the first point is a research field of its own and a lot of work is being done there. However, there is still a need for new optimization algorithms and strategies that are robust and efficient [96].

AGILE Paradigm

Of course new software is developed as part of the AGILE project to aid MDO processes. However the most important proposition put forth by the project is not a set of tools. Rather it is a philosophy of how to approach MDO processes in order to overcome the difficulties faced in the industry. This philosophy is referred to as the *AGILE Paradigm* [15]. Its purpose is not limited to any specific industry where MDO might be applied. It does not limit the usecase.

Being a paradigm, the AGILE paradigm approaches MDO processes within a predefined framework. In [15] the description of this framework starts with the definition and terminology of the general MDO based process. Ciampa and Nagel describe that three phases can always be identified in such a process: setup, operation, and solution. These three phases are shown in fig. 2.11, which has been taken from [15]. The figure shows, that over the course of the design process abstraction is exchanged

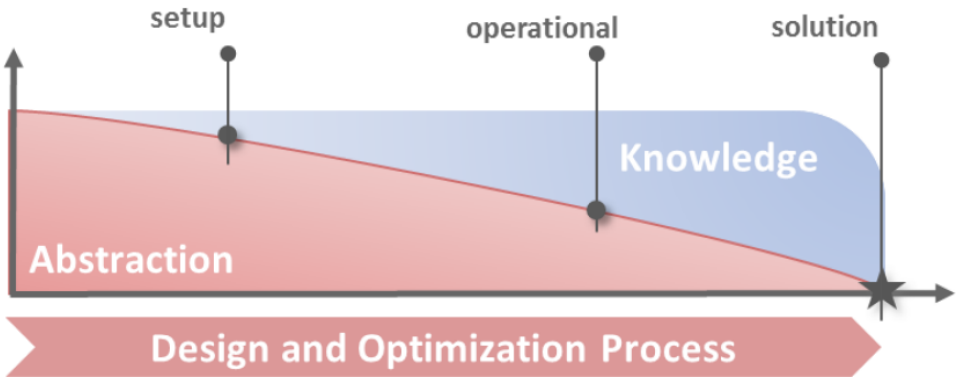



Figure 2.11: The general MDO based process and its three phases. Copied from [15].

for knowledge. A nice analogy for this is the exchange of potential for kinetic energy during freefall. An object starting at rest at a finite altitude has only potential energy (abstraction; the potential to innovate) and no kinetic energy (knowledge; the ability to attain a solution). When it is released it gains speed (knowledge) in exchange for altitude (abstraction) until it hits the ground (solution). When it does hit the ground, the kinetic energy is released again until it is at a full stop (convergence to the solution). When a solution is approached the knowledge that was gained during the process is invested until a final design is obtained. At this point all abstraction has been removed. Hence, the three phases of the design process: first a framework has to be set-up to drive the process. This limits the possibilities (abstraction) slightly, but yields an increment in the knowledge of the problem(s). Then, during the operational phase, a large amount of knowledge is gained about the design, and the design space (abstraction) is shrunk more and more. Finally, the design has been fully specified and no abstraction is left. All knowledge then accumulates into the final design.

These phases were also discussed in section 2.2.4. It was noted that the time spent in the setup phase increases when applying MDO. Ciampa and Nagel explain that the main challenges of this phase lie in the design task and MDO problem formulation, pre-selecting the design drivers, identifying and tying together all of the different analysis competencies that are required, and deploying the necessary IT infrastructure [15]. The AGILE Paradigm aims to overcome exactly these challenges and thereby reduce the setup time. In fact, one of the high level objectives of the AGILE project is to reduce the time needed to setup and solve MDO problems in a team of heterogeneous experts [15] by 40%. Next to this, the project aims to reduce the time needed to converge the optimization of an aircraft by 20%.

To achieve these goals and tackle the practical problems of MDO in the aerospace industry the AGILE paradigm attacks the three topics listed at the start of this section. Extracted from these, three main supporting pillars are defined [15]:

- Design Competences;

- Knowledge Architecture;
- Collaborative Architecture;

The entire concept of the paradigm is summarized in fig. 2.12, which was copied from [15]. In this

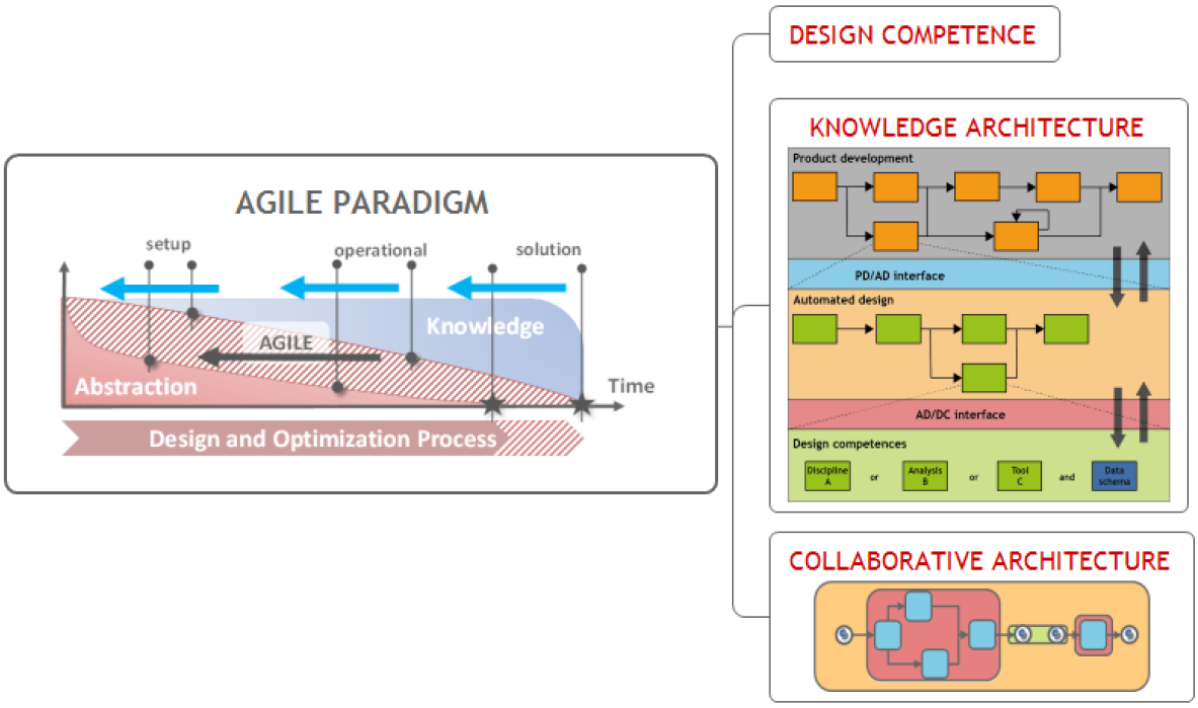



Figure 2.12: The concept of the AGILE Paradigm. Copied from [15].

figure a modified version of the general MDO based design process that was depicted in fig. 2.11 can be seen. These modifications represent what the AGILE project aims to attack in the design process. It pushes all three phases back, thereby reducing the total time required to obtain a solution. It does so by making the setup phase more efficient. The setup phase is not only streamlined in terms of time costs, but it also dramatically reduces the abstraction very early in the process. It is able to do this, because a wealth of knowledge can immediately be utilized from the pre-existing design competences. Those were previously hard to or even impossible to tap into, at least in such an early phase of the process, because there was a lack of a robust knowledge database and collaborative environment. Next, by investing in new, smart architectures and algorithms, by making it easier to store knowledge, and to communicate between different actors in the design process, the operational phase is accelerated as well. Finally, by developing and deploying advanced optimization techniques the solution time is reduced furthermore. The figure indicates which chunk of the process the AGILE paradigm aims to remove, which is quite significant.

Knowledge Architecture

The knowledge architecture begins by modeling the product development as a hierarchic structure. An interface between the product development and automated design formalizes hows this hierarchy can be translated into an executable workflow. This workflow, in turn, utilizes the design competences. A special interface allows the automated design layer to communicate with the design competences in a unified language.

The latter is important and deserves some attention. A crucial difficulty in setting up an MDO process that combines the knowledge of different disciplines is to tie the different disciplines to one another. This mainly stems from the fact that they do not speak the same language; i.e., they require input and give output in different forms.

To give a relevant example, consider the definition of an aircraft. Perhaps its design is stored in the form of some CAD file. Now if we want to analyze its induced drag with a Vortex Lattice Method (VLM) we need to discretize and represent the aircraft in an entirely different way. The geometry contained in

the CAD file somehow needs to be translated into a set of panels. If, on the other hand, a CFD analysis would have to be performed, the geometry would have to be translated to a proper grid, boundary conditions, etc.

Without a uniform interface definition the tying together of the different design competences in an automated manner becomes impractical, inefficient, and basically a mess. To address this problem, the AGILE paradigm defines a common language that forms a uniform interface between the different design competences and the automated design layer. Within the AGILE project this common language takes the form of the *Common Language for Aircraft Design*, or *CPACS* for short [18]. CPACS is an XML schema that allows for all the properties of an aircraft to be described. It has a broad base definition that captures all common parameters and utilities to fully specify the design and configuration of an aircraft. On top of geometrical parameters, CPACS also allows for performance related data to be recorded within the same unified data schema. Finally, CPACS is also flexible and expandable, because it allows for software specific data to be recorded within the same XML file as well. These properties make CPACS a powerful tool to improve communication between tools, disciplines, and engineers.

Collaborative Architecture

The collaborative architecture enables the distribution of an automated design workflow over multiple, separated collaborators. This separation can be organizational (different team or division) but also physical (different office). An important notion in the AGILE paradigm is the control of knowledge. It should be possible to have extensive collaboration between different companies while still protecting each company's intellectual property [16]. To achieve this, it has to be possible to execute parts of the worlflow at physically different locations. Furthermore, such a part, which can be a design competence or a sub-workflow, should always remain under the control and authority of the owner. Therefore, the collaborative architecture needs to offer the possibility of granting and/or revoking licenses. As described by Ciampa et al. in [16], a framework has been developed for this purpose. Figure 2.13 shows how it works schematically. This figure describes what happens in the situation that a simula-

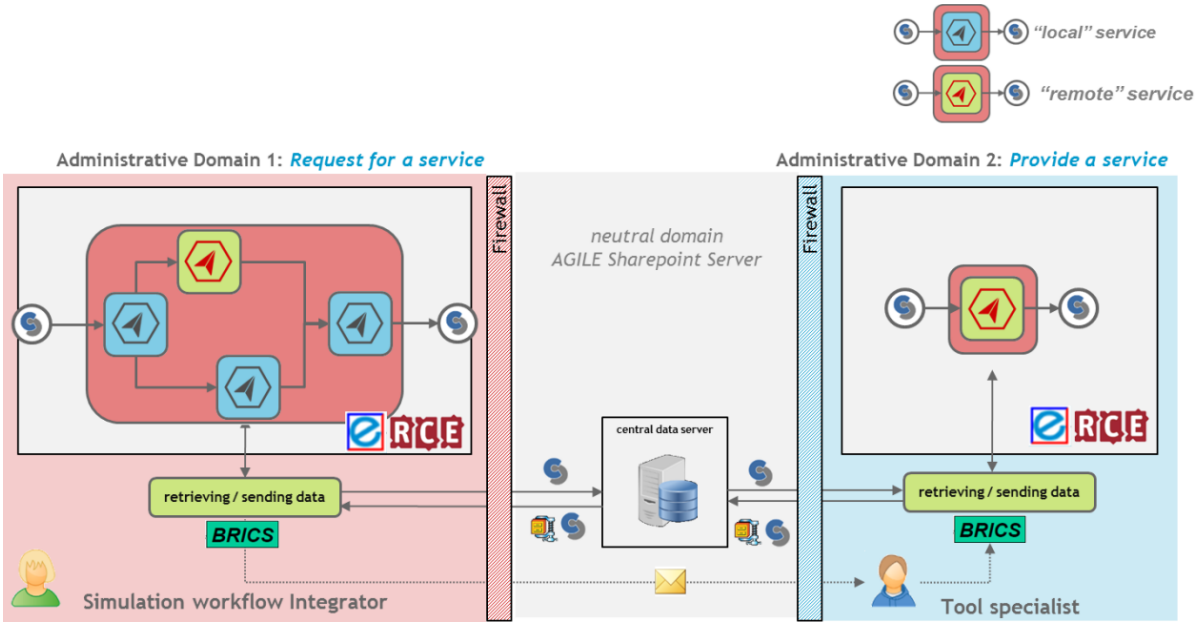



Figure 2.13: The collaborative architecture within AGILE. Copied from [15].

tion workflow integrator from one administrative domain wants to use a remote service, provided by a tool specialist operating in a different administrative domain. Both domains are shielded from the outside by a firewall. Two channels are now defined that cross the administrative domain borders: a data communication channel, and a messaging channel. The first is mediated by agreeing to store data on a central data server in a neutral domain, accessible to both parties. This server allows two way communications of data to and from it, but does not allow a direct connection between two administrative domains. The messaging channel can be mediated by, for example, an email server/service.

Regardless of its packaging, this channel should be based on a messaging protocol that, again, does not allow direct connections between the sending and receiving end. The Bricks protocol forms an interface between the data communication and messaging channels [9]. This is where the tool specialist enters the loop. It gives the tool specialist full control over if, how, when, and where to provide the requested service. Conceptually, this could mean the tool specialist gets a prompt asking him/her to approve the a service he/she provides to run at the request of a simulation workflow integrator. If he/she does, the data required to run the service is automatically downloaded from the central data server and the tool is run. Then the output is uploaded to the central data server, and a message is send to the workflow integrator through the messaging channel. This message is picked up by the Bricks interface on the simulation workflow integrator's end, which automatically downloads the output data from the central data server and injects it back into the workflow. During this process the borders between the administrative domains are never violated and the tool specialist's intellectual property is protected, but the knowledge of the tool specialist is still shared with the workflow integrator.

Figure 2.13 displays the logo's of three software products that are used extensively in the AGILE project. The blue and gray intertwined half-circles within a white circle is the logo of CPACS, which was already described. This logo is shown as in- and output, as well as along the data communication channel. The two logos in the bottom right corner of the gray areas within either administrative domains belong to the Process Integration and Design (PIDO) environments Optimus [65], a proprietary environment developed by NOESIS, and RCE [21] (Remote Component Environment), developed by the DLR. The gray areas in fact represent the inside of a PIDO environment. The red rounded rectangles represented services/workflows defined within these environments. The green or blue blocks within these, in their turn, represent an actual analysis tool (design competence).

KADMOS

To bridge the gap from the design competences, via the product development hierarchy, to the automated design workflows, and to provide tools for visualization and manipulation of MDO architectures a system named KADMOS is being developed by Delft University of Technology and RWTH Aachen University based on graph theory [3, 96, 97]. It is a framework written in Python that is aimed at the reducing the time required for the setup phase as discussed before. It has five levels or layers. It starts with a repository of analysis tools/design competences and ends with an executable workflow.

Figure 2.14 shows a top-level, conceptual overview of KADMOS, depicting these five layers. From top to bottom the five layers operate as follows [97].

1. The Repository Connectivity Graph (RCG) allows for the potential connections between all design competences to be visualized. To allow for this to work, the requirement is that all tools specify their in- and outputs in CPACS format. If done so, this layer of KADMOS is able to generate a directed graph representing the possible ways to couple one or more analysis tools to another. This allows the user to get a clear overview of the possibilities and shortcomings of the repository. It immediately becomes clear which tools could be used together in an MDO architecture and what additional knowledge is required;

2. The second layer allows for a user to specify which tools will be used within the MDO problem that is to be solved. Basically, the user makes a selection from the repository of available tools that will form a sub-repository for the MDO problem at hand. By removing unused tools from this sub-repository the RCG can be uncluttered and the unused tools and variables cleared from it. The resulting, problem specific graph is referred to as the MDO Problem Graph (MPG);

3. The next layer allows a user to label in- and outputs as belonging to the different categories present in an MDO problem; e.g., design variables, coupling variables, etc., as discussed in section 2.2.2. After doing so KADMOS is able to wrap any given MDO architecture available from a predefined database around the problem automatically. This step yields two graphs: the MDO Data Graph (MDG), which is the MPG augmented with the necessary architectural blocks (optimizers, MDA's, etc.), and the MDO Process Graph, which is in fact an N2 diagram describing the order in which the different nodes of the MDG are executed. When combining the MDG and MDO Process Graph, one obtains an XDSM describing the specific MDO problem to be solved, with the specified disciplines and function blocks, cast into the specified MDO architecture;

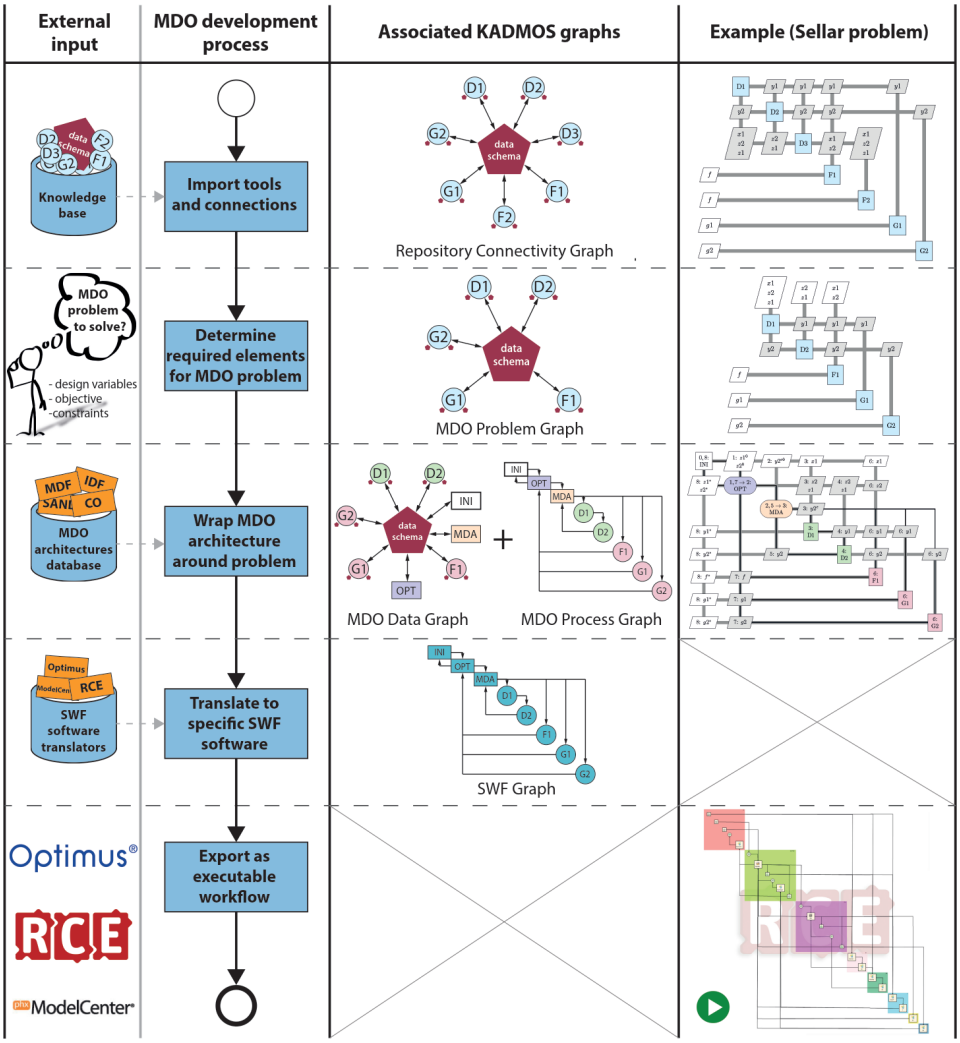



Figure 2.14: Top-level overview of the KADMOS framework. Copied from [97].

4. Layer four has the task of automatically translating the combination of the MDG and MDO process graph to an Simulation Workflow (SWF) Graph. This means the KADMOS representation of the problem and its architecture are translated to a specific PIDO environment (RCE, Optimus, etc.);

5. The fifth and final layer exports the translated SFW Graph to an executable workflow. This workflow can directly be opened in the corresponding PIDO environment that was targeted in the fourth layer.

### 2.2.6. The OpenMDAO Framework

In [97], Gent et al. describe that the challenges of integrating tools into an MDO architecture hinder the implementation of new tools and the reuse of existing tools. The reason being, according to [97], that all tools need to be refitted for every new MDO workflow. This means an MDO architecture has to be selected early and cannot easily be swapped later on, disallowing MDO engineers to explore different architectures in search for the one most suitable for the problem. Gent et al. make the side note that there is one framework for performing MDO that is an exception to this: OpenMDAO [32].

The development of the Open-source Multidisciplinary Design Analysis and Optimization (OpenMDAO) framework was first described in a 2008 paper by Moore, Naylor, and Gray [63]. Although the framework had not received its final name yet at this point, this paper contains a detailed account of its development process. First the authors discuss the context and motivation for developing an MDAO framework. The project was founded by NASA's Fundamental Aeronautics Program and aimed at developing MDAO capabilities to aid "[...] a seamless transition between single-discipline and mul-

tidisciplinary analyses by providing systematic process and an intelligent computational environment for managing multidisciplinary variable-fidelity tools that enable system analysis and optimization at primarily the conceptual and preliminary design stages for all flight regimes of conventional and unconventional vehicle classes." [63]. This description fits well into the general aim of the AGILE project as well. The developers of OpenMDAO define two areas where work will be performed: developing analysis tools, and developing an underlying framework that interconnects different analysis tools. Again, this can easily be translated into the lingo of the AGILE paradigm as the design competences and the knowledge architecture. Next the paper provides the reader with an extensive account of the requirements set and decisions made during the development of the framework. In the authors' own words, it provides an overview "[...] of the ideas that will form the backbone of the open source MDAO framework [...]" [63]. They also mention that a lot of work still needs to be performed.

In 2010 the same authors report on their progress in the development of the OpenMDAO framework [32]. This is the first time the framework is referred to by this specific name. This paper is a very interesting and useful document. It starts with a detailed description of the general Open-Source development process. To the author's knowledge, such an account has not been published in a scientific publication focusing on MDAO and computer aided engineering (CAE) in general. Besides the author's personal interest in open source development, it is relevant to this thesis, since all tools developed as part of it will be published under open-source licenses as well. Therefore a short summary of the description as given in [32] will be given here.

Figure 2.15 shows the lifecycle a ticket goes through within the open source development process of OpenMDAO.

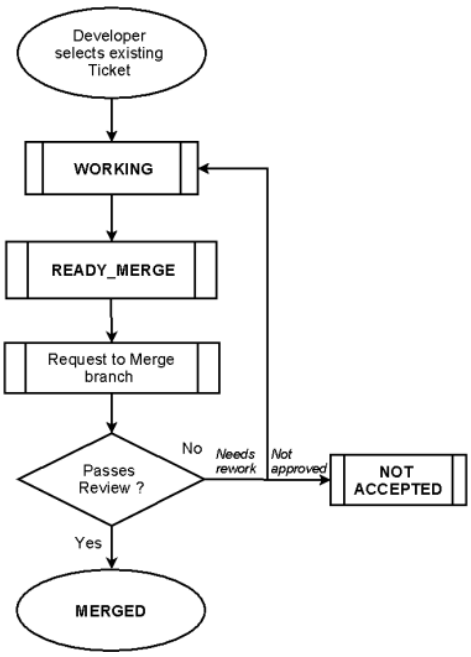



Figure 2.15: The lifecycle of an OpenMDAO ticket. Copied from [32].

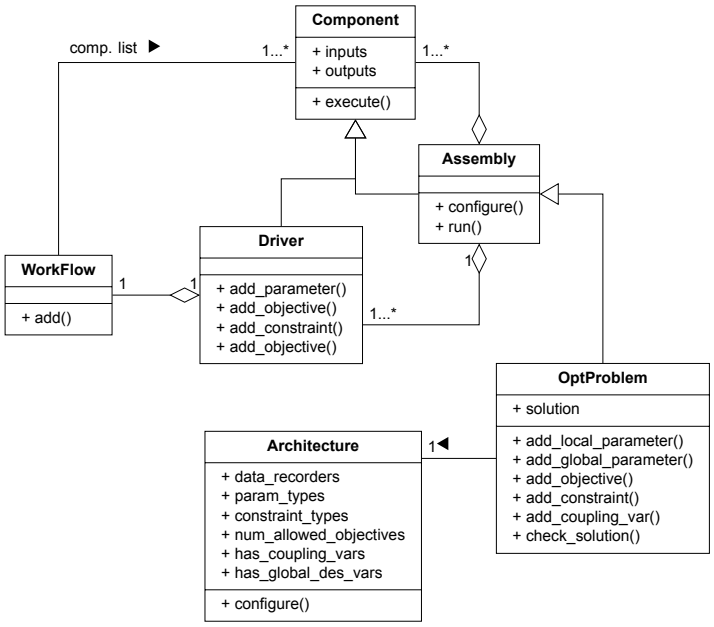



Figure 2.16: Class diagram of the core OpenMDAO classes (pre 1.0). Based on [33].

A ticket is a database entry describing some problem or desired improvement/addition to the code, a number of which are stored in a database. These tickets can be created by anyone with access to the open source development platform. Tickets can then be picked up by anyone in the development community to be worked on. Once a developer has completed the work required by the ticket he requests for his work to be merged into the main branch of the code. Before this request is accepted other developers review the proposed changes/additions to the code. If the work is accepted the work is merged with the existing code so it becomes available in the main branch. If not, the developer can redo his code, or the code can be blocked from entering the main branch. In this way anyone is able to contribute to the project, but the project is protected from people intentionally or unintentionally breaking it by the very same community that contributes.

After describing the open source development process [32] discusses the capabilities of the framework. The OpenMDAO infrastructure as it was at the time [32] was published, defines 4 entities:

- *Components*;
- *Drivers*;
- *Assemblies*; and
- *Sockets*.

*Components* are the basic building blocks of a problem in OpenMDAO. They are entities that have inputs and outputs that perform some operation on the inputs to calculate the outputs. *Drivers* can direct a chain of events within the problem. They are capable of performing optimizations, convergence iterations, and design of experiments. *Drivers* use *Components* to perform the analyses during these iterations. A system of *Drivers* and *Components* can make up an *Assembly*. As described by [32], *Assemblies* and *Drivers* are actually sub-classes of *Components* themselves, which allows nesting several of these to create more complex blocks of operations that take inputs to outputs. Finally, a *Socket* is an interface that can be held by *Components*, *Drivers* and *Assemblies* to perform parts of their internal operation. By implementing this interface, users can influence the way *Components*, *Drivers*, and *Assemblies* function. The OpenMDAO framework clearly makes extensive use of the Object Oriented Paradigm (OOP), which makes it flexible and robust. In the generalized MDO lingo, *Components* and *Drivers* correspond to discipline analyses/function blocks and optimizer/MDA blocks respectively.

In a second layer of abstraction, OpenMDAO defines one more important object: the WorkFlow class. A *WorkFlow* determines in which order a group of *Components* will be executed [32]. As such, every *Driver* contains precisely one *WorkFlow* object. Following the OOP fashion, it is possible to contain *Drivers* inside *WorkFlows*, which contain *WorkFlows* of their own, to created intricate, nested iterations.

The next noteworthy paper about OpenMDAO is a 2012 work by Gray et al. [33]. It describes using OpenMDAO as a standard platform to benchmark different MDO architectures. Being from 2012, this paper describes a more recent version of the OpenMDAO framework. Two more classes were added to the core: the *OptProblem* class and the *Architecture* class. The *OptProblem* class is a sub-class of the *Assembly* class which allows for an optimization problem to be explicitly defined in terms of the mathematical entities of table 2.1 (design variables, coupling variables, constraints, etc.). As described in [33], this class wraps all the necessary functionality to be able to optimize a given problem. Every *OptProblem* object has exactly one instance of the *Architecture* class. This class is able to take an optimization problem in the fundamental form and transform it into an equivalent problem formulation wrapped in the specific MDO architecture it defines.

In [33] the complete class diagram with these two additional classes is shown. It also includes some of the important functions of the classes. It is shown here in fig. 2.16.

After describing these two new classes the paper turns to describe the different architectures they implemented for the benchmark: IDF, MDF, CO, BLISS, and BLISS-2000. The meaning of these acronyms was already listed in table 2.2. To compare there architectures to one another the well known Sellar problem is solved [82]. Furthermore, a scalable problem is defined that has a variable number of variables, constraints, coupling variables, and disciplines. The latter has been developed by Martins et al. in 2002 [61]. The rest of [33] describes the results of the benchmark. Their overall conclusion is that the classic monolithic architectures (IDF and MDF) generally outperform the three tested distributed architectures (CO, BLISS, and BLISS-2000). Their performance metrics are proximity to the optimum at termination, and the number of iterations till convergence. Nonetheless, there are cases where the distributed architectures start to overtake the monolithic architectures' performance. The main purpose of the paper, however, was to demonstrate that OpenMDAO can be used as a common testbed for evaluating the performance of MDO architectures.

The final paper that will be addressed here in relation to the OpenMDAO framework is the 2012 work by Heath and Gray [41]. This paper describes how the OpenMDAO framework is suitable as well for more advanced architectures, involving adaptive sampling and surrogate modeling. This is relevant for this thesis, because these techniques are often applied when performing global optimizations, as is part of the aim here. The paper describes that two new concepts have been added to the framework

to provide these functionalities. First of all, the concept of *iteration hierarchies* are introduced. These ease the definition of complex processes in MDO architectures, and can be maintained during runtime, making the problem setup more flexible [41]. The second concept introduced are *MetaModels*. These are components that can mimic the behavior of an expensive analysis module by employing a meta model such as a response surface. Both of these concepts are very relevant to this thesis, as they will offer the flexibility and tools to develop new, complex architectures.

Finally, some recent changes in the OpenMDAO framework have to be addressed. The version of OpenMDAO as described by the four papers [32, 33, 41, 63] was deprecated after its final release, version 0.13.0, in April of 2015. On July 14th, 2015 the release of version 1.0 Alpha was announced on the OpenMDAO website [66]. Then, on July 20th another announcement was made describing this new release [67]. In this announcement it was described that this version departed from the previous versions. It had been moved to a new repository, and brought significant, backwards-incompatible API changes. As far as the author knows, no paper has been published describing this new release. However, it is important to discuss how it is different from the deprecated version, because it will be used in this thesis. The reference for this description is the source code of the latest version as of writing this section, taken directly from the public repository of the framework, version 1.7.3 [70].

Figure 2.17 shows a class diagram of the core classes of the new OpenMDAO framework. As can

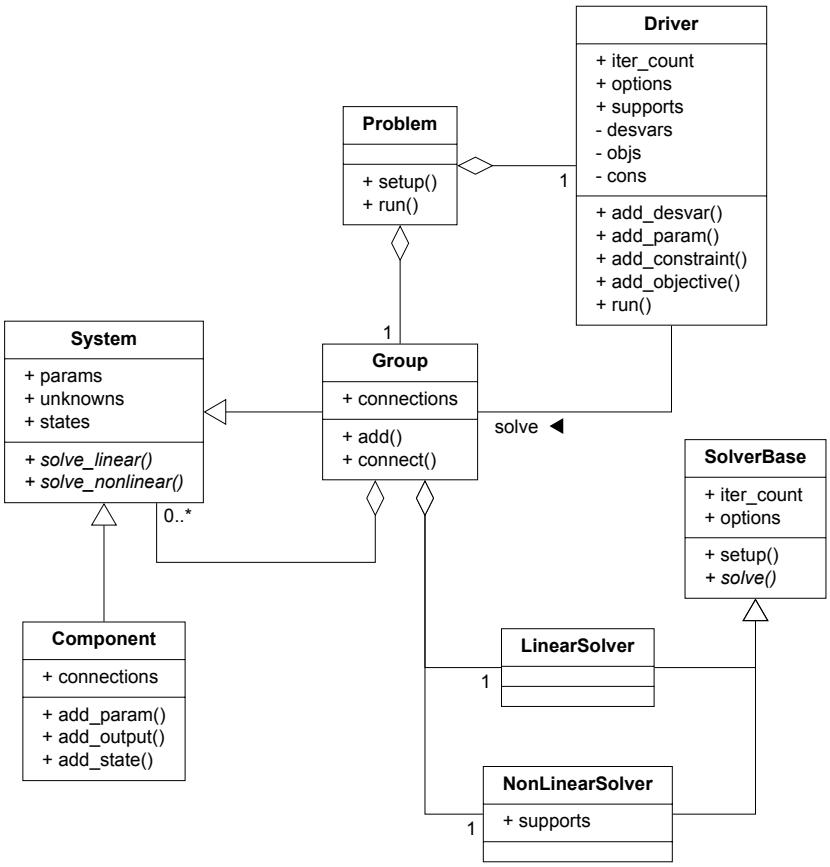



Figure 2.17: Class diagram of the core OpenMDAO classes. Derived from the source code of v.1.7.3.

be seen, it is indeed quite different from the old version presented in fig. 2.16. Note that only the most important classes have been shown, and only a selection of their variables and methods. This diagram serves the purpose of aiding the description of the main framework class hierarchy and to compare it to the old version of OpenMDAO.

Within the new framework an MDAO problem is set-up by defining a *Problem* object. This object contains one *Driver* and one *Group*. The *Driver* basically has the same functionality as the original *Driver* class in the old framework; it drives the iterative solution process of, for example, an optimization. The *Group* class takes the place of the *Assembly* class from the old framework. It can contain any

number of *Components* and *Groups*. Its main difference from *Assembly* is that it can only be run if it exists within a *Problem* and only after the *Problem*'s *setup()* method has been called. The *System* class is the common interface for *Groups* and *Components*. This level of abstraction allows *Group* and *Component* to have different interfaces and behave much differently. That is, in the old model *Assemblies* are *Components* that can hold *Components*, in the new model *Groups* are *Systems* that can hold *Systems*, but *Groups* are not *Components* and cannot have their own variables. Finally, in the old model a *Driver* can describe any iterative process, such as an optimizer or a solver. In the new model the *Solver* is completely decoupled from the *Driver*. Only a *Problem* has a *Driver*, and each *Group* has a *LinearSolver* and *NonLinearSolver*. The last two have a common interface by inheriting from *SolverBase*. The latter is not, however, used explicitly by any of the other classes.

There are a number of things about the OpenMDAO framework that have not been addressed here, such as how data flow is modeled and handled and the concept of *Recorders*, to name a few. These concepts are important to fully understand how OpenMDAO works, but are not considered relevant for this literature review. The interested reader is referred to the website of OpenMDAO for more information on these topics [32].

# 3

# Methodology

As mentioned in chapter 1, the proposed solution to the research questions is a third end to the AGILE pipeline. In this thesis a new software framework is proposed to take this place. From a high level point of view, this framework has the purpose of connecting KADMOS/CMDOWS with OpenMDAO. Since both KADMOS and OpenMDAO are written in Python, the new framework was written in Python as well. At the time of writing, the latest version of OpenMDAO that was released was version 2.0.2. This version was targeted by the new framework. The framework was named *OpenLEGO*, which is an acronym for Open-source Link Between AGILE and OpenMDAO. It also has a symbolic meaning, however, referring to the plastic building bricks, because it effectively enables a process integration engineer to plug-and-play analysis modules as if they were LEGO bricks.

This chapter will describe the concepts and implementation of the OpenLEGO framework, referring back to the concepts and problems described in the previous chapters.

## 3.1. Requirements

It is important to establish a common starting point before OpenLEGO's implementation is discussed. Therefore, a set of requirements are put in place. They are listed below.

- **CMDOWS:** OpenLEGO should operate on a CMDOWS file. A CMDOWS file is the main output of a KADMOS run, however, it should be emphasized that the CMDOWS file need not have been created by KADMOS, as CMDOWS is a standalone, neutral data schema;
- **OpenMDAO:** OpenLEGO should interface with OpenMDAO through its API. OpenLEGO should *use* this API, not change it. Initially it is intended for OpenLEGO to function as a package which depends on OpenMDAO, but is separate from it;
- **Architecture:** OpenLEGO should have a robust, well-defined architecture. It should be modular, with clear, definite tasks for each part of the larger system. This is important to increase code reusability and make it easier to maintain as well as update the code [30];
- **Usability:** OpenLEGO's API should be clear and easy to use. The user should be able to parse a CMDOWS file with a single line of code. After a CMDOWS file has been parsed into an OpenMDAO problem, the user should be able to use the result together with any standard OpenMDAO tools and procedures;
- **Representativity:** The OpenLEGO tools need to ensure the OpenMDAO problem parsed from a CMDOWS file always represents the problem defined in the CMDOWS file. The user should be able to change anything which is not specified in the CMDOWS file, however, without affecting the structure of the underlying problem.

The first two requirements follow from the description of the the problem. That is, a new end to the AGILE pipeline is required, because the current ends do not provide a satisfactory answer to the main research question of this thesis. As was discussed before, OpenMDAO is proposed as a third

end to the pipeline, because it is Open-source and because of its strength when it comes to handling analytical gradients. Since the before-last link in the chain that is the AGILE pipeline is KADOS' output, CMDOWS, it logically follows that the proposed solution should interface with this file, as the current two ends, RCE and Optimus, do.

As was mentioned in chapter 1, the choice for OpenMDAO as the optimization framework to end the new pipeline in itself is motivated by first and third requirements and sub-questions listed in chapter 1. OpenMDAO, like RCE, KADMOWS, and CMDOWS, naturally lends itself to extensibility, because it is open-source. Furthermore, OpenMDAO's codebase is clearly object-oriented, as was discussed in section 2.2.6. Its features are encapsulated within a clear set of Python classes. Therefore OpenMDAO holds promise to answer the question of how an MDAO system can be developed to be extensible; the first sub-question of this thesis.

As was mentioned in chapter 1 and elaborated on in section 2.2.6, OpenMDAO is especially powerful in terms of system level gradient evaluation. It allows the user to specify analytical gradients for any or all components of a model. The framework then automatically performs finite differencing where no gradients are provided and carries out all the chain rules to obtain the system level gradients. Hence, the the question of how gradient information can be incorporated into a system is answered adequately by OpenMDAO.

The second and third requirement listed above come from the first requirement listed in chapter 1, namely, the requirement of extensibility. Having a clear, well defined software architecture is key to creating extensible software [cite, design patterns]. Furthermore, having a clear, easy to use API is attractive to users, because it makes it easier to understand how the framework can be used. It also lowers the threshold for new developers to dive into the code, because a clear API should allow them to quickly discover how the framework fits together and functions.

The fifth and final requirement may seem trivial, but is nonetheless important. In order to attract users with little to no programming background, the framework needs to be as fool-proof as possible. Users need to be able to prod and probe a parsed problem, without accidentally breaking it. Anything not specified in a CMDOWS file needs to be changeable, but these changes should never change the underlying problem description unexpectedly.

## 3.2. Workflow

The tasks that OpenLEGO will have to perform in order to take a CMDOWS file and yield a runnable OpenMDAO problem were identified and ordered. The resulting worlfow diagram for OpenLEGO is shown in fig. 3.1. First the CMDOWS XML file needs to be read and parsed. As can be seen, four

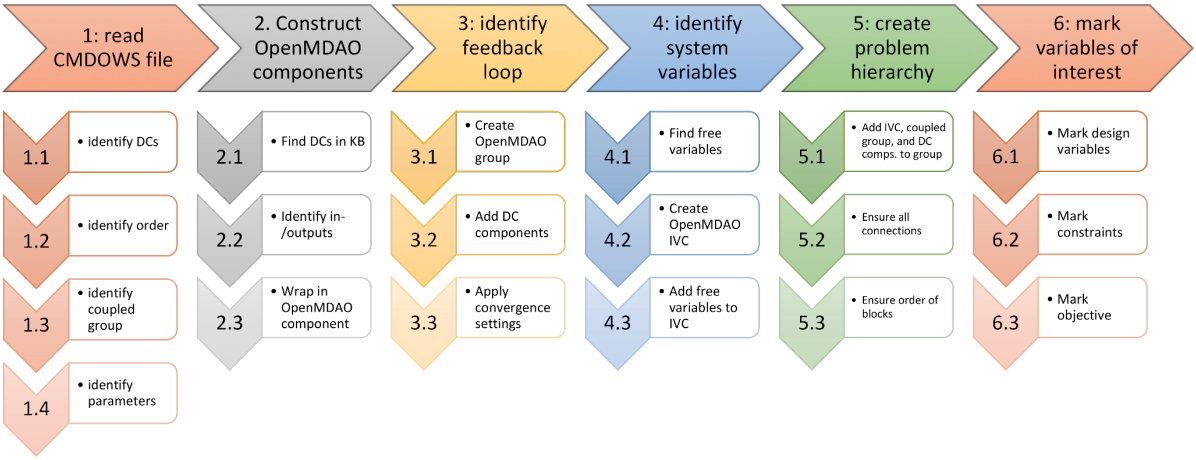



Figure 3.1: Workflow diagram for OpenLEGO

different sub-tasks are defined for this part. First, all Discipline Components defined in the CMDOWS file need to be identified by name. Then their order has to be read and stored. If a coupled group is defined in the CMDOWS file, its members need to be identified and ordered properly. Finally, the parameters which are part of the problem need to be identified.

Next, the information from the CMDOWS file along with a given Knowledge Base need to be used

to construct an OpenMDAO `Component` for each DC in the KB. To achieve this, the DCs specified in the CMDOWS file need to be found in the given KB first. Then, their in- and outputs need to be identified. Once this information is known, the functions of each DC should be wrapped in an OpenMDAO `Component` subclass.

Then, a feedback loop has to be handled, if it exists. To do this, an instance of OpenMDAO's `Group` class needs to be created. Then the OpenMDAO `Component` representations of the DCs which are part of this coupled group need to be added to the OpenMDAO `Group`. They need to be ordered and coupled to one another correctly. Finally, the convergence settings of the `Group` should be set such that the group is representative of the information specified in the CMDOWS file.

The next step in the process is the identification of free system variables. In OpenMDAO, these need to be handled separately and specified as part of an `IndepVarComp` (IVC) class. From the information stored in the CMDOWS file, it should be determined which variables should be marked as free variables. An instance of the IVC should then be created, and the free variables should be declared on it.

Following these first four steps, the problem is assembled in step 5. The IVC, coupled group, pre-coupling DCs, and post-coupling DCs are added to an instance of OpenMDAO's `Group` class. The in- and outputs of all subclasses of the `Component` class added in this way need to be properly connected. Furthermore, they should be ordered in the same way as is specified in the CMDOWS file.

Finally, in step 6, the variables of interest should be marked. Thus, the design variables, along with their lower and upper bounds, should be marked as specified in the CMDOWS file. Then, the constraints should be marked, along with the correct reference values and type (inequality, equality). Finally, the objective of the problem should be marked.

## 3.3. Knowledge Base

As was mentioned in chapter 1, the notion of the Knowledge Base (KB) as it was put forward within AGILE had to be extended, adding provisions for analytical gradients. Within AGILE a KB is populated with a set of analysis tools, or Discipline Components (DCs). A DC consists of four parts:

1. Input template XML file;
2. Output template XML file;
3. Meta information JSON file;
4. Analysis tool executable.

In- and output of a DC is specified by two XML files. Each valued XML element in these files corresponds to an input, resp. output. A similar approach is taken to define the gradients of a DC. That is, a third template XML file is defined, specifying the partial derivatives from which output with respect to which input variables are supplied by the DC.

Such an XML file needs to link specific outputs and inputs to one another. In order to formalize how this is done, an XML schema was specified. Since partial derivatives are defined in the form ”derivative of y w.r.t. x”, where y is a dependent variable (output) and x a free variable (input), the XML schema lists outputs first, and inputs as their dependencies. In other words, a 'partials' XML file contains a list of output variables, each of which lists the input s it depends on.

Within the AGILE pipeline, in- and output data is stored in XML files with the same structure as the template in- and output XML files that define the in- and output variables of DCs. Once more, the same approach is followed for the partials XML files. As such, a template partials XML file *qualifies* which outputs depend on which inputs by listing them with dummy values. In turn, a data carrying partials XML file, with the exact same structure as the template file, *quantifies* the partial derivatives of the same set of outputs w.r.t. the same set of inputs by storing their magnitudes as values in the XML tree.

In the partials XML schema, inputs and outputs are represented with the same complex type: `parameterType`. A `parameterType` must contain a single text valued child element with the tag name `uid` declaring the Unique Identifier (UID) of the parameter. The type can optionally contain a value, represented by a child element with `value` as tag name.

A valid partials XML file has a single root element with tag name `partials`. The root element contains any number of `dependentParameterType` elements with tag name `parameter`. A `dependentParameterType` is an extension to the `parameterType`, adding an obligatory child element of

complex type with tag name `partials`. The `partials` element, in turn can contain any number of elements with tag name `partial` of type `parameterType`. This setup allows for a list of dependent parameters, which each list the parameters they depend.

The generic structure of a partials XML file is shown in code frag. 3.1.

```
<partials>
    <parameter>
        <uid>output1_uid</uid>
        <value>output1_value</value>
        <partials>
            <partial>
                <uid>input1_uid</uid>
                <value>output1_input1_value</value>
            </partial>
            .
            .
            .
            <partial>
                <uid>inputN_uid</uid>
                <value>output1_inputN_value</value>
            </partial>
        </partials>
    </parameter>
    .
    .
    .
    <parameter>
        <uid>outputM_uid</uid>
        <value>outputM_value</value>
        <partials>
            <partial>
                <uid>input1_uid</uid>
                <value>outputM_input1_value</value>
            </partial>
            .
            .
            .
            <partial>
                <uid>inputN_uid</uid>
                <value>outputM_inputN_value</value>
            </partial>
        </partials>
    </parameter>
</partials>
```

Code frament 3.1: Generic structure of a partials XML file

The partial XML schema has been formalized using the XSD format. The contents of this XSD file are listed in appendix A.2.1.

A Python class, `Partials`, has been created to simplify handling partials XML files. Instances of this class can be written to an XML file or instantiated from an XML file The class exposes a function to declare partial derivatives of outputs w.r.t. inputs, including their values. The data of the class can also be turned into a text representation of the XML file, and a Python dictionary representation of the XML file. The code of the `Partials` class can be found in appendix A.2.2.

## 3.4. High Level Strategy

Despite the requirements and constraints imposed on the link, and regardless of the workflow, there is still a lot of design freedom. To reduce the design space further a high level strategy for OpenLEGO was devised. The overall focus during the design and development of the link was to maximize re-usability an modularity. All new components and features developed for this link need to be truly functional tools, which also have meaning outside of just this bridge between CMDOWS and OpenMDAO. This approach increases the impact and overall relevance of the link beyond a single purpose.

### 3.4.1. Coupling Strategy

The first decision that was made was whether the link should operate closer to the KADMOS end or the OpenMDAO end of the pipeline. This statement requires clarification: one can imagine a setup where a CMDOWS file is parsed and the outcome is some sort of executable, fully independent and self sufficient, containing all logic of an OpenMDAO problem's creation and execution. For example, the parser could output a Python script which sets up an OpenMDAO Problem and executes it. In this case the link can be said to be closer to KADMOS' side of the gap than it is to OpenMDAO, because the parser does not use OpenMDAO's API when parsing the CMDOWS file. Only once the created Python script is run is the API evoked. In other words, the link is further removed from OpenMDAO in this case, because the CMDOWS parser by itself does not have a direct dependency on the actual OpenMDAO Python package during execution, but it does directly depend on the CMDOWS file. Of course in reality it does depend on the OpenMDAO package, because knowledge of its API is still necessary to write the output Python script. However, no special code or classes are written in this case extending or enriching OpenMDAO's functionality. This coupling strategy is shown schematically in fig. 3.2.

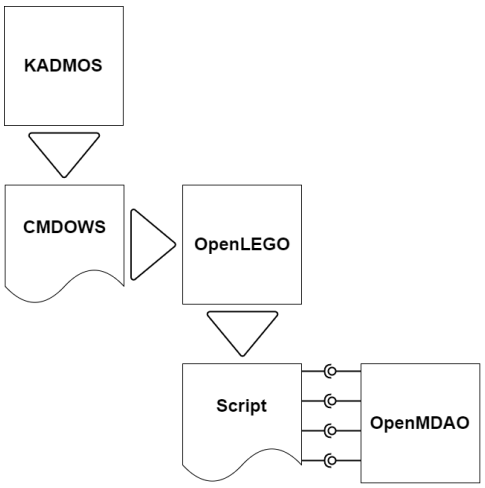



Figure 3.2: Coupling strategy which keeps the link detached from OpenMDAO.

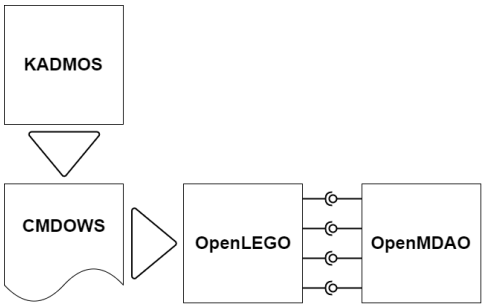



Figure 3.3: Coupling strategy which ties the link directly into OpenMDAO.

If, on the other hand, the link would reside closer to OpenMDAO the result would be quite different. In this case the result of the parser would not be some form of an executable file. Instead special classes could be written inheriting functionality from OpenMDAO, constructing themselves based on a given CMDOWS file. For example, a subclass of OpenMDAO's *Problem* class could be written which is a direct representation of a CMDOWS file it is provided with. Such a setup would be intrinsically connected to and intertwined with OpenMDAO's API endpoints through inheritance and aggregation. Instead of having a notion of executing a tool which simply transforms a CMDOWS file into an executable file, this approach has the notion of using new classes to setup and manipulate an OpenMDAO problem using the CMDOWS file as direct input. In other words, the first approach in essence hard-codes the setup of an OpenMDAO problem in a script, whereas the second approach constructs the problem dynamically with a CMDOWS file as starting point. This second approach is shown schematically in fig. 3.3.

Another way to describe the difference between these two approaches, is that the first yields a kind of converter or translator between CMDOWS and OpenMDAO, whereas the second provides a front-end for OpenMDAO able to take CMDOWS directly as input. Both of these approaches have

advantages and disadvantages. A clear disadvantage of the first approach is that it yields a hard-coded script file. The user can only adjust the OpenMDAO problem by manually editing this file. However, this is also an advantage, because in this way the problem can be stored and shared independently of KADMOS and CMDOWS and can be run directly at any time. A disadvantage of the second approach is that it is directly dependent on OpenMDAO. Furthermore, the OpenMDAO problem is not stored directly, and needs to be regenerated from the CMDOWS file every time a new session is started. Hence, the CMDOWS file remains the primary definition of the problem. However, this also has major advantages over the first approach. For example, after the an OpenMDAO problem is constructed from a CMDOWS file, the user has the opportunity to manipulate it directly using OpenMDAO's API, without the need to edit a hard-coded file. The CMDOWS file which is written by KADMOS already encodes the full problem and solution strategy. The script generated by the link in the first approach therefore contains no new information, making it a redundant step. For this reason the second approach is clearly the cleaner, more direct approach of the two, if only because it needs less intermediate steps to get from KADMOS to a functioning OpenMDAO run. Therefore, the second approach was selected as the coupling strategy for the link.

### 3.4.2. Construction Strategy

Viewed from a high level, the function of the link is to construct an OpenMDAO problem from a CMDOWS file. Within OOP, there are two major strategies to construct objects: encapsulating the construction in a class' constructor, using inheritance, or encapsulating the construction in a factory pattern. In essence, the difference between these two approaches is comparable to the difference between the two coupling strategies that were just discussed. A factory pattern is most akin to the first coupling strategy, whereas the constructor method is most akin to the second coupling strategy. However, in this context the advantages and disadvantages of these two approaches are different.

In the factory pattern approach the construction is detached from the classes being constructed. In the context of OpenMDAO, the construction of an optimization problem involves populating a tree hierarchy of *Group* and *Component* classes. The first of these is itself a collection of *Component*s and/or other *Groups*. More precisely, both the *Group* and *Component* classes are subclasses of the *System* class. This is a clear example of a composition pattern (also known as a tree and leaf pattern) [30]. This setup is very suitable for the purpose of OpenMDAO, because it is a direct model of a multidisciplinary design optimization problem. That is, each *Component* represents a discipline, and multiple disciplines can be connected to one another to form multi-discipline analyses, which themselves can in turn be connected to other multi-discipline analyses and/or other disciplines, forming complex analysis systems. If the factory pattern approach would be used to construct an OpenMDAO problem given a CMDOWS file, it would be straightforward to create a *Component* for every design competence listed in the CMDOWS file, and tie them together as specified. However, it would be a very code intensive, linear, functionally oriented type of programming. For example, every *Component* would have to be supplied with code individually to wrap the analysis tool it represents. The advantage of this approach is that no special, extra classes are needed besides the factory. That is, vanilla OpenMDAO classes would be used to build the entire problem with. In that sense this is a very clean approach, be it not a very elegant one due to the code-heavy construction.

The other approach, on the other hand, offers much more opportunities to exploit the object-oriented nature that Python has to offer. If done in a clever way, inheritance allows for the same end result with much less code. Not only does this lead to more intuitive code bases, it also makes the result much more robust because code duplication and boilerplate code can be avoided. One of the strengths of this approach which makes it so powerful, is the concept of *delegation* [30]. By delegating certain tasks to others, the responsibilities of a single class can be limited. This division of responsibility is protected by encapsulation. This makes it easier to understand what a single class does and does not do. It is also more intuitive in the sense that it models real world multi-part systems. A famous example of delegation in software development is the model-view-controller (MVC) pattern [30] which is applied when implementing graphical user interfaces (GUI). In this patters the model is responsible for handling and storing data, the view is responsible for using this data to display something to the user, and the controller is responsible for responding to user input by manipulating the model. In the interest of minimizing code duplication while maximizing the robustness of the link, this strategy favoring inheritance over the factory pattern was chosen.

### 3.4.3. Architectural Strategy

Having defined how the link will tie into OpenMDAO and how it will construct the problem from a given CMDOWS file, the architecture of the link can be described. Having decided to tightly couple the link with OpenMDAO, making extensive use of inheritance and delegation, the centerpiece of the link will be a subclass of the OpenMDAO *Problem* class. The *Problem* class lies at the heart of OpenMDAO. It therefore makes sense to stick to this approach, and create a specialized OpenMDAO *Problem* class representing a given CMDOWS file: a *CMDOWSProblem* class. This class will be responsible for constructing and assembling the OpenMDAO problem when given a CMDOWS file. Besides the CMDOWS file, the class also needs to be told where it can find the knowledge base in which all the disciplines used by the problem reside, since this information is not included in the CMDOWS file.

With this in mind, the next big question to address is how to handle the disciplines. That is: how should each discipline be turned into an OpenMDAO *Component*, such that it can be connected and included in the problem? Within AGILE, specifically within KADMOS, a discipline (also referred to as a design competence), should be represented by four files: an input XML file, an output XML file, a JSON file containing meta-data about the tool, and an executable [96]. The first three are straightforward and well defined. The in- and output XML files should contain only those elements which are read, resp. written, by a discipline during execution. The JSON file contains information like the name of the tool and its author, a version number, description, etc. However, it is not prescribed how the executables should be defined. That is, there is no uniform, standard interface prescribed for them. This is critical, because without a uniform interface there is no way a software package can interact with a discipline without prior knowledge of how it works. Therefore it was decided to define such a standard interface for the purpose of creating this link: the *AbstractDiscipline* class. However, to make this interface usable outside of the link as well, the requirement was put on it that it should not be dependent on OpenMDAO. Hence, the interface needs to be neutral.

Using this strategy, a first picture can be drawn of what the overall architecture will look like. From a bird's eye view there are two parts to the link. On the one hand the link provides a standard, neutral interface for all disciplines, independent of OpenMDAO, but following the definition of a design competence as seen within KADMOS. As such, every discipline will be represented by a class implementing this interface. On the other hand, the link provides a specialized OpenMDAO *Problem* class which is able to interact with all disciplines in the same way through their shared interface, and is able to connected them to one another as such. The last piece of the puzzle to be put into place is what happens in between these two ends, transforming the neutral disciplines into representative OpenMDAO *Component*s. In the interest of modularity and re-usability, this step is split up into two parts. Firstly a specialized OpenMDAO *Component* will be made which is able to use XML for the definition and storage of its variables: an *XMLComponent*. This component, in contrast to the standard interface for the disciplines, should be neutral within the context of OpenMDAO and independent of CMDOWS and KADMOS. As such, this component can also be used directly by the user within OpenMDAO when the need arises to incorporate an XML component in a problem. Clearly this indeed achieves the goal of increasing the links relevance outside of tying CMDOWS to OpenMDAO.

The last part is actually fairly straightforward. A class will be made that brings together the added functionalities of both the standard discipline interface and the *XMLComponent*: the *DisciplineComponent*. It does so through a combination of inheritance and composition. This class is an inherit part of the link between CMDOWS and OpenMDAO. Therefore, following the link's overall coupling strategy as described before, it inherits from the *XMLComponent* class, and is given an instance of a discipline class, which in turn implements the standard interface. The class uses the XML files of the specific discipline it is given to define its own in- and outputs, and uses the functions exposed by the discipline's standard interface to execute it. As such any discipline, given it has been implemented using the standard interface provided by the link, can be effortlessly represented by an OpenMDAO component by giving an instance of it to an instance of this coupling class. This approach has the advantage that all disciplines look and behave exactly the same way as seen from OpenMDAO's perspective.

This high level strategy of OpenLEGO is shown schematically in fig. 3.4. As can be seen, the domains of OpenMDAO and KADMOS/CMDOWS overlap in the vertical middle of the diagram due to the bridge connecting the XML OpenMDAO executable to the standard DC interface and the parsing core connecting a CMDOWS file to the OpenMDAO problem API. In contrast, the standard DC interface and XML OpenMDAO component parts depend only on KADMOS/CMDOWS and OpenMDAO respectively. The CMDOWS file and KADMOS DC representation are part of the KADMOS/CMDOWS

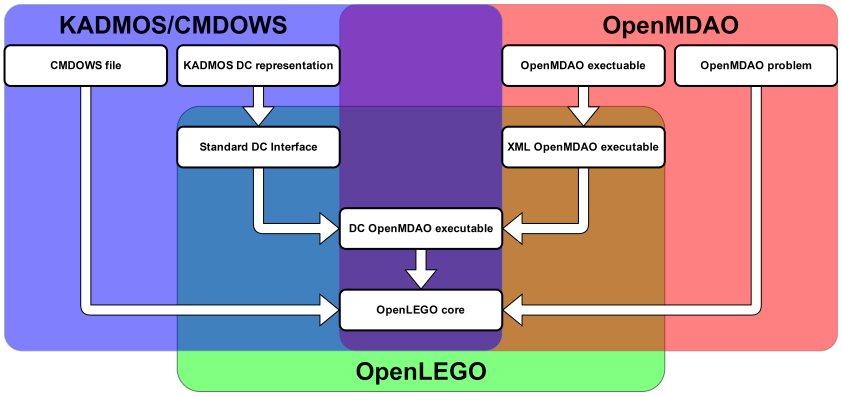



Figure 3.4: OpenLEGO strategy schematic

domain only; the OpenMDAO executable and problem are part of the API of OpenMDAO only. They are unaffected by OpenLEGO. Finally, OpenLEGO's domain is shown to contain the standard DC interface, XML OpenMDAO executable, DC OpenMDAO executable, and OpenLEGO core parts.

## 3.5. Software Architecture

Given the high level strategy described in the previous section, and given the fact that OpenMDAO is an Object-Oriented Programming (OOP) framework, it is immediately apparent that the XML OpenMDAO executable and OpenLEGO core can inherit from classes exposed by OpenMDAO's API to patch into OpenMDAO. A simplified class diagram of the core of OpenMDAO 2.0 is shown in fig. 3.5. The

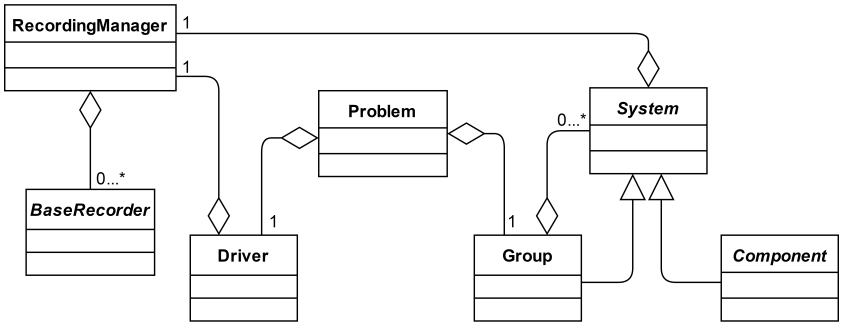



Figure 3.5: Simplified class diagram of the most important classes of OpenMDAO 2.0

`Problem` class is at the heart of OpenMDAO. It holds an instance of the `Driver` and `Group` classes. The latter is part of a structural software design pattern known as the *composite* or *tree-and-leaf* pattern, together with the `System` and `Component` classes [30]. This is a powerful design pattern, which simplifies the definition and construction of complex hierarchies of objects. In the case of OpenMDAO (in the context of MDAO architectures) this pattern allows for the definition of hierarchies of individual analyses (represented by the `Component` class) and Multi-Disciplinary Analyses (MDA) (represented by the `Group` class). Note that groups are also referred to as models in the following.

In OpenMDAO 2.0, variables of interest (VOI), such as design variables, constraints, and objectives, are defined on the `Group` class. Therefore, everything except for the solution strategy of the problem, such as a specific optimization algorithm, is fully contained within the `Group` class. This means no extra functionality is needed by OpenLEGO from the `Problem` and `Driver` classes. Therefore, the OpenLEGO core part of the high level strategy was chosen to be a subclass of OpenMDAO's `Group` class. This subclass was called `LEGOModel`.

Without a `Driver` class, the model of a `Problem` can only be used as a simple input-output executable block. That is, it is only possible to simply run all the analyses and MDA's once to get outputs for a given set of inputs. In order to perform more complex runs, such as an optimization, a specific subclass of `Driver` needs to be attached to the `Problem`. For example, the `ScipyOptimizer`

class, included in the OpenMDAO package, which inherits from the base `Driver` class and wraps the optimization algorithms exposed by the SciPy package [47].

The selection of an optimization algorithm or other solution strategy, however, is not prescribed in a CMDOWS file. Therefore this is left completely up to the user and no subclasses of the `Driver` class are needed directly by OpenLEGO. The same is true for the `BaseRecorder` class, instances and subclasses of which can be attached to `Driver`s and/or `Group`s to perform tasks like saving data to a file or displaying a plot during a run.

The OpenMDAO executable part of the high level strategy corresponds to the `Component` class. Therefore the XML OpenMDAO executable part is represented by a subclass of the `Component` class: the `XMLComponent` class. On the other end, the standard DC interface part will be represented by an Abstract Base Class (ABC) named `AbstractDiscipline` in OpenLEGO.

There is an ambiguity for the definition of the DC OpenMDAO executable part. It could be a subclass of either `XMLComponent`, `AbstractDiscipline`, or even both. However, multiple inheritance was chosen to be avoided. The main purpose of the DC OpenMDAO executable part is to automatically construct an `XMLComponent` using the in- and output files from a given DC, using the standard DC interface to obtain these. More so, the `LEGOModel` class uses the DC OpenMDAO executable part to define its OpenMDAO `Component`s. Therefore, it was decided that the DC OpenMDAO executable part should behave just like an `XMLComponent`, but that is should be constructed and executed using a subclass of `AbstractDiscipline`. This subclass of `XMLComponent` was called `DisciplineComponent`. It is connected to the `AbstractDiscipline` through aggregation. An instance of `DisciplineComponent` holds precisely one subclass of `AbstractDiscipline`.

Figure 3.6 shows a simplified class diagram of this architecture and how it connects to OpenMDAO. As can be seen, this architecture corresponds well with the the schematic of the high level strategy. In

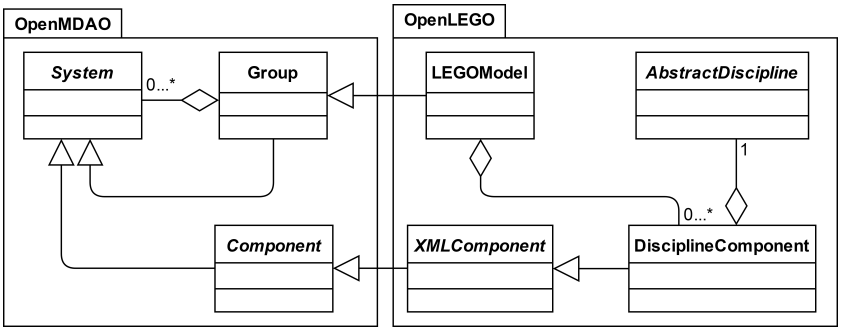



Figure 3.6: Simplified class diagram of OpenLEGO's architecture and its connections with OpenMDAO

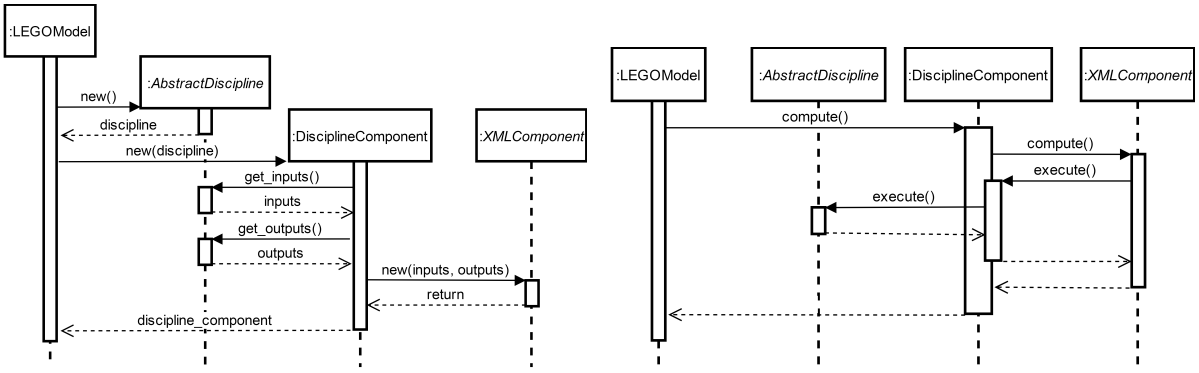
the remainder of this section, the implementation of OpenLEGO's four core classes, `AbstractDiscipline`, `XMLComponent`, `DisciplineComponent`, and `LEGOModel`, will be discussed in more detail.

To clarify the collaboration between these four core classes, sequence diagrams depicting the creation and computation of a `DisciplineComponent` instance by the `LEGOModel` class are shown in fig. 3.7a and fig. 3.7b respectively.

## 3.6. Implementation

After having discussed the software architecture of OpenLEGO in the previous section, the implementation of OpenLEGO will be discussed now. The code of OpenLEGO has been divided into 3 Python modules: `openlego.core`, `openlego.partials`, `openlego.recorders`, and `openlego.util`. The four code classes of OpenLEGO, which were introduced in the previous section, are stored in the first module. These classes rely heavily on the last two utility modules. The structure of this section follows the file structure of the OpenLEGO code base. That is, a sub-section has been dedicated to each of the six Python modules. Each of these sub-sections will give a general overview of the contents and purpose of its corresponding module first. Then a paragraph is dedicated to each important class and function contained in it. All source code of the project is attached to this report in appendix A.

This section will describe the `openlego.core` module first, in section 3.6.1. Then the XML utility functions of the `openlego.utils.xml_utils` module will be described, in section 3.6.2. Finally, the recorder classes will be described in section 3.6.3. The `openlego.partials` module will not be




(a) Construction sequence (b) Computation sequence

Figure 3.7: Sequence diagrams of the creation and computation of `DisciplineComponent`s

discussed here. The `Partials` class was already introduced in section 3.3.

### 3.6.1. Core Module

The `openlego.core` module contains the core of the OpenLEGO API. That is, the four main classes are contained within this module. Each class has its own sub-module. The source code of all sub-modules of the `openlego.core` module can be found in appendix A.1.

`openlego.core.abstract_discipline`:
Including the extension of the notion of the Knowledge Base described in the previous section, a Design Competence (DC) is formally represented by a set of 5 separate entities:

1. Input template XML file;
2. Output template XML file;
3. Partials template XML file;
4. Meta information JSON file;
5. Analysis tool executable.

The DC is further formalized by defining a standard interface, wrapping the properties and content of these 5 files. This interface comes in the form of an abstract Python class: `AbstractDiscipline`.

The `openlego.abstract_discipline` module contains the definition of this class. As was described before, the `AbstractDiscipline` class is independent from the OpenMDAO API. Therefore it is defined in a dedicated module. In this way this module can be imported even if OpenMDAO has not been installed.

The `AbstractDiscipline` class forms the Abstract Base Class (ABC) for all DCs within OpenLEGO. It standardizes the definition of a DC as prescribed by KADMOS. In this definition a DC is comprised of four parts: an input XML file, an output XML file, a JavaScript Object Notation (JSON) file with meta information about the DC, and an executable. KADMOS actually only handles the first three of these four parts directly; that is, it only deals with the in- and outputs, as well as meta-information (such as the name and execution time of a tool), but does not actually need the executable. Therefore, no specifications were given as to how this executable should be represented. However, OpenLEGO will need to be able to run the executables of all DCs. If there is no standard representation for them, then OpenLEGO cannot know how to execute a DC. This is the main reason why a standard interface in the form of the `AbstractDiscipline` class was created.

The class defines a set of properties (using Python's `@property` decorator) corresponding to the most common entries of the information JSON files of DCs, such as the name, creator, version, and description of the DC. These properties have standard implementations defined for them, but can be overridden by a specific subclass to better represent the DC it wraps.

Besides the information JSON file properties, the class also defines properties for the locations of the in-/output XML and information JSON files, as well as the path at which the DC is stored. These properties have standard implementations pointing to the directory in which the Python module of the implementing is stored, with standard names based on the name property of the subclass for the in-/output XML and information JSON files. Furthermore, three abstract methods are declared by `AbstractDiscipline`: `generate_input_xml(self) -> str`, `generate_output_xml(self) -> str`, and `execute(in_file, out_file)`. These methods must be implemented by all subclasses. The first two should return a string with the contents of the in- and output XML files of the DC respectively. The last is a static method, which should run the DC the subclass wraps using given in- and output files. The `execute(in_file, out_file)` method is static to reflect the fact that it represents an isolated tool. A fourth method, `generate_info_json(self) -> str`, returns a string representing the DCs information JSON file. This method is not abstract, however, because it uses the properties of the class as well as default values to generate a standard information JSON file for the DC. It should be overridden by a subclass if more specific, custom information should be included in the file. Finally, the `AbstractDiscipline` class defines the `deploy(self)` method, which calls the three generation methods and stores the results in files at the paths specified by the three corresponding path properties.

Due to the file generation methods, the interface can be used to generate the in-/output and info files dynamically. This enables the possibility of defining DCs which depend on high level parameters or settings for their in- and/or output XML files. An example of this is a tool performing an analysis based on the geometry of an aircraft defined in a CPACS file. Because of the strict definition that a given input XML file contains only those XML elements read by the tool, and the output file only those written, the topology of the geometrical model cannot change. That is, if an input CPACS file of a DC contains just one wing, with just one wing segment, and a second file contains the same one wing, but with two wing segments, then they will have different variables. However, the analysis tool behind the DC is most likely able to handle CPACS files with an arbitrary number of wings and wing segments. To reconcile the strict in-/output variable definition and the flexibility of a tool, it should be possible to generate different in- and output XML files based on some parameters. By using bound class methods to dynamically generate these files, this is can easily be achieved by using extra class properties.

`openlego.core.xml_component`:
The `XMLComponent` class is a subclass of OpenMDAO's `ExplicitComponent` class. Two new concrete methods are added: `set_inputs_from_xml(self, file_path)` and `set_outputs_from_xml(self, file_path)`. They allow for input and output OpenMDAO variables to be defined given in- and output XML files respectively. These methods are used in the `__init__(self, in_file, out_file)` method to immediately define in- and output variables if in- and output XML files are supplied to it, but can be called at any time before the `setup(self)` method is called to overwrite these. The in- and output variable names and sizes are stored in simple Python dictionaries until the `setup(self)` method is called.

`XMLComponent` implements the `setup(self)` method to actually add OpenMDAO in- and output variables to the `Component` given the dictionaries created earlier. Furthermore, an abstract method, `execute(self, in_file, out_file)`, is declared which must be implemented by a subclass to perform the computation of the component given an input and an output XML file. Finally, the abstract `compute(self, inputs, outputs)` method declared by the `ExplicitComponent` class is implemented. It writes all inputs to a temporary XML file, calls the abstract `execute(in_file, out_file)` method with this input file, and parses the resulting output XML file, storing the new values in the appropriate OpenMDAO output variables.

This class is a straightforward translation of the high level description of the XML OpenMDAO executable part into an actual implementation. As was prescribed by the requirements, it only depends on OpenMDAO and works for any valid XML file. It is not specific to KADMOS/CMDOWS. Therefore it can be used outside of the context of KADMOS/CMDOWS as well when a tool is used which uses XML in- and output files.

`openlego.core.discipline_component`:
Because of the robust definition of the `AbstractDiscipline` and `XMLComponent` classes, the implementation of the `DisciplineComponent` is the simplest of the four core classes of OpenLEGO.

As was mentioned before, this class is a subclass of the `XMLComponent` class. Therefore it inherits all of its functionality automatically. It holds an instance of a subclass of `AbstractDiscipline` to define its in- and outputs and uses it to execute. As such, the `__init__(self, discipline)` method is overridden and extended to take an instance of `AbstractDiscipline` as input, storing it in a class attribute. The `__init__(self, in_file, out_file)` of the `XMLComponent` class it inherits from is then called using the in- and output XML file paths defined by the instance of `AbstractDiscipline` as input arguments. Corresponding OpenMDAO in- and output variables are then automatically created. Finally, the abstract `execute(in_file, out_file)` method of the `XMLComponent` class is implemented. It simply calls the `execute(in_file, out_file)` method of the instance of `AbstractDiscipline` the `DisciplineComponent` holds.

`openlego.core.model`:
In contrast to the `DisciplineComponent` class, the `LEGOModel` class has the most elaborate implementation of the four core classes. This is to be expected, because it performs most of the parsing and construction tasks listed in section 3.2. It is a subclass of OpenMDAO's `Group` class. It defines no new methods which should be used directly by OpenMDAO. Therefore it can be treated and used like any other instance or subclass of `Group`. To construct itself, the `LEGOModel`'s `__init__(self, cmdows_file, kb_path)` method allows the user to specify the path to a CMDOWS file which should be parsed, as well as the path at which the KB of the problem is stored. The latter should point to the folder containing the Python files in which the subclasses of `AbstractDiscipline` are stored.

The strategy used to implement the `LEGOModel` class relies heavily on the concept of dependent/-computed properties. For the majority of these, it uses a simple subclass of Python's builtin `property` decorator class called `CachedProperty`. This class stores the value of a computed property and bypasses its actual computation as long as it is still marked as valid. This avoids recomputation of properties which only need to be computed once, but are accessed repeatedly. This is especially beneficial for properties who's computation is time and/or resource expensive.

All of these dependent/computed properties depend on the the CMDOWS file, KB path, or both. Therefore, if either of those two is set all dependent/computed properties are automatically invalided and will be recomputed the next time they are read. Furthermore, if any of them is attempted to be read when either no CMDOWS file and/or no KB path is set, an exception is raised. This approach ensures the requirement of representativity whilst minimizing computational cost. It also has the advantage of allowing the code to be split up into smaller, isolated sub-tasks, which reduces code duplication.

### 3.6.2. Utilities Module

OpenLEGO relies heavily on the ability to read and write the values of OpenMDAO variables to and from XML files. This ability is required throughout the core classes, and should be available to the user as well, to make it easier to wrap external analysis tools in subclasses of `AbstractDiscipline`. Therefore a suite of shared XML utility functions have been written and stored in a dedicated module: `openlego.utils.xml_utils`.

Besides the XML utilities, a set of general utility functions have been written as well. These are stored in the `openlego.utils.general_utils` module. However, these function will not be explicitly discussed here. The interested reader is referred to the source code, given in appendix A.4.1.

This section will discuss the important functions defined in the `openlego.utils.xml_utils` module. The source code of this module can be found in appendix A.4.2.

`xpath_to_param` & `param_to_xpath`
The `xpath_to_param(xpath) -> str` and `param_to_xpath(param) -> str` functions perform the necessary operations to convert a valid XPath to a valid OpenMDAO variable name and vice versa. These functions are actually remnants of the version of OpenLEGO targeting OpenMDAO 1.x. They were necessary there, because OpenMDAO 1.x put restrictions on what characters could be used in variable names. In OpenMDAO 2.0, however, these restrictions were lifted.

The decimal point, however, still needs to be replaced if it appears in an XPath in the new version, because OpenMDAO uses decimal points as path separators between subsystems. In the interest of backward compatibility with the version of OpenLEGO targeting OpenMDAO 1.x, and to yield the minimum impact on the rest of the code, these functions were kept to perform this one translation.

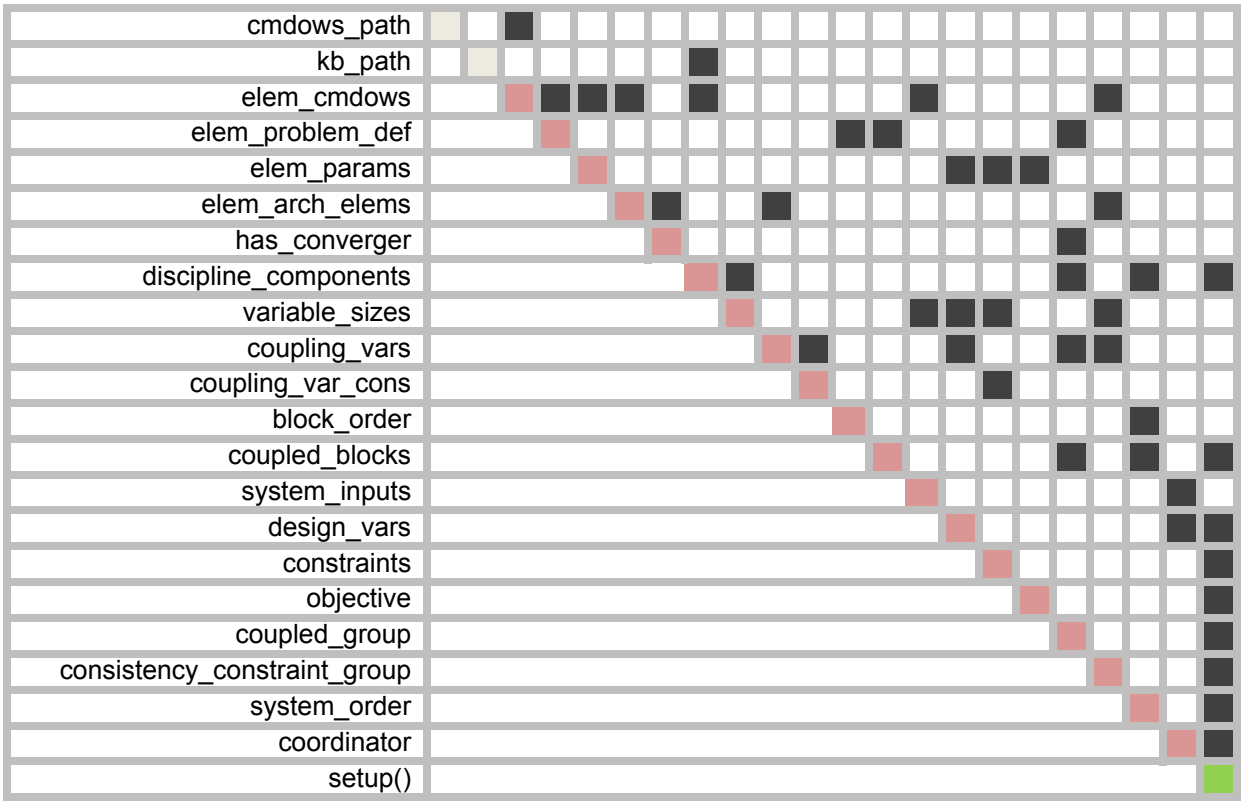



Figure 3.8: N2 diagram showing the dependency of all computed properties on one another.

In this version of OpenLEGO, decimal points appearing in XPaths are replaced with the sequence `':_:'`. This sequence was chosen because it is unlikely to appear directly within an XPath, even though it is a legal sequence within an XPath. Therefore it is unlikely that an XPath cannot be translated to a valid OpenMDAO variable name in a reversible way.

`xml_to_dict`

The `xml_to_dict(xml) -> OrderedDict` function is the key to turning an XML file into a set of OpenMDAO variables. It takes either the path of an XML file, or an instance of the `_ElementTree` class from the `lxml.etree` module and returns a representative instance of the `OrderedDict` class. The dictionary returned by this function contains an entry for each simple, valued XML element in the XML tree, with its full XPath as key.

This function uses the `etree.XPath` function to quickly obtain a list of all text elements in a given XML tree using the XPath expression `'//text()'`. It then iterates over the list of all elements that are returned. To obtain the full XPath of an element, the function backtracks from the text element up the tree until the XML file's root element is reached. At each node all XML attributes and indices are appended to the node's tag using the square bracket notation of XPaths. This ensures that the final XPath points to exactly the right element, which is especially important if there are multiple nodes sharing the same tag under a single parent. The value of each text element is parsed using the `parse_cmdows_value(value) -> Union[str,float,np.ndarray]` function.

`xml_safe_create_element`

The `xml_safe_create_element(tree, xpath, value=None) -> _Element` function is used to create an XML element in a given XML tree at a specific, absolute XPath. All intermediate elements - that is, every node between the root element and target element - are created if they do not yet exist. If an integer index is specified at a given node in the XPath, the correct number of sibling elements are created correspondingly to ensure the integer index position exists in the tree. If attributes are specified at a given node these are created accordingly as well. Finally, the given value is written to the target of the XPath, if one is supplied. As such, this function ensures that the specified XPath exists in the given

XML tree when it returns. It finally returns an instance of the `_Element` class from the `lxml.etree` module corresponding to the target element that was created.

`xml_merge`
The `xml_merge(base, merger, out_file=None)` function can be used to merge two XML files into one. This function is used regularly by the `XMLComponent` class to merge the outputs of a one analysis tool into a base XML file. The function merges the content of the `merger` into the `base`. By default the merged XML is written to the `base` file, but if the `out_file` argument is given, it will leave the `base` alone and instead write to this file. If there are conflicting elements which appear both in the `base` and in the `merger`, the values in the `merger` will overwrite those of the `base`.

This function makes use of the `xml_to_dict` function, section 3.6.2.2, to create a dictionary of the tree of the `merger`. Then it iterates over all key-value pairs and uses the `xml_safe_create_element` function, section 3.6.2.3, to write them to the `base` or `out_file` tree.

### 3.6.3. Recorders Module

Aside from the core functionality of OpenLEGO a set of OpenMDAO recorders have been developed. These have been stored in the `openlego.recorders` module, following OpenMDAO's naming convention. The concept of recorders was discussed briefly at the start of this section. The leftmost two classes in the class diagram of OpenMDAO's core, which was shown in fig. 3.5, represent this concept. As can be seen, recorders are managed by instances of `RecordingManager` held by both the `Driver` and `System` classes. The `RecordingManager` can hold any number of subclasses of the `BaseRecorder` ABC. Specific recorders can be created by defining a subclass of this ABC.

During the course of the development and testing of OpenLEGO a set of recorders was used repeatedly to monitor optimizations as they progressed. Matplotlib was used to generate and update plots for this purpose [44]. This Python plotting library is powerful, but has some important disadvantages. The most challenging issue that was encountered was that Matplotlib's figure windows freeze when they are used in interactive, non-blocking mode. The figures work well when they are used to display a single window to the user, pausing the code until the user closes the window. However, when a figure needs to be continually updated and refreshed the window becomes unresponsive and therefore unusable.

Eventually a general solution was found to this problem using Python's multiprocessing toolbox. The loop handling the figure window was moved to a dedicated, separate process. The Tkinter library was used to control the figure, which allows for direct control of the figure's main loop [86]. In order to allow the main process to send updates to the process handling the plot, a pipe and polling mechanism was used. This mechanism polls one end of a Python multiprocessing `Pipe` in a loop, handling any input as it is received. The added benefit of this approach is that all the code handling and updating the figure is run on a separate process. Therefore plotting does not block the computational thread performing the optimization. This method was found to work so well that it was generalized and wrapped within a subclass of OpenMDAO's `BaseRecorder`, `BaseIterationPlotter`, such that it can be reused to simply create a non-blocking, non-freezing, continually updated plot during an optimization run.

A set of specialized iteration plotters were created, building on the capabilities of `BaseIterationPlotter`. Because these recorders are generally useful, and because they solve common problems, they were considered relevant enough to include in the OpenLEGO package. They are listed and described briefly below.

`VOIPlotter`:
This recorder allows the user to plot any number of VOIs as a function of the number of iterations. The user can specify which VOIs need to be recorded by using the native OpenMDAO `options['includes']` options for recorders. Each VOI gets its own vertical axis to allow for VOIs of different ranges and scales to be displayed clearly in one figure.

`SimpleObjectivePlotter`:
This is a simple convenience recorder which automatically plots the normalized objective function value as a function of the number of iterations when it is attached.

`BaseLanePlotter`:
This recorder implements the concept of the *lane plot*, such as the ones shown by Bartoli et al. in [10]. This type of plot uses a colormap to map values of variables to a color range. Each variable

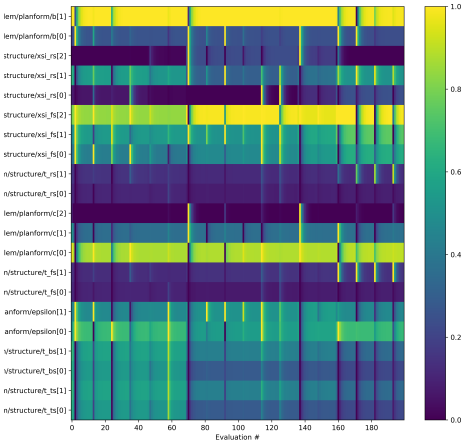



Figure 3.9: Part of the design variable lane plot during the wing optimization test case run.

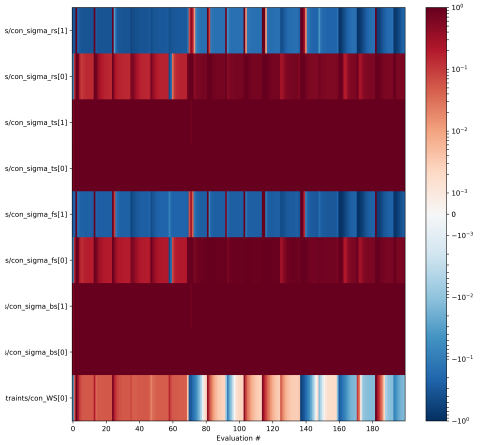



Figure 3.10: Part of the constraint variable lane plot during the wing optimization test case run.

which is plotted is represented by its own horizontal lane. In the context of optimizations, the horizontal axis corresponds to the number of iterations and each lane gets a colored rectangle at each iteration corresponding to its value at that point. These plots are particularly useful to display the design vectors of optimization problems, normalized to their bounds, or the values of the constraints as a function of the number of iterations. Examples of this are shown in figs. 3.9 and 3.10. The `BaseLanePlotter` is an ABC which should be implemented to make a specific plot, such as the next two in this list.

`NormalizedDesignVarPlotter`:
This is a subclass of the `BaseLanePlotter` ABC which plots all the design variables of the optimization problem using the lane plot style. All design variables are normalized to their bounds. That is, a variable at its upper bound maps to 1 and a variable at its lower bound maps to 0.

An example of such a lane plot depicting the design variables of an optimization problem is shown in fig. 3.9. This plot was created during the first part of the aeroelastic wing optimization problem test case, which will be described in section 4.3. This example uses the default settings for the `NormalizedDesignVarPlotter` recorder. It uses the perceptually uniform sequential colormap Viridis, which comes with matplotlib [44].

`ConstraintsPlotter`:
This is also a subclass of the `BaseLanePlotter` ABC which plots all the constraint values of the optimization problem using the lane plot style. All constraints are assumed to be satisfied if they are zero. It is up to the user whether inequality constraints are considered satisfied when they are $\leq$ or $\geq$ than zero. A symmetric logarithmic colormap is used by default in order to capture the variation constraints' values equally well when they are close to zero and further away from it.

An example of a plot created with this recorder is shown in fig. 3.10. Like fig. 3.9, this plot was created during the first part of the aeroelastic wing optimization problem test case described in section 4.3. This example plot was created using the default settings of the `ConstraintsPlotter` recorder. As can be seen, a symmetric logarithmic scale is used with a diverging colormap called 'RdBu', which comes with standard matplotlib [44]. The lighter the color, the closer to active a constraint is considered to be like this.

# 4

# Results

To demonstrate the capabilities of OpenLEGO two test cases were set up: the Sellar problem and a low-fidelity aerostructural wing optimization problem. This section will present these test cases and their implementations in OpenLEGO.

The Sellar test case will be discussed first, in section 4.2. This test case serves as a proof of concept of OpenLEGO. Besides constructing and solving the optimization problem, the Sellar problem will be used to demonstrate OpenLEGO's ability to handle different MDO architectures. Next, in section 4.3, the wing optimization test case will be discussed. This case demonstrates the applicability of OpenLEGO to a more realistic MDO problem in the context of aircraft design. Although this is still a problem with a relatively small number of disciplines, it has a much larger number of variables and serves as a first demonstration of OpenLEGO's ability to handle bigger problems using external tools.

## 4.1. Scalable Optimization Problem

To demonstrate the advantage of making use of analytical gradients if they are made available by analysis tools, a scalable, mathematical optimization problem was studied. The approach and setup presented in [90] was followed. The mathematical description of the optimization problem studied here is given in eq. (4.1).

$$
\begin{aligned}
\text{minimize} \quad & f && = \boldsymbol{z}^T\boldsymbol{z} + \sum_{i=1}^{n_d} \mathbf{y}_i^T\mathbf{y}_i, \\
\text{w.r.t.} \quad & \mathbf{x}_i, \boldsymbol{z}, && \\
\text{subject to} \quad & \mathbf{g}_i && = 1 - C_{g,i}\mathbf{y}_i \geq 0, \\
\text{where} \quad & D_i : \hat{C}_{y,i}\mathbf{y}_i && = C_{x,i}\mathbf{x}_i - \sum_{j=1, j\neq i}^{n_d} C_{y,j}\mathbf{y}_i + C_z\boldsymbol{z}, \\
& \hat{C}_{y,i} && = \left[\left(n_y + n_x\right) n_d + n_z\right] C_{y,i}^{\circ -1}, \\
& C_{x,i} && \in \mathbb{R}^{n_y \times n_x}, \\
& C_{y,i}, C_{g,i} && \in \mathbb{R}^{n_y \times n_y}, \\
& C_z && \in \mathbb{R}^{n_y \times n_z}, \\
& i && \in [1, n_d], \\
& n_d, n_x, n_y, n_z && \in \mathbb{R},
\end{aligned}
\tag{4.1}
$$

where $C_{x,i}$, $C_{y,i}$, $C_{g,i}$, and $C_z$ are matrices with random coefficients, determined prior to the start of an optimization. The notation $A^{\circ -1}$ represents the Hadamard inverse [73], defined by eq. (4.2).

$$
B = A^{\circ -1} \iff B_{ij} = A_{ij}^{-1}, \tag{4.2}
$$

i.e., the element-wise inverse of a matrix. The number of disciplines, $n_d$, local design variables per discipline, $n_x$, local coupling variables per discipline, $n_y$, and global design variables, $n_z$, can be varied.

This allows the problem to be tweaked to any given characteristic. For this study, as was done in [90], the coefficients of all matrices but the ones related to the coupling variables were set to unity. Furthermore, all off-diagonal coefficients of $C_{g,i}$ were set to zero. Hence, the equation was simplified to eq. (4.3):

$$\begin{aligned}
\text{minimize} \quad & f && = \mathbf{z}^T\mathbf{z} + \sum_{i=1}^{n_d} \mathbf{y}_i^T\mathbf{y}_i, \\
\text{w.r.t.} \quad & \mathbf{x}_i, \mathbf{z}, \\
\text{subject to} \quad & \mathbf{g}_i && = 1 - \text{diag}\left(\mathbf{c}_{g,i}\right)\mathbf{y}_i \geq 0, \\
\text{where} \quad & D_i : \hat{C}_{y,i}\mathbf{y}_i && = J_{n_y,n_x}\mathbf{x}_i - \sum_{j=1,j\neq i}^{n_d} C_{y,j}\mathbf{y}_i + J_{n_y,n_z}\mathbf{z}, \\
& \hat{C}_{y,i} && = \left[\left(n_y + n_x\right)n_d + n_z\right] C_{y,i}^{\circ -1}, \\
& C_{y,i} && \in \mathbb{R}^{n_y \times n_y}, \\
& i && \in [1, n_d], \\
& n_d, n_x, n_y, n_z && \in \mathbb{R},
\end{aligned} \tag{4.3}$$

where the notation $J_{m,n}$ denotes an $m \times n$ matrix of ones [43], and $\mathbf{c}_{g,i}$ is a vector of length $n_y$ with random values.

For this study, the number of disciplines and the number of global design variables were fixed to $n_d = n_z = 3$, as was done in [90]. The number of local design variables and coupling variables per discipline were varied. All combinations of the values 2, 20, and 200 for each of them were analyzed, making for 9 different configurations. With each of these combinations, 100 random problems were generated and solved once with analytical gradients and once using finite differencing to approximate the gradients. The amount of function evaluations of each discipline was recorded for both the run with and without analytical gradients. An MDF architecture was used for all runs. All 'analysis' disciplines, $D_i$, are coupled to one another due to the coupling variables, $\mathbf{y}_i$. Therefore they are all part of the MDA of the MDF architecture. Since they all have the same number of local design variables, $\mathbf{x}_i$, non-local coupling variables, $\mathbf{y}_{j\neq i}$, and global design variables, $\mathbf{z}$, as inputs, they are all executed the same number of times as each other, regardless of whether analytical gradients are used or not. Therefore it was sufficient to note only the amount of function evaluations of the first discipline.

The average number of function evaluations across all 100 runs of every configuration are tabulated in table 4.1. The differences between the two runs for each configuration are tabulated in table 4.2.

Table 4.1: Average number of discipline function evaluations for across 100 complete optimization runs using finite differencing (upper left hand corners), and using analytical gradients (lower right hand corners) for all combinations of $n_x$ and $n_y$.

| $n_x$ \ $n_y$ | 2 | 20 | 200 |
|---|---|---|---|
| 2 | 102.97<br>1086.24 | 96.10<br>3191.95 | 31.30<br>8743.15 |
| 20 | 89.51<br>2994.13 | 95.74<br>4816.18 | 65.88<br>16051.50 |
| 200 | 93.89<br>36735.18 | 99.29<br>46133.00 | 100.85<br>51671.79 |

Finally, the fractions of these differences w.r.t. the numbers corresponding to the runs using finite

Table 4.2: Differences between the average number of discipline function evaluations per configuration

| $n_x$ \ $n_y$ | 2 | 20 | 200 |
|---|---|---|---|
| 2 | 983.27 | 3095.85 | 8711.85 |
| 20 | 2904.62 | 4720.44 | 15 985.62 |
| 200 | 36 641.29 | 46 033.71 | 51 570.93 |

differencing are given in table 4.3. It is immediately clear from these results that making use of analytical

Table 4.3: Percent differences between the average number of discipline function evaluations per configuration

| $n_x$ \ $n_y$ | 2 | 20 | 200 |
|---|---|---|---|
| 2 | 90.52% | 96.99% | 99.64% |
| 20 | 97.01% | 98.01% | 99.59% |
| 200 | 99.74% | 99.78% | 99.80% |

gradients has a dramatic, positive impact on the number of discipline function evaluations required to complete an optimization run. Even in the lightest case, with the smallest number of local design and coupling variables per discipline, the average number of discipline function evaluations is reduced by more than 90%. As can be seen, the reduction is larger the more complex the configurations get. For all configurations where the number of local design variables or the number of coupling variables per discipline were set to 200, the reduction is well over 99%.

Hence, making use of analytical gradients if they are provided by analysis tools can profoundly reduce the computational cost of any optimization run. Therefore it is indeed very important to make it easier to achieve analytical gradient utilization in any modern MDAO system, such as the one developed as part of this thesis. This serves as further justification of the work done.

## 4.2. Sellar Problem

The Sellar problem is a common test problem used to demonstrate and test MDO solution strategies [82]. It is used as an example by both OpenMDAO [68] as well as KADMOS [95]. It is a purely mathematical problem which can be written as eq. (4.4).

$$
\begin{aligned}
\text{minimize} \quad & f_1(x_1, y_1, y_2, z_2) && = x_1^2 + z_2 + y_1 + \mathrm{e}^{-y_2} \\
\text{w.r.t.} \quad & x_1, z_1, z_2, \\
\text{for} \quad & 0 \leq x_1 \leq 10, \\
& -10 \leq z_1 \leq 10, \\
& 0 \leq z_2 \leq 10, \\
\text{subject to} \quad & g_1(y_1) && = 1 - \frac{y_1}{3.16} \leq 0, \\
& g_2(y_2) && = \frac{y_2}{24} - 1 \leq 0, \\
\text{where} \quad & y_1 = D_1(x_1, y_2, z_1, z_2) && = x_1 + z_1^2 + z_2 - 0.2y_2, \\
& y_2 = D_2(y_1, z_1, z_2) && = z_1 + z_2 + \sqrt{y_1}.
\end{aligned}
\tag{4.4}
$$

When written in this way, it can be understood as a single objective, constrained, multidisciplinary optimization problem with two coupled disciplines, $D_1$ and $D_2$. This problem has two local minima,

$$
\begin{aligned}
&x_1^* = 0.11,\ y_1^* = 3.16,\ y_2^* = 0.20,\ z_1^* = -1.72,\ z_2^* = 0.14,\ f^* = 4.13,\ g_1^* = 0.00,\ g_2^* = -0.99, \text{ and} \\
&x_1^* = 0.00,\ y_1^* = 3.16,\ y_2^* = 3.76,\ z_1^* = +1.98,\ z_2^* = 0.00,\ f^* = 3.18,\ g_1^* = 0.00,\ g_2^* = -0.84,
\end{aligned}
$$

the second of which is the problem's global minimum. At both local minima the first constraint, $g_1$, is active and the second, $g_2$, is inactive. Both $x_1$ and $z_2$ are at their respective lower bounds at both minima. Being defined purely in terms of simple algebraic functions, the Sellar problem can be solved quickly on any modern computer. However, its solution is not trivial, because of the strong coupling between $D_1$ and $D_2$ through coupling variables $y_1$ and $y_2$. Together, these qualities make the Sellar problem a good test case for MDO systems.

Since the problem is purely mathematical, the gradients can be expressed as simple mathematical functions as well. To obtain these, the derivatives of the objective function, $f_1$, constraint functions, $g_1$ and $g_2$, and the two coupling variables (disciplines), $y_1$ and $y_2$, need to be carried out. They are given in eqs. (4.5a) to (4.9c).

$$\frac{\partial f_1}{\partial x_1} = 2x_1 \quad (4.5a)$$
$$\frac{\partial f_1}{\partial y_1} = 1 \quad (4.5b)$$
$$\frac{\partial f_1}{\partial y_2} = -e^{-y_2} \quad (4.5c)$$
$$\frac{\partial f_1}{\partial z_2} = 1 \quad (4.5d)$$

$$\frac{\partial g_1}{\partial y_1} = -\frac{1}{3.16} \quad (4.6)$$
$$\frac{\partial g_2}{\partial y_2} = \frac{1}{24} \quad (4.7)$$

$$\frac{\partial y_1}{\partial x_1} = 1 \quad (4.8a)$$
$$\frac{\partial y_1}{\partial y_2} = -0.2 \quad (4.8b)$$
$$\frac{\partial y_1}{\partial z_1} = 2z_1 \quad (4.8c)$$
$$\frac{\partial y_1}{\partial z_2} = 1 \quad (4.8d)$$

$$\frac{\partial y_2}{\partial y_1} = \frac{1}{2\sqrt{y_1}} \quad (4.9a)$$
$$\frac{\partial y_2}{\partial z_1} = 1 \quad (4.9b)$$
$$\frac{\partial y_2}{\partial z_2} = 1 \quad (4.9c)$$

To create a KB with which the Sellar problem can be solved, six subclasses of OpenLEGO's `AbstractDiscipline` class were created: `D1`, `D2`, `F1`, `G1`, and `G2`. They represent the two coupled disciplines, the objective function, and the two constraint functions respectively. An example implementation of `D1` is shown in code frag. 4.1.

```
1  from lxml import etree
2  from openlego.api import AbstractDiscipline
3  from openlego.xml import xml_safe_create_element
4  from openlego.partials.partials import Partials
5
6
7  class D1(AbstractDiscipline):
8
9      @property
10     def supplies_partials(self):
11         return True
12
13     def generate_input_xml(self):
14         return '<data>' + \
15                     '<x1>0.0</x1>' + \
16                     '<y2>0.0</y1>' + \
17                     '<z1>0.0</z1>' + \
18                     '<z2>0.0</z2>' + \
19                '</data>'
20
21     def generate_output_xml(self):
22         return '<data>' + \
23                     '<y1>0.0</y1>' + \
24                '</data>'
25
26     def generate_partials_xml(self):
27         partials = Partials()
28         partials.declare_partials(x_y1, [x_x1, x_y2, x_z1, x_z2])
29         return partials.get_string()
30
31     @staticmethod
32     def execute(in_file, out_file):
33         doc = etree.parse(in_file)
34         x1 = float(doc.xpath('/data/x1')[0].text)
35         y2 = float(doc.xpath('/data/y2')[0].text)
36         z1 = float(doc.xpath('/data/z1')[0].text)
37         z2 = float(doc.xpath('/data/z2')[0].text)
38
39         y1 = x1 + z1**2 + z2 - .2*y2
40
41         root = etree.Element('data')
```

```
42          doc = etree.ElementTree(root)
43          xml_safe_create_element(doc, '/data/y1', y1)
44          doc.write(out_file)
45
46      @staticmethod
47      def linearize(in_file, partials_file):
48          doc = etree.parse(in_file)
49          z1 = float(doc.xpath(x_z1)[0].text)
50
51          partials = Partials()
52          partials.declare_partials(x_y1,
53              [x_x1, x_y2, x_z1, x_z2],
54              [  1.,  -.2, 2*z1,   1.])
55          partials.write(partials_file)
```

Code frament 4.1: Example implementation of discipline $D_1$ of the Sellar problem in OpenLEGO

This class implements the three abstract methods of `AbstractDiscipline`, `generate_input_xml()`, `generate_output_xml()`, and `execute()`. The first two return simple, static Python strings representing XML files with elements corresponding to the discipline's four inputs, `x1, y2, z1, z2`, and single output, `y1`, respectively. The `execute()` method uses the `lxml` Python package and the `xml` utilities module, included in the OpenLEGO package, to read the input values from the given input XML file, `in_file`, calculate the value of `y1`, and write the resulting value of `y1` to the given output XML file, `out_file`.

The class also overrides the standard behavior of the `supplies_partials` property, `generate_partials_xml()` function, and `linearize()` function. The first is simply set to `True`, to signify that analytical gradients are provided by the discipline. In the second, the `Partials` class is used to generate a template partials XML file declaring which partials the discipline provides. Finally, the `linearize` function calculates the values of the gradients given an input XML file.

The other disciplines were implemented similarly. Note that this implementation is very verbose and inefficient for the simple disciplines of the Sellar problem, but is required because generally analysis tools are more complex and are often represented by external programs.

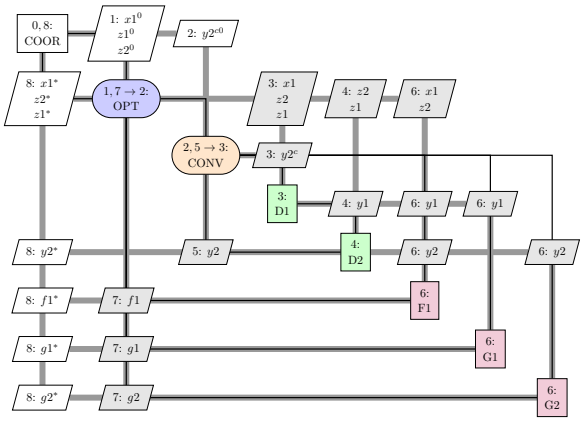
After generating all in-, output, and information files, KADMOS was used to define the problem and generate a corresponding CMDOWS file. For this test case, the Multi-Discipline Feasible (MDF) MDO architecture was used with a Gauss-Siedel (GS) converger [60]. An eXtended Design Structure Matrix (XDSM) [60] of the resulting problem definition, generated by KADMOS, is shown in fig. 4.1a.

The resulting CMDOWS file could then be used to generate the OpenMDAO model using a single line of code (second line of code frag. 4.2). Finally, in order the solve the optimization problem, this model was attached to an instance of OpenMDAO's `Problem` class, along with an appropriate subclass of `Driver`. A minimal example using the `ScipyOptimizer` driver with the default settings is given in code frag. 4.2.

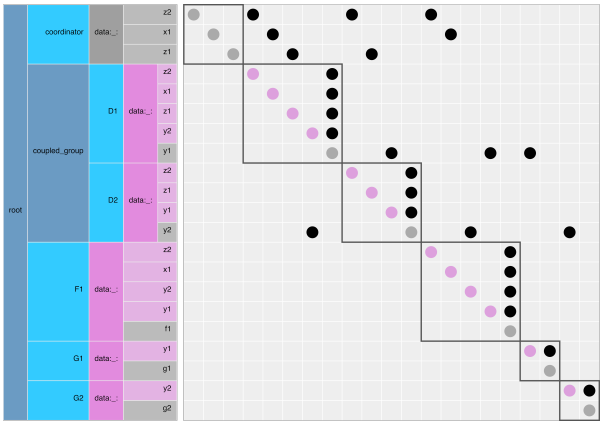
```
1 prob = Problem()
2 prob.model = LEGOModel('path/to/cmdows/file.xml', 'path/to/kb')
3 prob.driver = ScipyOptimizer()
4
5 prob.setup()
6 prob.run_driver()
```

Code frament 4.2: Minimal example for the Sellar test case

Once the OpenMDAO `Problem` was properly constructed and set up, the OpenMDAO `view_model()` function was used to generate an N2 diagram of the problem, shown in fig. 4.1b. Comparing this with the XDSM diagram generated by KADMOS, it is clear that the OpenMDAO representation corresponds perfectly with the intended problem definition. The OpenMDAO problem was run with initial values $x_1^0 = 5$, $z_1^0 = 0$, $z_2^0 = 5$, and converged to the known global minimum after 5 iterations. Hence, Open-

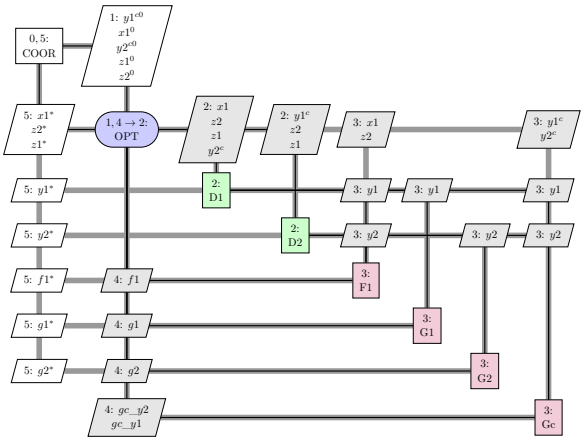


(a) XDSM, generated by KADMOS

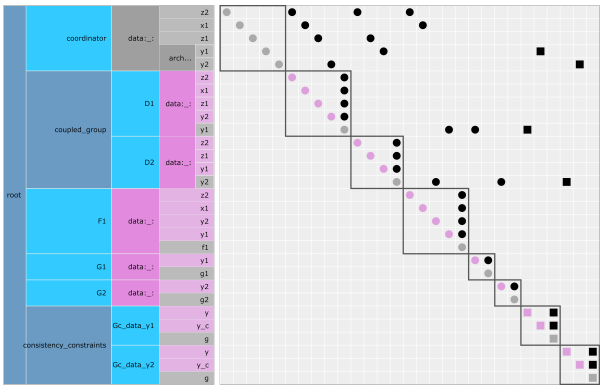


(b) N2 diagram, generated by OpenMDAO

Figure 4.1: XDSM and N2 diagram of the Sellar problem using the MDF architecture



(a) XDSM, generated by KADMOS



(b) N2 diagram, generated by OpenMDAO

Figure 4.2: XDSM and N2 diagram of the Sellar problem using the IDF architecture

LEGO was able to generate a correct and functioning OpenMDAO representation of the CMDOWS file, which converges to the correct solution when run.

The problem can also be solved using an Individual Discipline Feasible (IDF) [60] MDO architecture. KADMOS is able to generate a CMDOWS file for the problem wrapped in the IDF architecture by simply changing the `mdao_architecture` setting, and OpenLEGO can handle the resulting file just as well as the MDF variant. An XDSM of the problem using the IDF architecture is shown in fig. 4.2a alongside the N2 diagram generated by the OpenMDAO model constructed by OpenLEGO, fig. 4.2b. When running this problem using $y_1^{c0} = 5$, $y_2^{c0} = 5$ as initial values of the coupling variables, and the same initial values for the design variables as for the MDF case, this problem converges to the global optimum after 6 iterations.

When comparing fig. 4.2b to fig. 4.1b, it can be seen that OpenLEGO has added two new design variables, corresponding to $y_1^c$ and $y_2^c$, and a new group, `consistency_constraints`, with two new components, `Gc_data_y1` and `Gc_data_y2`, corresponding to the consistency constraint functions represented by the *Gc* block in fig. 4.2a. The feedback between disciplines `D1` and `D2`, which is present in the MDF architecture (represented by the single black circle positioned below the diagonal of the N2 diagram of fig. 4.1b) has been removed by the addition of copies of the coupling variables and the consistency constraint functions. Indeed, no feedback connections are present in the N2 diagram of the IDF architecture, as can be seen in fig. 4.2b. This lack of feedback between the coupled disciplines is also precisely the difference between the MDF and IDF architectures [60], and is evident as well when comparing the XDSMs in figs. 4.1a and 4.2a. This proves OpenLEGO is able to handle both the

MDF and the IDF architectures.

## 4.3. Aerostructural Wing Optimization

The aerostructural wing optimization test case is a low-fidelity, conceptual/preliminary design space exploration of a wing. The problem uses planform and structural parameters as design variables. The dAEDalus framework is used to perform the aerostructural analyses [12, 83, 84]. This is an open-source tool written for Matlab, which uses a Vortex Lattice Method (VLM) as aerodynamic, and a linear structural model, based on straight, connected beam segments.

New functions were written for dAEDalus first, allowing it to be initialized using the CPACS data schema. Next, Matlab functions were written to wrap the functionality of dAEDalus and yield a set of distinct operations which could in turn be used to define distinct disciplines for the MDO problem. The dAEDalus framework uses a hierarchy of Matlab objects to store the model of an aircraft. Functions can be called on these objects to perform analyses. These analyses then enrich the objects with their results, after which they can be retrieved. For example, when performing the aerodynamic analysis on the model, the aerodynamic forces and moments are imposed on the grid, and stored on each panel of the aerodynamic model.

Within dAEDalus, the aircraft is represented by three related, but distinct objects representing a geometric, structural, and aerodynamic model respectively. The geometric model is constructed directly from the input file and is the basis for the construction of the other two. The structural model is created from the geometric model given a set of discretization settings, such as the maximum spanwise length of beam elements. By imposing forces and moments on each beam element, the structural model computes the resulting displacements of the grid points. The aerodynamic model is created from the geometric model, taking into account the deflections computed by the structural model. Given a set of panels, the aerodynamic model computes the aerodynamic forces and moments acting on each. These can then be used to compute the new resulting deflections. Each time the grid is deflected, the aerodynamic model needs to be updated and the aerodynamic forces and moments change slightly, leading in turn back to slightly different deflections. This strong coupling between the aerodynamics and structures is, the phenomenon known as aeroelasticity [56].

The construction of the dAEDalus models is costly in terms of computation time and memory. Therefore they should be created as least often as possible and should preferably not be moved around in memory. Much of the data contained in these models is not used directly by the optimization, but rather by Matlab to perform the analyses. Since communicating large amounts of data between Matlab and Python would lead to a lot of overhead, it was decided to store the models created by dAEDalus during a run in the Matlab workspace and only communicate the data needed by the optimization. With this in mind, four DCs were created to encapsulate the generation of the models and the performance of dAEDalus' analyses:

1. `dSMI` (dAEDalus Steady Model Initializer), which constructs the geometric and structural models and returns the initial structural grid and the weight of the wing;
2. `dSAMI` (dAEDalus Steady Aerodynamic Model Initializer), which constructs the aerodynamic model and returns the lift coefficient and friction drag coefficient of the wing;
3. `dSAA` (dAEDalus Steady Aerodynamic Analysis), which runs the VLM on the aerodynamic model and returns the induced drag coefficient; and
4. `dSSA` (dAEDalus Steady Structural Analysis), which runs the structural analysis and returns the stresses and deflected structural grid.

For this low-fidelity test case, a very simple model of the wing was used, based on a fixed number, $n_{\text{ws}}$, of tapered wing segments. In this reduced model, the geometry of a single wing segment at index $i$ is completely described by two chord lengths, $c_i$ and $c_{i+1}$, two twist angles, $\epsilon_i$ and $\epsilon_{i+1}$, two thickness over chord ratios, $\left({}^{t}/_{c}\right)_i$ and $\left({}^{t}/_{c}\right)_{i+1}$, a span, $b_i$, sweep angle, $\Lambda_i$, and dihedral angle, $\Gamma_i$. The structure of each wing segment is comprised solely of a wingbox with a trapezoidal cross-section. This wingbox has constant front spar, rear spar, top skin, and bottom skin thicknesses along each segment, $t_{\text{fs}_i}$, $t_{\text{rs}_i}$, $t_{\text{ts}_i}$, and $t_{\text{bs}_i}$. The cross-sections are defined at each end of a wing segment by front and rear spar chordwise locations, $\xi_{\text{fs}_i}$, $\xi_{\text{fs}_{i+1}}$, $\xi_{\text{rs}_i}$, and $\xi_{\text{rs}_{i+1}}$. Furthermore, it is prescribed that two adjacent wing

segments share the same section. Hence, there cannot be a jump in chord length, and spars cannot have cuts, for example. Additionally, a set of reference values are used, namely: a non load-bearing skin thickness and density, $t_{\text{skin}}$ and $\rho_{\text{skin}}$, a fixed aircraft mass, $m_{\text{fixed}}$, payload mass, $m_{\text{PL}}$, maximum landing mass, $m_{\text{MLW}}$, a systems mass fraction, $f_{m_{\text{sys.}}}$, and a wings mass fraction, $f_{m_{\text{wings}}}$. These are used to estimate the new total weight of the aircraft as the weight of the wing changes during the optimization. A discipline named `WOM` (Wing Object Model) was created which takes all of these parameters as input and yields the aircraft in CPACS format as output.

The objective of the optimization problem is to minimize the fuel burn for a given range and payload by changing the geometric and structural wing parameters. A simple estimation of the fuel burn based on the Breguet range equation is used [76]. The mission is simplified to a single, constant speed, constant altitude cruise leg. A discipline named `FWE` (Fuel Weight Estimator) was created to perform this estimation.

The stresses are calculated at three different load cases: 1g cruise flight, a 2.5g symmetric pull-up maneuver, and a -1.0g symmetric push-over maneuver. They are collected by a load collector discipline, `dLC`, which computes the maximum stress out of all cases in each part of the wingbox, in each wing segment. These collected stresses are then constrained to be below a given yield stress, $\sigma_{\text{yield}}$. Additionally, the wing loading ($W/S$) is constrained to be smaller than or equal to the original value.

The innermost twist angle is considered to be the wing's overall incidence angle. It is subtracted from all the other twist angles and kept fixed at its initial value. If it were not fixed, there would be an ambiguity between it and the angle of attack of the wing. Furthermore, the effect of the incidence angle on the longitudinal stability as well as any interference effects between the wing and the fuselage are not captured by the analyses. The thickness over chord ratios are kept constant as well, because their impact on the aerodynamic characteristics of the wing cannot be captured with a VLM. The same is true for the sweep angles, because compressibility effects are not captured accurately. Finally, directional stability is not considered, so the dihedral angles are also left unchanged too. Therefore the design vector is comprised of the variables $c_i$, $\epsilon_j$, $b_j$, $\xi_{\text{fs}_i}$, $\xi_{\text{rs}_i}$, $t_{\text{fs}_j}$, $t_{\text{rs}_j}$, $t_{\text{ts}_j}$, and $t_{\text{bs}_j}$, for $i \in [0, n_{\text{ws}} + 1]$ and $j \in [0, n_{\text{ws}}]$.

Once all disciplines were defined in OpenLEGO by creating a subclass of the `AbstractDisciipline` class for each, the problem could easily be constructed using KADMOS. An XDSM of the problem using the MDF architecture, generated by KADMOS, is shown in fig. 4.3. Note the feedback

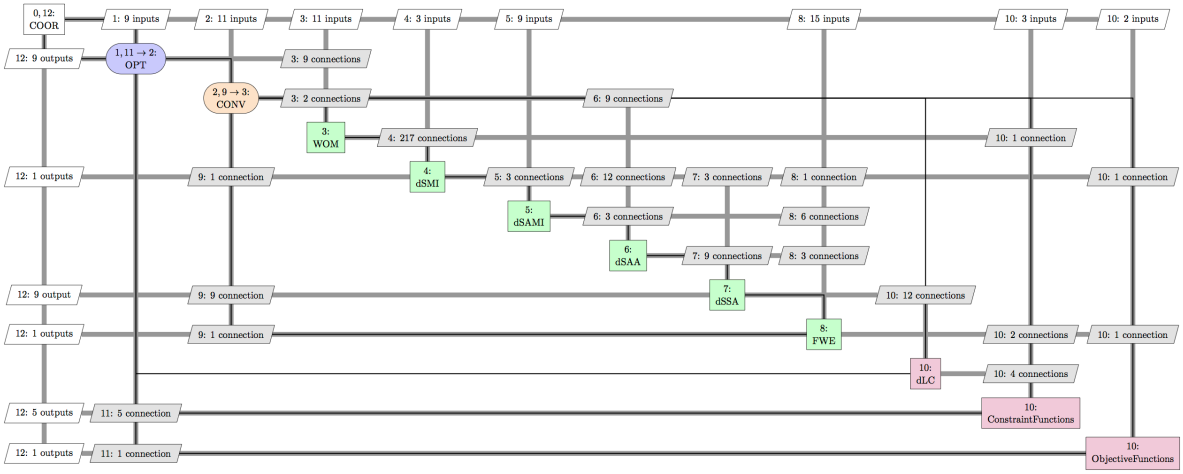



Figure 4.3: XDSM of the wing optimization problem using MDF, generated by KADMOS

from the model initializer, `dSMI`, structural analysis, `dSSA`, and fuel weight estimator, `FWE`. The first calculates the wing weight, and the latter calculates the fuel weight. Both of these need to be used by the wing object model, `WOM`, to calculate the updated takeoff weight. The aerodynamic and structural analyses are coupled to one another, which is, of course, the crux of the aeroelastic problem. As such, the feedback from the structural analysis block corresponds to the deflected grid, which is used by the aerodynamic model to compute the forces for the updated, deflected wing shape. As can be seen,

these feedbacks all connect to the converger block, and the coupled analyses are run consecutively, since an MDF architecture is used. As such, all coupled analyses can be seen as sub-disciplines of a higher-level multidisciplinary analysis (MDA).

Also note that the black lines indicating the order of execution in this diagram are actually incorrect for the last four blocks. According to the XDSM, all three post-coupling blocks can be run in parallel. Yet, the load collector discipline, `dLC`, supplies inputs to the `ConstraintFunctions`, and the fuel weight estimator, `FWE`, supplies inputs to the `ConstraintFunctions` and `ObjectiveFunctions` blocks. Therefore they would have to be run sequentially and not in parallel. This is a mistake made by KADMOS, because it assumes post-coupling blocks do not provide input to one another. However, when parsing a CMDOWS file OpenLEGO does not use this information. Therefore it is not affected by this mistake and the CMDOWS file can still be used to generate a correct OpenMDAO representation of the problem.

OpenLEGO was able to generate a representative OpenMDAO problem from the CMDOWS file created by KADMOS. A collapsed view of the N2 diagram created by OpenMDAO's `view_model()` method of this problem is shown in fig. 4.4. When comparing fig. 4.4 with fig. 4.3 it is evident that

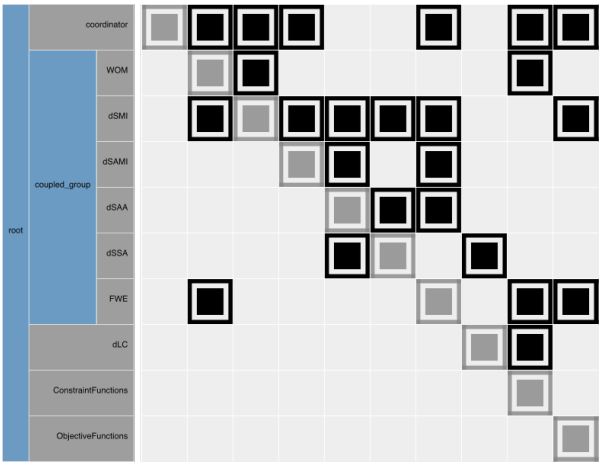



Figure 4.4: N2 diagram of the wing optimization problem using MDF, generated by OpenMDAO

the disciplines were ordered correctly, and connections were created between the correct couples of disciplines. Upon closer inspection of the model, it was concluded that it matched the problem description.

Unfortunately, no gradient information was available for the dAEDalus tools. Therefore this part of the pipeline can not be demonstrated with this problem. The importance of having gradient information available, however, was made apparent by this problem with fervor. Note the number of connections between the coupled analyses in fig. 4.3. Especially between `WOM` and `dSMI` there are 217 connections listed. This is more than the maximum number of coupling variables that were studied with the scalable optimization problem, as was described in section 4.1, which went up to 200. There each discipline had to execute up to tens of thousands of times to complete an optimization. To demonstrate the significance of this issue with an industrially relevant problem, the gradients of the full wing optimization model, corresponding to the XDSM of fig. 4.3, were estimated once using OpenMDAO's functions. The number of function evaluations of the four dAEDalus analyses, `dSMI`, `dSAMI`, `dSAA`, and `dSSA`, were recorded. They are given in table 4.4. As can be seen, these numbers are very large. Especially the

Table 4.4: Number of function evaluations of the dAEDalus Steady Model Initializer (`dSMI`), Aerodynamic Model Initializer (`dSAMI`), Aerodynamic Analysis (`dSAA`), and Structural Analysis (`dSSA`) for one gradient evaluation.

| **Discipline** | dSMI | dSAMI | dSAA | dSSA | total |
|---|---|---|---|---|---|
| **Number of evaluations** | 9244 | 24 | 24963 | 16668 | 50899 |

aerodynamic and structural analysis stand out. This is due to the deflected grids being passed between them as a result of the structural deformations. These are represented by vectors in the model, which is why the number of connections between `dSAA` end `dSSA` are listed as only 9 ($x$, $y$, and $z$ vectors for

Table 4.5: Geometric design variables

| **symbol** | $c_\mathsf{r}$ | $c_\mathsf{k}$ | $c_\mathsf{t}$ | $\epsilon_\mathsf{k}$ | $\epsilon_\mathsf{t}$ | $b_\mathsf{ib}$ | $b_\mathsf{ob}$ |
|---|---|---|---|---|---|---|---|
| **units** | m | m | m | rad | rad | m | m |
| **initial** | 13.71 | 7.26 | 2.73 | −0.10 | −0.18 | 12.72 | 22.70 |
| **final** | 14.63 | 8.51 | 2.21 | −0.004 | −0.035 | 8.76 | 24.97 |

three load cases). However, to calculate the gradients, each individual element of these vectors are varied independently, leading to the enormous amounts of evaluations listed in the table.

On the system that this run was performed on, this single gradient evaluation took 5 hours and 25 minutes. If the gradients need to be computed 150 times for a single optimization, a conservative amount for an optimization of this size, it would take more than a month to complete. This demonstrates the importance of making gradient information available, and using it in an optimization framework.

In order to be able to run the problem within a feasible amount of time, the problem was simplified. This was done by collapsing all of the4 coupled disciplines, except for the fuel weight estimator, into a since, multidisciplinary analysis tool. This has the effect of removing the thousands of coupling variables between the aerodynamic and structural analysis, and between the wing object model and the model initializers. As such, the number of function evaluations required for the estimation of the gradients was reduced drastically. An XDSM of the simplified problem is shown in fig. 4.5.

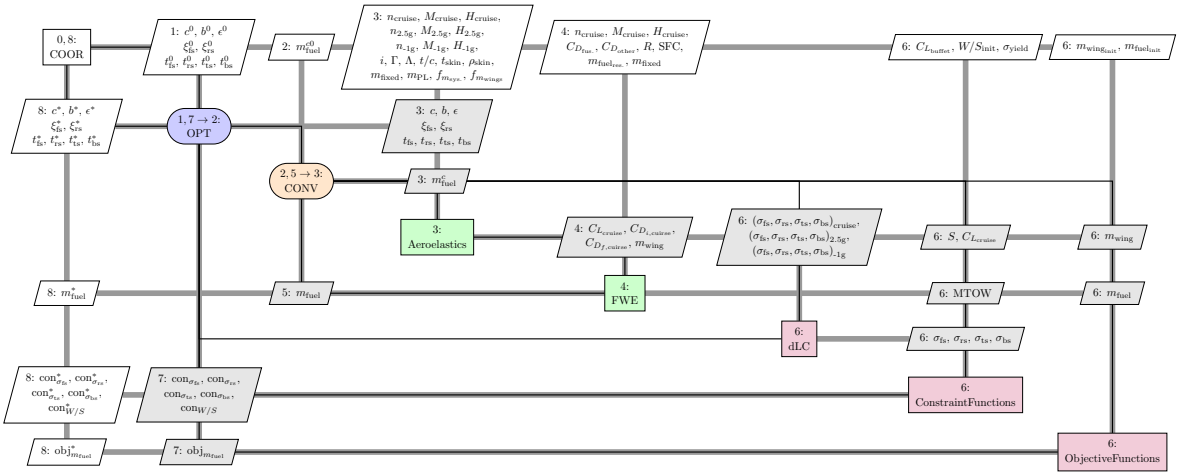
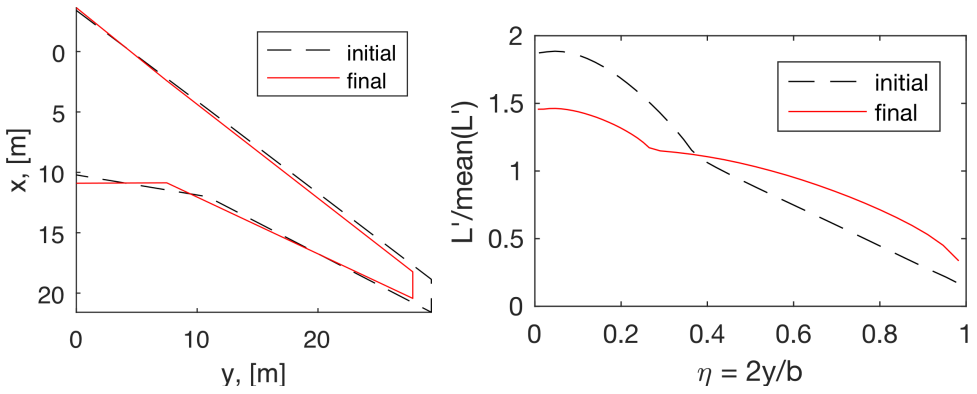



Figure 4.5: XDSM of the simplified wing optimization problem using MDF, generated by KADMOS

The optimization problem was run using an initial design based on the NASA Common Research Model (CRM) [98], using $n_\mathsf{ws} = 2$. The run had to be started back up three times due to the memory overflows. Unfortunately, the optimization finally exited abnormally due to a positive derivative in the search direction after 201 iterations, having run continuously for over a week. The final design point was infeasible, with most constraints violated.

The geometric and structural design variables at the initial and final point are tabulated in table 4.5 and table 4.6 respectively. The initial and final deflected wing shape and lift distributions (1g cruise flight) are shown in figs. 4.6a and 4.6b. As can be seen, the overall taper ratio of the wing was increased. This is to be expected when a VLM is used, because this increases the Oswald efficiency factor and consequently the aerodynamic efficiency. The twist was increased both at the kink and at the tip, increasing the local angle of attack. The inboard span of the wing was decreased, but the outboard span was increased. Overall, the optimizer clearly tried to make the lift distribution more elliptical, judging from fig. 4.6b, and moved the generation of lift further outboard. In a high-fidelity optimization, however, it is expected that the wing span would increase, but previous design studies using a VLM have also yielded a short, increased taper ratio wing [20]. In table 4.6 it can be seen that the optimizer Did not move the the spars by much. The front spar was moved aft slightly, whereas the rear spar was left nearly identical. The thicknesses of the outboard front spar and rear spar were increased; the thicknesses of all skins were decreased. However, as was mentioned before, most constraints




(a) Wing geometry (b) Lift distributions

Figure 4.6: Wing geometry and lift distributions of the initial and final wings

Table 4.6: Structural design variables. Note: The second to last row corresponds to the initial values, the last to the final values.

| $\xi_{\mathrm{fs_r}}$ | $\xi_{\mathrm{fs_k}}$ | $\xi_{\mathrm{fs_t}}$ | $\xi_{\mathrm{rs_r}}$ | $\xi_{\mathrm{rs_k}}$ | $\xi_{\mathrm{rs_t}}$ | $t_{\mathrm{fs_{ib}}}$ | $t_{\mathrm{fs_{ob}}}$ | $t_{\mathrm{rs_{ib}}}$ | $t_{\mathrm{rs_{ob}}}$ | $t_{\mathrm{ts_{ib}}}$ | $t_{\mathrm{ts_{ob}}}$ | $t_{\mathrm{bs_{ib}}}$ | $t_{\mathrm{bs_{ob}}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | mm | mm | mm | mm | mm | mm | mm | mm |
| 0.10 | 0.19 | 0.35 | 0.60 | 0.80 | 0.60 | 4.5 | 4.6 | 4.5 | 4.6 | 25.5 | 22.4 | 25.5 | 22.4 |
| 0.17 | 0.20 | 0.32 | 0.62 | 0.80 | 0.60 | 4.5 | 5.4 | 4.5 | 5.9 | 19.3 | 20.8 | 19.3 | 20.8 |

were violated. This includes almost all the stress constraints. Therefore these thicknesses are in no way sufficient to support this design. In table 4.7, a set of performance characteristics are listed for the initial and final design point. As can be seen, both the fuel and wing mass have been reduced significantly. The lift coefficient during cruise is reduced slightly due to the reduced total weight. There is a very slight decrease in the friction drag estimate, but since this parameter is determined using a very crude estimation method, this can be disregarded. The real improvement comes from the induced drag coefficient, which the VLM is able to predict well. It was reduced by 28.3%, and the aerodynamic efficiency during cruise was increased by 5.2%. Despite the smaller wing, the wing loading was reduced by 4.2%. This is due to the reduction in takeoff weight, caused by the reductions in fuel and wing weight.

Finally, the values of the constraints are listed in table 4.8. As can be seen, the front spar stress, rear spar stress, and wing loading constraints are all inactive and satisfied. However, the top and bottom skin stress constraints are significantly violated. Hence, the final design is clearly infeasible indeed.

Table 4.7: Performance characteristics

| **symbols** | $m_{\mathrm{fuel}}$ | $m_{\mathrm{wing}}$ | $C_{\mathrm{L}}$ | $C_{\mathrm{D_f}}$ | $C_{\mathrm{D_i}}$ | $C_{\mathrm{D}}$ | $L/D$ | $W/S$ |
|---|---|---|---|---|---|---|---|---|
| **units** | kg | kg | – | – | – | – | – | $\mathrm{kg\,m^{-2}}$ |
| **initial** | 108 508 | 49 591 | 0.4229 | 0.0048 | 0.0060 | 0.0218 | 19.4 | 537.7 |
| **final** | 96 758 | 37 603 | 0.4088 | 0.0047 | 0.0043 | 0.0200 | 20.4 | 515.1 |

Table 4.8: Constraint values

| **symbols** | $\mathrm{con}_{\sigma_{\mathrm{fs}}}$ | | $\mathrm{con}_{\sigma_{\mathrm{rs}}}$ | | $\mathrm{con}_{\sigma_{\mathrm{ts}}}$ | | $\mathrm{con}_{\sigma_{\mathrm{bs}}}$ | | $\mathrm{con}_{W/S}$ |
|---|---|---|---|---|---|---|---|---|---|
| **segment** | i.b. | o.b. | i.b. | o.b. | i.b. | o.b. | i.b. | o.b. | – |
| **initial** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **final** | −0.07 | −0.26 | −0.07 | −0.32 | 0.64 | 0.56 | 0.64 | 0.56 | −0.04 |

# 5

# Conclusions & Recommendations

We have come to the end of the main content of this thesis report. At this point it is worthwhile to reflect back on the report and on the work done, and draw conclusions from it. Section 5.1 is dedicated to this.

Of course, research is never done. So too this work is only a lap in the much longer race. Advances have been made, questions have been answered, but there are also open ends to this work. These provide provide opportunities for future research. Therefore it is important to reflect on this shortly. This will be done in the form of a brief set of recommendations, presented in section 5.2.

## 5.1. Conclusions

After having studied the field of MDAO extensively for the 9 months of this thesis, the author has to conclude that the maturity of MDAO is not industry ready yet as this point. As was discussed in chapter 1, the problems lie in both technical and non-technical difficulties and barriers. However, as became apparent when investigating the literature of this field, as presented in chapter 2, a lot of research has been and is being done to improve this. Projects like AGILE are helping to evolve MDO towards a point where the industry can no longer ignore it. Hence, it can be concluded that this field has the potential to reach an adequate level of maturity to entertain the idea of widespread industrial implementation in the very near future.

As was discussed in chapters 1 and 2, the AGILE project focuses mainly on moving from 2nd to 3rd generation MDO frameworks. These frameworks rely heavily on supporting physically separated teams of experts, working together on an integrated multidisciplinary design study. AGILE's role as an enabler of this move to distributed systems, where human engineers are being put back into the loop, involves the development and dissemination of new tools and protocols. Still, from experience gained in the industry over the past months, the author has to conclude that the industry is actually not even fully convinced of 1st generation MDO at this point. The first generation is, of course, the foundation upon which the technology is built. Therefore it is important that the technology is matured at its base as well.

It was found that many of the tools developed within AGILE and other projects could help achieve this. This is a powerful notion: methods developed for higher level MDO systems can help evolve lower level systems as well! This conclusion lead to the development of a new software framework, which consequently formed the last link in a end-to-end 1st generation MDO 'pipeline'. The concept of this pipeline was inspired upon the 3rd generation tool-chain developed within AGILE, which takes a database of analysis tools through several protocols and software frameworks to yield a runnable optimization workflow. The difference between this pipeline as it is meant within AGILE and the one developed in this thesis, is that the first is geared towards distributed, collaborative frameworks, whereas the latter considers only local, monolithic frameworks.

At this point, it is important to reflect back on the research questions posited in chapter 1. The main question to be answered was:

How can MDAO be made more applicable for conceptual design in the industry?

The answer should now be clear:

> By maturing its foundation, that is, by evolving the state-of-the-art of 1st generation MDAO frameworks.

The author found that the extensible, open-source frameworks are most suitable and wanted by the industry. So it was concluded that, regardless of its form, for any new framework to be a move in the right direction it should be extensible and open-source. Furthermore, from previous research it was concluded that it is important that any new framework would allow the user to rapidly (re)configure MDAO problems. And finally, as was proofed experimentally in section 4.1, it was concluded that the inclusion of gradient information is key. To achieve these goals, a new software framework was developed, adding to the tools coming from the AGILE project, and the OpenMDAO optimization framework. The strategy, architecture, and implementation of this new pipeline and software framework were discussed in chapter 3.

From the investigation into the effect of including analytical gradient information, presented in section 4.1, it has to be concluded that this is indeed essential to the successful industrialization of an MDAO system, as was posited in chapter 1. Using gradient information allowed for dramatic reductions in computational costs for a wide range of different problems. In the next two section, sections 4.2 and 4.3, the application of the new framework was demonstrated. From these results, it can be concluded that the high level requirements put forth in chapter 1 and the detailed software requirements listed in section 3.1, were successfully translated into a functioning software system. Section 4.2 showcased the software's ability to allow for an optimization problem to be easily (re)configured. This comes cutesy of the interface with the KADMOS/CMDOWS tools from AGILE, and the seamless integration with the OpenMDAO framework. After having completed the KADMOS link of the chain, an OpenMDAO model could be constructed with a single line of code.

The new framework was also used successfully to construct an OpenMDAO problem representing an aeroelastic wing optimization use case, as was discussed in section 4.3. The dAEDalus framework for aeroelastic analysis was integrated into this framework. From this it can be concluded that the new tool can be used for industry relevant problems too. Unfortunately no feasible results could be presented from this optimization. However, the author would like to stress that this was never the main goal here. What was demonstrated with the problem, is that it is possible to configure an optimization relevant to aircraft design using the new pipeline. Regardless of the outcomes of the optimization run, it has to be concluded that this was indeed done successfully.

## 5.2. Recommendations

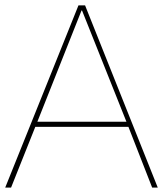
As was said at the start of this chapter, the author recognizes his role as only a part of a much lager process. Based on the results presented in this report, it can be said the work done during the nine months of this thesis have helped evolve the field of MDO a step closer towards widespread industrialization. As such, important questions have been answered. However, the answering of questions rarely does not lead to new questions.

One of these questions is the following: could the extension to the notion of the knowledge base proposed in this thesis be better integrated into the AGILE pipeline? As it stands now, the extension to add support for analytical gradient specification exists completely separated from the KADMOS/CMDOWS definition of the knowledge base. Ideally, this idea should be integrated deeply within the definition of KADMOS and the structure of CMDOWS. If this were done, the extended notion of the knowledge base could even be adopted in the 3rd generation MDO tool-chain AGILE provides. As was seen in section 4.1, this could have dramatically positive effects on the computational cost of optimizations. Therefore this could help the AGILE project attain, or even surpass its goal of a 40% reduction in the time required to converge a large-scale optimization problem. It is therefore strongly recommended this be investigated in the near future.

The author is aware of a more elaborate aircraft design problem, using a much larger suite of analysis tools, described within the AGILE project. To investigate the true potential of the pipeline described and software developed during this thesis, it is recommended that this much larger problem be implemented and run with these new tools. Not only should this demonstrate whether the tools developed are truly scalable, but it should also serve as an import showcase of the powers of modern MDO systems. This could move the industry a step closer to true acceptance of MDO in their processes.

As was mentioned in sections 4.3 and 5.1, the wing optimization test case performed considered in this work did not yield a feasible solution. It goes without saying that the author recommends that this problem is adjusted and properly seen to conclusion if resources are available for this.

Last, but certainly not least, the author recommends possible additions to the software framework OpenLEGO, developed as part of this thesis. In this thesis, the main function of OpenLEGO is to connect a 1st generation, monolithic version of the full 3rd generation MDO pipeline developed within AGILE to the OpenMDAO framework. As was discussed in chapters 1 and 2, OpenMDAO has advantages over PIDO systems such as RCE and Optimus. However, as it stands now it has one major disadvantage: it cannot make use of AGILE's full suite of tools enabling 3rd generation MDO, such as the BRICS protocol. It would be interesting to investigate if OpenLEGO could be extended to bring it from the 1st generation MDO domain into the 3rd generation MDO domain that AGILE targets. Enabling BRICS capabilities is seen by the author as an important step for this, because it would open up the realm of collaborative optimization within OpenMDAO. The author would like to make two suggestions for making this possible. Firstly, a special OpenMDAO component could be developed which allows for BRICS analyses to be easily included in models. The author is aware of the remote component provisions already existent in OpenMDAO currently, so it should be possible to target BRICS too. Secondly, an interface between OpenMDAO and the PIDO systems RCE and Optimus could be developed. This would give users the freedom to use the powers of the graphical user interfaces of the PIDO systems to create workflows of cooperating tools and engineer, while allowing them to tap into the strengths of the OpenMDAO framework. This would effectively also enable the link between OpenMDAO and the BRICS protocol, because the latter is already available in these PIDO systems.



# Code

## A.1. OpenLEGO core

### A.1.1. openlego.core.abstract_discipline

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definition of the AbstractDiscipline interface class.
"""
from __future__ import absolute_import, division, print_function

import abc
import inspect
import json
import os

from openlego.partials.partials import Partials


class AbstractDiscipline(object):
    """Defines the common interface for all disciplines within ''OpenLEGO''."""
    __metaclass__ = abc.ABCMeta

    @property
    def name(self):
        # type: () -> str
        """:obj:'str': Name of this discipline."""
        return self.__class__.__name__

    @property
    def version(self):
        # type: () -> float
        """:obj:'str': Version number of this discipline."""
        return 1.0

    @property
    def creator(self):
```

```
48          # type: () -> str
49          """:obj:'str': Name of the person that created this discipline."""
50          return 'Placeholder'
51
52      @property
53      def description(self):
54          # type: () -> str
55          """:obj:'str': Description of this discipline."""
56          return 'Abstract discipline'
57
58      @property
59      def precision(self):
60          # type: () -> int
61          """:obj:'int': Precision of this discipline."""
62          return 0
63
64      @property
65      def path(self):
66          # type: () -> str
67          """:obj:'str': Path at which this discipline resides."""
68          return os.path.dirname(inspect.getfile(self.__class__))
69
70      @property
71      def in_file(self):
72          # type: () -> str
73          """:obj:'str': Path of the template input XML file of this discipline."""
74          return os.path.join(self.path, self.name + '-input.xml')
75
76      @property
77      def out_file(self):
78          # type: () -> str
79          """:obj:'str': Path of the template output XML file of this discipline."""
80          return os.path.join(self.path, self.name + '-output.xml')
81
82      @property
83      def json_file(self):
84          # type: () -> str
85          """:obj:'str': Path of the information JSON file of this discipline."""
86          return os.path.join(self.path, self.name + '-info.json')
87
88      @property
89      def partials_file(self):
90          # type: () -> str
91          """:obj:'str': Path of the partials XML file of this discipline."""
92          return os.path.join(self.path, self.name + '-partials.xml')
93
94      @property
95      def supplies_partials(self):
96          # type: () -> bool
97          """Set to True to indicate this discipline supplies gradients."""
98          return False
99
100     @abc.abstractmethod
101     def generate_input_xml(self):
102         # type: () -> str
103         """Generate the template input XML for this discipline.
104
105         This method should be implemented to define the input template of a specific
    ↪  discipline.
106
107         Returns
108         -------
109             str
110                 String representation of the template input XML.
111         """
112         raise NotImplementedError
113
114     @abc.abstractmethod
115     def generate_output_xml(self):
116         # type: () -> str
117         """Generate the template output XML for this discipline.
```

```
118
119         This method should be implemented to define the output template of a specific discipline.
120
121         Returns
122         -------
123             str
124                 String representation of the template output XML file.
125         """
126         raise NotImplementedError
127
128     def generate_info_json(self):
129         # type: () -> str
130         """Generate the information JSON file for this discipline.
131
132         This method should be overridden or extended to specify a non-standard info JSON file for a specific discipline.
133
134         Returns
135         -------
136             str
137                 String representation of the info JSON file
138         """
139         return json.dumps({'general_info': {'name': self.name,
140                                             'version': self.version,
141                                             'creator': self.creator,
142                                             'description': self.description},
143                            'execution_info': [{'mode': 'main',
144                                                'description': 'main execution mode',
145                                                'precision': self.precision}]}, indent=4)
146
147     def generate_partials_xml(self):
148         # type: () -> str
149         """Generate the template partials XML file for this discipline.
150
151         This method should be implemented to define for which inputs this discipline can provide the sensitivities.
152
153         Returns
154         -------
155             str
156                 String representation of the template partials XML file.
157         """
158         return Partials().get_string()
159
160     def deploy(self):
161         # type: () -> None
162         """Deploy this discipline's template in-/output, partials XML files and its information JSON file."""
163         with open(self.in_file, 'w') as f:
164             f.write(self.generate_input_xml())
165         with open(self.out_file, 'w') as f:
166             f.write(self.generate_output_xml())
167         with open(self.json_file, 'w') as f:
168             f.write(self.generate_info_json())
169         with open(self.partials_file, 'w') as f:
170             f.write(self.generate_partials_xml())
171
172     @staticmethod
173     @abc.abstractmethod
174     def execute(in_file, out_file):
175         # type: (str, str) -> None
176         """Execute this discipline with the given in- and output XML files.
177
178         This method should be implemented to define the execution of a specific discipline.
179
180         Parameters
181         ----------
182             in_file : str
183                 Path to the input XML file.
184
```

```
185             out_file : str
186                 Path to the output XML file.
187         """
188         raise NotImplementedError
189 
190     @staticmethod
191     def linearize(in_file, partials_file):
192         # type: (str, str) -> None
193         """Compute the sensitivities of a given input XML file and write them to a given
    ↪  partials XML file.
194 
195         This method should be implemented to define the linearization of a specific
    ↪  discipline. By default a discipline
196         is considered a 'black box', and no sensitivities are provided.
197 
198         Parameters
199         ----------
200             in_file : str
201                 Path to the input XML file.
202 
203             partials_file : str
204                 Path to the sensitivities XML file.
205         """
206         Partials().write(partials_file)
```

Code frament A.1: Code of the `openlego.core.abstract_discipline` Python module.

### A.1.2. `openlego.core.discipline_component`

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5 
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9 
10     http://www.apache.org/licenses/LICENSE-2.0
11 
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17 
18 This file contains the definition the 'DisciplineComponent' class.
19 """
20 from __future__ import absolute_import, division, print_function
21 
22 from typing import Optional
23 
24 from .abstract_discipline import AbstractDiscipline
25 from .xml_component import XMLComponent
26 
27 
28 class DisciplineComponent(XMLComponent):
29     """Specialized 'XMLComponent' wrapping an 'AbstractDiscipline'.
30 
31     This version of 'XMLComponent' defines in- and output variables based on the in- and
    ↪  output template XML files
32     generated by a subclass of 'AbstractDiscipline'. The 'execute()' method simply forwards to
    ↪  that of the discipline.
33 
34     Attributes
35     ----------
36         discipline
37     """
38 
39     def __init__(self, discipline, data_folder='', keep_files=False, base_file=None):
```

```
40          # type: (AbstractDiscipline, Optional[str]) -> None
41          """Initialize a 'Component' using a given 'discipline'.
42
43          Stores a reference to the given 'discipline'. The in- and output XML templates should
    ↪ already exist at the paths
44          specified in the 'discipline'. This constructor uses those files to create the
    ↪ ''OpenMDAO'' 'params' and
45          'unknowns' using the methods exposed by the 'XMLComponent' class this class inherits
    ↪ from.
46
47          Parameters
48          ----------
49              discipline : :obj:'AbstractDiscipline'
50                  Instance of a subclass of 'AbstractDiscipline' this 'Component' will represent.
51
52              data_folder : str(''), optional
53                  Path to a folder in which to store (temporary) data of this 'Component' during
    ↪ execution.
54
55              keep_files : bool(False), optional
56                  Set to 'True' to keep the data files generated by this 'Component' during
    ↪ execution.
57
58              base_file : str, optional
59                  Path to an XML file which should be kept up-to-date with the latest data, if
    ↪ required.
60
61          Notes
62          -----
63              Although this constructor could use the supplied 'discipline' to also
    ↪ automatically generate its in- and
64              output XML templates on the fly, the user is left in control of their generation.
    ↪ This is to allow for a
65              'discipline' to generate different in- and output templates dynamically based on
    ↪ certain parameters. During
66              execution only the static methods of the 'discipline's are used. Hence, any
    ↪ instance variables will not be
67              accessible then. Therefore it is impossible to guarantee consistency if the in-
    ↪ and output XML files are
68              generated here.
69          """
70          self._discipline = discipline
71          if discipline.supplies_partials:
72              super(DisciplineComponent, self).__init__(self._discipline.in_file,
73                                                        self._discipline.out_file,
74                                                        self._discipline.partials_file,
75                                                        data_folder, keep_files, base_file)
76          else:
77              super(DisciplineComponent, self).__init__(self._discipline.in_file,
78                                                        self._discipline.out_file,
79                                                        None,
80                                                        data_folder, keep_files, base_file)
81              self.partials_from_xml = None
82
83      @property
84      def discipline(self):
85          # type: () -> AbstractDiscipline
86          """:obj:'AbstractDiscipline': Read-only reference to the specific discipline this
    ↪ 'Component' wraps."""
87          return self._discipline
88
89      def execute(self, input_xml=None, output_xml=None):
90          # type: (str, str) -> None
91          """Call the 'execute()' method of this 'Component''s discipline.
92
93          Parameters
94          ----------
95              input_xml : str
96                  Path to the input XML file.
97
98              output_xml : str
```

```
99                 Path to the output XML file.
100
101         Raises
102         ------
103             ValueError
104                 If either no 'input_xml' or 'output_xml' path was specified.
105
106         Notes
107         -----
108             Since this class inherits from 'XMLComponent' the interface, including the
    ↪ optionality of its arguments, are
109             left untouched. For this method this means the 'input_xml' and 'output_xml'
    ↪ parameters are strictly
110             optional. However, in the context of the 'DisciplineComponent' they should always
    ↪ be given. Therefore an
111             exception is raised here when one of them or both are omitted.
112         """
113         if input_xml is None or output_xml is None:
114             raise ValueError('Both an input_xml and output_xml path are expected.')
115         self.discipline.execute(input_xml, output_xml)
116
117     def linearize(self, input_xml=None, partials_xml=None):
118         # type: (str, str) -> None
119         """Call the 'linearize()' method of this 'Component''s discipline.
120
121         Parameters
122         ----------
123             input_xml : str
124                 Path to the input XML file.
125
126             partials_xml : str
127                 Path to the partials XML file.
128
129         Raises
130         ------
131             ValueError
132                 If either no 'input_xml' or 'partials_xml' path was specified.
133         """
134         if self.discipline.supplies_partials:
135             if input_xml is None or partials_xml is None:
136                 raise ValueError('Both an input_xml and a partials_xml path are expected.')
137             self.discipline.linearize(input_xml, partials_xml)
```

Code frament A.2: Code of the `openlego.core.discipline_component` Python module.

### A.1.3. `openlego.core.model`

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the 'LEGOModel' class.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  import imp
23  import re
```

```
24  import warnings
25
26  import numpy as np
27  from lxml import etree
28  from lxml.etree import _Element, _ElementTree
29  from openmdao.api import Group, IndepVarComp, LinearBlockGS, NonlinearBlockGS,
      ↪ LinearBlockJac, NonlinearBlockJac, \
30      LinearRunOnce, NonLinearRunOnce, ExecComp
31  from typing import Union, Optional, List, Any, Dict, Tuple
32
33  from openlego.utils.general_utils import CachedProperty, parse_cmdows_value
34  from openlego.utils.xml_utils import xpath_to_param, xml_to_dict
35  from .abstract_discipline import AbstractDiscipline
36  from .discipline_component import DisciplineComponent
37
38  re_sys_name_char = re.compile(r'[^_a-zA-Z0-9]')
39  re_sys_name_starts = re.compile(r'^[a-zA-Z]')
40
41
42  class LEGOModel(Group):
43      """Specialized OpenMDAO Group class representing the problem specified by a CMDOWS file.
44
45      An important note about this class in the context of OpenMDAO is that the aggregation
      ↪  pattern of the root Group
46      class the base Problem class has is changed into a stronger composition pattern. This is
      ↪  because this class directly
47      controls the creation and assembly of this class by making use of Python's @property
      ↪  decorator. It is not possible,
48      nor should it be attempted, to manually inject a different instance of Group  in place of
      ↪  these, because the
49      correspondence between the CMDOWS file and the Problem can then no longer be guaranteed.
50
51      Attributes
52      ----------
53          cmdows_path
54          kb_path
55          discipline_components
56          block_order
57          coupled_blocks
58          system_order
59          system_variables
60          system_inputs
61          driver
62          coordinator
63
64          data_folder : str, optional
65              Path to the folder in which to store all data generated during the 'Problem''s
      ↪  execution.
66
67          base_xml_file : str, optional
68              Path to an XML file which should be kept up-to-date with the latest data
      ↪  describing the problem.
69      """
70
71      def __init__(self, cmdows_path=None, kb_path=None, data_folder=None, base_xml_file=None,
      ↪  **kwargs):
72          # type: (Optional[str], Optional[str], Optional[str], Optional[str]) -> None
73          """Initialize a CMDOWS Problem from a given CMDOWS file and knowledge base.
74
75          It is also possible to specify where (temporary) data should be stored, and if a base
      ↪  XML
76          file should be kept up-to-data.
77
78          Parameters
79          ----------
80          cmdows_path : str, optional
81              Path to the CMDOWS file.
82
83          kb_path : str, optional
84              Path to the knowledge base.
85
```

```
86          data_folder : str, optional
87              Path to the data folder in which to store all files and output from the problem.
88
89          base_xml_file : str, optional
90              Path to a base XML file to update with the problem data.
91          """
92          self._cmdows_path = cmdows_path
93          self._kb_path = kb_path
94          self.data_folder = data_folder
95          self.base_xml_file = base_xml_file
96
97          super(LEGOModel, self).__init__(**kwargs)
98          self.linear_solver = LinearRunOnce()
99          self.nonlinear_solver = NonLinearRunOnce()
100
101     def __getattribute__(self, name):
102         # type: (str) -> Any
103         """Check the integrity before returning any of the cached variables.
104
105         Parameters
106         ----------
107         name : str
108             Name of the attribute to read.
109
110         Returns
111         -------
112             any
113                 The value of the requested attribute.
114         """
115         if name != '__class__' and name != '__dict__':
116             if name in [_name for _name, value in self.__class__.__dict__.items() if
    ↪  isinstance(value, CachedProperty)]:
117                 self.__integrity_check()
118         return super(LEGOModel, self).__getattribute__(name)
119
120     def __setattr__(self, name, value):
121         # type: (str, Any) -> None
122         """Bypass setting coordinator and coupled_group attributes.
123
124         Parameters
125         ----------
126             name : str
127                 Name of the attribute.
128
129             value : any
130                 Value to set the attribute to.
131         """
132         if name not in ['coordinator', 'coupled_group']:
133             super(LEGOModel, self).__setattr__(name, value)
134
135     def __integrity_check(self):
136         # type: () -> None
137         """Ensure both a CMDOWS file and a knowledge base path have been supplied.
138
139         Raises
140         ------
141             ValueError
142                 If either no CMDOWS file or no knowledge base path has been supplied
143         """
144         a = self._cmdows_path is None
145         b = self._kb_path is None
146         if a or b:
147             raise ValueError('No ' + a * 'CMDOWS file ' + (a & b) * 'and ' + b * 'knowledge
    ↪  base path ' + 'specified!')
148
149     def invalidate(self):
150         # type: () -> None
151         """Invalidate the instance.
152
153         All computed (cached) properties will be recomputed upon being read once the instance
    ↪  has been invalidated."""
```

```
154             for value in self.__class__.__dict__.values():
155                 if isinstance(value, CachedProperty):
156                     value.invalidate()
157 
158     def does_value_fit(self, name, val):
159         # type: (str, Union[str, float, np.ndarray]) -> bool
160         """Check whether a given value has the correct size to be assigned to a given variable.
161 
162             Parameters
163             ----------
164                 name : str
165                     Name of the variable.
166 
167                 val : str or float or np.ndarray
168                     Value to check.
169 
170             Returns
171             -------
172                 bool
173                     'True' if the value fits, 'False' if not.
174             """
175         return (isinstance(val, np.ndarray) and val.size == self.variable_sizes[name]) \
176             or (not isinstance(val, np.ndarray) and self.variable_sizes[name] == 1)
177 
178     @property
179     def cmdows_path(self):
180         # type: () -> str
181         """:obj:'str': Path to the CMDOWS file this class corresponds to.
182 
183             When this property is set the instance is automatically invalidated.
184             """
185         return self._cmdows_path
186 
187     @cmdows_path.setter
188     def cmdows_path(self, cmdows_path):
189         # type: (str) -> None
190         self._cmdows_path = cmdows_path
191         self.invalidate()
192 
193     @property
194     def kb_path(self):
195         # type: () -> str
196         """:obj:'str': Path to the knowledge base.
197 
198             When this property is set the instance is automatically invalidated.
199             """
200         return self._kb_path
201 
202     @kb_path.setter
203     def kb_path(self, kb_path):
204         # type: (str) -> None
205         self._kb_path = kb_path
206         self.invalidate()
207 
208     @CachedProperty
209     def elem_cmdows(self):
210         # type: () -> _Element
211         """:obj:'etree._Element': Root element of the CMDOWS XML file."""
212         return etree.parse(self.cmdows_path).getroot()
213 
214     @CachedProperty
215     def elem_problem_def(self):
216         # type: () -> _Element
217         """:obj:'etree._Element': The problemDefition element of this problem's CMDOWS file."""
218         return self.elem_cmdows.find('problemDefinition')
219 
220     @CachedProperty
221     def elem_params(self):
222         # type: () -> _Element
223         """:obj:'etree._Element': The problemRoles/parameters element of the CMDOWS file."""
224         params = self.elem_cmdows.find('problemDefinition/problemRoles/parameters')
```

```
225         if params is None:
226             raise Exception('cmdows does not contain (valid) parameters in the problemRoles')
227         return params
228 
229     @CachedProperty
230     def elem_arch_elems(self):
231         # type: () -> _Element
232         """:obj:`etree._Element`: The architectureElements element of the CMDOWS file."""
233         arch_elems = self.elem_cmdows.find('architectureElements')
234         if arch_elems is None:
235             raise Exception('cmdows does not contain (valid) architecture elements')
236         return arch_elems
237 
238     @CachedProperty
239     def has_converger(self):
240         # type: () -> bool
241         """:obj:`bool`: True if there is a converger, False if not."""
242         if self.elem_arch_elems.find('executableBlocks/convergers/converger') is not None:
243             return True
244         return False
245 
246     @CachedProperty
247     def discipline_components(self):
248         # type: () -> Dict[str, DisciplineComponent]
249         """:obj:`dict`: Dictionary of discipline components by their design competence ``uID``
    ↪  from CMDOWS.
250 
251         Raises
252         ------
253             RuntimeError
254                 If a ``designCompetence`` specified in the CMDOWS file does not correspond to
    ↪  an `AbstractDiscipline`.
255         """
256         _discipline_components = dict()
257         for design_competence in self.elem_cmdows.iter('designCompetence'):
258             uid = design_competence.attrib['uID']
259             name = design_competence.find('ID').text
260             try:
261                 fp, pathname, description = imp.find_module(name, [self.kb_path])
262                 mod = imp.load_module(name, fp, pathname, description)
263                 cls = getattr(mod, name)  # type: AbstractDiscipline.__class__
264                 if not issubclass(cls, AbstractDiscipline):
265                     raise RuntimeError
266             except Exception:
267                 raise RuntimeError(
268                     'Unable to process CMDOWS file: no proper discipline found for design
    ↪  competence with name %s'
269                     % name)
270             finally:
271                 if 'fp' in locals():
272                     fp.close()
273 
274             component = DisciplineComponent(cls(), data_folder=self.data_folder,
    ↪  base_file=self.base_xml_file)
275             _discipline_components.update({uid: component})
276         return _discipline_components
277 
278     @CachedProperty
279     def variable_sizes(self):
280         # type: () -> Dict[str, int]
281         """:obj:`dict`: Dictionary of the sizes of all variables by their names."""
282         variable_sizes = {}
283         for component in self.discipline_components.values():
284             for name, value in component.variables_from_xml.items():
285                 variable_sizes.update({name: np.atleast_1d(value).size})
286         return variable_sizes
287 
288     @CachedProperty
289     def coupling_vars(self):
290         # type: () -> Dict[str, Dict[str, str]]
291         """:obj:`dict`: Dictionary with coupling variables."""
```

```
292         coupling_vars = dict()
293 
294         # First create a map between related param and coupling copy var
295         for var in self.elem_arch_elems.iter('couplingCopyVariable'):
296             related_param = var.find('relatedParameterUID').text
297             coupling_vars.update({xpath_to_param(related_param):
    ↳ xpath_to_param(var.attrib['uID'])})
298 
299         # Then update dict with corresponding consitency constraint var
300         for convar in self.elem_arch_elems.iter('consistencyConstraintVariable'):
301             param = xpath_to_param(convar.find('relatedParameterUID').text)
302             if param not in coupling_vars:
303                 raise RuntimeError('invalid cmdows file')
304 
305             coupling_vars.update({param: {'copy': coupling_vars[param], 'con':
    ↳ xpath_to_param(convar.attrib['uID'])}})
306         return coupling_vars
307 
308     @CachedProperty
309     def coupling_var_copies(self):
310         # type: () -> Dict[str, str]
311         """:obj:'dict': Dictionary with coupling variable copies."""
312         coupling_var_copies = dict()
313         for var, value in self.coupling_vars.items():
314             coupling_var_copies.update({var: value['copy']})
315         return coupling_var_copies
316 
317     @CachedProperty
318     def coupling_var_cons(self):
319         # type: () -> Dict[str, str]
320         """:obj:'dict': Dictionary with coupling variable constraints."""
321         coupling_var_cons = None
322         if 'con' in self.coupling_vars.values()[0]:
323             coupling_var_cons = dict()
324             for var, value in self.coupling_vars.items():
325                 coupling_var_cons.update({var: value['con']})
326         return coupling_var_cons
327 
328     @CachedProperty
329     def block_order(self):
330         # type: () -> List[str]
331         """:obj:'list' of :obj:'str': List of executable block ''uIDs'' in the order specified
    ↳ in the CMDOWS file."""
332         positions = list()
333         uids = list()
334         for block in
    ↳ self.elem_problem_def.iterfind('problemFormulation/executableBlocksOrder/executableBlock'):
335             uid = block.text
336             positions.append(int(block.attrib['position']))
337             uids.append(uid)
338         return [uid for position, uid in sorted(zip(positions, uids))]
339 
340     @CachedProperty
341     def coupled_blocks(self):
342         # type: () -> List[str]
343         """:obj:'list' of :obj:'str': List of ''uIDs'' of the coupled executable blocks
    ↳ specified in the CMDOWS file."""
344         _coupled_blocks = []
345         for block in
    ↳ self.elem_problem_def.iterfind('problemRoles/executableBlocks/coupledBlocks/coupledBlock'):
346             _coupled_blocks.append(block.text)
347         return _coupled_blocks
348 
349     @CachedProperty
350     def system_inputs(self):
351         # type: () -> Dict[str, int]
352         """:obj:'dict': Dictionary containing the system input sizes by their names."""
353         system_inputs = {}
354         for value in self.elem_cmdows.xpath(
355 
    ↳ r'workflow/dataGraph/edges/edge[fromExecutableBlockUID="Coordinator"]/toParameterUID/text()'):
```

```
356             if 'architectureNodes' not in value or 'designVariables' in value:
357                 name = xpath_to_param(value)
358                 system_inputs.update({name: self.variable_sizes[name]})
359 
360         return system_inputs
361 
362     @CachedProperty
363     def design_vars(self):
364         # type: () -> Dict[str, Dict[str, Any]]
365         """:obj:`dict`: Dictionary containing the design variables' initial values, lower
    ↪ bounds, and upper bounds."""
366         desvars = self.elem_params.find('designVariables')
367         if desvars is None:
368             raise Exception('cmdows does not contain (valid) design variables')
369 
370         design_vars = {}
371         for desvar in desvars:
372             name = xpath_to_param(desvar.find('parameterUID').text)
373 
374             # Obtain the initial value
375             initial = desvar.find('nominalValue')
376             if initial is not None:
377                 initial = parse_cmdows_value(initial)
378                 if not self.does_value_fit(name, initial):
379                     raise ValueError('incompatible size of nominalValue for design variable
    ↪ "%s"' % name)
380             else:
381                 warnings.warn('no nominalValue given for designVariable "%s". Default is all
    ↪ zeros.' % name)
382                 initial = np.zeros(self.variable_sizes[name])
383 
384             if name in self.coupling_vars:
385                 # If this is a coupling variable the bounds are -1e99 and 1e99 and it should
    ↪ not be normalized
386                 design_vars.update(
387                     {self.coupling_vars[name]['copy']: {'initial': initial,
388                                                         'lower':
    ↪ -1e99*np.ones(self.variable_sizes[name]),
389                                                         'upper':
    ↪ 1e99*np.ones(self.variable_sizes[name]),
390                                                         'ref0': None, 'ref': None}})
391             else:
392                 # Obtain the lower and upper bounds
393                 bounds = 2 * [None]  # type: List[Optional[str]]
394                 limit_range = desvar.find('validRanges/limitRange')
395                 if limit_range is not None:
396                     for index, bnd, in enumerate(['minimum', 'maximum']):
397                         elem = limit_range.find(bnd)
398                         if elem is not None:
399                             bounds[index] = parse_cmdows_value(elem)
400                             if not self.does_value_fit(name, bounds[index]):
401                                 raise ValueError('incompatible size of %s for design variable
    ↪ %s' % (bnd, name))
402 
403                 # Add the design variable to the dict
404                 design_vars.update({name: {'initial': initial,
405                                            'lower': bounds[0], 'upper': bounds[1],
406                                            'ref0': bounds[0], 'ref': bounds[1]}})
407         return design_vars
408 
409     @CachedProperty
410     def constraints(self):
411         # type: () -> Dict[str, Dict[str, Any]]
412         """:obj:`dict`: Dictionary containing the constraints' lower, upper, and equals
    ↪ reference values."""
413         convars = self.elem_params.find('constraintVariables')
414         constraints = {}
415         if convars is not None:
416             for convar in convars:
417                 con = {'lower': None, 'upper': None, 'equals': None}
418                 name = xpath_to_param(convar.find('parameterUID').text)
```

```
419
420                 if self.coupling_var_cons is not None and name in
    ↪ self.coupling_var_cons.values():
421                     # If this is a coupling variable consistency constraint, equals should just
    ↪ be zero
422                     for key, value in self.coupling_var_cons.items():
423                         if name == value:
424                             size = self.variable_sizes[key]
425                             if size == 1:
426                                 con['equals'] = 0.
427                             else:
428                                 con['equals'] = np.zeros(self.variable_sizes[key])
429                             break
430                 else:
431                     # Obtain the reference value of the constraint
432                     constr_ref = convar.find('referenceValue')  # type: etree._Element
433                     if constr_ref is not None:
434                         ref = parse_cmdows_value(constr_ref)
435                         if isinstance(ref, str):
436                             raise ValueError('referenceValue for constraint "%s" is not
    ↪ numerical' % name)
437                         elif not self.does_value_fit(name, ref):
438                             warnings.warn('incompatible size of constraint "%s". Will assume
    ↪ the same for all.' % name)
439                             ref = np.ones(self.variable_sizes[name]) * np.atleast_1d(ref)[0]
440                     else:
441                         warnings.warn('no referenceValue given for constraint "%s". Default is
    ↪ all zeros.' % name)
442                         ref = np.zeros(self.variable_sizes[name])
443
444                     # Process the constraint type
445                     constr_type = convar.find('constraintType')
446                     if constr_type is not None:
447                         if constr_type.text == 'inequality':
448                             constr_oper = convar.find('constraintOperator')
449                             if constr_oper is not None:
450                                 oper = constr_oper.text
451                                 if oper == '>=' or oper == '>':
452                                     con['lower'] = ref
453                                 elif oper == '<=' or oper == '<':
454                                     con['upper'] = ref
455                                 else:
456                                     raise ValueError('invalid constraintOperator "%s" for
    ↪ constraint "%s"' % (oper, name))
457                             else:
458                                 warnings.warn(
459                                     'no constraintOperator given for inequality constraint.
    ↪ Default is "&lt;=".')
460                                 con['upper'] = ref
461                         elif constr_type.text == 'equality':
462                             if convar.find('constraintOperator') is not None:
463                                 warnings.warn('constraintOperator given for an
    ↪ equalityConstraint will be ignored')
464                             con['equals'] = ref
465                         else:
466                            raise ValueError('invalid constraintType "%s" for constraint "%s".'
    ↪ % (constr_type.text, name))
467                     else:
468                        warnings.warn('no constraintType specified for constraint "%s". Default
    ↪ is a <= inequality.')
469                         con['upper'] = ref
470
471                 # Add constraint to the dictionary
472                 constraints.update({name: con})
473         return constraints
474
475     @CachedProperty
476     def objective(self):
477         # type: () -> str
478         """:obj:'str': Name of the objective variable."""
479         objvars = self.elem_params.find('objectiveVariables')
```

```
480         if objvars is None:
481             raise Exception('cmdows does not contain (valid) objective variables')
482         if len(objvars) > 1:
483             raise Exception('cmdows contains multiple objectives, but this is not supported')
484 
485         return xpath_to_param(objvars[0].find('parameterUID').text)
486 
487     @CachedProperty
488     def coupled_group(self):
489         # type: () -> Optional[Group]
490         """:obj:`Group`, optional: Group wrapping the coupled blocks with a converger
    ↪ specified in the CMDOWS file.
491 
492         If no coupled blocks are specified in the CMDOWS file this property is `None`.
493         """
494         if self.coupled_blocks:
495             coupled_group = Group()
496             for uid in self.coupled_blocks:
497                 # Get the correct DisciplineComponent
498                 discipline_component = self.discipline_components[uid]
499 
500                 # Change input variable names if they are provided as copies of coupling
    ↪ variables
501                 promotes = ['*']  # type: List[Union[str, Tuple[str, str]]]
502                 if not self.has_converger:
503                     for i in discipline_component.inputs_from_xml.keys():
504                         if i in self.coupling_vars:
505                             promotes.append((i, self.coupling_vars[i]['copy']))
506 
507                 # Add the DisciplineComponent to the group
508                 coupled_group.add_subsystem(uid, self.discipline_components[uid], promotes)
509 
510             # Find the convergence type of the coupled group
511             if self.has_converger:
512                conv_type = self.elem_problem_def.find('problemFormulation/convergerType').text
513                 if conv_type == 'Gauss-Seidel':
514                     coupled_group.linear_solver = LinearBlockGS()
515                     coupled_group.nonlinear_solver = NonlinearBlockGS()
516                 elif conv_type == 'Jacobi':
517                     coupled_group.linear_solver = LinearBlockJac()
518                     coupled_group.nonlinear_solver = NonlinearBlockJac()
519                 else:
520                     raise RuntimeError('Specified convergerType "%s" is not supported.' %
    ↪ conv_type)
521             else:
522                 coupled_group.linear_solver = LinearRunOnce()
523                 coupled_group.nonlinear_solver = NonLinearRunOnce()
524             return coupled_group
525         return None
526 
527     @CachedProperty
528     def consistency_constraint_group(self):
529         # type: () -> Optional[Group]
530         """:obj:`Group`, optional: Group containing ExecComps for the consistency
    ↪ constraints."""
531         elem_ccf =
    ↪ self.elem_arch_elems.find('executableBlocks/consistencyConstraintFunctions')
532         if elem_ccf is not None:
533             group = Group()
534 
535             # Loop over all consistencyConstraintFunction elements
536             for child in elem_ccf:
537                 uid = child.attrib['uID']
538                 xpaths = []
539 
540                 # Loop over all coupling variables which need to be constraint by this
    ↪ consistencyConstraintFunction
541                 for value in self.elem_cmdows.xpath(
542                         'workflow/data-
    ↪ Graph/edges/edge[toExecutableBlockUID="{}"]/fromParameterUID/text()'.format(uid)):
543                     # Only add a given variable once
```

```
544                     if 'architectureNodes' not in value and value not in xpaths:
545                         xpaths.append(value)
546
547                         name = xpath_to_param(value)
548                         size = self.variable_sizes[name]
549                         coupling_var = self.coupling_vars[name]
550
551                         if size == 1:
552                             val = 0.
553                         else:
554                             val = np.zeros(size)
555
556                         sys_name = re_sys_name_char.sub('', self.elem_arch_elems.xpath(
557                             'parameters/consistencyConstraintVariables/' +
558
    'consistencyConstraintVariable[@uID="{}"]/label/text()'.format(coupling_var['con']))[0])
559                         while not re_sys_name_starts.match(sys_name):
560                             sys_name = sys_name[1:]
561
562                         # Add an ExecComp to the Group for this equality constraint
563                         group.add_subsystem(
564                             sys_name,
565                             ExecComp('g = y_c - y', g=val, y_c=val, y=val),
566                             [('g', coupling_var['con']), ('y_c', coupling_var['copy']), ('y',
    name)])
567                 return group
568         return None
569
570     @CachedProperty
571     def system_order(self):
572         # type: () -> List[str]
573         """:obj:'list' of :obj:'str': List system names in the order specified in the CMDOWS
    file."""
574         _system_order = ['coordinator']
575         coupled_group_set = False
576         for block in self.block_order:
577             if block in self.coupled_blocks:
578                 if not coupled_group_set:
579                     _system_order.append('coupled_group')
580                     coupled_group_set = True
581             elif block in self.discipline_components:
582                 _system_order.append(block)
583         if self.consistency_constraint_group is not None:
584             _system_order.append('consistency_constraints')
585         return _system_order
586
587     @CachedProperty
588     def coordinator(self):
589         # type: () -> IndepVarComp
590         """:obj:'IndepVarComp': An 'IndepVarComp' representing the system's ''Coordinator''
    block.
591
592         This 'IndepVarComp' takes care of all system input parameters and initial values of
    design variables.
593         """
594         coordinator = IndepVarComp()
595
596         # Add design variables
597         for name, value in self.design_vars.items():
598             coordinator.add_output(name, value['initial'])
599
600         # Add system constants
601         for name, shape in self.system_inputs.items():
602             if name not in self.design_vars.keys():
603                 coordinator.add_output(name, shape=shape)
604
605         return coordinator
606
607     def setup(self):
608         # type: () -> None
609         """Assemble the LEGOModel using the the CMDOWS file and knowledge base."""
```

```
610             # Add the coordinator
611             self.add_subsystem('coordinator', self.coordinator, ['*'])
612 
613             # Add all pre-coupling and post-coupling components
614             for name, component in self.discipline_components.items():
615                 if name not in self.coupled_blocks:
616                     self.add_subsystem(name, component, ['*'])
617 
618             # Add the coupled group
619             if self.coupled_group is not None:
620                 self.add_subsystem('coupled_group', self.coupled_group, ['*'])
621 
622             # Add the consistency constraint group
623             if self.consistency_constraint_group is not None:
624                 self.add_subsystem('consistency_constraints', self.consistency_constraint_group,
    ↪  ['*'])
625 
626             # Put the blocks in the correct order
627             self.set_order(list(self.system_order))
628 
629             # Add the design variables
630             for name, value in self.design_vars.items():
631                 self.add_design_var(name, lower=value['lower'], upper=value['upper'],
    ↪  ref0=value['ref0'], ref=value['ref'])
632 
633             # Add the constraints
634             for name, value in self.constraints.items():
635                 self.add_constraint(name, lower=value['lower'], upper=value['upper'],
    ↪  equals=value['equals'])
636 
637             # Add the objective
638             self.add_objective(self.objective)
639 
640         def initialize_from_xml(self, xml):
641             # type: (Union[str, _ElementTree]) -> None
642             """Initialize the problem with initial values from an XML file.
643 
644             This function can only be called after the problem's setup method has been called.
645 
646             Parameters
647             ----------
648                 xml : str or :obj:`etree._ElementTree`
649                     Path to an XML file or an instance of `etree._ElementTree` representing it.
650             """
651             for xpath, value in xml_to_dict(xml).items():
652                 name = xpath_to_param(xpath)
653                 if name in self._outputs:
654                     self._outputs[name] = value
655                 elif name in self._inputs:
656                     self._inputs[name] = value
```

Code frament A.3: Code of the `openlego.core.model` Python module.

### A.1.4. `openlego.core.xml_component`

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5 
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9 
10     http://www.apache.org/licenses/LICENSE-2.0
11 
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
```

```
16  limitations under the License.
17
18  This file contains the definition the 'XMLComponent' class.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  import abc
23  import os
24  from abc import abstractmethod
25  from datetime import datetime
26
27  import numpy as np
28  from lxml import etree
29  from openmdao.api import Group, IndepVarComp, ExplicitComponent
30  from openmdao.vectors.vector import Vector
31  from typing import Optional, List, Union, Iterable
32
33  from openlego.utils.xml_utils import xml_safe_create_element, xml_to_dict, xpath_to_param,
      ↪ param_to_xpath, xml_merge
34  from openlego.partials.partials import Partials
35
36  dir_path = os.path.dirname(os.path.realpath(__file__))
37
38
39  class XMLComponent(ExplicitComponent):
40      """Abstract base class exposing an interface to use XML files for its in- and output.
41
42      This subclass of 'PromotingComponent' can automatically create ''OpenMDAO'' inputs and
      ↪  outputs based on given in-
43      and output XML template files. For maximum flexibility it is possible to only specify
      ↪  inputs from an XML file and
44      retain direct control over the definition of the outputs, or vice versa. It is also
      ↪  perfectly valid to add inputs
45      even when an XML file is used to generate a set of inputs, or outputs when an XML file it
      ↪  used to generate outputs.
46      It is even possible to generate in- and/or output parameters based on more than one XML
      ↪  file.
47
48      This class exposes the functions 'set_inputs_from_xml()' and 'set_outputs_from_xml()' for
      ↪  this purpose. Lists of all
49      parameters obtained from XML files are stored by this class for later inspection.
50
51      The 'solve_nonlinear()' method of the 'Component' class is implemented to wrap the XML
      ↪  related operations such as
52      reading in- and output data from the corresponding XML files during execution and storing
      ↪  it in this 'Component''s
53      parameter dictionaries.
54
55      A new abstract method is defined by this class, 'execute()', which assumes the role of the
      ↪  'solve_nonlinear()'
56      function, in essence. A specific case of this class should implement this method to
      ↪  perform the actual calculations
57      of an analysis tool using XML in- and/or output.
58
59      Attributes
60      ----------
61          inputs_from_xml, outputs_from_xml, partials_from_xml : dict
62              List of inputs, resp. outputs, resp. partials, taken from XML.
63
64          data_folder : str('')
65              Path to a folder in which to store data generated during the execution of this
      ↪  'XMLComponent'.
66
67          keep_files : bool(False)
68              Set to 'True' to keep all temporary XML files generated by the 'XMLComponent'
      ↪  during execution.
69
70              This attribute is 'False' by default, in which case all temporary in- and output
      ↪  XML files will be deleted
71              after they are no longer needed by this component.
72
```

```
73          base_file : str, optional
74              Path to an XML file to keep up-to-date with the latest data from executions.
75      """
76      __metaclass__ = abc.ABCMeta
77
78      def __init__(self,
79                   input_xml=None,            # type: Optional[Union[str, etree._ElementTree]]
80                   output_xml=None,           # type: Optional[Union[str, etree._ElementTree]]
81                   partials_xml=None,         # type: Optional[Union[str, etree._ElementTree]]
82                   data_folder='',            # type: str
83                   keep_files=False,          # type: bool
84                   base_file=None             # type: Optional[str]
85                   ):
86          # type: (...) -> None
87          """Initialize the 'XMLComponent'.
88
89          Parameters
90          ----------
91              input_xml, output_xml, partials_xml : str or :obj:'etree._ElementTree', optional
92                  Paths to or an 'etree._ElementTree' of input, resp. output, resp. partial, XML
    ↪  files.
93
94              data_folder : str
95                  Path to the folder in which to store temporary data files generated by this
    ↪  'XMLComponent'.
96
97              keep_files : bool(False)
98                  Set to 'True' to keep the temporary XML files after they are no longer needed.
99
100             base_file : str, optional
101                 Path to a base XML file to keep up-to-date with all latest data from this
    ↪  'XMLComponent'.
102         """
103         super(XMLComponent, self).__init__()
104
105         self.inputs_from_xml = dict()
106         self.outputs_from_xml = dict()
107         self.partials_from_xml = dict()
108
109         if input_xml is not None:
110             self.set_inputs_from_xml(input_xml)
111
112         if output_xml is not None:
113             self.set_outputs_from_xml(output_xml)
114
115         if partials_xml is not None:
116             self.declare_partials_from_xml(partials_xml)
117
118         self.data_folder = data_folder
119         self.keep_files = keep_files
120         self.base_file = base_file
121
122     def set_inputs_from_xml(self, input_xml):
123         # type: (Union[str, etree._ElementTree]) -> None
124         """Set inputs to the 'Component' based on an input XML template file.
125
126         Parameter names correspond to their XML elements' full XPaths, converted to valid
    ↪  ''OpenMDAO'' names using the
127         'xpath_to_param()' method.
128
129         Parameters
130         ----------
131             input_xml : str or :obj:'etree._ElementTree'
132                 Path to or an 'etree._ElementTree' of an input XML file.
133         """
134         self.inputs_from_xml.clear()
135         for xpath, value in xml_to_dict(input_xml).items():
136             name = xpath_to_param(xpath)
137             self.inputs_from_xml.update({name: value})
138
139     def set_outputs_from_xml(self, output_xml):
```

```
140         # type: (Union[str, etree._ElementTree]) -> None
141         """Set outputs to the 'Component' based on an output XML template file.
142 
143         Parameter names correspond to their XML elements' full XPaths, converted to valid
    ↪  ''OpenMDAO'' names using the
144         'xpath_to_param()' method.
145 
146         Parameters
147         ----------
148             output_xml : str or :obj:'etree._ElementTree'
149                 Path to or an 'etree._ElementTree' of an output XML file.
150         """
151         self.outputs_from_xml.clear()
152         for xpath, value in xml_to_dict(output_xml).items():
153             name = xpath_to_param(xpath)
154             self.outputs_from_xml.update({name: value})
155 
156     def declare_partials_from_xml(self, partial_xml):
157         # type: (Union[str, etree._ElementTree]) -> None
158         """Declare partials to the 'Component' based on a partials XML template file.
159 
160         Parameters
161         ----------
162             partial_xml : str or :obj:'etree._ElementTree'
163                 Path to or an 'etree._ElementTree' of a partials XML file.
164         """
165         self.partials_from_xml.clear()
166         if partial_xml is not None:
167             partials = Partials(partial_xml)
168             self.partials_from_xml = partials.get_partials().copy()
169 
170     @property
171     def variables_from_xml(self):
172         # type: () -> dict
173         """:obj:'dict': Dictionary of all XML inputs and outputs."""
174         variables = self.inputs_from_xml.copy()
175         variables.update(self.outputs_from_xml.copy())
176         return variables
177 
178     def setup(self):
179         for name, value in self.inputs_from_xml.items():
180             if not isinstance(value, float) and not isinstance(value, np.ndarray):
181                 # TODO: pass_by_obj
182                 # raise NotImplementedError('pass-by-object variables are not yet supported by
    ↪  OpenMDAO 2.0')
183                 pass
184             else:
185                 self.add_input(name, value)
186 
187         for name, value in self.outputs_from_xml.items():
188             if not isinstance(value, float) and not isinstance(value, np.ndarray):
189                 # TODO: pass_by_obj
190                 # raise NotImplementedError('pass-by-object variables are not yet supported by
    ↪  OpenMDAO 2.0')
191                 pass
192             else:
193                 self.add_output(name, value)
194 
195         if self.partials_from_xml:
196             for src, partials in self.partials_from_xml.items():
197                 if src is not None and partials is not None:
198                     self.declare_partials(src, partials.keys())
199         else:
200             self.declare_partials('*', '*', method='fd')
201             # if self.outputs_from_xml and self.inputs_from_xml:
202             #     for src in self.outputs_from_xml.keys():
203             #         self.declare_partials(src, self.inputs_from_xml.keys(), method='fd')
204 
205     @abstractmethod
206     def execute(self, input_xml=None, output_xml=None):
207         # type: (Optional[str], Optional[str]) -> None
```

```
208         """Execute the tool using the given input XML file. Write the results to the given
    ↪  output XML file.
209
210         Parameters
211         ----------
212             input_xml, output_xml : str, optional
213                 Path to the input, resp. output, XML file.
214         """
215         raise NotImplementedError
216
217     @abstractmethod
218     def linearize(self, input_xml=None, partials_xml=None):
219         # type: (Optional[str], Optional[str]) -> None
220         """Compute the partials of a tool using the given XML file. Write the results to the
    ↪  given partials XML file.
221
222         Parameters
223         ----------
224             input_xml, partials_xml : str, optional
225                 Path to the input, resp. partials, XML file.
226         """
227         raise NotImplementedError
228
229     def generate_file_names(self):
230         # type: () -> (str, str, str)
231         """Generate temporary file names for the input, output, and partials XML files.
232
233         Returns
234         -------
235             str
236                 Input XML file path.
237
238             str
239                 Output XML file path.
240
241             str
242                 Partials XML file path.
243
244         """
245         salt = datetime.now().strftime('%Y%m%d%H%M%f')
246         input_xml = os.path.join(self.data_folder, self.name + '_in_%s.xml' % salt)
247         output_xml = os.path.join(self.data_folder, self.name + '_out_%s.xml' % salt)
248         partials_xml = os.path.join(self.data_folder, self.name + '_partials_%s.xml' % salt)
249
250         return input_xml, output_xml, partials_xml
251
252     def write_input_file(self, file, inputs):
253         # type: (Union[str, etree._ElementTree], Vector) -> None
254         """Write the current input values to an input XML file.
255
256         Parameters
257         ----------
258             file : str or :obj:'etree._ElementTree'
259                 Path to or :obj:'etree._ElementTree' of an input XML file.
260
261             inputs : Vector
262                 Input vector of this 'Component'.
263         """
264         # Create new root element and an ElementTree
265         root = etree.Element(param_to_xpath(self.inputs_from_xml.keys()[0]).split('/')[1])
266         doc = etree.ElementTree(root)
267
268        # Convert all XML param names to XPaths and add new elements to the tree correspondingly
269         for param in self.inputs_from_xml:
270             if param in inputs:
271                 xml_safe_create_element(doc, param_to_xpath(param), inputs[param])
272
273         # Write the tree to an XML file
274         doc.write(file, pretty_print=True, xml_declaration=True, encoding='utf-8')
275
276     def read_outputs_file(self, file, outputs):
```

```
277         # type: (Union[str, etree._ElementTree], Vector) -> None
278         """Read the outputs from a given XML file and store them in this 'Component''s
    ↪ variables.
279 
280         Parameters
281         ----------
282             file : str or :obj:'etree._ElementTree'
283                 Path to or :obj:'etree._ElementTree' of an output XML file.
284 
285             outputs : Vector
286                 Output vector of this 'Component'.
287         """
288         # Extract the results from the output xml
289         for xpath, value in xml_to_dict(file).items():
290             name = xpath_to_param(xpath)
291             if name in self.outputs_from_xml and name in outputs:
292                 outputs[name] = value
293 
294     def read_partials_file(self, file, partials):
295         # type: (Union[str, etree._ElementTree], Vector) -> None
296         """Read the partials from a given XML file and store them in this 'Component''s
    ↪ variables.
297 
298         Parameters
299         ----------
300             file : str or :obj:'etree._ElementTree'
301                 Path to or :obj:'etree._ElementTree' of a partials XML file.
302 
303             partials : Vector
304                 Partials vector of this 'Component'.
305 
306         """
307         _partials = Partials(file)
308         for src, partial in _partials.get_partials().items():
309             for tgt, val in partial.items():
310                 if [src, tgt] in partials:
311                     try:
312                         partials[src, tgt] = val
313                     except Exception as e:
314                         print(e.message)
315 
316     def compute(self, inputs, outputs):
317         # type: (Vector, Vector) -> None
318         """Write the input XML file, call 'execute()', and read the output XML file to obtain
    ↪ the results.
319 
320         Parameters
321         ----------
322             inputs : 'Vector'
323                 Input parameters.
324 
325             outputs : 'Vector'
326                 Output parameters.
327         """
328 
329         input_xml, output_xml, _ = self.generate_file_names()
330 
331         if self.inputs_from_xml:
332             self.write_input_file(input_xml, inputs)
333             if self.base_file is not None:
334                 xml_merge(self.base_file, input_xml)
335 
336         # Call execute
337         if self.base_file is not None:
338             self.execute(self.base_file, output_xml)
339             xml_merge(self.base_file, output_xml)
340         else:
341             self.execute(input_xml, output_xml)
342 
343         # If files should not be kept, delete the input XML file
344         if not self.keep_files:
```

```
345             try:
346                 os.remove(input_xml)
347             except OSError:
348                 pass
349 
350         if self.outputs_from_xml:
351             self.read_outputs_file(output_xml, outputs)
352 
353             # If files should not be kept, delete the output XML file
354             if not self.keep_files:
355                 try:
356                     os.remove(output_xml)
357                 except OSError:
358                     pass
359 
360     def compute_partials(self, inputs, partials):
361         # type: (Vector, Vector) -> None
362         """Write the input XML file, call 'linearize()', and read the sensitivities from the
    ↪  resulting XML file.
363 
364         Parameters
365         ----------
366             inputs : 'Vector'
367                 Input parameters.
368 
369             partials: 'Vector'
370                 Partials.
371         """
372         if self.partials_from_xml:
373             input_xml, _, partials_xml = self.generate_file_names()
374 
375             self.write_input_file(input_xml, inputs)
376             self.linearize(input_xml, partials_xml)
377 
378             if not self.keep_files:
379                 try:
380                     os.remove(input_xml)
381                 except OSError:
382                     pass
383 
384             self.read_partials_file(partials_xml, partials)
385 
386             if not self.keep_files:
387                 try:
388                     os.remove(partials_xml)
389                 except OSError:
390                     pass
391 
392     def xml_params_as_indep_vars(self, group, params, values, aliases=None):
393         # type: (Group, List[str], Union[np.ndarray, Iterable], Optional[List[str]]) -> None
394         """Create 'IndepVarComp's for given input params of this 'XMLComponent'.
395 
396         Parameters
397         ----------
398             group : :obj:'Group'
399                 'Group' to add the 'IndepVarComp's to.
400 
401             params : list of str
402                 List of param names. These need to exist in this 'XMLComponent'.
403 
404             values : :obj:'np.ndarray' or list of numbers
405                 List of (initial) values for all 'IndepVarComp's.
406 
407             aliases : list of str, optional
408                 List of aliases (promoted names) to give the 'IndepVarComp's.
409         """
410         if len(params) != len(values) or (aliases is None and len(params) != len(aliases)):
411             raise ValueError('number of params, values and optionally aliases needs to be the
    ↪  same')
412 
413         for param in params:
```

```
414             if param not in self.inputs_from_xml:
415                 raise ValueError('at least one param given is not a param of this XMLComponent
    ↪  (%s)' % param)
416
417         for index, param in enumerate(params):
418             if aliases is None:
419                 alias = 'INDEP_' + param_to_xpath(param).split('/')[-1].split('[')[0]
420             else:
421                 alias = aliases[index]
422
423             group.add(alias, IndepVarComp(alias, val=values[index]), promotes=[alias])
424             group.connect(alias, param)
```

Code frament A.4: Code of the `openlego.core.xml_component` Python module.

# A.2. Partials

## A.2.1. XMLSchema

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <xs:schema
3          xmlns:xs="http://www.w3.org/2001/XMLSchema"
4          elementFormDefault="qualified"
5  >
6
7      <!-- parameterType: an element with a parameterUID child element -->
8      <xs:complexType name="parameterType">
9          <xs:sequence>
10             <xs:element name="uid" type="xs:string">
11                 <xs:annotation>
12                     <xs:documentation xml:lang="en">
13                         Unique identifier of the parameter this element refers to.
14                     </xs:documentation>
15                 </xs:annotation>
16             </xs:element>
17             <xs:element name="value" minOccurs="0">
18                 <xs:annotation>
19                     <xs:documentation>
20                         Value of the sensitivity of the current parameter to this variable.
21                     </xs:documentation>
22                 </xs:annotation>
23                 <xs:complexType>
24                     <xs:simpleContent>
25                         <xs:extension base="xs:string">
26                             <xs:attribute name="mapType" />
27                         </xs:extension>
28                     </xs:simpleContent>
29                 </xs:complexType>
30             </xs:element>
31         </xs:sequence>
32     </xs:complexType>
33
34     <!-- dependentParamType: an extended parameterType with a number of partialType children
    ↪  -->
35     <xs:complexType name="dependentParamType">
36         <xs:annotation>
37             <xs:documentation>
38                 This element defines the sensitivities of a parameter.
39             </xs:documentation>
40         </xs:annotation>
41         <xs:complexContent>
42             <xs:extension base="parameterType">
43                 <xs:sequence>
44                     <xs:element name="partials">
45                         <xs:annotation>
46                             <xs:documentation>
47                                 A list of sensitivities of this parameter w.r.t. parameters it
    ↪  depends on.
48                             </xs:documentation>
49                         </xs:annotation>
```

```
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="partial" minOccurs="0" maxOccurs="unbounded"
    ↪  type="parameterType" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Definition of the overall schema -->
    <xs:element name="partials">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="parameter" minOccurs="0" maxOccurs="unbounded"
    ↪  type="dependentParamType" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Code frament A.5: Partials XSD schema code.

### A.2.2. openlego.partials.partials

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import os
import warnings

from lxml import etree
from typing import Union, List, Optional, Any

from openlego.utils.general_utils import parse_string
from openlego.utils.xml_utils import value_to_xml


dir_path = os.path.dirname(os.path.abspath(__file__))
xsd_file_path = os.path.join(dir_path, 'partials.xsd')
xsi_schema_location = 'file:///' + xsd_file_path

schema = etree.XMLSchema(file=xsi_schema_location)
parser = etree.XMLParser(schema=schema)


class Partials(object):

    def __init__(self, file=None):
        # type: (Optional[str]) -> None
        """Initialize 'Partials' object.

        Parameters
        ----------
            file : str, optional
                Path to the partials XML file to initialize from.
        """
        super(Partials, self).__init__()

        if file is None:
            self._tree = etree.ElementTree(etree.Element('partials'), parser=parser)    #
    ↪  type: etree._ElementTree
        else:
            self._tree = etree.parse(file, parser)

    @property
```

```
42      def _elem_root(self):
43          # type: () -> etree._Element
44          """Root '_Element' of the partials XML file."""
45          return self._tree.getroot()
46
47      def get_partials(self, src=None):
48          # type: (Optional[src]) -> dict
49          """Get a dictionary with the partials stored in the XML file.
50
51          Parameters
52          ----------
53              src : str, optional
54                  Name of the source parameter to get the partials from
55
56          Returns
57          -------
58              dict
59                  In the form partials['from_param_name']['to_param_name'] = sensitivity_value
    ↪  if no 'src' is given, or
60                  in the form partials['to_param_name'] = sensitivity_value if 'src' is given.
61          """
62          partials = dict()
63
64          if src is not None:
65              elem_param = self._tree.xpath('/partials/parameter[uid="{}"]'.format(src))
66              if len(elem_param):
67                  for elem_partial in elem_param[1]:
68                      param = elem_partial[0].text
69                      value = parse_string(elem_partial[1].text)
70                      partials.update({param: value})
71          else:
72              for elem_param in self._elem_root:
73                  uid = elem_param[0].text
74
75                  if uid not in partials:
76                      partials.update({uid: dict()})
77
78                  for elem_partial in elem_param[1]:
79                      param = elem_partial[0].text
80                      if len(elem_partial) > 1:
81                          value = parse_string(elem_partial[1].text)
82                      else:
83                          value = 0.
84                      partials[uid].update({param: value})
85
86          return partials
87
88      def declare_partials(self, src, tgt, val=None):
89          # type: (str, Union[str, List[str]], Optional[Any]) -> None
90          """Declare a set of partials that is provided.
91
92          Parameters
93          ----------
94              src : str
95                  Name of the source parameter.
96
97              tgt : str or Iterable[str]
98                  Name(s) of target parameters.
99
100             val : any, optional
101                 Optional value(s) of partials.
102
103         Notes
104         -----
105             If 'val' is given and 'tgt' is a list, 'val' should have the same length as 'tgt'.
106         """
107         if not isinstance(tgt, list):
108             tgt = [tgt]
109             if val is not None:
110                 val = [val]
111
```

```
112         elem_root = self._elem_root
113 
114         x_param = "/partials/parameter[uid='{}']".format(src)
115         elem_param = self._tree.xpath(x_param)
116 
117         if not len(elem_param):
118             elem_param = etree.SubElement(elem_root, 'parameter')
119             elem_param_uid = etree.SubElement(elem_param, 'uid')
120             elem_param_uid.text = src
121 
122             elem_partials = etree.SubElement(elem_param, 'partials')
123         else:
124             elem_partials = elem_param[0][1]
125 
126         for i, t in enumerate(tgt):
127             x_partial = '/'.join([x_param, "partials/partial[uid='{}']"]).format(t)
128             elem_partial = self._tree.xpath(x_partial)
129 
130             if not len(elem_partial):
131                 elem_partial = etree.SubElement(elem_partials, 'partial')
132                 elem_param_uid = etree.SubElement(elem_partial, 'uid')
133                 elem_param_uid.text = t
134             else:
135                 warnings.warn(
136                     'Partial from {} to {} is defined more than once. Last occurrence take
    ↪ precedence.'
137                     .format(src, t))
138 
139             if val is not None:
140                 elem_value = etree.SubElement(elem_partial, 'value')
141                 value_to_xml(elem_value, val[i])
142 
143     def add_partials(self, partials):
144         # type: (dict) -> None
145         """Add a set of partials to the XML file.
146 
147         Parameters
148         ----------
149             partials : dict
150                 Dictionary of the partials.
151         """
152         for param_uid, param in partials.items():
153             self.declare_partials(param_uid, param.keys(), param.values())
154 
155     def write(self, file):
156         # type: (str) -> None
157         """Write the current state of the class to a partials XML file.
158 
159         Parameters
160         ----------
161             file : str
162                 Path of the file to write to.
163         """
164         if not schema.validate(self._tree):
165             raise RuntimeError('Something is wrong.. XML is not a valid partials file.')
166 
167         self._tree.write(file, encoding='utf-8', pretty_print=True, xml_declaration=True)
168 
169     def get_string(self):
170         # type: () -> str
171         """Return the current state of the class as a partials XML string.
172 
173         Returns
174         -------
175             str
176                 String representation of a partials XML file.
177         """
178         if not schema.validate(self._tree):
179             raise RuntimeError('Something is wrong.. XML is not a valid partials file.')
180 
```

```
181         return etree.tostring(self._tree, encoding='utf-8', pretty_print=True,
    ↪ xml_declaration=True)
```

Code frament A.6: Code of the `openlego.partials.partials` Python module.

# A.3. Recorders

## A.3.1. `openlego.recorders.base_iteration_plotter`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definitions of the `BaseIterationPlotter` class.
"""
from __future__ import absolute_import, division, print_function

import sys

if sys.version_info[0] == 3:
    import tkinter as tk
else:
    import Tkinter as Tk
    tk = Tk

import abc
import time
import matplotlib

from abc import abstractmethod
from multiprocessing import Process, Pipe
from openmdao.core.driver import Driver
from openmdao.solvers.solver import Solver
from openmdao.core.system import System
from openmdao.recorders.base_recorder import BaseRecorder
from typing import Optional, Any, Union

matplotlib.use('TkAgg')
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib import pyplot as plt
from matplotlib.figure import Figure

from openlego.utils.general_utils import try_hard


class BaseIterationPlotter(BaseRecorder):
    """Base class enabling continually updated plots.

    This is an abstract base class which enables data from an OpenMDAO run to be plotted and
    ↪ for that plot to be updated
    for each iteration/function evaluation. This class uses matplotlib for all the plotting
    ↪ functions. A separate
    process is used to host and manage the plot. This avoids the blocking and stalling
    ↪ behavior of the main loop by
    matplotlib.

    Attributes
```

```
59          ----------
60              save_settings : dict
61                  Default setting sused when saving the plot as an image file.
62          """
63          __metaclass__ = abc.ABCMeta
64
65          def __init__(self):
66              # type: () -> None
67              """Initialize the class."""
68              super(BaseIterationPlotter, self).__init__()
69              self.options.declare('save_on_close', False, desc='Set to True to save figure when
    ↪  closing the Recorder')
70              self.save_settings = {'path': self.__class__.__name__ + '_figure.png',
71                                    'dpi': 600, 'ar': 1.61803398875, 'width': 4000}
72
73              self._in, self._out = Pipe()
74              self._callback_in, self._callback_out = Pipe()
75              self._process = None
76
77              self._tk = None
78              self._canvas = None
79              self._toolbar = None
80              self._fig = None
81
82              self.options['record_objectives'] = True
83              self.options['record_constraints'] = True
84
85          @abstractmethod
86          def init_fig(self, fig):
87              # type: (Figure) -> None
88              """Initialize the figure.
89
90              A plotting recorder implementing this method should use it to setup parts of the
    ↪  figure that should only be
91              initialized once, such as titles, labels, and legends.
92
93              Parameters
94              ----------
95                  fig : :obj:'Figure'
96                      Instance of 'Figure' on which the plot should be initialized.
97              """
98              raise NotImplementedError
99
100         @abstractmethod
101         def _update_plot(self, *args):
102             # type: (Any) -> None
103             """Update the plot with data from the next iteration/function evaluation.
104
105             This method should be implemented to insert new data into the plot. This method is
    ↪  called within the plot
106             handling 'Process' and should never be called directly.
107
108             Parameters
109             ----------
110                 *args : any
111                     Data to update or enrich the plot with.
112              """
113             raise NotImplementedError
114
115         def startup(self, object_requesting_recording):
116             # type: (Union[Driver, System, Solver]) -> None
117             """Call the 'BaseRecorder' 'startup()' method and start the 'Process' handling the
    ↪  figure.
118
119             Parameters
120             ----------
121                 object_requesting_recording : Union[Driver, System, Solver]
122                     The Object to which this recorder is attached.
123             """
124             super(BaseIterationPlotter, self).startup(object_requesting_recording)
125
```

```
126         self._process = Process(target=self._process_run)
127         self._process.daemon = True
128         self._process.start()
129 
130     def _process_run(self):
131         # type: () -> None
132         """Perform the figure handling operations.
133 
134         This funciton is the entry point for the plot handling 'Process'. It should not be
    ↪ used directly.
135 
136         Notes
137         -----
138             This function is executed within a separate 'Process'. Any parameters assigned
    ↪ from within this scope can
139             only be accessed by function also running on this 'Process'.
140         """
141         self._tk = tk.Tk()
142         self._tk.protocol('WM_DELETE_WINDOW', self._tk.quit())
143 
144         def plt_figure():
145             # type: () -> Figure
146             """Open a 'Figure' and wait for a short time.
147 
148             This is part of a workaround to ensure a figure window always opens.
149 
150             Returns
151             -------
152                 :obj:'Figure'
153                     Instance of the openened 'Figure'.
154             """
155             fig = plt.figure()
156             time.sleep(1e-4)
157             return fig
158         self._fig = try_hard(plt_figure, try_hard_limit=4)  # type: Figure
159 
160         self._canvas = FigureCanvasTkAgg(self._fig, master=self._tk)
161         self._toolbar = NavigationToolbar2TkAgg(self._canvas, self._tk)
162         self._toolbar.update()
163         self._canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
164 
165         self.init_fig(self._fig)
166 
167         self._tk.after(1, self._loop())
168         self._tk.mainloop()
169 
170     def _loop(self):
171         # type: () -> None
172         """Continually handle the figure on the dedicated 'Process'.
173 
174         This function is the loop of the 'Process' handling the plot. It is executed on the
    ↪ dedicated 'Process' handling
175         the figure and therefore has access to the parameters that were assigned in
    ↪ '_process_run()'.
176 
177         Notes
178         -----
179             This method should never be called directly. Any instructions to manipulate the
    ↪ figure are communicated
180             through 'Pipe's. Convenience methods have been set up for this purpose.
181         """
182         while self._out.poll():
183             out = self._out.recv()
184             if out is None or not isinstance(out, tuple):
185                 raise ValueError('packet sent to _update() is not a tuple')
186             elif 'update' in out[0] or 'save' in out[0]:
187                 if 'update' in out[0]:
188                     self._update_plot(*out[1:])
189                 elif 'save' in out[0]:
190                     path, dpi, ar, width = out[1:]
191                     self._fig.set_size_inches(ar * width / dpi, ar * width / dpi)
```

```
192                     self._fig.savefig(path, dpi=dpi)
193
194                 if 'block' in out[0]:
195                     self._callback_in.send(True)
196             elif 'close' in out[0]:
197                 self._tk.destroy()
198                 if 'block' in out[0]:
199                     self._callback_in.send(True)
200                 return
201
202         self._canvas.draw()
203         self._tk.after(1, self._loop)
204
205     def _blocking_call(self, instr, *args):
206         # type: (str, *Any) -> Any
207         """Send an instruction to the 'Process' handling the plot and wait for a reply.
208
209         Parameters
210         ----------
211             instr : str
212                 Instruction to send to the 'Process'.
213
214             *args
215                 Any arguments to send along with the instruction.
216
217         Returns
218         -------
219             any
220                 A reply from the 'Process'.
221         """
222         self._in.send(('block %s' % instr,) + args)
223         while not self._callback_out.poll():
224             time.sleep(1.e-4)
225         return self._callback_out.recv()
226
227     def record_metadata_driver(self, object_requesting_recording):
228         pass
229
230     def record_metadata_system(self, object_requesting_recording):
231         pass
232
233     def record_metadata_solver(self, object_requesting_recording):
234         pass
235
236     def record_iteration_driver_passing_vars(self, object_requesting_recording, desvars,
    ↪  responses, objectives,
237                                              constraints, sysvars, metadata):
238         super(BaseIterationPlotter,
    ↪  self).record_iteration_driver_passing_vars(object_requesting_recording,
239                                                                                 desvars,
    ↪  responses, objectives,
240                                                                                 constraints,
    ↪  sysvars, metadata)
241         self._record_iteration_driver(metadata)
242
243     def record_iteration_driver(self, object_requesting_recording, metadata):
244         super(BaseIterationPlotter,
    ↪  self).record_iteration_driver(object_requesting_recording, metadata)
245         self._record_iteration_driver(metadata)
246
247     def record_iteration_system(self, object_requesting_recording, metadata):
248         super(BaseIterationPlotter,
    ↪  self).record_iteration_system(object_requesting_recording, metadata)
249         self._record_iteration_system(metadata)
250
251     def record_iteration_solver(self, object_requesting_recording, metadata, **kwargs):
252         super(BaseIterationPlotter,
    ↪  self).record_iteration_solver(object_requesting_recording, metadata, **kwargs)
253         self._record_iteration_solver(metadata)
254
255     def _record_iteration_driver(self, metadata):
```

```
256             self._in.send(('update',
257                            self._desvars_values,
258                            self._responses_values,
259                            self._objectives_values,
260                            self._constraints_values,
261                            metadata))
262             self.save_figure()
263 
264         def _record_iteration_system(self, metadata):
265             self._in.send(('update',
266                            self._inputs,
267                            self._outputs,
268                            self._resids,
269                            metadata))
270             self.save_figure()
271 
272         def _record_iteration_solver(self, metadata):
273             self._in.send(('update',
274                            self._abs_error,
275                            self._rel_error,
276                            self._outputs,
277                            self._resids,
278                            metadata))
279             self.save_figure()
280 
281         def save_figure(self, path=None, dpi=None, ar=None, width=None):
282             # type: (Optional[str], Optional[int], Optional[float], Optional[int]) -> None
283             """Save the current plot to an image file.
284 
285             Parameters
286             ----------
287                 path : str, optional
288                     Path of the image file.
289 
290                 dpi : int, optional
291                     Resolution of the image file in dots per inch.
292 
293                 ar : float, optional
294                     Aspect ratio of the image file.
295 
296                 width : int, optional
297                     Width of the image file in pixels.
298 
299             Notes
300             -----
301                 If any of the parameters are not given the defaults stored in the 'save_settings'
    ↪ dictionary will be used.
302 
303                 This function makes a blocking call to the 'Process' to ensure the figure really
    ↪ is saved once this function
304                 returns.
305             """
306             if path is None:
307                 path = self.save_settings['path']
308             if dpi is None:
309                 dpi = self.save_settings['dpi']
310             if ar is None:
311                 ar = self.save_settings['ar']
312             if width is None:
313                 width = self.save_settings['width']
314             self._blocking_call('save', path, dpi, ar, width)
315 
316         def save_and_close(self, path=None, dpi=None, ar=None, width=None):
317             # type: (Optional[str], Optional[int], Optional[float], Optional[int]) -> None
318             """Save the current plot to an image file and close this 'Recorder'.
319 
320             Parameters
321             ----------
322                 path : str, optional
323                     Path of the image file.
324 
```

```
325             dpi : int, optional
326                 Resolution of the image file in dots per inch.
327
328             ar : float, optional
329                 Aspect ratio of the image file.
330
331             width : int, optional
332                 Width of the image file in pixels.
333
334         Notes
335         -----
336             If any of the parameters are not given the defaults stored in the 'save_settings'
    ↪ dictionary will be used.
337
338             This function makes a blocking call to the 'Process' to ensure the figure really
    ↪ is saved once this function
339             returns.
340         """
341         self.save_figure(path, dpi, ar, width)
342
343         # Prevent save_on_close option from saving the figure again, then call self.close()
344         self.options['save_on_close'] = False
345         self.close()
346
347     def close(self):
348         """Close the figure, then calls the 'BaseRecorder' 'close()' method.
349
350         If the option ''save_on_close'' is set to 'True', this function first saves the figure
    ↪ and waits for
351         confirmation before it closes it and this 'Recorder'.
352         """
353         # Potentially save the figure before closing
354         if self.options['save_on_close']:
355             self.save_figure()
356
357         # Close the figure and call super
358         self._blocking_call('close')
359         super(BaseIterationPlotter, self).close()
```

Code frament A.7: Code of the `openlego.recorders.base_iteration_plotter` Python module.

### A.3.2. `openlego.recorders.base_lane_plotter`

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9
10     http://www.apache.org/licenses/LICENSE-2.0
11
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17
18 This file contains the definition of the 'BaseLanePlotter' class.
19 """
20 from __future__ import absolute_import, division, print_function
21
22 import abc
23 from abc import abstractmethod
24
25 import matplotlib.colors as colors
26 import numpy as np
27 from matplotlib import ticker as ticker
```

```
28  from matplotlib.figure import Figure
29  from openmdao.core.driver import Driver
30  from typing import Optional
31
32  from .base_iteration_plotter import BaseIterationPlotter
33
34
35  class BaseLanePlotter(BaseIterationPlotter):
36      """Specialized 'BaseIterationPlotter' wrapping a ''lane plot'' style visualization of
      ↪  variables.
37
38      Abstract base class enabling OpenMDAO data to be visualized using colored, horizontal
      ↪  lanes. Each variable to be
39      visualized this way has its own lane. The x-axis corresponds to the number of
      ↪  iterations/function evaluations. A
40      colorbar is used to indicate the value of a design variable.
41
42      Attributes
43      ----------
44          n_vars : int
45              The number variables.
46
47          var_names : :obj:'list' of :obj:'str'
48              List of all variable names.
49
50          xs, ys, cs: :obj:'np.ndarray'
51              Arrays containing the x-, y-, and color data of the figure.
52
53          iter : int
54              Number of the last iteration.
55
56          ax : :obj:'Axes'
57              Matplotlib 'Axes' of the plot.
58
59          max_iter : int
60              Maximum number of iterations.
61
62          quad : :obj:'matplotlib.collections.QuadMesh'
63              Instance of 'QuadMesh' that represents the actual plot.
64
65          vmin, vmax : float
66              Lower and upper cutoff for  values along the colorbar.
67
68          cmap : str
69              Name of the colormap to use.
70
71          norm : :obj:'colors.Normalize', optional
72              Which normalization scheme to use for the colorbar.
73      """
74      __metaclass__ = abc.ABCMeta
75
76      def __init__(self, vmin=0., vmax=1., cmap='viridis', norm=None):
77          # type: (float, float, str, Optional[colors.Normalize]) -> None
78          """Initialize a new 'BaseLanePlotter' instance.
79
80          Parameters
81          ----------
82              vmin, vmax : float
83                  Lower and upper cutoff for the values along the colorbar.
84
85              cmap : str('viridis')
86                  Name of the colormap to use for the plot.
87
88              norm : :obj:'colors.Normalize', optional
89                  Instance of 'colors.Normalize' can be supplied to use a normalization scheme
      ↪  for the colorbar.
90          """
91          super(BaseLanePlotter, self).__init__()
92
93          self.n_vars = None
94          self.var_names = None
```

```
95
96          self.xs = None
97          self.ys = None
98          self.cs = None
99
100         self.iter = 0
101         self.ax = None
102         self.max_iter = 1000
103
104         self.quad = None
105
106         self.vmin = vmin
107         self.vmax = vmax
108         self.cmap = cmap
109         self.norm = norm
110
111     def startup(self, object_requesting_recording):
112         # type: (Driver) -> None
113         """Make sure this 'Recorder' is attached to a 'Driver' and obtain the maximum number
    ↪ of iterations.
114
115         Parameters
116         ----------
117             object_requesting_recording : :obj:'Driver'
118                 Instance of 'Driver' to which this 'Recorder' is attached.
119         """
120         if not isinstance(object_requesting_recording, Driver):
121             raise ValueError('This Recorder should be attached to a Driver.')
122
123         if 'maxiter' in object_requesting_recording.options:
124             self.max_iter = object_requesting_recording.options['maxiter']
125
126         super(BaseLanePlotter, self).startup(object_requesting_recording)
127
128     @abstractmethod
129     def init_vars(self):
130         # type: () -> None
131         """Initialize the variables of the plot.
132
133         This method should be implemented by subclasses such that they can control how
    ↪ variables are initialized.
134         """
135         raise NotImplementedError
136
137     def init_fig(self, fig):
138         # type: (Figure) -> None
139         """Initialize the figure, setting up axes, labels, the colorbar, etc.
140
141         Parameters
142         ----------
143             fig : :obj:'Figure'
144                 Instance of the 'Figure' which should be populated.
145         """
146         self.init_vars()
147
148         self.xs, self.ys = np.meshgrid(np.arange(0., self.max_iter+.5)-.5, np.arange(0.,
    ↪ self.n_vars+.5)-.5)
149         self.cs = np.zeros((self.n_vars, self.max_iter))
150
151         self.ax = fig.add_subplot(111)
152         self.ax.xaxis.set_major_locator(ticker.MaxNLocator(integer=True))
153         self.ax.yaxis.set_ticks(np.arange(0, self.n_vars))
154         self.ax.yaxis.set_ticklabels(self.var_names)
155
156         self.ax.set_xlim([-.5, .5])
157         self.ax.set_ylim([-.5, self.n_vars-.5])
158         self.quad = self.ax.pcolormesh(self.xs, self.ys, self.cs,
159                                        vmin=self.vmin, vmax=self.vmax, cmap=self.cmap,
    ↪ norm=self.norm)
160
161         fig.colorbar(self.quad)
```

```
162
163         self.ax.set_xlabel('Evaluation #')
164
165     @abstractmethod
166     def _compute_new_data(self, desvars, responses, objectives, constraints, metadata):
167         # type: (dict, dict, dict, dict, dict) -> np.ndarray
168         """Return a 1D numpy.ndarray containing the new data points.
169
170         Parameters
171         ----------
172             desvars, responses, objectives, constraints, metadata : dict
173                 Dictionaries of the new design, response, objective, and constraint variables,
    ↪  as well as metadata.
174
175         Returns
176         -------
177             np.ndarray
178                 A 1D numpy array containing the new data.
179         """
180         raise NotImplementedError
181
182     def _update_plot(self, *args):
183         # type: (dict, dict, dict, dict, dict) -> None
184         """Insert the new data into the plot and refresh it.
185
186         Parameters
187         ----------
188             desvars, responses, objectives, constraints, metadata : dict
189                 Dictionaries of the new design, response, objective, and constraint variables,
    ↪  as well as metadata.
190         """
191         if len(args) != 5 and not any([isinstance(arg, dict) for arg in args]):
192             raise ValueError('Illegal arguments for _update_plot of %s' % self.__name__)
193         desvars, responses, objectives, constraints, metadata = args
194
195         data = self._compute_new_data(desvars, responses, objectives, constraints, metadata)
196         self.cs[:, self.iter] = data[:]
197         self.quad.set_array(self.cs.ravel())
198         self.ax.set_xlim([-.5, self.iter+.5])
199         self.iter += 1
```

Code frament A.8: Code of the `openlego.recorders.base_lane_plotter` Python module.

### A.3.3. `openlego.recorders.constraint_plotter`

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9
10     http://www.apache.org/licenses/LICENSE-2.0
11
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17
18 This file contains the definition of the 'ConstraintsPlotter' class.
19 """
20 from __future__ import absolute_import, division, print_function
21
22 import matplotlib.colors as colors
23 import numpy as np
24 from openmdao.core.driver import Driver
25 from typing import Optional
```

```
26
27  from .base_lane_plotter import BaseLanePlotter
28
29
30  class ConstraintsPlotter(BaseLanePlotter):
31      """Specific case of the 'BaseLanePlotter' plotting all the constraint variables of a
      ↪  'Problem.
32
33      A symmetric logarithmic colorbar is used by default by this plottter.
34
35      Attributes
36      ----------
37          constr_meta : dict
38              A copy of the constraint metadata of the 'Driver' this 'Recorder' is associated
      ↪  with.
39      """
40
41      def __init__(self, vmin=-1., vmax=1., cmap='RdBu_r',
42                   norm=colors.SymLogNorm(linthresh=1.e-3, linscale=.5, vmin=-1., vmax=1.)):
43          # type: (float, float, str, Optional[colors.Normalize]) -> None
44          """Initialize the 'BaseLanePlotter' and store the ''constraint_metadata'' from the
      ↪  'Driver'.
45
46          Parameters
47          ----------
48              vmin, vmax : float
49                  Lower and upper cutoff for the values along the colorbar.
50
51              cmap : str('RdBu_r')
52                  Name of the colormap to use for the plot.
53
54              norm : :obj:'colors.Normalize'('colors.SymLogNorm'), optional
55                  A symmetric logarithmic colorbar is used by default by this plottter.
56          """
57          super(ConstraintsPlotter, self).__init__(vmin, vmax, cmap, norm)
58          self.constr_meta = None
59
60      def startup(self, object_requesting_recording):
61          # type: (Driver) -> None
62          """Make sure this 'Recorder' is attached to a 'Driver' and obtain the constraint
      ↪  variable metadata.
63
64          Parameters
65          ----------
66              object_requesting_recording : :obj:'Driver'
67                  Instance of 'Driver' to which this 'Recorder' is attached.
68          """
69          self.constr_meta = object_requesting_recording._cons.copy()
70          super(ConstraintsPlotter, self).startup(object_requesting_recording)
71
72      def init_vars(self):
73          # type: () -> None
74          """Initialize the list of constraint variable names and obtain the number of
      ↪  constraints."""
75          self.var_names = list()
76          self.n_vars = 0
77          for key in self.constr_meta.keys():
78              size = self.constr_meta[key]['size']
79              self.n_vars += size
80              self.var_names.extend(['%s[%d]' % (key, i) for i in range(size)])
81
82      def _compute_new_data(self, desvars, responses, objectives, constraints, metadata):
83          # type: (dict, dict, dict, dict, dict) -> np.ndarray
84          """Compute the new data points for the lane plot from the constraints.
85
86          Parameters
87          ----------
88              desvars, responses, objectives, constraints, metadata : dict
89                  Dictionaries of the new design, response, objective, and constraint variables,
      ↪  as well as metadata.
90
```

```
        Returns
        -------
            np.ndarray
                A 1D numpy array containing the new data.
        """
        parts = [constraints[key] for key in self.constr_meta.keys()]
        for index, part in enumerate(parts):
            parts[index] = np.atleast_1d(part).flatten()
        return np.concatenate(parts)
```

Code frament A.9: Code of the `openlego.recorders.constraint_plotter` Python module.

### A.3.4. `openlego.recorders.normalized_desvar_plotter`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definition of the 'NormalizedDesignVarPlotter' class.
"""
from __future__ import absolute_import, division, print_function

import matplotlib.colors as colors
import numpy as np
from openmdao.core.driver import Driver
from typing import Optional

from .base_lane_plotter import BaseLanePlotter


class NormalizedDesignVarPlotter(BaseLanePlotter):
    """Specific case of the 'BaseLanePlotter' which plots all normalized design variables of a
    'Problem'.

    Design variable values are normalized using the ''ref0'' and ''ref'' properties of the
    design variables as
    specified in the ''metadata''.
    """

    def __init__(self, vmin=0., vmax=1., cmap='viridis', norm=None):
        # type: (float, float, str, Optional[colors.Normalize]) -> None
        """Initialize the 'BaseLanePlotter' and stores the ''desvar_metadata'' from the
    'Driver'.

        Parameters
        ----------
            vmin, vmax : float
                Lower and upper cutoff for the values along the colorbar.

            cmap : str('viridis')
                Name of the colormap to use for the plot.

            norm : :obj:'colors.Normalize', optional
                Instance of 'colors.Normalize' can be supplied to use a normalization scheme
    for the colorbar.
        """
        super(NormalizedDesignVarPlotter, self).__init__(vmin, vmax, cmap, norm)
```

```
53          self.desvar_meta = None
54
55      def startup(self, object_requesting_recording):
56          # type: (Driver) -> None
57          """Make sure this 'Recorder' is attached to a 'Driver' and obtain the design variable
    ↪  metadata.
58
59          Parameters
60          ----------
61              object_requesting_recording : :obj:'Driver'
62                  Instance of 'Driver' to which this 'Recorder' is attached.
63          """
64          self.desvar_meta = object_requesting_recording._designvars.copy()
65          super(NormalizedDesignVarPlotter, self).startup(object_requesting_recording)
66
67      def init_vars(self):
68          # type: () -> None
69          """Initialize the lists of design variable names and obtain the number of design
    ↪  variables."""
70          self.var_names = list()
71          self.n_vars = 0
72          for key in self.desvar_meta.keys():
73              ref0 = self.desvar_meta[key]['ref0']
74              if isinstance(ref0, np.ndarray):
75                  size = ref0.size
76              else:
77                  size = 1
78              self.n_vars += size
79              self.var_names.extend(['%s[%d]' % (key, i) for i in range(size)])
80
81      def _compute_new_data(self, desvars, responses, objectives, constraints, metadata):
82          # type: (dict, dict, dict, dict, dict) -> np.ndarray
83          """Compute the new data points of the plot from the design variable values.
84
85          Parameters
86          ----------
87              desvars, responses, objectives, constraints, metadata : dict
88                  Dictionaries of the new design, response, objective, and constraint variables,
    ↪  as well as metadata.
89
90          Returns
91          -------
92              np.ndarray
93                  A 1D numpy array containing the new data.
94          """
95          parts = [(desvars[key] - self.desvar_meta[key]['lower']) /
96                   (self.desvar_meta[key]['upper'] - self.desvar_meta[key]['lower']) for key in
    ↪  self.desvar_meta.keys()]
97          for index, part in enumerate(parts):
98              parts[index] = np.atleast_1d(part).flatten()
99          return np.concatenate(parts)
```

Code frament A.10: Code of the `openlego.recorders.normalized_desvar_plotter` Python module.

### A.3.5. `openlego.recorders.simple_objective_plotter`

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the 'SimpleObjectivePlotter' class.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  import numpy as np
23  from matplotlib import ticker as ticker
24  from matplotlib.figure import Figure
25  from openmdao.core.driver import Driver
26
27  from .base_iteration_plotter import BaseIterationPlotter
28
29
30  class SimpleObjectivePlotter(BaseIterationPlotter):
31      """Specialized 'BaseIterationPlotter' which simply plots the normalized objective function
      ↪  value against iteration.
32
33      Attributes
34      ----------
35          obj_name : str
36              Name of the objective function value.
37
38          obj_init : float
39              Initial value of the objective function
40
41          xdata, ydata : :obj:'np.ndarray'
42              The x- and y-data of the plot.
43
44          ax : :obj:'Axes'
45              The axis of the plot.
46
47          line : :obj:'Line'
48              The 'Line' object of the plot.
49
50          first_run : bool
51              Flag signifying whether this is the first run of the 'Recorder'. Flipped to
      ↪  'False' after first iteration.
52      """
53
54      def __init__(self):
55          # type: (Driver) -> None
56          """Initialize the 'SimpleObjectivePlotter'."""
57          super(SimpleObjectivePlotter, self).__init__()
58
59          self.obj_name = None
60          self.obj_init = None
61
62          self.xdata = None
63          self.ydata = None
64
65          self.ax = None
66          self.line = None
67
68          self.first_run = True
69
70      def startup(self, object_requesting_recording):
71          # type: (Driver) -> None
72          """Obtain the name of the objective function variable from the 'Driver' before calling
      ↪  the 'super()'.
73
74          Parameters
75          ----------
76              object_requesting_recording : :obj:'Driver'
77                  'Driver' that owns this 'Recorder'.
78          """
79          if not isinstance(object_requesting_recording, Driver):
80              raise ValueError('This Recorder must be attached to a Driver.')
81
82          super(SimpleObjectivePlotter, self).startup(object_requesting_recording)
```

```
83
84      def init_fig(self, fig):
85          # type: (Figure) -> None
86          """Initialize the axes and line of the plot.
87
88          Parameters
89          ----------
90              fig : :obj:'Figure'
91                  Instance of the 'Figure' which should be populated.
92          """
93          self.ax = fig.add_subplot(111)
94          self.ax.xaxis.set_major_locator(ticker.MaxNLocator(integer=True))
95          self.ax.set_xlabel('Iteration #')
96          self.ax.set_ylabel('Normalized objective function value')
97
98          self.line, = self.ax.plot([], [], color='black')
99
100     def _update_plot(self, *args):
101         # type: (dict, dict, dict, dict, dict) -> None
102         """Insert the new data into the plot and refresh it.
103
104         Parameters
105         ----------
106             desvars, responses, objectives, constraints, metadata : dict
107                 Dictionaries of the new design, response, objective, and constraint variables,
    ↪  as well as metadata.
108         """
109         if len(args) != 5 and all([isinstance(arg, dict) for arg in args]):
110             raise ValueError('Illegal arguments for method _update_plot() of %s' %
    ↪  self.__name__)
111         _, _, objectives, _, _ = args
112
113         if self.first_run:
114             self.first_run = False
115             self.obj_init = objectives.values()[0]
116             self.xdata = np.array([0.])
117             self.ydata = np.array([1.])
118         else:
119             _iter = self.xdata[-1] + 1
120             self.xdata = np.append(self.xdata, [_iter])
121             self.ydata = np.append(self.ydata, [objectives.values()[0]/self.obj_init])
122             self.ax.set_xlim([0, _iter])
123             self.ax.set_ylim([0, 1])
124
125         self.line.set_data(self.xdata, self.ydata)
126         self.ax.relim()
127         self.ax.autoscale_view()
```

Code frament A.11: Code of the `openlego.recorders.simple_objective_plotter` Python module.

### A.3.6. `openlego.recorders.voi_plotter`

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definitions of the 'VOIPlotter' class.
```

```
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from collections import OrderedDict
23
24  import numpy as np
25  from matplotlib import ticker as ticker
26  from matplotlib.figure import Figure
27  from mpl_toolkits import axisartist as aa
28  from mpl_toolkits.axes_grid1.parasite_axes import host_subplot_class_factory
29  from openmdao.core.system import System
30
31  from .base_iteration_plotter import BaseIterationPlotter
32
33
34  class VOIPlotter(BaseIterationPlotter):
35      """Specialized 'BaseIterationPlotter' plotting several variables of interest with a
    ↪  regular line plot.
36
37      Only those variables specified by settings this 'Recorder''s 'includes' property will be
    ↪  plotted. If no variables
38      are specified this way an error is thrown.
39
40      Attributes
41      ----------
42          xdata : :obj:'np.ndarray'
43              Numpy array holding the x-data of the plot.
44
45          vois : dict
46              Python dictionary representing the information of all variables of interest.
47
48          lines : :obj:'list' of 'Line's
49              The 'Line' objects of all variable of interest plots.
50
51      """
52
53      def __init__(self):
54          # type: () -> None
55          """Initializes the 'VOIPlotter'."""
56          super(VOIPlotter, self).__init__()
57
58          self.options.add_option('legend', [],
59                                  desc="List of legend entries for each VOI to be plotted. "
60                                       "If given, this list needs to have the same length as the
    ↪  options['includes']")
61          self.options.add_option('labels', [],
62                                  desc="List of labels to give to each VOI to be plotted. "
63                                       "If given, this list needs to have the same length as the
    ↪  options['includes']")
64          self.options['includes'] = []
65
66          self.xdata = None
67          self.vois = None
68
69          self.lines = None
70
71      def startup(self, object_requesting_recording):
72          # type: (System) -> None
73          """Check 'includes', 'legend', and 'labels' options for validity and compatibility and
    ↪  create 'vois' dictionary.
74
75          Parameters
76          ----------
77              object_requesting_recording : :obj:'System'
78                  'Sytem' that owns this 'Recorder'.
79
80          Raises
81          ------
82              AttributeError
83                  If the 'includes', 'legend', and/or 'labels' are not valid or incompatible
    ↪  with one another.
```

```
84          """
85          if not isinstance(object_requesting_recording, System):
86              raise ValueError('This Recorder must be attached to a System.')
87
88          includes = self.options['includes']
89          legend = self.options['legend']
90          labels = self.options['labels']
91          if len(includes) == 0 or includes[0] == '*':
92              raise AttributeError("At least one variable needs to be put into
    ↪ options['includes']")
93          elif len(legend) > 0 and len(legend) != len(includes):
94              raise AttributeError('Number of legend entries does not match number of included
    ↪ variables')
95          elif len(labels) > 0 and len(labels) != len(includes):
96              raise AttributeError('Number of labels does not match number of included
    ↪ variables')
97
98          if len(labels) == 0:
99              self.options['labels'] = includes[:]
100         if len(legend) == 0:
101             self.options['legend'] = legend[:]
102
103         self.vois = OrderedDict()
104
105         super(VOIPlotter, self).startup(object_requesting_recording)
106
107     def init_fig(self, fig):
108         # type: (Figure) -> None
109         """Initialize the variables of interest figure.
110
111         Parameters
112         ----------
113             fig : :obj:'Figure'
114                 Instance of the 'Figure' which should be populated.
115         """
116         host_subplot_class = host_subplot_class_factory(aa.Axes)
117
118         ax_main = host_subplot_class(fig, 111)
119         fig.add_subplot(ax_main)
120         ax_main.xaxis.set_major_locator(ticker.MaxNLocator(integer=True))
121
122         offset = 60
123         flag = False
124         pos = ['left', 'right']
125
126         for index, key in enumerate(self.options['includes']):
127             if not index:
128                 ax = ax_main
129                 ax.set_xlim([0, 1])
130                 ax.set_xlabel('Evaluation #')
131             else:
132                 ax = ax_main.twinx()
133
134             ax.autoscale(True, 'y')
135             ax.set_ylabel(self.options['labels'][index])
136
137             if index > 1:
138                 new_fixed_axis = ax.get_grid_helper().new_fixed_axis
139                 ax.axis[pos[flag]] = new_fixed_axis(loc=pos[flag], axes=ax,
    ↪ offset=((index//2)*offset, 0))
140                 ax.axis[pos[flag]].toggle(all=True)
141
142             line, = ax.plot([], [], label=self.options['legend'][index])
143             color = line.get_color()
144             ax.axis[pos[flag]].label.set_color(color)
145             ax.spines[pos[flag]].set_color(color)
146             ax.tick_params(axis='y', color=color)
147
148             self.vois.update({key: {'ax': ax, 'line': line, 'data': np.array([])}})
149
150             flag ^= True
```

```
151
152         ax_main.legend(bbox_to_anchor=(.5, 1.), loc='lower center', ncol=4)
153
154     def _update_plot(self, *args):
155         # type: (dict, dict, dict, dict) -> None
156         """Insert the new data into the plot and refresh it.
157
158         Parameters
159         ----------
160             inputs, outputs, resids, metadata : dict
161                 Dictionaries containing inputs, outputs, residuals, and metadata of the system.
162         """
163         inputs, outputs, resids, _ = args
164
165         if self.xdata is None:
166             self.xdata = np.array([0.])
167         else:
168             self.xdata = np.append(self.xdata, [self.xdata[-1] + 1])
169
170         for key in self.vois.keys():
171             if key in inputs:
172                 data = inputs[key]
173             elif key in outputs:
174                 data = outputs[key]
175             elif key in resids:
176                 data = resids[key]
177             else:
178                 raise ValueError('Variable of interest "%s" does not belong to the System
    ↪  holding this Recorder.' % key)
179
180             self.vois[key]['data'] = np.append(self.vois[key]['data'], [data])
181             self.vois[key]['line'].set_data(self.xdata, self.vois[key]['data'])
182             self.vois[key]['ax'].relim()
183             self.vois[key]['ax'].autoscale_view()
```

Code frament A.12: Code of the `openlego.recorders.voi_plotter` Python module.

# A.4. Utilities

## A.4.1. `openlego.utils.general_utils`

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9
10     http://www.apache.org/licenses/LICENSE-2.0
11
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17
18 This file contains the definition of general utility functions.
19 """
20 from __future__ import absolute_import, division, print_function
21
22 import re
23 import warnings
24
25 import numpy as np
26 from lxml import etree
27 from openmdao.core.driver import Driver
28 from typing import Callable, Any, Optional, Union, Type
29
```

```
30
31  def try_hard(fun, *args, **kwargs):
32      # type: (Callable, *Any, **Any) -> Any
33      """Try repeatedly to call a function until it returns successfully.
34
35      Utility function that repeatedly tries to call a given function with the given arguments
    ↪  until that function
36      successfully returns. It is possible to limit the maximum number of attempts by setting
    ↪  the try_hard_limit argument.
37
38
39      Parameters
40      ----------
41          fun : function
42              The function to try to call.
43
44          *args
45              Any ordered arguments to pass to the function.
46
47          **kwargs
48              Any named arguments to pass to the function.
49
50
51      Returns
52      -------
53          any
54              The return value of the function to be called.
55
56      Notes
57      -----
58          This function is part of a workaround to deal with APIs crashing in a non-predictable
    ↪  manner, seemingly random
59          way. It was found that, when simply trying to call such functions again, the problem
    ↪  seemed to not exist
60          anymore.
61      """
62      warnings.simplefilter('always', UserWarning)
63      try_hard_limit = -1
64      if kwargs is not None:
65          if 'try_hard_limit' in kwargs.keys():
66              try_hard_limit = kwargs['try_hard_limit']
67              del kwargs['try_hard_limit']
68
69      msg = None
70      return_value = None
71      successful = False
72      attempts = 0
73      while not successful:
74          try:
75              return_value = fun(*args, **kwargs)
76              successful = True
77          except Exception as e:
78              attempts += 1
79
80              if msg is None:
81                  msg = 'Had to try again to evaluate: %s(' % fun.__name__ + ', '.join(['%s' %
    ↪  arg for arg in args])
82                  if kwargs is not None:
83                      msg += ', '.join(['%s=%s' % (key, value) for key, value in kwargs.items()])
84                  msg += '). The following exception was raised: "%s"' % e.message
85
86              if 0 < try_hard_limit <= attempts:
87                  raise
88              else:
89                  warnings.warn(msg)
90
91      return return_value
92
93
94  class CachedProperty(property):
```

```
95      """Subclass of 'property' using a cache to avoid recalculating an expensive 'property'
    ↪  every time it is read.
96
97      An attribute can be decorated with this class is the same way as with a normal 'property'.
    ↪  It adds the possibility
98      to invalidate the cache when necessary.
99      """
100
101     def __init__(self, fget=None, fset=None, fdel=None, doc=None):
102         # type: (Optional[Callable], Optional[Callable], Optional[Callable], Optional[str]) ->
    ↪  None
103         """Initialize the 'CachedProperty'.
104
105         Parameters
106         ----------
107             fget, fset, fdel : function, optional
108                 Getter, setter, and deleter functions.
109
110             doc : str, optional
111                 Docstring of the property.
112         """
113         super(CachedProperty, self).__init__(fget, fset, fdel, doc)
114         self.__cache = None
115         self.__dirty = True
116
117     def __get__(self, instance, owner=None):
118         # type: (Any, Optional[type]) -> Any
119         """Get the value of the property.
120
121         This method checks if the cache of this property is still valid first. If it is, it
    ↪  simply returns the cached
122         value. If it isn't, it calls the 'super()' to recompute the cached variable, stores
    ↪  it, and then returns the
123         newly calculated value.
124
125         Parameters
126         ----------
127             instance : any
128                 The instance through which the attribute is accessed.
129
130             owner : type, optional
131                 The owner of the attribute.
132
133         Returns
134         -------
135             any
136                 The value of the attribute.
137         """
138         if self.__dirty:
139             self.__cache = super(CachedProperty, self).__get__(instance, owner)
140             self.__dirty = False
141         return self.__cache
142
143     def invalidate(self):
144         # type: () -> None
145         """Marks the cache of the property as invalid, prompting its recomputation the next
    ↪  time it is accessed."""
146         self.__dirty = True
147
148
149 def parse_string(s):
150     # type: (str) -> Union[str, np.ndarray, float]
151     """Convert a string to a numpy array of floats or float if possible.
152
153     The string is returned unchanged if it cannot be converted to a numpy array of floats or
    ↪  float.
154
155     Parameters
156     ----------
157         s : str
158             String to be converted.
```

```
159
160         Returns
161         -------
162             str or np.ndarray or float
163                 Parsed string or the string itself.
164         """
165         v = re.sub(r'[\[\]]', '', s)
166
167         if ',' in v:
168             v = v.split(',')
169         elif ';' in v:
170             v = v.split(';')
171
172         try:
173             v = np.atleast_1d(np.array(v, dtype=float))
174             if v.size == 1:
175                 v = v[0]
176             return v
177         except ValueError:
178             return s
179
180
181     def parse_cmdows_value(elem):
182         # type: (etree._Element) -> Union[str, np.ndarray, float]
183         """Convert an XML element from a CMDOWS file to a value.
184
185         Parameters
186         ----------
187             elem : :obj:'_Element'
188                 'etree._Element' to convert.
189
190         Returns
191         -------
192             str or np.ndarray or float
193                 Converted element.
194         """
195         if len(list(elem)) > 1:
196             return np.array([parse_string(child.text) for child in elem])
197         else:
198             return parse_string(elem.text)
199
200
201     def normalized_to_bounds(driver):
202         # type: (Type[Driver]) -> Type[NormalizedDriver]
203         """Decorate a 'Driver' to adjust its ''adder''/''scaler'' attributes normalizing the
        ↪  ''desvar''s.
204
205         This decorator automatically adjusts the adder and scalar attributes of the design
        ↪  variables belonging to the
206         targeted ''OpenMDAO'' 'Driver' class such that the design variables are normalized to
        ↪  their bounds.
207
208         Parameters
209         ----------
210             driver : :obj:'Driver'
211                 'Driver' to normalize the design variables of.
212
213         Returns
214         -------
215             :obj:'NormalizedDriver'
216                 Instance of 'NormalizedDriver' which inherits from the given 'Driver'.
217
218         Examples
219         --------
220             @normalized_to_bounds\n
221             class MyNormalizedDriver(Driver):
222                 # My design variables will now automatically be normalized to their bounds.
223                 pass
224         """
225
226         class NormalizedDriver(driver):
```

```
227         """Wrapper class for the 'normalized_to_bounds' decorator.
228 
229         This class adds a static function to the 'Driver' it inherits from, which will
    ↪ intercept all 'add_desvar()'
230         calls to the wrapped 'Driver' class to change its ''adder''/''scaler'' attributes
    ↪ depending on the given
231         upper and lower bounds.
232         """
233 
234         @staticmethod
235         def normalize_to_bounds(func):
236             # type: (Callable) -> Callable
237             """Wrap the function handle of the 'add_desvar()' function.
238 
239             Parameters
240             ----------
241                 func : function
242                     Function handle of the 'add_desvar()' function.
243 
244             Returns
245             -------
246                 func : function
247                     Function wrapping the 'add_desvar()' function, which adds logic to
    ↪ calculate ''adder''/''scaler''.
248             """
249 
250             def new_function(name,          # type: str
251                              lower=None,    # type: Optional[Union[float, np.ndarray]]
252                              upper=None,    # type: Optional[Union[float, np.ndarray]]
253                              *args, **kwargs):
254                 # type: (...) -> None
255                 """Wrap the 'add_desvar()' function call.
256 
257                 Inner wrapper function which will set the ''adder'' and ''scaler'' 'kwargs' of
    ↪ the wrapped
258                 'add_desvar()' method before calling it.
259 
260                 Parameters
261                 ----------
262                     name : str
263                         Name of the design variable to add.
264 
265                     lower : float or list of float, optional
266                         Lower bound(s) of the design variable.
267 
268                     upper : float or list of float, optional
269                         Upper bound(s) of the design variable.
270 
271                     *args
272                         Any extra, ordered arguments to pass to the 'add_desvar()' method.
273 
274                     **kwargs
275                         Any extra, named arguments to pass to the 'add_desvar()' method.
276                 """
277                 if lower is not None:
278                     adder = -lower
279                 else:
280                     adder = 0.
281 
282                 if upper is not None:
283                     scaler = 1./(upper + adder)
284                 else:
285                     scaler = 1.
286 
287                 if len(args) > 4:
288                     args = args[:-1]
289                 elif len(args) > 3:
290                     args = args[:-1]
291 
292                 if 'adder' in kwargs:
293                     del kwargs['adder']
```

```
294                 if 'scaler' in kwargs:
295                     del kwargs['scaler']
296
297                 func(name, lower, upper, adder=adder, scaler=scaler, *args, **kwargs)
298
299             return new_function
300
301         def __getattribute__(self, item):
302             """Intercept any calls to the 'add_desvar()' method of the Driver class.
303
304             This ''hook'' checks if 'add_desvar()' is called. If so, it returns the wrapped
    ↪ function instead of the
305             clean 'add_desvar()' call.
306
307             Parameters
308             ----------
309                 item : str
310                     Name of the attribute.
311
312             Returns
313             -------
314                 any
315                     The attribute that was requested or the wrapped call to 'add_desvar()' if
    ↪ it is requested.
316             """
317             x = super(NormalizedDriver, self).__getattribute__(item)
318             if item in ['add_desvar']:
319                 return self.normalize_to_bounds(x)
320             else:
321                 return x
322
323     return NormalizedDriver
```

Code frament A.13: Code of the `openlego.utils.general_utils` Python module.

### A.4.2. `openlego.utils.xml_utils`

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains a set of XML utility functions.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  import re
23  from collections import OrderedDict
24  from shutil import copyfile
25
26  import numpy as np
27  from lxml import etree
28  from typing import Optional, Union, List
29
30  # Patterns for XML attribute names and values
31  pttrn_attr_val = r'([-.0-9:A-Z_a-z]+?)'
32  pttrn_attr_name = r'([:A-Z_a-z][0-9:A-Z_a-z]*?)'
33
```

```
# Expressions used to replace illegal characters in an XPath to legal characters within an OpenMDAO variable name.
repl_dot = ':_:'                # A dot (.) becomes :_:
repl_dot_inv = '.'

# Regular expressions to match attributes and indices within valid XPaths
re_atr = re.compile(r'\[@' + pttrn_attr_name + "=['\"]" + pttrn_attr_val + "['\"]\]")
re_ind = re.compile(r'\[([0-9]+?)\]')

parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
find_text = etree.XPath('//text()')


def xpath_to_param(xpath):
    # type: (str) -> str
    """Convert an XML XPath to a valid ''OpenMDAO'' parameter name.

    Parameters
    ----------
        xpath : str
            XPath to convert.

    Returns
    -------
        str
            Valid ''OpenMDAO'' parameter name.
    """
    param = xpath.replace(repl_dot_inv, repl_dot)
    return param


def param_to_xpath(param):
    # type: (str) -> str
    """Convert an ''OpenMDAO'' parameter name to the corresponding XML XPath.

    This function is the inverse of 'xpath_to_param()'.

    Parameters
    ----------
        param : str
            Valid ''OpenMDAO'' parameter name.

    Returns
    -------
        str
            Corresponding XML XPath.
    """
    xpath = param.replace(repl_dot, repl_dot_inv)
    return xpath


def value_to_xml(elem, value):
    if isinstance(value, np.ndarray):
        value = np.atleast_1d(value).flatten()

    if isinstance(value, np.ndarray):
        if value.size == 1:
            elem.text = str(value[0])
        else:
            elem.text = ';'.join([str(v) for v in value[:]])
            elem.attrib.update({'mapType': 'vector'})
    else:
        elem.text = str(value)


def xml_to_dict(xml):
    # type: (Union[str, etree._ElementTree]) -> OrderedDict
    """Convert an XML file to a python dictionary with all valued elements as values with their full XPaths as keys.

    Parameters
```

```
103         ----------
104             xml : str or :obj:'etree._ElementTree'
105                 Path to or 'etree._ElementTree' of an XML file.
106 
107         Returns
108         -------
109             :obj:'OrderedDict'
110                 'OrderedDict' representing the XML file in file order.
111         """
112         if isinstance(xml, str):
113             xml = etree.parse(xml, parser)
114 
115         _dict = OrderedDict()
116         for text in find_text(xml):
117             # Construct 'augmented' XPath for this element, including attributes
118             xpath = ''
119             child = text.getparent()
120             while child is not None:
121                 parent = child.getparent()
122 
123                 tag = child.tag
124                 for name, value in child.items():
125                     # Exclude special purpose attribute: mapType
126                     if name != 'mapType':
127                         tag += r'[@%s="%s"]' % (name, value)
128 
129                 if parent is not None:
130                     siblings = parent.findall(tag)
131                     if len(siblings) > 1:
132                         tag += '[%d]' % (siblings.index(child) + 1)
133 
134                 xpath = '/'.join([tag, xpath])
135                 child = parent
136 
137             # Try to convert the text into a float or a list of floats
138             try:
139                 value = float(text)
140             except ValueError:
141                 try:
142                     value = np.array(text.split(';'), dtype=float)
143                 except ValueError:
144                     value = str(text)
145 
146             # Update the dict with this element
147             _dict.update({'/' + xpath[:-1]: value})
148 
149         return _dict
150 
151 
152 def xml_safe_create_element(
153             tree,       # type: etree._ElementTree
154             xpath,      # type: str
155             value=None  # type: Optional[Union[str, int, float, List[Union[str, int, float]],
      ↪  np.ndarray]]
156 ):
157         # type: (...) -> etree._Element
158         """Create an element at the given XML XPath with the given value.
159 
160         This method ensures that all elements implied by the given X-Path exist.
161 
162         Supplying a value is optional. If no value is supplied an empty XML node is created at the
      ↪  deepest level implied by
163         the XPath.
164 
165         Parameters
166         ----------
167             tree : :obj:'etree._ElementTree'
168                 'etree._ElementTree' in which to create the element.
169 
170             xpath : str
171                 XPath to ensure.
```

```
172
173         value : str or int or float or list of str or list of int or list of float or
    ↪  :obj:'np.ndarray'
174             Optional value to write at the deepest node of the ensured XPath.
175
176     Returns
177     -------
178         :obj:'etree._Element'
179             Instance of 'etree._Element' corresponding to the newly created element.
180     """
181     # Split the xpath to get the intermediate nodes as a list
182     xpath_list = xpath.split('/')
183     n = len(xpath_list)
184
185     # Loop over the elements in the XPath from tip to root until the XPath is found to already
    ↪  exist
186     elem = None
187     i = 0
188     for i in range(0, n - 1):
189         xpath = '/'.join(xpath_list[0:(n - i)])
190         try:
191             elems = tree.xpath(xpath)
192             if len(elems):
193                 elem = elems[0]
194                 break
195         except etree.XPathError:
196             raise ValueError('Specified XPath is invalid')
197
198     # If no existing element was found the root elements of the tree and XPath don't match
199     if elem is None:
200         raise ValueError("Specified XPath is incompatible with the given XML tree: root tags
    ↪  don't match")
201
202     # Loop over the part of the XPath beyond this point and create all intermediate elements
    ↪  including attributes
203     for j in range(n - i, n):
204         tag = xpath_list[j]
205
206         # See if this node has an integer index specified
207         match_ind = re_ind.search(tag)
208         if match_ind:
209             tag = tag[:match_ind.start()] + tag[match_ind.end():]
210             index = int(match_ind.group(1)) - 1
211         else:
212             index = 0
213
214         # Find any attributes at this node
215         attrib = {}
216         match_attr = list(re_atr.finditer(tag))
217         if match_attr:
218             # Loop over all attributes on this node
219             for match in match_attr:
220                 if match.start() < len(tag):
221                     tag = tag[:match.start()]
222                 attrib.update({match.group(1): match.group(2)})
223
224         # Check if there are siblings with the same name
225         siblings = elem.findall(tag)
226         n_siblings = len(siblings)
227
228         # Check if there's a sibling with the same name at this index without conflicting
    ↪  attributes
229         if index < n_siblings and not any(
230                 [siblings[index].attrib[key] != attrib[key] for key in attrib.keys() if key in
    ↪  siblings[index].attrib]):
231             # If so, use it instead of adding a new one
232             siblings[index].attrib.update(attrib)
233             elem = siblings[index]
234         elif index <= n_siblings:
235             # In this case just append a new element
236             _elem = etree.Element(tag, attrib)
```

```
237             elem.append(_elem)
238             elem = _elem
239         else:
240             # In the last case, insert as many empty siblings until this node's index
241             sibling = None
242             for i in range(index - n_siblings):
243                 _sibling = etree.Element(tag)
244                 if i == 0:
245                     if not n_siblings:
246                         elem.append(_sibling)
247                     else:
248                         siblings[-1].addnext(_sibling)
249                 else:
250                     sibling.addnext(_sibling)
251                 sibling = _sibling
252 
253             # Finally at a new element at the right index with all attributes
254             elem = etree.Element(tag, attrib)
255             sibling.addnext(elem)
256 
257         # Finally we can update the current XPath, since it has been assured to exist at this
    ↪ point
258         xpath = '/'.join([xpath, xpath_list[j]])
259 
260     # If a value was supplied assign it to the deepest element in the XPath
261     if value is not None:
262         value_to_xml(elem, value)
263 
264     return elem
265 
266 
267 def xml_merge(base, merger, out_file=None):
268     # type: (Union[str, etree._ElementTree], Union[str, etree._ElementTree], Optional[str])
    ↪  -> None
269     """Merge an XML file into another.
270 
271     First two parameters can be either a path to an XML file or an instance of
    ↪  'etree._ElementTree' corresponding to an
272     XML tree. All content from the merger will be merger into the base. The third parameter is
    ↪  optional. If set, the
273     result of the merger will be written to the file at this path.
274 
275     This function does not return anything. If  base is an instance of 'etree._ElementTree'
    ↪  this object will be changed,
276     if it is a 'str' the file at that location will be changed. However, if ''out_file'' is
    ↪  set, the file at that
277     location will be changed instead, and not the one at base.
278 
279     Parameters
280     ----------
281         base : str or :obj:'etree._ElementTree'
282             Path to or 'etree._ElementTree' of an XML file into which the merger should be
    ↪  merged.
283 
284         merger : str or :obj:'etree._ElementTree'
285            Path to or 'etree._ElementTree' of an XML file which should be merged into the base.
286 
287         out_file : str, optional
288             Path to a file into which the result of the merger should be written. If not
    ↪  given, the result will
289             overwrite the base.
290 
291     Notes
292     -----
293         If conflicting elements exist the value of the merger will overwrite the one in the
    ↪  base.
294     """
295     if isinstance(base, str):
296         try:
297             doc = etree.parse(base, parser)
298         except IOError:
```

```
299             if out_file is None:
300                 out_file = base
301 
302             if isinstance(merger, str):
303                 copyfile(merger, out_file)
304             else:
305                 merger.write(out_file, encoding='utf-8', pretty_print=True,
    ↪  xml_declaration=True)
306 
307             return
308     else:
309         doc = base
310 
311     merger_dict = xml_to_dict(merger)
312     for xpath, value in merger_dict.items():
313         xml_safe_create_element(doc, xpath, value)
314 
315     if out_file is not None:
316         doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
317     elif isinstance(base, str):
318         doc.write(base, encoding='utf-8', pretty_print=True, xml_declaration=True)
```

Code frament A.14: Code of the `openlego.utils.xml_utils` Python module.

# A.5. Test Suite

## A.5.1. Sellar Problem

### Sellar Test Case

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5 
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9 
10     http://www.apache.org/licenses/LICENSE-2.0
11 
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17 
18 This file contains the test case for the Sellar example problem.
19 """
20 from __future__ import absolute_import, division, print_function
21 
22 import unittest
23 
24 from openmdao.api import Problem, ScipyOptimizer
25 
26 from openlego.api import LEGOModel
27 from openlego.recorders import NormalizedDesignVarPlotter, ConstraintsPlotter,
    ↪  SimpleObjectivePlotter
28 
29 
30 def solve_sellar(cmdows_file):
31     """Solve the Sellar problem using the given CMDOWS file."""
32     # 1. Create Problem
33     prob = Problem()                                        # Create an instance of the
    ↪  Problem class
34     prob.set_solver_print(0)                                  # Turn off printing of
    ↪  solver information
35 
36     # 2. Create the LEGOModel
37     model = prob.model = LEGOModel(cmdows_file,                    # CMDOWS file
```

```
38                                      'kb',                                   # Knowledge base path
39                                      '',                                     # Output directory
40                                      'sellar-output.xml')                    # Output file
41
42      # 3. Create the Driver
43      driver = prob.driver = ScipyOptimizer()                                 # Use a SciPy for the
    ↪  optimization
44      driver.options['optimizer'] = 'SLSQP'                                   # Use the SQP algorithm
45      driver.options['disp'] = True                                           # Print the result
46      driver.opt_settings = {'disp': True, 'iprint': 2}                       # Display iterations
47
48      # 4. Setup the Problem
49      prob.setup()                                                           # Call the OpenMDAO setup()
    ↪  method
50      model.coupled_group.linear_solver.options['maxiter'] = 17               # Increase maxiter of the
    ↪  linear solver
51      model.coupled_group.nonlinear_solver.options['maxiter'] = 17            # Increase maxiter of the
    ↪  nonlinear solver
52      prob.run_model()                                                        # Run the model once to
    ↪  init. the variables
53      model.initialize_from_xml('sellar-input.xml')                           # Set the initial values
    ↪  from an XML file
54
55      # 5. Create and attach some Recorders (Optional)
56      desvar_plotter = NormalizedDesignVarPlotter()                          # Create a plotter for the
    ↪  design variables
57      desvar_plotter.options['save_on_close'] = True                         # Should this plot be saved
    ↪  automatically?
58      desvar_plotter.save_settings['path'] = 'desvar.png'                     # Set the filename of the
    ↪  image file
59
60      convar_plotter = ConstraintsPlotter()                                  # Create a plotter for the
    ↪  constraints
61      convar_plotter.options['save_on_close'] = True                         # Should this plot be saved
    ↪  automatically?
62      convar_plotter.save_settings['path'] = 'convar.png'                     # Set the filename of the
    ↪  image file
63
64      objvar_plotter = SimpleObjectivePlotter()                              # Create a plotter for the
    ↪  objective
65      objvar_plotter.options['save_on_close'] = True                         # Should this plot be saved
    ↪  automatically?
66      objvar_plotter.save_settings['path'] = 'objvar.png'                     # Set the filename of the
    ↪  image file
67
68      # driver.add_recorder(desvar_plotter)                                     # Attach the design
    ↪  variable plotter
69      # driver.add_recorder(convar_plotter)                                     # Attach the constraint
    ↪  variable plotter
70      # driver.add_recorder(objvar_plotter)                                     # Attach the objective
    ↪  variable plotter
71
72      # 6. Solve the Problem
73      prob.run_driver()                                                       # Run the optimization
74
75      # 7. Print results
76      from .kb import x_f1, x_x1, x_z1, x_z2, x_y1, x_y2, x_g1, x_g2
77      print('Optimum found! Objective function value: f = {}'.format(prob[x_f1]))
78      print('Design variables at optimum: x = {}, z1 = {}, z2 = {}'.format(prob[x_x1],
    ↪  prob[x_z1], prob[x_z2]))
79      print('Coupling variables at optimum: y1 = {}, y2 = {}'.format(prob[x_y1], prob[x_y2]))
80      print('Constraints at optimum: g1 = {}, g2 = {}'.format(prob[x_g1], prob[x_g2]))
81
82      # 8. Cleanup the Problem afterwards
83      prob.cleanup()                                                          # Clear all resources and
    ↪  close the plots
84      model.invalidate()                                                      # Clear the cached
    ↪  properties of the LEGOModel
85
86
87  class TestSellar(unittest.TestCase):
```

```
88
89      def test_mdf_gs(self):
90          """Solve the Sellar problem using the MDF architecture and a Gauss-Siedel converger."""
91          solve_sellar('sellar-MDG_MDF-GS.xml')
92
93      def test_mdf_j(self):
94          """Solve the Sellar problem using the MDF architecture and a Jacobi converger."""
95          solve_sellar('sellar-MDG_MDF-J.xml')
96
97      def test_idf(self):
98          """Solve the Sellar problem using the IDF architecture."""
99          solve_sellar('sellar-MDG_IDF.xml')
100
101
102 if __name__ == '__main__':
103     unittest.main()
```

Code frament A.15: Code of the `test_sellar` Python script.

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <cmdows xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     ↪ xsi:noNamespaceSchemaLocation="https://bitbucket.org/imcovangent/cmdows/raw/master/schema/0.7/cmdows.xsd">
3    <header>
4      <creator>D. de Vries</creator>
5      <description>Sellar problem MPG file</description>
6      <timestamp>2017-10-09T11:33:54.624000</timestamp>
7      <fileVersion>0.1</fileVersion>
8      <cmdowsVersion>0.7</cmdowsVersion>
9      <updates>
10       <update>
11         <modification>KADMOS export of a mdao data graph (MDG).</modification>
12         <creator>D. de Vries</creator>
13         <timestamp>2017-10-09T11:33:54.624000</timestamp>
14         <fileVersion>0.1</fileVersion>
15         <cmdowsVersion>0.7</cmdowsVersion>
16       </update>
17     </updates>
18   </header>
19   <executableBlocks>
20     <designCompetences>
21       <designCompetence uID="F1">
22         <ID>F1</ID>
23         <modeID>main</modeID>
24         <instanceID>1</instanceID>
25         <version>1.0</version>
26         <label>F1</label>
27         <inputs>
28           <input>
29             <parameterUID>/data_schema/x1</parameterUID>
30           </input>
31           <input>
32             <parameterUID>/data_schema/y2</parameterUID>
33           </input>
34           <input>
35             <parameterUID>/data_schema/z2</parameterUID>
36           </input>
37           <input>
38             <parameterUID>/data_schema/y1</parameterUID>
39           </input>
40         </inputs>
41         <outputs>
42           <output>
43             <parameterUID>/data_schema/f1</parameterUID>
44           </output>
45           <output>
46             <parame-
     ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</parameterUID>
47           </output>
48         </outputs>
49         <metadata>
```

```
50          <generalInfo>
51            <description>main execution mode</description>
52          </generalInfo>
53        </metadata>
54       </designCompetence>
55       <designCompetence uID="D2">
56        <ID>D2</ID>
57        <modeID>main</modeID>
58        <instanceID>1</instanceID>
59        <version>1.0</version>
60        <label>D2</label>
61        <inputs>
62          <input>
63            <parameterUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</parameterUID>
64          </input>
65          <input>
66            <parameterUID>/data_schema/z2</parameterUID>
67          </input>
68          <input>
69            <parameterUID>/data_schema/z1</parameterUID>
70          </input>
71        </inputs>
72        <outputs>
73          <output>
74            <parameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</parameterUID>
75          </output>
76          <output>
77            <parameterUID>/data_schema/y2</parameterUID>
78          </output>
79        </outputs>
80        <metadata>
81          <generalInfo>
82            <description>main execution mode</description>
83          </generalInfo>
84        </metadata>
85       </designCompetence>
86       <designCompetence uID="G2">
87        <ID>G2</ID>
88        <modeID>main</modeID>
89        <instanceID>1</instanceID>
90        <version>1.0</version>
91        <label>G2</label>
92        <inputs>
93          <input>
94            <parameterUID>/data_schema/y2</parameterUID>
95          </input>
96        </inputs>
97        <outputs>
98          <output>
99            <parameterUID>/data_schema/g2</parameterUID>
100         </output>
101         <output>
102           <parameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</parameterUID>
103         </output>
104       </outputs>
105       <metadata>
106         <generalInfo>
107           <description>main execution mode</description>
108         </generalInfo>
109       </metadata>
110      </designCompetence>
111      <designCompetence uID="G1">
112       <ID>G1</ID>
113       <modeID>main</modeID>
114       <instanceID>1</instanceID>
115       <version>1.0</version>
116       <label>G1</label>
117       <inputs>
```

```
118           <input>
119             <parameterUID>/data_schema/y1</parameterUID>
120           </input>
121         </inputs>
122         <outputs>
123           <output>
124             <parame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</parameterUID>
125           </output>
126           <output>
127             <parameterUID>/data_schema/g1</parameterUID>
128           </output>
129         </outputs>
130         <metadata>
131           <generalInfo>
132             <description>main execution mode</description>
133           </generalInfo>
134         </metadata>
135       </designCompetence>
136       <designCompetence uID="D1">
137         <ID>D1</ID>
138         <modeID>main</modeID>
139         <instanceID>1</instanceID>
140         <version>1.0</version>
141         <label>D1</label>
142         <inputs>
143           <input>
144             <parameterUID>/data_schema/x1</parameterUID>
145           </input>
146           <input>
147             <parameterUID>/data_schema/z2</parameterUID>
148           </input>
149           <input>
150             <parameterUID>/data_schema/z1</parameterUID>
151           </input>
152           <input>
153             <parame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</parameterUID>
154           </input>
155         </inputs>
156         <outputs>
157           <output>
158             <parame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</parameterUID>
159           </output>
160           <output>
161             <parameterUID>/data_schema/y1</parameterUID>
162           </output>
163         </outputs>
164         <metadata>
165           <generalInfo>
166             <description>main execution mode</description>
167           </generalInfo>
168         </metadata>
169       </designCompetence>
170     </designCompetences>
171   </executableBlocks>
172   <parameters>
173     <parameter uID="/data_schema/g1">
174       <label>g1</label>
175     </parameter>
176     <parameter uID="/data_schema/g2">
177       <label>g2</label>
178     </parameter>
179     <parameter uID="/data_schema/f1">
180       <label>f1</label>
181     </parameter>
182     <parameter uID="/data_schema/y2">
183       <label>y2</label>
184     </parameter>
185     <parameter uID="/data_schema/y1">
```

```
      <label>y1</label>
    </parameter>
    <parameter uID="/data_schema/x1">
      <label>x1</label>
    </parameter>
    <parameter uID="/data_schema/z2">
      <label>z2</label>
    </parameter>
    <parameter uID="/data_schema/z1">
      <label>z1</label>
    </parameter>
  </parameters>
  <problemDefinition uID="IDFNone">
    <problemFormulation>
      <mdaoArchitecture>IDF</mdaoArchitecture>
      <executableBlocksOrder>
        <executableBlock position="1">D1</executableBlock>
        <executableBlock position="2">D2</executableBlock>
        <executableBlock position="3">F1</executableBlock>
        <executableBlock position="4">G1</executableBlock>
        <executableBlock position="5">G2</executableBlock>
      </executableBlocksOrder>
      <allowUnconvergedCouplings>false</allowUnconvergedCouplings>
    </problemFormulation>
    <problemRoles>
      <parameters>
        <designVariables>
          <designVariable uID="__desVar__/data_schema/y2">
            <parameterUID>/data_schema/y2</parameterUID>
            <nominalValue>0.0</nominalValue>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/y1">
            <parameterUID>/data_schema/y1</parameterUID>
            <nominalValue>0.0</nominalValue>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/x1">
            <parameterUID>/data_schema/x1</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
              <limitRange>
                <minimum>0.0</minimum>
                <maximum>10.0</maximum>
              </limitRange>
            </validRanges>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/z2">
            <parameterUID>/data_schema/z2</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
              <limitRange>
                <minimum>0.0</minimum>
                <maximum>10.0</maximum>
              </limitRange>
            </validRanges>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/z1">
            <parameterUID>/data_schema/z1</parameterUID>
            <nominalValue>1.0</nominalValue>
            <validRanges>
              <limitRange>
                <minimum>-10.0</minimum>
                <maximum>10.0</maximum>
              </limitRange>
            </validRanges>
          </designVariable>
        </designVariables>
        <objectiveVariables>
          <objectiveVariable uID="__objVar__/data_schema/f1">
            <parameterUID>/data_schema/f1</parameterUID>
          </objectiveVariable>
        </objectiveVariables>
```

```
257         <constraintVariables>
258           <constraintVariable uID="__conVar__/data_schema/g1">
259             <parameterUID>/data_schema/g1</parameterUID>
260             <constraintType>inequality</constraintType>
261             <constraintOperator>&lt;=</constraintOperator>
262             <referenceValue>0.0</referenceValue>
263           </constraintVariable>
264           <constraintVariable uID="__conVar__/data_schema/g2">
265             <parameterUID>/data_schema/g2</parameterUID>
266             <constraintType>inequality</constraintType>
267             <constraintOperator>&lt;=</constraintOperator>
268             <referenceValue>0.0</referenceValue>
269           </constraintVariable>
270           <constraintVariable
    ↪ uID="__conVar__/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y2">
271             <parame-
    ↪ terUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y2</parameterUID>
272             <constraintType>equality</constraintType>
273             <constraintOperator>==</constraintOperator>
274             <referenceValue>0.0</referenceValue>
275           </constraintVariable>
276           <constraintVariable
    ↪ uID="__conVar__/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y1">
277             <parame-
    ↪ terUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y1</parameterUID>
278             <constraintType>equality</constraintType>
279             <constraintOperator>==</constraintOperator>
280             <referenceValue>0.0</referenceValue>
281           </constraintVariable>
282         </constraintVariables>
283       </parameters>
284       <executableBlocks>
285         <coupledBlocks>
286           <coupledBlock>D1</coupledBlock>
287           <coupledBlock>D2</coupledBlock>
288         </coupledBlocks>
289         <postCouplingBlocks>
290           <postCouplingBlock>F1</postCouplingBlock>
291           <postCouplingBlock>G1</postCouplingBlock>
292           <postCouplingBlock>G2</postCouplingBlock>
293           <postCouplingBlock>Gc</postCouplingBlock>
294         </postCouplingBlocks>
295       </executableBlocks>
296     </problemRoles>
297   </problemDefinition>
298   <workflow>
299     <problemDefinitionUID>IDFNone</problemDefinitionUID>
300     <dataGraph>
301       <name>MDG1</name>
302       <edges>
303         <edge>
304           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</fromParameterUID>
305           <toExecutableBlockUID>D2</toExecutableBlockUID>
306         </edge>
307         <edge>
308           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</fromParameterUID>
309           <toExecutableBlockUID>Gc</toExecutableBlockUID>
310         </edge>
311         <edge>
312           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</fromParameterUID>
313           <toExecutableBlockUID>Gc</toExecutableBlockUID>
314         </edge>
315         <edge>
316           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</fromParameterUID>
317           <toExecutableBlockUID>D1</toExecutableBlockUID>
318         </edge>
319         <edge>
```

```
320           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1</fromParameter
321           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
322         </edge>
323         <edge>
324           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</fromParameter
325           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
326         </edge>
327         <edge>
328           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</fromParameterUID>
329           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
330         </edge>
331         <edge>
332           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</fromParameterUID>
333           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
334         </edge>
335         <edge>
336           <fromExecutableBlockUID>Gc</fromExecutableBlockUID>
337           <toParame-
      ↪ terUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y2</toParame
338         </edge>
339         <edge>
340           <fromExecutableBlockUID>Gc</fromExecutableBlockUID>
341           <toParame-
      ↪ terUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y1</toParame
342         </edge>
343         <edge>
344           <fromParameterUID>/data_schema/y2</fromParameterUID>
345           <toExecutableBlockUID>F1</toExecutableBlockUID>
346         </edge>
347         <edge>
348           <fromParameterUID>/data_schema/y2</fromParameterUID>
349           <toExecutableBlockUID>Gc</toExecutableBlockUID>
350         </edge>
351         <edge>
352           <fromParameterUID>/data_schema/y2</fromParameterUID>
353           <toExecutableBlockUID>G2</toExecutableBlockUID>
354         </edge>
355         <edge>
356           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
357           <toParameterUID>/data_schema/g2</toParameterUID>
358         </edge>
359         <edge>
360           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
361           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</toParameterUID>
362         </edge>
363         <edge>
364           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
365           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</toParameterUID>
366         </edge>
367         <edge>
368           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
369           <toParameterUID>/data_schema/g1</toParameterUID>
370         </edge>
371         <edge>
372           <fromParameterUID>/data_schema/y1</fromParameterUID>
373           <toExecutableBlockUID>F1</toExecutableBlockUID>
374         </edge>
375         <edge>
376           <fromParameterUID>/data_schema/y1</fromParameterUID>
377           <toExecutableBlockUID>Gc</toExecutableBlockUID>
378         </edge>
379         <edge>
380           <fromParameterUID>/data_schema/y1</fromParameterUID>
381           <toExecutableBlockUID>G1</toExecutableBlockUID>
382         </edge>
```

```
      <edge>
        <fromParameterUID>/data_schema/g1</fromParameterUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/g2</fromParameterUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</fromParameterUID>
        <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/x1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/z2</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toParameterUID>/data_schema/z1</toParameterUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/f1</fromParameterUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</fromParameterUID>
        <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>F1</fromExecutableBlockUID>
        <toParameterUID>/data_schema/f1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>F1</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</toParameterUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/x1</fromParameterUID>
        <toExecutableBlockUID>F1</toExecutableBlockUID>
```

```
446         </edge>
447         <edge>
448           <fromParameterUID>/data_schema/x1</fromParameterUID>
449           <toExecutableBlockUID>D1</toExecutableBlockUID>
450         </edge>
451         <edge>
452           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</fromParameterUID>
453           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
454         </edge>
455         <edge>
456           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</fromParameterUID>
457           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
458         </edge>
459         <edge>
460           <fromParameterUID>/data_schema/z2</fromParameterUID>
461           <toExecutableBlockUID>F1</toExecutableBlockUID>
462         </edge>
463         <edge>
464           <fromParameterUID>/data_schema/z2</fromParameterUID>
465           <toExecutableBlockUID>D2</toExecutableBlockUID>
466         </edge>
467         <edge>
468           <fromParameterUID>/data_schema/z2</fromParameterUID>
469           <toExecutableBlockUID>D1</toExecutableBlockUID>
470         </edge>
471         <edge>
472           <fromParameterUID>/data_schema/z1</fromParameterUID>
473           <toExecutableBlockUID>D2</toExecutableBlockUID>
474         </edge>
475         <edge>
476           <fromParameterUID>/data_schema/z1</fromParameterUID>
477           <toExecutableBlockUID>D1</toExecutableBlockUID>
478         </edge>
479         <edge>
480           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
481           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1</toParameterUI
482         </edge>
483         <edge>
484           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
485           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</toParameterUID>
486         </edge>
487         <edge>
488           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
489           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</toParameterUI
490         </edge>
491         <edge>
492           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
493           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</toParameterUID>
494         </edge>
495         <edge>
496           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
497           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</toParameterUID>
498         </edge>
499         <edge>
500           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</fromParameterUI
501           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
502         </edge>
503         <edge>
504           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</fromParameterUI
505           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
506         </edge>
507         <edge>
```

```
        <fromParameterUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</fromParameterUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</fromParameterUID>
        <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</fromParameterUID>
        <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>D2</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>D2</fromExecutableBlockUID>
        <toParameterUID>/data_schema/y2</toParameterUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y2</fromParameterU
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromParameterUID>/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y1</fromParameterU
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>D1</fromExecutableBlockUID>
        <toParameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>D1</fromExecutableBlockUID>
        <toParameterUID>/data_schema/y1</toParameterUID>
      </edge>
    </edges>
  </dataGraph>
  <processGraph>
    <name>MPG1</name>
    <edges>
      <edge>
        <fromExecutableBlockUID>F1</fromExecutableBlockUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
        <processStepNumber>4</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
        <processStepNumber>5</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toExecutableBlockUID>D2</toExecutableBlockUID>
        <processStepNumber>2</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
        <toExecutableBlockUID>D1</toExecutableBlockUID>
        <processStepNumber>2</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>G2</fromExecutableBlockUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
        <processStepNumber>4</processStepNumber>
```

```
572         </edge>
573         <edge>
574           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
575           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
576           <processStepNumber>4</processStepNumber>
577         </edge>
578         <edge>
579           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
580           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
581           <processStepNumber>1</processStepNumber>
582         </edge>
583         <edge>
584           <fromExecutableBlockUID>Gc</fromExecutableBlockUID>
585           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
586           <processStepNumber>4</processStepNumber>
587         </edge>
588         <edge>
589           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
590           <toExecutableBlockUID>F1</toExecutableBlockUID>
591           <processStepNumber>3</processStepNumber>
592         </edge>
593         <edge>
594           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
595           <toExecutableBlockUID>Gc</toExecutableBlockUID>
596           <processStepNumber>3</processStepNumber>
597         </edge>
598         <edge>
599           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
600           <toExecutableBlockUID>G2</toExecutableBlockUID>
601           <processStepNumber>3</processStepNumber>
602         </edge>
603         <edge>
604           <fromExecutableBlockUID>D1</fromExecutableBlockUID>
605           <toExecutableBlockUID>F1</toExecutableBlockUID>
606           <processStepNumber>3</processStepNumber>
607         </edge>
608         <edge>
609           <fromExecutableBlockUID>D1</fromExecutableBlockUID>
610           <toExecutableBlockUID>Gc</toExecutableBlockUID>
611           <processStepNumber>3</processStepNumber>
612         </edge>
613         <edge>
614           <fromExecutableBlockUID>D1</fromExecutableBlockUID>
615           <toExecutableBlockUID>G1</toExecutableBlockUID>
616           <processStepNumber>3</processStepNumber>
617         </edge>
618       </edges>
619       <nodes>
620         <node>
621           <referenceUID>F1</referenceUID>
622           <processStepNumber>3</processStepNumber>
623           <diagonalPosition>4</diagonalPosition>
624         </node>
625         <node>
626           <referenceUID>Optimizer</referenceUID>
627           <processStepNumber>1</processStepNumber>
628           <convergerStepNumber>4</convergerStepNumber>
629           <diagonalPosition>1</diagonalPosition>
630         </node>
631         <node>
632           <referenceUID>G2</referenceUID>
633           <processStepNumber>3</processStepNumber>
634           <diagonalPosition>6</diagonalPosition>
635         </node>
636         <node>
637           <referenceUID>G1</referenceUID>
638           <processStepNumber>3</processStepNumber>
639           <diagonalPosition>5</diagonalPosition>
640         </node>
641         <node>
642           <referenceUID>Coordinator</referenceUID>
```

```
643             <processStepNumber>0</processStepNumber>
644             <convergerStepNumber>5</convergerStepNumber>
645             <diagonalPosition>0</diagonalPosition>
646           </node>
647           <node>
648             <referenceUID>Gc</referenceUID>
649             <processStepNumber>3</processStepNumber>
650             <diagonalPosition>7</diagonalPosition>
651           </node>
652           <node>
653             <referenceUID>D2</referenceUID>
654             <processStepNumber>2</processStepNumber>
655             <diagonalPosition>3</diagonalPosition>
656           </node>
657           <node>
658             <referenceUID>D1</referenceUID>
659             <processStepNumber>2</processStepNumber>
660             <diagonalPosition>2</diagonalPosition>
661           </node>
662         </nodes>
663       </processGraph>
664     </workflow>
665     <architectureElements>
666       <parameters>
667         <initialGuessCouplingVariables>
668           <initialGuessCouplingVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1">
669             <relatedParameterUID>/data_schema/y1</relatedParameterUID>
670             <label>y1^{c0}</label>
671           </initialGuessCouplingVariable>
672           <initialGuessCouplingVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2">
673             <relatedParameterUID>/data_schema/y2</relatedParameterUID>
674             <label>y2^{c0}</label>
675           </initialGuessCouplingVariable>
676         </initialGuessCouplingVariables>
677         <finalCouplingVariables>
678           <finalCouplingVariable
    ↪ uID="/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2">
679             <relatedParameterUID>/data_schema/y2</relatedParameterUID>
680             <label>y2^*</label>
681           </finalCouplingVariable>
682           <finalCouplingVariable
    ↪ uID="/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1">
683             <relatedParameterUID>/data_schema/y1</relatedParameterUID>
684             <label>y1^*</label>
685           </finalCouplingVariable>
686         </finalCouplingVariables>
687         <couplingCopyVariables>
688           <couplingCopyVariable
    ↪ uID="/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1">
689             <relatedParameterUID>/data_schema/y1</relatedParameterUID>
690             <label>y1^c</label>
691           </couplingCopyVariable>
692           <couplingCopyVariable
    ↪ uID="/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2">
693             <relatedParameterUID>/data_schema/y2</relatedParameterUID>
694             <label>y2^c</label>
695           </couplingCopyVariable>
696         </couplingCopyVariables>
697         <initialGuessDesignVariables>
698           <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1">
699             <relatedParameterUID>/data_schema/x1</relatedParameterUID>
700             <label>x1^0</label>
701           </initialGuessDesignVariable>
702           <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1">
703             <relatedParameterUID>/data_schema/z1</relatedParameterUID>
704             <label>z1^0</label>
705           </initialGuessDesignVariable>
```

```
706         <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2">
707           <relatedParameterUID>/data_schema/z2</relatedParameterUID>
708           <label>z2^0</label>
709         </initialGuessDesignVariable>
710       </initialGuessDesignVariables>
711       <finalDesignVariables>
712         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1">
713           <relatedParameterUID>/data_schema/x1</relatedParameterUID>
714           <label>x1^*</label>
715         </finalDesignVariable>
716         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2">
717           <relatedParameterUID>/data_schema/z2</relatedParameterUID>
718           <label>z2^*</label>
719         </finalDesignVariable>
720         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1">
721           <relatedParameterUID>/data_schema/z1</relatedParameterUID>
722           <label>z1^*</label>
723         </finalDesignVariable>
724       </finalDesignVariables>
725       <finalOutputVariables>
726         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1">
727           <relatedParameterUID>/data_schema/g1</relatedParameterUID>
728           <label>g1^*</label>
729         </finalOutputVariable>
730         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2">
731           <relatedParameterUID>/data_schema/g2</relatedParameterUID>
732           <label>g2^*</label>
733         </finalOutputVariable>
734         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1">
735           <relatedParameterUID>/data_schema/f1</relatedParameterUID>
736           <label>f1^*</label>
737         </finalOutputVariable>
738       </finalOutputVariables>
739       <consistencyConstraintVariables>
740         <consistencyConstraintVariable
    ↪ uID="/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y2">
741           <relatedParameterUID>/data_schema/y2</relatedParameterUID>
742           <label>gc_y2</label>
743         </consistencyConstraintVariable>
744         <consistencyConstraintVariable
    ↪ uID="/data_schema/architectureNodes/consistencyConstraintVariables/data_schemaCopy//gc_y1">
745           <relatedParameterUID>/data_schema/y1</relatedParameterUID>
746           <label>gc_y1</label>
747         </consistencyConstraintVariable>
748       </consistencyConstraintVariables>
749     </parameters>
750     <executableBlocks>
751       <coordinators>
752         <coordinator uID="Coordinator">
753           <label>COOR</label>
754         </coordinator>
755       </coordinators>
756       <optimizers>
757         <optimizer uID="Optimizer">
758           <label>OPT</label>
759           <designVariables>
760             <designVariable>
761               <designVariableUID>__desVar__/data_schema/x1</designVariableUID>
762             </designVariable>
763             <designVariable>
764               <designVariableUID>__desVar__/data_schema/y2</designVariableUID>
765             </designVariable>
766             <designVariable>
767               <designVariableUID>__desVar__/data_schema/z2</designVariableUID>
```

```
            </designVariable>
            <designVariable>
              <designVariableUID>__desVar__/data_schema/z1</designVariableUID>
            </designVariable>
            <designVariable>
              <designVariableUID>__desVar__/data_schema/y1</designVariableUID>
            </designVariable>
          </designVariables>
          <objectiveVariables>
            <objectiveVariable>
              <objectiveVariableUID>__objVar__/data_schema/f1</objectiveVariableUID>
            </objectiveVariable>
          </objectiveVariables>
          <constraintVariables>
            <constraintVariable>
              <constraintVariableUID>__conVar__/data_schema/g1</constraintVariableUID>
            </constraintVariable>
            <constraintVariable>
              <constraintVariableUID>__conVar__/data_schema/g2</constraintVariableUID>
            </constraintVariable>
          </constraintVariables>
        </optimizer>
      </optimizers>
      <consistencyConstraintFunctions>
        <consistencyConstraintFunction uID="Gc">
          <label>Gc</label>
        </consistencyConstraintFunction>
      </consistencyConstraintFunctions>
      <coupledAnalyses>
        <coupledAnalysis>
          <relatedExecutableBlockUID>D2</relatedExecutableBlockUID>
        </coupledAnalysis>
        <coupledAnalysis>
          <relatedExecutableBlockUID>D1</relatedExecutableBlockUID>
        </coupledAnalysis>
      </coupledAnalyses>
      <postCouplingAnalyses>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>G2</relatedExecutableBlockUID>
        </postCouplingAnalysis>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>G1</relatedExecutableBlockUID>
        </postCouplingAnalysis>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>F1</relatedExecutableBlockUID>
        </postCouplingAnalysis>
      </postCouplingAnalyses>
    </executableBlocks>
  </architectureElements>
</cmdows>
```

Code frament A.16: Sellar problem IDF CMDOWS file.

```
<?xml version='1.0' encoding='UTF-8'?>
<cmdows xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  ↪ xsi:noNamespaceSchemaLocation="https://bitbucket.org/imcovangent/cmdows/raw/master/schema/0.7/cmdows.xsd">
  <header>
    <creator>D. de Vries</creator>
    <description>Sellar problem MPG file</description>
    <timestamp>2017-10-09T11:35:18.577000</timestamp>
    <fileVersion>0.1</fileVersion>
    <cmdowsVersion>0.7</cmdowsVersion>
    <updates>
      <update>
        <modification>KADMOS export of a mdao data graph (MDG).</modification>
        <creator>D. de Vries</creator>
        <timestamp>2017-10-09T11:35:18.577000</timestamp>
        <fileVersion>0.1</fileVersion>
        <cmdowsVersion>0.7</cmdowsVersion>
      </update>
```

```
    </updates>
  </header>
  <executableBlocks>
    <designCompetences>
      <designCompetence uID="F1">
        <ID>F1</ID>
        <modeID>main</modeID>
        <instanceID>1</instanceID>
        <version>1.0</version>
        <label>F1</label>
        <inputs>
          <input>
            <parameterUID>/data_schema/x1</parameterUID>
          </input>
          <input>
            <parameterUID>/data_schema/y2</parameterUID>
          </input>
          <input>
            <parameterUID>/data_schema/z2</parameterUID>
          </input>
          <input>
            <parameterUID>/data_schema/y1</parameterUID>
          </input>
        </inputs>
        <outputs>
          <output>
            <parameterUID>/data_schema/f1</parameterUID>
          </output>
          <output>
            <parameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</parameterUID>
          </output>
        </outputs>
        <metadata>
          <generalInfo>
            <description>main execution mode</description>
          </generalInfo>
        </metadata>
      </designCompetence>
      <designCompetence uID="D2">
        <ID>D2</ID>
        <modeID>main</modeID>
        <instanceID>1</instanceID>
        <version>1.0</version>
        <label>D2</label>
        <inputs>
          <input>
            <parameterUID>/data_schema/z2</parameterUID>
          </input>
          <input>
            <parameterUID>/data_schema/z1</parameterUID>
          </input>
          <input>
            <parameterUID>/data_schema/y1</parameterUID>
          </input>
        </inputs>
        <outputs>
          <output>
            <parameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</parameterUID>
          </output>
          <output>
            <parameterUID>/data_schema/y2</parameterUID>
          </output>
        </outputs>
        <metadata>
          <generalInfo>
            <description>main execution mode</description>
          </generalInfo>
        </metadata>
      </designCompetence>
```

```
86        <designCompetence uID="G2">
87          <ID>G2</ID>
88          <modeID>main</modeID>
89          <instanceID>1</instanceID>
90          <version>1.0</version>
91          <label>G2</label>
92          <inputs>
93            <input>
94              <parameterUID>/data_schema/y2</parameterUID>
95            </input>
96          </inputs>
97          <outputs>
98            <output>
99              <parameterUID>/data_schema/g2</parameterUID>
100           </output>
101           <output>
102             <parameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</parameterUID>
103           </output>
104         </outputs>
105         <metadata>
106           <generalInfo>
107             <description>main execution mode</description>
108           </generalInfo>
109         </metadata>
110       </designCompetence>
111       <designCompetence uID="G1">
112         <ID>G1</ID>
113         <modeID>main</modeID>
114         <instanceID>1</instanceID>
115         <version>1.0</version>
116         <label>G1</label>
117         <inputs>
118           <input>
119             <parameterUID>/data_schema/y1</parameterUID>
120           </input>
121         </inputs>
122         <outputs>
123           <output>
124             <parameterUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</parameterUID>
125           </output>
126           <output>
127             <parameterUID>/data_schema/g1</parameterUID>
128           </output>
129         </outputs>
130         <metadata>
131           <generalInfo>
132             <description>main execution mode</description>
133           </generalInfo>
134         </metadata>
135       </designCompetence>
136       <designCompetence uID="D1">
137         <ID>D1</ID>
138         <modeID>main</modeID>
139         <instanceID>1</instanceID>
140         <version>1.0</version>
141         <label>D1</label>
142         <inputs>
143           <input>
144             <parameterUID>/data_schema/x1</parameterUID>
145           </input>
146           <input>
147             <parameterUID>/data_schema/z2</parameterUID>
148           </input>
149           <input>
150             <parameterUID>/data_schema/z1</parameterUID>
151           </input>
152           <input>
153             <parameterUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</parameterUID>
```

```
          </input>
        </inputs>
        <outputs>
          <output>
            <parameterUID>/data_schema/y1</parameterUID>
          </output>
        </outputs>
        <metadata>
          <generalInfo>
            <description>main execution mode</description>
          </generalInfo>
        </metadata>
      </designCompetence>
    </designCompetences>
  </executableBlocks>
  <parameters>
    <parameter uID="/data_schema/g1">
      <label>g1</label>
    </parameter>
    <parameter uID="/data_schema/g2">
      <label>g2</label>
    </parameter>
    <parameter uID="/data_schema/f1">
      <label>f1</label>
    </parameter>
    <parameter uID="/data_schema/y1">
      <label>y1</label>
    </parameter>
    <parameter uID="/data_schema/y2">
      <label>y2</label>
    </parameter>
    <parameter uID="/data_schema/x1">
      <label>x1</label>
    </parameter>
    <parameter uID="/data_schema/z2">
      <label>z2</label>
    </parameter>
    <parameter uID="/data_schema/z1">
      <label>z1</label>
    </parameter>
  </parameters>
  <problemDefinition uID="MDFGauss-Seidel">
    <problemFormulation>
      <mdaoArchitecture>MDF</mdaoArchitecture>
      <convergerType>Gauss-Seidel</convergerType>
      <executableBlocksOrder>
        <executableBlock position="1">D1</executableBlock>
        <executableBlock position="2">D2</executableBlock>
        <executableBlock position="3">F1</executableBlock>
        <executableBlock position="4">G1</executableBlock>
        <executableBlock position="5">G2</executableBlock>
      </executableBlocksOrder>
      <allowUnconvergedCouplings>false</allowUnconvergedCouplings>
    </problemFormulation>
    <problemRoles>
      <parameters>
        <designVariables>
          <designVariable uID="__desVar__/data_schema/x1">
            <parameterUID>/data_schema/x1</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
              <limitRange>
                <minimum>0.0</minimum>
                <maximum>10.0</maximum>
              </limitRange>
            </validRanges>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/z2">
            <parameterUID>/data_schema/z2</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
```

```
225             <limitRange>
226               <minimum>0.0</minimum>
227               <maximum>10.0</maximum>
228             </limitRange>
229           </validRanges>
230         </designVariable>
231         <designVariable uID="__desVar__/data_schema/z1">
232           <parameterUID>/data_schema/z1</parameterUID>
233           <nominalValue>1.0</nominalValue>
234           <validRanges>
235             <limitRange>
236               <minimum>-10.0</minimum>
237               <maximum>10.0</maximum>
238             </limitRange>
239           </validRanges>
240         </designVariable>
241       </designVariables>
242       <objectiveVariables>
243         <objectiveVariable uID="__objVar__/data_schema/f1">
244           <parameterUID>/data_schema/f1</parameterUID>
245         </objectiveVariable>
246       </objectiveVariables>
247       <constraintVariables>
248         <constraintVariable uID="__conVar__/data_schema/g1">
249           <parameterUID>/data_schema/g1</parameterUID>
250           <constraintType>inequality</constraintType>
251           <constraintOperator>&lt;=</constraintOperator>
252           <referenceValue>0.0</referenceValue>
253         </constraintVariable>
254         <constraintVariable uID="__conVar__/data_schema/g2">
255           <parameterUID>/data_schema/g2</parameterUID>
256           <constraintType>inequality</constraintType>
257           <constraintOperator>&lt;=</constraintOperator>
258           <referenceValue>0.0</referenceValue>
259         </constraintVariable>
260       </constraintVariables>
261     </parameters>
262     <executableBlocks>
263       <coupledBlocks>
264         <coupledBlock>D1</coupledBlock>
265         <coupledBlock>D2</coupledBlock>
266       </coupledBlocks>
267       <postCouplingBlocks>
268         <postCouplingBlock>F1</postCouplingBlock>
269         <postCouplingBlock>G1</postCouplingBlock>
270         <postCouplingBlock>G2</postCouplingBlock>
271       </postCouplingBlocks>
272     </executableBlocks>
273   </problemRoles>
274 </problemDefinition>
275 <workflow>
276   <problemDefinitionUID>MDFGauss-Seidel</problemDefinitionUID>
277   <dataGraph>
278     <name>MDG1</name>
279     <edges>
280       <edge>
281         <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</fromParameterUID>
282         <toExecutableBlockUID>D1</toExecutableBlockUID>
283       </edge>
284       <edge>
285         <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</fromParameterUID>
286         <toExecutableBlockUID>Converger</toExecutableBlockUID>
287       </edge>
288       <edge>
289         <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
290         <toParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</toParameterUID>
291       </edge>
292       <edge>
```

```
293           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</fromParameterUID>
294           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
295         </edge>
296         <edge>
297           <fromParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</fromParameterUID>
298           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
299         </edge>
300         <edge>
301           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
302           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</toParameterUID>
303         </edge>
304         <edge>
305           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
306           <toParameterUID>/data_schema/x1</toParameterUID>
307         </edge>
308         <edge>
309           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
310           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</toParameterUID>
311         </edge>
312         <edge>
313           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
314           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</toParameterUID>
315         </edge>
316         <edge>
317           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
318           <toParameterUID>/data_schema/z2</toParameterUID>
319         </edge>
320         <edge>
321           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
322           <toParameterUID>/data_schema/z1</toParameterUID>
323         </edge>
324         <edge>
325           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
326           <toParameterUID>/data_schema/g2</toParameterUID>
327         </edge>
328         <edge>
329           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
330           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</toParameterUID>
331         </edge>
332         <edge>
333           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
334           <toParame-
      ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</toParameterUID>
335         </edge>
336         <edge>
337           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
338           <toParameterUID>/data_schema/g1</toParameterUID>
339         </edge>
340         <edge>
341           <fromParameterUID>/data_schema/y1</fromParameterUID>
342           <toExecutableBlockUID>F1</toExecutableBlockUID>
343         </edge>
344         <edge>
345           <fromParameterUID>/data_schema/y1</fromParameterUID>
346           <toExecutableBlockUID>D2</toExecutableBlockUID>
347         </edge>
348         <edge>
349           <fromParameterUID>/data_schema/y1</fromParameterUID>
350           <toExecutableBlockUID>G1</toExecutableBlockUID>
351         </edge>
352         <edge>
353           <fromParameterUID>/data_schema/g1</fromParameterUID>
354           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
355         </edge>
356         <edge>
```

```
357           <fromParameterUID>/data_schema/g2</fromParameterUID>
358           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
359         </edge>
360         <edge>
361           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</fromParameterUID>
362           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
363         </edge>
364         <edge>
365           <fromParameterUID>/data_schema/y2</fromParameterUID>
366           <toExecutableBlockUID>F1</toExecutableBlockUID>
367         </edge>
368         <edge>
369           <fromParameterUID>/data_schema/y2</fromParameterUID>
370           <toExecutableBlockUID>G2</toExecutableBlockUID>
371         </edge>
372         <edge>
373           <fromParameterUID>/data_schema/y2</fromParameterUID>
374           <toExecutableBlockUID>Converger</toExecutableBlockUID>
375         </edge>
376         <edge>
377           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</fromParameterUID>
378           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
379         </edge>
380         <edge>
381           <fromExecutableBlockUID>F1</fromExecutableBlockUID>
382           <toParameterUID>/data_schema/f1</toParameterUID>
383         </edge>
384         <edge>
385           <fromExecutableBlockUID>F1</fromExecutableBlockUID>
386           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</toParameterUID>
387         </edge>
388         <edge>
389           <fromParameterUID>/data_schema/x1</fromParameterUID>
390           <toExecutableBlockUID>F1</toExecutableBlockUID>
391         </edge>
392         <edge>
393           <fromParameterUID>/data_schema/x1</fromParameterUID>
394           <toExecutableBlockUID>D1</toExecutableBlockUID>
395         </edge>
396         <edge>
397           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</fromParameterUID>
398           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
399         </edge>
400         <edge>
401           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</fromParameterUID>
402           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
403         </edge>
404         <edge>
405           <fromParameterUID>/data_schema/z2</fromParameterUID>
406           <toExecutableBlockUID>F1</toExecutableBlockUID>
407         </edge>
408         <edge>
409           <fromParameterUID>/data_schema/z2</fromParameterUID>
410           <toExecutableBlockUID>D2</toExecutableBlockUID>
411         </edge>
412         <edge>
413           <fromParameterUID>/data_schema/z2</fromParameterUID>
414           <toExecutableBlockUID>D1</toExecutableBlockUID>
415         </edge>
416         <edge>
417           <fromParameterUID>/data_schema/z1</fromParameterUID>
418           <toExecutableBlockUID>D2</toExecutableBlockUID>
419         </edge>
420         <edge>
421           <fromParameterUID>/data_schema/z1</fromParameterUID>
422           <toExecutableBlockUID>D1</toExecutableBlockUID>
```

```
423         </edge>
424         <edge>
425           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
426           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</toParameterUID>
427         </edge>
428         <edge>
429           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
430           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</toParameterUI
431         </edge>
432         <edge>
433           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
434           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</toParameterUID>
435         </edge>
436         <edge>
437           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
438           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</toParameterUID>
439         </edge>
440         <edge>
441           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</fromParameterUI
442           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
443         </edge>
444         <edge>
445           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</fromParameterUI
446           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
447         </edge>
448         <edge>
449           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</fromParameterUI
450           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
451         </edge>
452         <edge>
453           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</fromParameterUID>
454           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
455         </edge>
456         <edge>
457           <fromParameterUID>/data_schema/f1</fromParameterUID>
458           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
459         </edge>
460         <edge>
461           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
462           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</toParameterUID>
463         </edge>
464         <edge>
465           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
466           <toParameterUID>/data_schema/y2</toParameterUID>
467         </edge>
468         <edge>
469           <fromExecutableBlockUID>D1</fromExecutableBlockUID>
470           <toParameterUID>/data_schema/y1</toParameterUID>
471         </edge>
472       </edges>
473     </dataGraph>
474     <processGraph>
475       <name>MPG1</name>
476       <edges>
477         <edge>
478           <fromExecutableBlockUID>F1</fromExecutableBlockUID>
479           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
480           <processStepNumber>7</processStepNumber>
481         </edge>
482         <edge>
483           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
484           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
```

```
485           <processStepNumber>8</processStepNumber>
486         </edge>
487         <edge>
488           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
489           <toExecutableBlockUID>Converger</toExecutableBlockUID>
490           <processStepNumber>2</processStepNumber>
491         </edge>
492         <edge>
493           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
494           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
495           <processStepNumber>7</processStepNumber>
496         </edge>
497         <edge>
498           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
499           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
500           <processStepNumber>7</processStepNumber>
501         </edge>
502         <edge>
503           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
504           <toExecutableBlockUID>F1</toExecutableBlockUID>
505           <processStepNumber>6</processStepNumber>
506         </edge>
507         <edge>
508           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
509           <toExecutableBlockUID>G2</toExecutableBlockUID>
510           <processStepNumber>6</processStepNumber>
511         </edge>
512         <edge>
513           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
514           <toExecutableBlockUID>G1</toExecutableBlockUID>
515           <processStepNumber>6</processStepNumber>
516         </edge>
517         <edge>
518           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
519           <toExecutableBlockUID>D1</toExecutableBlockUID>
520           <processStepNumber>3</processStepNumber>
521         </edge>
522         <edge>
523           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
524           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
525           <processStepNumber>1</processStepNumber>
526         </edge>
527         <edge>
528           <fromExecutableBlockUID>D2</fromExecutableBlockUID>
529           <toExecutableBlockUID>Converger</toExecutableBlockUID>
530           <processStepNumber>5</processStepNumber>
531         </edge>
532         <edge>
533           <fromExecutableBlockUID>D1</fromExecutableBlockUID>
534           <toExecutableBlockUID>D2</toExecutableBlockUID>
535           <processStepNumber>4</processStepNumber>
536         </edge>
537       </edges>
538       <nodes>
539         <node>
540           <referenceUID>F1</referenceUID>
541           <processStepNumber>6</processStepNumber>
542           <diagonalPosition>5</diagonalPosition>
543         </node>
544         <node>
545           <referenceUID>Optimizer</referenceUID>
546           <processStepNumber>1</processStepNumber>
547           <convergerStepNumber>7</convergerStepNumber>
548           <diagonalPosition>1</diagonalPosition>
549         </node>
550         <node>
551           <referenceUID>G2</referenceUID>
552           <processStepNumber>6</processStepNumber>
553           <diagonalPosition>7</diagonalPosition>
554         </node>
555         <node>
```

```
556           <referenceUID>G1</referenceUID>
557           <processStepNumber>6</processStepNumber>
558           <diagonalPosition>6</diagonalPosition>
559         </node>
560         <node>
561           <referenceUID>Converger</referenceUID>
562           <processStepNumber>2</processStepNumber>
563           <convergerStepNumber>5</convergerStepNumber>
564           <diagonalPosition>2</diagonalPosition>
565         </node>
566         <node>
567           <referenceUID>Coordinator</referenceUID>
568           <processStepNumber>0</processStepNumber>
569           <convergerStepNumber>8</convergerStepNumber>
570           <diagonalPosition>0</diagonalPosition>
571         </node>
572         <node>
573           <referenceUID>D2</referenceUID>
574           <processStepNumber>4</processStepNumber>
575           <diagonalPosition>4</diagonalPosition>
576         </node>
577         <node>
578           <referenceUID>D1</referenceUID>
579           <processStepNumber>3</processStepNumber>
580           <diagonalPosition>3</diagonalPosition>
581         </node>
582       </nodes>
583       <metadata>
584         <loopNesting>
585           <loopElements>
586             <loopElement relatedUID="Optimizer">
587               <loopElements>
588                 <loopElement relatedUID="Converger">
589                   <functionElements>
590                     <functionElement>D2</functionElement>
591                     <functionElement>D1</functionElement>
592                   </functionElements>
593                 </loopElement>
594               </loopElements>
595               <functionElements>
596                 <functionElement>F1</functionElement>
597                 <functionElement>G2</functionElement>
598                 <functionElement>G1</functionElement>
599               </functionElements>
600             </loopElement>
601           </loopElements>
602         </loopNesting>
603       </metadata>
604     </processGraph>
605   </workflow>
606   <architectureElements>
607     <parameters>
608       <initialGuessCouplingVariables>
609         <initialGuessCouplingVariable
      ↪ uID="/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2">
610           <relatedParameterUID>/data_schema/y2</relatedParameterUID>
611           <label>y2^{c0}</label>
612         </initialGuessCouplingVariable>
613       </initialGuessCouplingVariables>
614       <finalCouplingVariables>
615         <finalCouplingVariable
      ↪ uID="/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2">
616           <relatedParameterUID>/data_schema/y2</relatedParameterUID>
617           <label>y2^*</label>
618         </finalCouplingVariable>
619       </finalCouplingVariables>
620       <couplingCopyVariables>
621         <couplingCopyVariable
      ↪ uID="/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2">
622           <relatedParameterUID>/data_schema/y2</relatedParameterUID>
623           <label>y2^c</label>
```

```
        </couplingCopyVariable>
      </couplingCopyVariables>
      <initialGuessDesignVariables>
        <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1">
          <relatedParameterUID>/data_schema/x1</relatedParameterUID>
          <label>x1^0</label>
        </initialGuessDesignVariable>
        <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1">
          <relatedParameterUID>/data_schema/z1</relatedParameterUID>
          <label>z1^0</label>
        </initialGuessDesignVariable>
        <initialGuessDesignVariable
    ↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2">
          <relatedParameterUID>/data_schema/z2</relatedParameterUID>
          <label>z2^0</label>
        </initialGuessDesignVariable>
      </initialGuessDesignVariables>
      <finalDesignVariables>
        <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1">
          <relatedParameterUID>/data_schema/x1</relatedParameterUID>
          <label>x1^*</label>
        </finalDesignVariable>
        <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2">
          <relatedParameterUID>/data_schema/z2</relatedParameterUID>
          <label>z2^*</label>
        </finalDesignVariable>
        <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1">
          <relatedParameterUID>/data_schema/z1</relatedParameterUID>
          <label>z1^*</label>
        </finalDesignVariable>
      </finalDesignVariables>
      <finalOutputVariables>
        <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1">
          <relatedParameterUID>/data_schema/g1</relatedParameterUID>
          <label>g1^*</label>
        </finalOutputVariable>
        <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2">
          <relatedParameterUID>/data_schema/g2</relatedParameterUID>
          <label>g2^*</label>
        </finalOutputVariable>
        <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1">
          <relatedParameterUID>/data_schema/f1</relatedParameterUID>
          <label>f1^*</label>
        </finalOutputVariable>
      </finalOutputVariables>
    </parameters>
    <executableBlocks>
      <coordinators>
        <coordinator uID="Coordinator">
          <label>COOR</label>
        </coordinator>
      </coordinators>
      <optimizers>
        <optimizer uID="Optimizer">
          <label>OPT</label>
          <designVariables>
            <designVariable>
              <designVariableUID>__desVar__/data_schema/x1</designVariableUID>
            </designVariable>
            <designVariable>
              <designVariableUID>__desVar__/data_schema/z2</designVariableUID>
            </designVariable>
            <designVariable>
```

```
              <designVariableUID>__desVar__/data_schema/z1</designVariableUID>
            </designVariable>
          </designVariables>
          <objectiveVariables>
            <objectiveVariable>
              <objectiveVariableUID>__objVar__/data_schema/f1</objectiveVariableUID>
            </objectiveVariable>
          </objectiveVariables>
          <constraintVariables>
            <constraintVariable>
              <constraintVariableUID>__conVar__/data_schema/g1</constraintVariableUID>
            </constraintVariable>
            <constraintVariable>
              <constraintVariableUID>__conVar__/data_schema/g2</constraintVariableUID>
            </constraintVariable>
          </constraintVariables>
        </optimizer>
      </optimizers>
      <convergers>
        <converger uID="Converger">
          <label>CONV</label>
        </converger>
      </convergers>
      <coupledAnalyses>
        <coupledAnalysis>
          <relatedExecutableBlockUID>D2</relatedExecutableBlockUID>
        </coupledAnalysis>
        <coupledAnalysis>
          <relatedExecutableBlockUID>D1</relatedExecutableBlockUID>
        </coupledAnalysis>
      </coupledAnalyses>
      <postCouplingAnalyses>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>G2</relatedExecutableBlockUID>
        </postCouplingAnalysis>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>G1</relatedExecutableBlockUID>
        </postCouplingAnalysis>
        <postCouplingAnalysis>
          <relatedExecutableBlockUID>F1</relatedExecutableBlockUID>
        </postCouplingAnalysis>
      </postCouplingAnalyses>
    </executableBlocks>
  </architectureElements>
</cmdows>
```

Code frament A.17: Sellar problem MDF-GS CMDOWS file.

```
<?xml version='1.0' encoding='UTF-8'?>
<cmdows xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  ↪ xsi:noNamespaceSchemaLocation="https://bitbucket.org/imcovangent/cmdows/raw/master/schema/0.7/cmdows.
  <header>
    <creator>D. de Vries</creator>
    <description>Sellar problem MPG file</description>
    <timestamp>2017-10-09T11:34:39.674000</timestamp>
    <fileVersion>0.1</fileVersion>
    <cmdowsVersion>0.7</cmdowsVersion>
    <updates>
      <update>
        <modification>KADMOS export of a mdao data graph (MDG).</modification>
        <creator>D. de Vries</creator>
        <timestamp>2017-10-09T11:34:39.674000</timestamp>
        <fileVersion>0.1</fileVersion>
        <cmdowsVersion>0.7</cmdowsVersion>
      </update>
    </updates>
  </header>
  <executableBlocks>
    <designCompetences>
      <designCompetence uID="F1">
```

```
22          <ID>F1</ID>
23          <modeID>main</modeID>
24          <instanceID>1</instanceID>
25          <version>1.0</version>
26          <label>F1</label>
27          <inputs>
28            <input>
29              <parameterUID>/data_schema/x1</parameterUID>
30            </input>
31            <input>
32              <parameterUID>/data_schema/y2</parameterUID>
33            </input>
34            <input>
35              <parameterUID>/data_schema/z2</parameterUID>
36            </input>
37            <input>
38              <parameterUID>/data_schema/y1</parameterUID>
39            </input>
40          </inputs>
41          <outputs>
42            <output>
43              <parameterUID>/data_schema/f1</parameterUID>
44            </output>
45            <output>
46              <parame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</parameterUID>
47            </output>
48          </outputs>
49          <metadata>
50            <generalInfo>
51              <description>main execution mode</description>
52            </generalInfo>
53          </metadata>
54        </designCompetence>
55        <designCompetence uID="D2">
56          <ID>D2</ID>
57          <modeID>main</modeID>
58          <instanceID>1</instanceID>
59          <version>1.0</version>
60          <label>D2</label>
61          <inputs>
62            <input>
63              <parame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</parameterUID>
64            </input>
65            <input>
66              <parameterUID>/data_schema/z2</parameterUID>
67            </input>
68            <input>
69              <parameterUID>/data_schema/z1</parameterUID>
70            </input>
71          </inputs>
72          <outputs>
73            <output>
74              <parame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</parameterUID>
75            </output>
76            <output>
77              <parameterUID>/data_schema/y2</parameterUID>
78            </output>
79          </outputs>
80          <metadata>
81            <generalInfo>
82              <description>main execution mode</description>
83            </generalInfo>
84          </metadata>
85        </designCompetence>
86        <designCompetence uID="G2">
87          <ID>G2</ID>
88          <modeID>main</modeID>
89          <instanceID>1</instanceID>
```

```
90          <version>1.0</version>
91          <label>G2</label>
92          <inputs>
93            <input>
94              <parameterUID>/data_schema/y2</parameterUID>
95            </input>
96          </inputs>
97          <outputs>
98            <output>
99              <parameterUID>/data_schema/g2</parameterUID>
100           </output>
101           <output>
102             <parame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</parameterUID>
103           </output>
104         </outputs>
105         <metadata>
106           <generalInfo>
107             <description>main execution mode</description>
108           </generalInfo>
109         </metadata>
110       </designCompetence>
111       <designCompetence uID="G1">
112         <ID>G1</ID>
113         <modeID>main</modeID>
114         <instanceID>1</instanceID>
115         <version>1.0</version>
116         <label>G1</label>
117         <inputs>
118           <input>
119             <parameterUID>/data_schema/y1</parameterUID>
120           </input>
121         </inputs>
122         <outputs>
123           <output>
124             <parame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</parameterUID>
125           </output>
126           <output>
127             <parameterUID>/data_schema/g1</parameterUID>
128           </output>
129         </outputs>
130         <metadata>
131           <generalInfo>
132             <description>main execution mode</description>
133           </generalInfo>
134         </metadata>
135       </designCompetence>
136       <designCompetence uID="D1">
137         <ID>D1</ID>
138         <modeID>main</modeID>
139         <instanceID>1</instanceID>
140         <version>1.0</version>
141         <label>D1</label>
142         <inputs>
143           <input>
144             <parameterUID>/data_schema/x1</parameterUID>
145           </input>
146           <input>
147             <parameterUID>/data_schema/z2</parameterUID>
148           </input>
149           <input>
150             <parameterUID>/data_schema/z1</parameterUID>
151           </input>
152           <input>
153             <parame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</parameterUID>
154           </input>
155         </inputs>
156         <outputs>
157           <output>
```

```
            <parameterUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</parameterUID>
          </output>
          <output>
            <parameterUID>/data_schema/y1</parameterUID>
          </output>
        </outputs>
        <metadata>
          <generalInfo>
            <description>main execution mode</description>
          </generalInfo>
        </metadata>
      </designCompetence>
    </designCompetences>
  </executableBlocks>
  <parameters>
    <parameter uID="/data_schema/g1">
      <label>g1</label>
    </parameter>
    <parameter uID="/data_schema/g2">
      <label>g2</label>
    </parameter>
    <parameter uID="/data_schema/f1">
      <label>f1</label>
    </parameter>
    <parameter uID="/data_schema/y2">
      <label>y2</label>
    </parameter>
    <parameter uID="/data_schema/y1">
      <label>y1</label>
    </parameter>
    <parameter uID="/data_schema/x1">
      <label>x1</label>
    </parameter>
    <parameter uID="/data_schema/z2">
      <label>z2</label>
    </parameter>
    <parameter uID="/data_schema/z1">
      <label>z1</label>
    </parameter>
  </parameters>
  <problemDefinition uID="MDFJacobi">
    <problemFormulation>
      <mdaoArchitecture>MDF</mdaoArchitecture>
      <convergerType>Jacobi</convergerType>
      <executableBlocksOrder>
        <executableBlock position="1">D1</executableBlock>
        <executableBlock position="2">D2</executableBlock>
        <executableBlock position="3">F1</executableBlock>
        <executableBlock position="4">G1</executableBlock>
        <executableBlock position="5">G2</executableBlock>
      </executableBlocksOrder>
      <allowUnconvergedCouplings>false</allowUnconvergedCouplings>
    </problemFormulation>
    <problemRoles>
      <parameters>
        <designVariables>
          <designVariable uID="__desVar__/data_schema/x1">
            <parameterUID>/data_schema/x1</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
              <limitRange>
                <minimum>0.0</minimum>
                <maximum>10.0</maximum>
              </limitRange>
            </validRanges>
          </designVariable>
          <designVariable uID="__desVar__/data_schema/z2">
            <parameterUID>/data_schema/z2</parameterUID>
            <nominalValue>5.0</nominalValue>
            <validRanges>
```

```
228                 <limitRange>
229                   <minimum>0.0</minimum>
230                   <maximum>10.0</maximum>
231                 </limitRange>
232               </validRanges>
233             </designVariable>
234             <designVariable uID="__desVar__/data_schema/z1">
235               <parameterUID>/data_schema/z1</parameterUID>
236               <nominalValue>1.0</nominalValue>
237               <validRanges>
238                 <limitRange>
239                   <minimum>-10.0</minimum>
240                   <maximum>10.0</maximum>
241                 </limitRange>
242               </validRanges>
243             </designVariable>
244           </designVariables>
245           <objectiveVariables>
246             <objectiveVariable uID="__objVar__/data_schema/f1">
247               <parameterUID>/data_schema/f1</parameterUID>
248             </objectiveVariable>
249           </objectiveVariables>
250           <constraintVariables>
251             <constraintVariable uID="__conVar__/data_schema/g1">
252               <parameterUID>/data_schema/g1</parameterUID>
253               <constraintType>inequality</constraintType>
254               <constraintOperator>&lt;=</constraintOperator>
255               <referenceValue>0.0</referenceValue>
256             </constraintVariable>
257             <constraintVariable uID="__conVar__/data_schema/g2">
258               <parameterUID>/data_schema/g2</parameterUID>
259               <constraintType>inequality</constraintType>
260               <constraintOperator>&lt;=</constraintOperator>
261               <referenceValue>0.0</referenceValue>
262             </constraintVariable>
263           </constraintVariables>
264         </parameters>
265         <executableBlocks>
266           <coupledBlocks>
267             <coupledBlock>D1</coupledBlock>
268             <coupledBlock>D2</coupledBlock>
269           </coupledBlocks>
270           <postCouplingBlocks>
271             <postCouplingBlock>F1</postCouplingBlock>
272             <postCouplingBlock>G1</postCouplingBlock>
273             <postCouplingBlock>G2</postCouplingBlock>
274           </postCouplingBlocks>
275         </executableBlocks>
276       </problemRoles>
277     </problemDefinition>
278     <workflow>
279       <problemDefinitionUID>MDFJacobi</problemDefinitionUID>
280       <dataGraph>
281         <name>MDG1</name>
282         <edges>
283           <edge>
284             <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</fromParameterUID>
285             <toExecutableBlockUID>D2</toExecutableBlockUID>
286           </edge>
287           <edge>
288             <fromParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</fromParameterUID>
289             <toExecutableBlockUID>D1</toExecutableBlockUID>
290           </edge>
291           <edge>
292             <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1</fromParameter
293             <toExecutableBlockUID>Converger</toExecutableBlockUID>
294           </edge>
295           <edge>
```

```
296           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</fromParameterUID>
297           <toExecutableBlockUID>Converger</toExecutableBlockUID>
298         </edge>
299         <edge>
300           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
301           <toParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1</toParameterUID>
302         </edge>
303         <edge>
304           <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
305           <toParame-
    ↪ terUID>/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2</toParameterUID>
306         </edge>
307         <edge>
308           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</fromParameterUID>
309           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
310         </edge>
311         <edge>
312           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</fromParameterUID>
313           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
314         </edge>
315         <edge>
316           <fromParameterUID>/data_schema/y2</fromParameterUID>
317           <toExecutableBlockUID>F1</toExecutableBlockUID>
318         </edge>
319         <edge>
320           <fromParameterUID>/data_schema/y2</fromParameterUID>
321           <toExecutableBlockUID>G2</toExecutableBlockUID>
322         </edge>
323         <edge>
324           <fromParameterUID>/data_schema/y2</fromParameterUID>
325           <toExecutableBlockUID>Converger</toExecutableBlockUID>
326         </edge>
327         <edge>
328           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
329           <toParameterUID>/data_schema/g2</toParameterUID>
330         </edge>
331         <edge>
332           <fromExecutableBlockUID>G2</fromExecutableBlockUID>
333           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2</toParameterUID>
334         </edge>
335         <edge>
336           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
337           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1</toParameterUID>
338         </edge>
339         <edge>
340           <fromExecutableBlockUID>G1</fromExecutableBlockUID>
341           <toParameterUID>/data_schema/g1</toParameterUID>
342         </edge>
343         <edge>
344           <fromParameterUID>/data_schema/y1</fromParameterUID>
345           <toExecutableBlockUID>F1</toExecutableBlockUID>
346         </edge>
347         <edge>
348           <fromParameterUID>/data_schema/y1</fromParameterUID>
349           <toExecutableBlockUID>G1</toExecutableBlockUID>
350         </edge>
351         <edge>
352           <fromParameterUID>/data_schema/y1</fromParameterUID>
353           <toExecutableBlockUID>Converger</toExecutableBlockUID>
354         </edge>
355         <edge>
356           <fromParameterUID>/data_schema/g1</fromParameterUID>
357           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
358         </edge>
359         <edge>
```

```
360           <fromParameterUID>/data_schema/g2</fromParameterUID>
361           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
362         </edge>
363         <edge>
364           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</fromParameterUID>
365           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
366         </edge>
367         <edge>
368           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
369           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</toParameterUID>
370         </edge>
371         <edge>
372           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
373           <toParameterUID>/data_schema/x1</toParameterUID>
374         </edge>
375         <edge>
376           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
377           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</toParameterUID>
378         </edge>
379         <edge>
380           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
381           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</toParameterUID>
382         </edge>
383         <edge>
384           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
385           <toParameterUID>/data_schema/z2</toParameterUID>
386         </edge>
387         <edge>
388           <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
389           <toParameterUID>/data_schema/z1</toParameterUID>
390         </edge>
391         <edge>
392           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</fromParameterUID>
393           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
394         </edge>
395         <edge>
396           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1</fromParameterUID>
397           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
398         </edge>
399         <edge>
400           <fromExecutableBlockUID>F1</fromExecutableBlockUID>
401           <toParameterUID>/data_schema/f1</toParameterUID>
402         </edge>
403         <edge>
404           <fromExecutableBlockUID>F1</fromExecutableBlockUID>
405           <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1</toParameterUID>
406         </edge>
407         <edge>
408           <fromParameterUID>/data_schema/x1</fromParameterUID>
409           <toExecutableBlockUID>F1</toExecutableBlockUID>
410         </edge>
411         <edge>
412           <fromParameterUID>/data_schema/x1</fromParameterUID>
413           <toExecutableBlockUID>D1</toExecutableBlockUID>
414         </edge>
415         <edge>
416           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2</fromParameterUID>
417           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
418         </edge>
419         <edge>
420           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1</fromParameterUID>
421           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
```

```
422         </edge>
423         <edge>
424           <fromParameterUID>/data_schema/z2</fromParameterUID>
425           <toExecutableBlockUID>F1</toExecutableBlockUID>
426         </edge>
427         <edge>
428           <fromParameterUID>/data_schema/z2</fromParameterUID>
429           <toExecutableBlockUID>D2</toExecutableBlockUID>
430         </edge>
431         <edge>
432           <fromParameterUID>/data_schema/z2</fromParameterUID>
433           <toExecutableBlockUID>D1</toExecutableBlockUID>
434         </edge>
435         <edge>
436           <fromParameterUID>/data_schema/z1</fromParameterUID>
437           <toExecutableBlockUID>D2</toExecutableBlockUID>
438         </edge>
439         <edge>
440           <fromParameterUID>/data_schema/z1</fromParameterUID>
441           <toExecutableBlockUID>D1</toExecutableBlockUID>
442         </edge>
443         <edge>
444           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
445           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1</toParameterUID>
446         </edge>
447         <edge>
448           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
449           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</toParameterUID>
450         </edge>
451         <edge>
452           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
453           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2</toParameterUID>
454         </edge>
455         <edge>
456           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
457           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</toParameterUID>
458         </edge>
459         <edge>
460           <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
461           <toParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</toParameterUID>
462         </edge>
463         <edge>
464           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1</fromParameterUID>
465           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
466         </edge>
467         <edge>
468           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1</fromParameterUID>
469           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
470         </edge>
471         <edge>
472           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2</fromParameterUID>
473           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
474         </edge>
475         <edge>
476           <fromParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</fromParameterUID>
477           <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
478         </edge>
479         <edge>
480           <fromParameterUID>/data_schema/f1</fromParameterUID>
481           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
482         </edge>
483         <edge>
```

```
          <fromExecutableBlockUID>D2</fromExecutableBlockUID>
          <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2</toParameterUID>
        </edge>
        <edge>
          <fromExecutableBlockUID>D2</fromExecutableBlockUID>
          <toParameterUID>/data_schema/y2</toParameterUID>
        </edge>
        <edge>
          <fromExecutableBlockUID>D1</fromExecutableBlockUID>
          <toParame-
    ↪ terUID>/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1</toParameterUID>
        </edge>
        <edge>
          <fromExecutableBlockUID>D1</fromExecutableBlockUID>
          <toParameterUID>/data_schema/y1</toParameterUID>
        </edge>
      </edges>
    </dataGraph>
    <processGraph>
      <name>MPG1</name>
      <edges>
        <edge>
          <fromExecutableBlockUID>F1</fromExecutableBlockUID>
          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
          <processStepNumber>6</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
          <processStepNumber>7</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
          <toExecutableBlockUID>Converger</toExecutableBlockUID>
          <processStepNumber>2</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>G2</fromExecutableBlockUID>
          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
          <processStepNumber>6</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>G1</fromExecutableBlockUID>
          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
          <processStepNumber>6</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
          <toExecutableBlockUID>F1</toExecutableBlockUID>
          <processStepNumber>5</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
          <toExecutableBlockUID>D2</toExecutableBlockUID>
          <processStepNumber>3</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
          <toExecutableBlockUID>G2</toExecutableBlockUID>
          <processStepNumber>5</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
          <toExecutableBlockUID>G1</toExecutableBlockUID>
          <processStepNumber>5</processStepNumber>
        </edge>
        <edge>
          <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
          <toExecutableBlockUID>D1</toExecutableBlockUID>
          <processStepNumber>3</processStepNumber>
```

```
      </edge>
      <edge>
        <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
        <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
        <processStepNumber>1</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>D2</fromExecutableBlockUID>
        <toExecutableBlockUID>Converger</toExecutableBlockUID>
        <processStepNumber>4</processStepNumber>
      </edge>
      <edge>
        <fromExecutableBlockUID>D1</fromExecutableBlockUID>
        <toExecutableBlockUID>Converger</toExecutableBlockUID>
        <processStepNumber>4</processStepNumber>
      </edge>
    </edges>
    <nodes>
      <node>
        <referenceUID>F1</referenceUID>
        <processStepNumber>5</processStepNumber>
        <diagonalPosition>5</diagonalPosition>
      </node>
      <node>
        <referenceUID>Optimizer</referenceUID>
        <processStepNumber>1</processStepNumber>
        <convergerStepNumber>6</convergerStepNumber>
        <diagonalPosition>1</diagonalPosition>
      </node>
      <node>
        <referenceUID>G2</referenceUID>
        <processStepNumber>5</processStepNumber>
        <diagonalPosition>7</diagonalPosition>
      </node>
      <node>
        <referenceUID>G1</referenceUID>
        <processStepNumber>5</processStepNumber>
        <diagonalPosition>6</diagonalPosition>
      </node>
      <node>
        <referenceUID>Converger</referenceUID>
        <processStepNumber>2</processStepNumber>
        <convergerStepNumber>4</convergerStepNumber>
        <diagonalPosition>2</diagonalPosition>
      </node>
      <node>
        <referenceUID>Coordinator</referenceUID>
        <processStepNumber>0</processStepNumber>
        <convergerStepNumber>7</convergerStepNumber>
        <diagonalPosition>0</diagonalPosition>
      </node>
      <node>
        <referenceUID>D2</referenceUID>
        <processStepNumber>3</processStepNumber>
        <diagonalPosition>4</diagonalPosition>
      </node>
      <node>
        <referenceUID>D1</referenceUID>
        <processStepNumber>3</processStepNumber>
        <diagonalPosition>3</diagonalPosition>
      </node>
    </nodes>
    <metadata>
      <loopNesting>
        <loopElements>
          <loopElement relatedUID="Optimizer">
            <loopElements>
              <loopElement relatedUID="Converger">
                <functionElements>
                  <functionElement>D2</functionElement>
                  <functionElement>D1</functionElement>
```

```
                </functionElements>
              </loopElement>
            </loopElements>
            <functionElements>
              <functionElement>F1</functionElement>
              <functionElement>G2</functionElement>
              <functionElement>G1</functionElement>
            </functionElements>
          </loopElement>
        </loopElements>
      </loopNesting>
    </metadata>
  </processGraph>
</workflow>
<architectureElements>
  <parameters>
    <initialGuessCouplingVariables>
      <initialGuessCouplingVariable
↪ uID="/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y1">
        <relatedParameterUID>/data_schema/y1</relatedParameterUID>
        <label>y1^{c0}</label>
      </initialGuessCouplingVariable>
      <initialGuessCouplingVariable
↪ uID="/data_schema/architectureNodes/initialGuessCouplingVariables/data_schemaCopy/y2">
        <relatedParameterUID>/data_schema/y2</relatedParameterUID>
        <label>y2^{c0}</label>
      </initialGuessCouplingVariable>
    </initialGuessCouplingVariables>
    <finalCouplingVariables>
      <finalCouplingVariable
↪ uID="/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y1">
        <relatedParameterUID>/data_schema/y1</relatedParameterUID>
        <label>y1^*</label>
      </finalCouplingVariable>
      <finalCouplingVariable
↪ uID="/data_schema/architectureNodes/finalCouplingVariables/data_schemaCopy/y2">
        <relatedParameterUID>/data_schema/y2</relatedParameterUID>
        <label>y2^*</label>
      </finalCouplingVariable>
    </finalCouplingVariables>
    <couplingCopyVariables>
      <couplingCopyVariable
↪ uID="/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y1">
        <relatedParameterUID>/data_schema/y1</relatedParameterUID>
        <label>y1^c</label>
      </couplingCopyVariable>
      <couplingCopyVariable
↪ uID="/data_schema/architectureNodes/couplingCopyVariables/data_schemaCopy/y2">
        <relatedParameterUID>/data_schema/y2</relatedParameterUID>
        <label>y2^c</label>
      </couplingCopyVariable>
    </couplingCopyVariables>
    <initialGuessDesignVariables>
      <initialGuessDesignVariable
↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/x1">
        <relatedParameterUID>/data_schema/x1</relatedParameterUID>
        <label>x1^0</label>
      </initialGuessDesignVariable>
      <initialGuessDesignVariable
↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z1">
        <relatedParameterUID>/data_schema/z1</relatedParameterUID>
        <label>z1^0</label>
      </initialGuessDesignVariable>
      <initialGuessDesignVariable
↪ uID="/data_schema/architectureNodes/initialGuessDesignVariables/data_schemaCopy/z2">
        <relatedParameterUID>/data_schema/z2</relatedParameterUID>
        <label>z2^0</label>
      </initialGuessDesignVariable>
    </initialGuessDesignVariables>
    <finalDesignVariables>
```

```
685         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/x1">
686           <relatedParameterUID>/data_schema/x1</relatedParameterUID>
687           <label>x1^*</label>
688         </finalDesignVariable>
689         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z2">
690           <relatedParameterUID>/data_schema/z2</relatedParameterUID>
691           <label>z2^*</label>
692         </finalDesignVariable>
693         <finalDesignVariable
    ↪ uID="/data_schema/architectureNodes/finalDesignVariables/data_schemaCopy/z1">
694           <relatedParameterUID>/data_schema/z1</relatedParameterUID>
695           <label>z1^*</label>
696         </finalDesignVariable>
697       </finalDesignVariables>
698       <finalOutputVariables>
699         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g1">
700           <relatedParameterUID>/data_schema/g1</relatedParameterUID>
701           <label>g1^*</label>
702         </finalOutputVariable>
703         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/g2">
704           <relatedParameterUID>/data_schema/g2</relatedParameterUID>
705           <label>g2^*</label>
706         </finalOutputVariable>
707         <finalOutputVariable
    ↪ uID="/data_schema/architectureNodes/finalOutputVariables/data_schemaCopy/f1">
708           <relatedParameterUID>/data_schema/f1</relatedParameterUID>
709           <label>f1^*</label>
710         </finalOutputVariable>
711       </finalOutputVariables>
712     </parameters>
713     <executableBlocks>
714       <coordinators>
715         <coordinator uID="Coordinator">
716           <label>COOR</label>
717         </coordinator>
718       </coordinators>
719       <optimizers>
720         <optimizer uID="Optimizer">
721           <label>OPT</label>
722           <designVariables>
723             <designVariable>
724               <designVariableUID>__desVar__/data_schema/x1</designVariableUID>
725             </designVariable>
726             <designVariable>
727               <designVariableUID>__desVar__/data_schema/z2</designVariableUID>
728             </designVariable>
729             <designVariable>
730               <designVariableUID>__desVar__/data_schema/z1</designVariableUID>
731             </designVariable>
732           </designVariables>
733           <objectiveVariables>
734             <objectiveVariable>
735               <objectiveVariableUID>__objVar__/data_schema/f1</objectiveVariableUID>
736             </objectiveVariable>
737           </objectiveVariables>
738           <constraintVariables>
739             <constraintVariable>
740               <constraintVariableUID>__conVar__/data_schema/g1</constraintVariableUID>
741             </constraintVariable>
742             <constraintVariable>
743               <constraintVariableUID>__conVar__/data_schema/g2</constraintVariableUID>
744             </constraintVariable>
745           </constraintVariables>
746         </optimizer>
747       </optimizers>
748       <convergers>
749         <converger uID="Converger">
```

```
750           <label>CONV</label>
751         </converger>
752       </convergers>
753       <coupledAnalyses>
754         <coupledAnalysis>
755           <relatedExecutableBlockUID>D2</relatedExecutableBlockUID>
756         </coupledAnalysis>
757         <coupledAnalysis>
758           <relatedExecutableBlockUID>D1</relatedExecutableBlockUID>
759         </coupledAnalysis>
760       </coupledAnalyses>
761       <postCouplingAnalyses>
762         <postCouplingAnalysis>
763           <relatedExecutableBlockUID>G2</relatedExecutableBlockUID>
764         </postCouplingAnalysis>
765         <postCouplingAnalysis>
766           <relatedExecutableBlockUID>G1</relatedExecutableBlockUID>
767         </postCouplingAnalysis>
768         <postCouplingAnalysis>
769           <relatedExecutableBlockUID>F1</relatedExecutableBlockUID>
770         </postCouplingAnalysis>
771       </postCouplingAnalyses>
772     </executableBlocks>
773   </architectureElements>
774 </cmdows>
```

Code frament A.18: Sellar problem MDF-J CMDOWS file.

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <data_schema>
3      <x1>5.0</x1>
4      <z1>1.0</z1>
5      <z2>5.0</z2>
6      <architectureNodes>
7          <couplingCopyVariables>
8              <data_schemaCopy>
9                  <y1>5.0</y1>
10                 <y2>5.0</y2>
11             </data_schemaCopy>
12         </couplingCopyVariables>
13     </architectureNodes>
14 </data_schema>
```

Code frament A.19: Sellar problem input XML file.

## Knowledge Base

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  """
4  Copyright 2017 D. de Vries
5
6  Licensed under the Apache License, Version 2.0 (the "License");
7  you may not use this file except in compliance with the License.
8  You may obtain a copy of the License at
9
10     http://www.apache.org/licenses/LICENSE-2.0
11
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17
18 This file contains the definition of the Sellar D1 discipline.
19 """
20 from __future__ import absolute_import, division, print_function
21
22 from lxml import etree
```

```
23
24 from openlego.api import AbstractDiscipline
25 from openlego.utils.xml_utils import xml_safe_create_element
26 from openlego.test_suite.test_examples.sellar.kb import root_tag, x_x1, x_y1, x_y2, x_z1, x_z2
27 from openlego.partials.partials import Partials
28
29 import openlego.test_suite.test_examples.sellar.store as store
30
31
32 class D1(AbstractDiscipline):
33
34     @property
35     def creator(self):
36         return u'D. de Vries'
37
38     @property
39     def description(self):
40         return u'First discipline of the Sellar problem'
41
42     @property
43     def supplies_partials(self):
44         return True
45
46     def generate_input_xml(self):
47         root = etree.Element(root_tag)
48         doc = etree.ElementTree(root)
49
50         xml_safe_create_element(doc, x_x1, 0.)
51         xml_safe_create_element(doc, x_z1, 0.)
52         xml_safe_create_element(doc, x_z2, 0.)
53         xml_safe_create_element(doc, x_y2, 0.)
54
55         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
56
57     def generate_output_xml(self):
58         root = etree.Element(root_tag)
59         doc = etree.ElementTree(root)
60
61         xml_safe_create_element(doc, x_y1, 0.)
62
63         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
64
65     def generate_partials_xml(self):
66         partials = Partials()
67         partials.declare_partials(x_y1, [x_x1, x_y2, x_z1, x_z2])
68         return partials.get_string()
69
70     @staticmethod
71     def execute(in_file, out_file):
72         store.count[0] += 1
73         store.sleep()
74         doc = etree.parse(in_file)
75         z1 = float(doc.xpath(x_z1)[0].text)
76         z2 = float(doc.xpath(x_z2)[0].text)
77         x1 = float(doc.xpath(x_x1)[0].text)
78         y2 = float(doc.xpath(x_y2)[0].text)
79
80         y1 = z1**2. + x1 + z2 - .2*y2
81
82         root = etree.Element(root_tag)
83         doc = etree.ElementTree(root)
84         xml_safe_create_element(doc, x_y1, y1)
85         doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
86
87     @staticmethod
88     def linearize(in_file, partials_file):
89         doc = etree.parse(in_file)
90         z1 = float(doc.xpath(x_z1)[0].text)
91
92         partials = Partials()
93         partials.declare_partials(x_y1, [x_x1, x_y2, x_z1, x_z2], [1., -.2, 2.*z1, 1.])
```

```
94          partials.write(partials_file)
```

Code frament A.20: Code of the Sellar `D1` Python module.

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the Sellar D2 discipline.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from lxml import etree
23
24  from openlego.api import AbstractDiscipline
25  from openlego.utils.xml_utils import xml_safe_create_element
26  from openlego.test_suite.test_examples.sellar.kb import root_tag, x_y1, x_y2, x_z1, x_z2
27  from openlego.partials.partials import Partials
28
29  import openlego.test_suite.test_examples.sellar.store as store
30
31
32  class D2(AbstractDiscipline):
33
34      @property
35      def creator(self):
36          return u'D. de Vries'
37
38      @property
39      def description(self):
40          return u'Second discipline of the Sellar problem'
41
42      @property
43      def supplies_partials(self):
44          return True
45
46      def generate_input_xml(self):
47          root = etree.Element(root_tag)
48          doc = etree.ElementTree(root)
49
50          xml_safe_create_element(doc, x_z1, 0.)
51          xml_safe_create_element(doc, x_z2, 0.)
52          xml_safe_create_element(doc, x_y1, 0.)
53
54          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
55
56      def generate_output_xml(self):
57          root = etree.Element(root_tag)
58          doc = etree.ElementTree(root)
59
60          xml_safe_create_element(doc, x_y2, 0.)
61
62          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
63
64      def generate_partials_xml(self):
65          partials = Partials()
66          partials.declare_partials(x_y2, [x_y1, x_z1, x_z2])
```

```
67          return partials.get_string()
68
69      @staticmethod
70      def execute(in_file, out_file):
71          store.count[1] += 1
72          store.sleep()
73          doc = etree.parse(in_file)
74          z1 = float(doc.xpath(x_z1)[0].text)
75          z2 = float(doc.xpath(x_z2)[0].text)
76          y1 = float(doc.xpath(x_y1)[0].text)
77
78          y2 = abs(y1)**.5 + z1 + z2
79
80          root = etree.Element(root_tag)
81          doc = etree.ElementTree(root)
82          xml_safe_create_element(doc, x_y2, y2)
83          doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
84
85      @staticmethod
86      def linearize(in_file, partials_file):
87          doc = etree.parse(in_file)
88          y1 = float(doc.xpath(x_y1)[0].text)
89
90          dy2_dy1 = .5 * float(((y1 > 0) - (y1 < 0))) / abs(y1)**.5
91
92          partials = Partials()
93          partials.declare_partials(x_y2, [x_y1, x_z1, x_z2], [dy2_dy1, 1., 1.])
94          partials.write(partials_file)
```

Code frament A.21: Code of the Sellar `D2` Python module.

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the Sellar F1 discipline.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from math import exp
23
24  from lxml import etree
25
26  from openlego.api import AbstractDiscipline
27  from openlego.utils.xml_utils import xml_safe_create_element
28  from openlego.test_suite.test_examples.sellar.kb import root_tag, x_x1, x_y1, x_y2, x_z2, x_f1
29  from openlego.partials.partials import Partials
30
31  import openlego.test_suite.test_examples.sellar.store as store
32
33
34  class F1(AbstractDiscipline):
35
36      @property
37      def creator(self):
38          return u'D. de Vries'
39
```

```
40      @property
41      def description(self):
42          return u'Objective function of the Sellar problem'
43
44      @property
45      def supplies_partials(self):
46          return True
47
48      def generate_input_xml(self):
49          root = etree.Element(root_tag)
50          doc = etree.ElementTree(root)
51
52          xml_safe_create_element(doc, x_z2, 0.)
53          xml_safe_create_element(doc, x_x1, 0.)
54          xml_safe_create_element(doc, x_y1, 0.)
55          xml_safe_create_element(doc, x_y2, 0.)
56
57          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
58
59      def generate_output_xml(self):
60          root = etree.Element(root_tag)
61          doc = etree.ElementTree(root)
62
63          xml_safe_create_element(doc, x_f1, 0.)
64
65          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
66
67      def generate_partials_xml(self):
68          partials = Partials()
69          partials.declare_partials(x_f1, [x_x1, x_y1, x_y2, x_z2])
70          return partials.get_string()
71
72      @staticmethod
73      def execute(in_file, out_file):
74          store.count[2] += 1
75          store.sleep()
76          doc = etree.parse(in_file)
77          z2 = float(doc.xpath(x_z2)[0].text)
78          x1 = float(doc.xpath(x_x1)[0].text)
79          y1 = float(doc.xpath(x_y1)[0].text)
80          y2 = float(doc.xpath(x_y2)[0].text)
81
82          f1 = x1**2. + z2 + y1 + exp(-y2)
83
84          root = etree.Element(root_tag)
85          doc = etree.ElementTree(root)
86          xml_safe_create_element(doc, x_f1, f1)
87          doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
88
89      @staticmethod
90      def linearize(in_file, partials_file):
91          doc = etree.parse(in_file)
92          x1 = float(doc.xpath(x_x1)[0].text)
93          y2 = float(doc.xpath(x_y2)[0].text)
94
95          partials = Partials()
96          partials.declare_partials(x_f1, [x_x1, x_y1, x_y2, x_z2], [2.*x1, 1., -exp(-y2), 1.])
97          partials.write(partials_file)
```

Code frament A.22: Code of the Sellar `F1` Python module.

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
```

```
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the Sellar G1 discipline.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from lxml import etree
23
24  from openlego.api import AbstractDiscipline
25  from openlego.utils.xml_utils import xml_safe_create_element
26  from openlego.test_suite.test_examples.sellar.kb import root_tag, x_y1, x_g1
27  from openlego.partials.partials import Partials
28
29  import openlego.test_suite.test_examples.sellar.store as store
30
31
32  class G1(AbstractDiscipline):
33
34      @property
35      def creator(self):
36          return u'D. de Vries'
37
38      @property
39      def description(self):
40          return u'First constraint function of the Sellar problem'
41
42      @property
43      def supplies_partials(self):
44          return True
45
46      def generate_input_xml(self):
47          root = etree.Element(root_tag)
48          doc = etree.ElementTree(root)
49
50          xml_safe_create_element(doc, x_y1, 0.)
51
52          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
53
54      def generate_output_xml(self):
55          root = etree.Element(root_tag)
56          doc = etree.ElementTree(root)
57
58          xml_safe_create_element(doc, x_g1, 0.)
59
60          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
61
62      def generate_partials_xml(self):
63          partials = Partials()
64          partials.declare_partials(x_g1, x_y1)
65          return partials.get_string()
66
67      @staticmethod
68      def execute(in_file, out_file):
69          store.count[3] += 1
70          store.sleep()
71          doc = etree.parse(in_file)
72          y1 = float(doc.xpath(x_y1)[0].text)
73
74          g1 = 1. - y1/3.16
75
76          root = etree.Element(root_tag)
77          doc = etree.ElementTree(root)
78          xml_safe_create_element(doc, x_g1, g1)
79          doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
80
```

```
81      @staticmethod
82      def linearize(in_file, partials_file):
83          partials = Partials()
84          partials.declare_partials(x_g1, x_y1, -1./3.16)
85          partials.write(partials_file)
```

Code frament A.23: Code of the Sellar `G1` Python module.

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3   """
4   Copyright 2017 D. de Vries
5
6   Licensed under the Apache License, Version 2.0 (the "License");
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the definition of the Sellar G2 discipline.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from lxml import etree
23
24  from openlego.api import AbstractDiscipline
25  from openlego.utils.xml_utils import xml_safe_create_element
26  from openlego.test_suite.test_examples.sellar.kb import root_tag, x_y2, x_g2
27  from openlego.partials.partials import Partials
28
29  import openlego.test_suite.test_examples.sellar.store as store
30
31
32  class G2(AbstractDiscipline):
33
34      @property
35      def creator(self):
36          return u'D. de Vries'
37
38      @property
39      def description(self):
40          return u'First constraint function of the Sellar problem'
41
42      @property
43      def supplies_partials(self):
44          return True
45
46      def generate_input_xml(self):
47          root = etree.Element(root_tag)
48          doc = etree.ElementTree(root)
49
50          xml_safe_create_element(doc, x_y2, 0.)
51
52          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
53
54      def generate_output_xml(self):
55          root = etree.Element(root_tag)
56          doc = etree.ElementTree(root)
57
58          xml_safe_create_element(doc, x_g2, 0.)
59
60          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
61
62      def generate_partials_xml(self):
```

```
        partials = Partials()
        partials.declare_partials(x_g2, x_y2)
        return partials.get_string()

    @staticmethod
    def execute(in_file, out_file):
        store.count[4] += 1
        store.sleep()
        doc = etree.parse(in_file)
        y2 = float(doc.xpath(x_y2)[0].text)

        g2 = y2/24. - 1.

        root = etree.Element(root_tag)
        doc = etree.ElementTree(root)
        xml_safe_create_element(doc, x_g2, g2)
        doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)

    @staticmethod
    def linearize(in_file, partials_file):
        partials = Partials()
        partials.declare_partials(x_g2, x_y2, 1./24.)
        partials.write(partials_file)
```

Code frament A.24: Code of the Sellar `G2` Python module.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of simple elements -->
<xs:element name="x1" type="xs:decimal"/>
<xs:element name="z1" type="xs:decimal"/>
<xs:element name="z2" type="xs:decimal"/>
<xs:element name="y1" type="xs:decimal"/>
<xs:element name="y2" type="xs:decimal"/>
<xs:element name="g1" type="xs:decimal"/>
<xs:element name="g2" type="xs:decimal"/>
<xs:element name="f1" type="xs:decimal"/>

<!-- definition of complex elements -->

<xs:element name="data_schema">
  <xs:complexType>
    <xs:all>
      <xs:element ref="x1" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="z1" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="z2" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="y1" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="y2" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="g1" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="g2" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="f1" minOccurs="0" maxOccurs="1"/>
    </xs:all>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Code frament A.25: Data schema of the Sellar problem.

## A.5.2. Wing Optimization

### Test Case

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
```

```
7   you may not use this file except in compliance with the License.
8   You may obtain a copy of the License at
9
10      http://www.apache.org/licenses/LICENSE-2.0
11
12  Unless required by applicable law or agreed to in writing, software
13  distributed under the License is distributed on an "AS IS" BASIS,
14  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  See the License for the specific language governing permissions and
16  limitations under the License.
17
18  This file contains the test case for the wing optimization example problem.
19  """
20  from __future__ import absolute_import, division, print_function
21
22  from datetime import datetime
23  import unittest
24  from os import path
25  from shutil import copyfile
26
27  from openmdao.api import Problem, ScipyOptimizer
28  from openmdao.recorders.base_recorder import BaseRecorder
29
30  from openlego.api import LEGOModel
31  from openlego.recorders import NormalizedDesignVarPlotter, ConstraintsPlotter,
      ↪ SimpleObjectivePlotter
32
33
34  class MyRecorder(BaseRecorder):
35      def record_metadata_system(self, object_requesting_recording):
36          pass
37
38      def record_metadata_solver(self, object_requesting_recording):
39          pass
40
41      def record_metadata_driver(self, object_requesting_recording):
42          pass
43
44      def record_iteration_driver_passing_vars(self, object_requesting_recording, desvars,
      ↪ responses, objectives,
45                                               constraints, sysvars, metadata):
46          super(MyRecorder,
      ↪ self).record_iteration_driver_passing_vars(object_requesting_recording, desvars,
47                                                                                responses,
      ↪ objectives, constraints, sysvars,
48                                                                                metadata)
49          src = path.abspath('base.xml')
50          dst = path.abspath('base-' + datetime.now().strftime('%Y%m%d%H%M%f') + '.xml')
51          copyfile(src, dst)
52
53
54  class TestWingOptimization(unittest.TestCase):
55
56      def test_wing_opt(self):
57          """Solve the wing optimization problem."""
58          # 1. Create a Problem
59          prob = Problem()
60          prob.set_solver_print(0)
61
62          base_path = path.abspath('base.xml')
63          copyfile(path.abspath('input.xml'), base_path)
64
65          # 2. Create the LEGOModel
66          model = prob.model = LEGOModel('MDG_MDF-GS-full.xml',           # CMDOWS file
67                                         'kb',                              # Knowledge base
68                                         '',                                # Output directory
69                                         base_path)               # Output file
70
71          # 3. Create a Driver object
72          driver = prob.driver = ScipyOptimizer()
73          driver.options['optimizer'] = 'SLSQP'
```

```
        driver.options['disp'] = True
        driver.options['tol'] = 1.0e-2
        driver.opt_settings = {'disp': True, 'iprint': 2, 'ftol': 1.0e-2}

        # 4. Setup the problem
        prob.setup()

        from openmdao.api import view_model
        view_model(prob)
        model.coupled_group.linear_solver.options['maxiter'] = 10  # Increase maxiter of the linear solver
        model.coupled_group.nonlinear_solver.options['maxiter'] = 1  # Increase maxiter of the nonlinear solver
        model.coupled_group.linear_solver.options['rtol'] = 1e-2
        model.coupled_group.nonlinear_solver.options['rtol'] = 1e-2
        model.approx_totals(method='fd')
        prob.final_setup()

        model.initialize_from_xml('input.xml')
        prob.run_model()

        # 5. Attach some Recorders
        desvar_plotter = NormalizedDesignVarPlotter()  # Create a plotter for the design variables
        desvar_plotter.options['save_on_close'] = True  # Should this plot be saved automatically?
        desvar_plotter.save_settings['path'] = 'desvar.png'  # Set the filename of the image file

        convar_plotter = ConstraintsPlotter()  # Create a plotter for the constraints
        convar_plotter.options['save_on_close'] = True  # Should this plot be saved automatically?
        convar_plotter.save_settings['path'] = 'convar.png'  # Set the filename of the image file

        objvar_plotter = SimpleObjectivePlotter()  # Create a plotter for the objective
        objvar_plotter.options['save_on_close'] = True  # Should this plot be saved automatically?
        objvar_plotter.save_settings['path'] = 'objvar.png'  # Set the filename of the image file

        driver.add_recorder(desvar_plotter)  # Attach the design variable plotter
        driver.add_recorder(convar_plotter)  # Attach the constraint variable plotter
        driver.add_recorder(objvar_plotter)  # Attach the objective variable plotter

        # 6. Solve the problem
        prob.run_driver()
        prob.cleanup()
```

Code frament A.26: Code of the wing optimization test Python script.

```
<?xml version='1.0' encoding='UTF-8'?>
<cmdows xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="https://bitbucket.org/imcovangent/cmdows/raw/master/schema/0.7/cmdows.xsd">
  <header>
    <creator>D. de Vries</creator>
    <description>Wing optimization MPG file</description>
    <timestamp>2017-11-07T15:13:10.036671</timestamp>
    <fileVersion>0.1</fileVersion>
    <cmdowsVersion>0.7</cmdowsVersion>
    <updates>
      <update>
        <modification>KADMOS export of a mdao data graph (MDG).</modification>
        <creator>D. de Vries</creator>
        <timestamp>2017-11-07T15:13:10.036671</timestamp>
        <fileVersion>0.1</fileVersion>
        <cmdowsVersion>0.7</cmdowsVersion>
      </update>
    </updates>
  </header>
```

```
<executableBlocks>
  <designCompetences>
    <designCompetence uID="dLC">
      <ID>dLC</ID>
      <modeID>main</modeID>
      <instanceID>1</instanceID>
      <version>1.0</version>
      <label>dLC</label>
      <inputs>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_rs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_bs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_ts</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_ts</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_ts</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_bs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_fs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_fs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_bs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_rs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_rs</parameterUID>
        </input>
        <input>
<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_fs</parameterUID>
        </input>
      </inputs>
      <outputs>
        <output>
          <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_ts</parameterUID>
        </output>
        <output>
          <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_bs</parameterUID>
        </output>
        <output>
          <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_rs</parameterUID>
        </output>
        <output>
          <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_fs</parameterUID>
        </output>
```

```
78          </outputs>
79          <metadata>
80            <generalInfo>
81              <description>main execution mode</description>
82            </generalInfo>
83          </metadata>
84        </designCompetence>
85        <designCompetence uID="FWE">
86          <ID>FWE</ID>
87          <modeID>main</modeID>
88          <instanceID>1</instanceID>
89          <version>1.0</version>
90          <label>FWE</label>
91          <inputs>
92            <input>
93
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_f</parameterUID>
94            </input>
95            <input>
96              <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n</parameterUID>
97            </input>
98            <input>
99              <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_other</parameterUID>
100           </input>
101           <input>
102
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/reference/R</parameterUID>
103           </input>
104           <input>
105
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_L</parameterUID>
106           </input>
107           <input>
108
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_i</parameterUID>
109           </input>
110           <input>
111             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H</parameterUID>
112           </input>
113           <input>
114
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_f</parameterUID>
115           </input>
116           <input>
117             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_fus</parameterUID>
118           </input>
119           <input>
120             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M</parameterUID>
121           </input>
122           <input>
123             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed</parameterUID>
124           </input>
125           <input>
126             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_res</parameterUID>
127           </input>
128           <input>
129
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/reference/SFC</parameterUID>
130           </input>
131           <input>
132             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M</parameterUID>
133           </input>
134           <input>
135             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H</parameterUID>
136           </input>
137           <input>
138             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H</parameterUID>
```

```
139           </input>
140           <input>
141             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M</parameterUID>
142           </input>
143           <input>
144             <parameterUID>/cpacs/toolspecific/dAEDalus/m_wing</parameterUID>
145           </input>
146           <input>
147
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_L</parameterUID>
148           </input>
149           <input>
150             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n</parameterUID>
151           </input>
152           <input>
153
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_i</parameterUID>
154           </input>
155           <input>
156
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_f</parameterUID>
157           </input>
158           <input>
159
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_i</parameterUID>
160           </input>
161           <input>
162             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n</parameterUID>
163           </input>
164           <input>
165
    ↪ <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_L</parameterUID>
166           </input>
167         </inputs>
168         <outputs>
169           <output>
170             <parameterUID>/cpacs/toolspecific/fuel_weight_estimator/C_L</parameterUID>
171           </output>
172           <output>
173             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MTOW</parameterUID>
174           </output>
175           <output>
176             <parameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</parameterUID>
177           </output>
178           <output>
179             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalCouplingVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m
180           </output>
181         </outputs>
182         <metadata>
183           <generalInfo>
184             <description>main execution mode</description>
185           </generalInfo>
186         </metadata>
187       </designCompetence>
188       <designCompetence uID="ObjectiveFunctions">
189         <ID>ObjectiveFunctions</ID>
190         <modeID>main</modeID>
191         <instanceID>1</instanceID>
192         <version>1.0</version>
193         <label>ObjectiveFunctions</label>
194         <inputs>
195           <input>
196             <parameterUID>/cpacs/toolspecific/dAEDalus/m_wing</parameterUID>
197           </input>
198           <input>
199             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_wing_init</parameterUID>
200           </input>
201           <input>
```

```
202             <parameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</parameterUID>
203           </input>
204           <input>
205             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_init</parameterUID>
206           </input>
207         </inputs>
208         <outputs>
209           <output>
210             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/objecti
211           </output>
212           <output>
213             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</parameterUID>
214           </output>
215         </outputs>
216         <metadata>
217           <generalInfo>
218             <description>main execution mode</description>
219           </generalInfo>
220         </metadata>
221       </designCompetence>
222       <designCompetence uID="ConstraintFunctions">
223         <ID>ConstraintFunctions</ID>
224         <modeID>main</modeID>
225         <instanceID>1</instanceID>
226         <version>1.0</version>
227         <label>ConstraintFunctions</label>
228         <inputs>
229           <input>
230             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_L_buffet</parameterUID>
231           </input>
232           <input>
233             <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_rs</parameterUID>
234           </input>
235           <input>
236             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/WS_init</parameterUID>
237           </input>
238           <input>
239             <parameterUID>/cpacs/toolspecific/fuel_weight_estimator/C_L</parameterUID>
240           </input>
241           <input>
242             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MTOW</parameterUID>
243           </input>
244           <input>
245             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/sigma_yield</parameterUID>
246           </input>
247           <input>
248             <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_fs</parameterUID>
249           </input>
250           <input>
251             <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_bs</parameterUID>
252           </input>
253           <input>
254             <parameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_ts</parameterUID>
255           </input>
256           <input>
257
    ↪ <parameterUID>/cpacs/vehicles/aircraft/model[@uID="model"]/reference/area</parameterUID>
258           </input>
259         </inputs>
260         <outputs>
261           <output>
262             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constra
263           </output>
```

```
264           <output>
265             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
266           </output>
267           <output>
268             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</parameterUID>
269           </output>
270           <output>
271             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</parameterUID>
272           </output>
273           <output>
274             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</parameterUID>
275           </output>
276           <output>
277             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
278           </output>
279           <output>
280             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
281           </output>
282           <output>
283             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</parameterUID>
284           </output>
285           <output>
286             <parame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
287           </output>
288           <output>
289             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</parameterUID>
290           </output>
291         </outputs>
292         <metadata>
293           <generalInfo>
294             <description>main execution mode</description>
295           </generalInfo>
296         </metadata>
297       </designCompetence>
298       <designCompetence uID="Aeroelastics">
299         <ID>Aeroelastics</ID>
300         <modeID>main</modeID>
301         <instanceID>1</instanceID>
302         <version>1.0</version>
303         <label>Aeroelastics</label>
304         <inputs>
305           <input>
306             <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n</parameterUID>
307           </input>
308           <input>
309             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/incidence</parameterUID>
310           </input>
311           <input>
312             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_wings</parameterUID>
313           </input>
314           <input>
315             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</parameterUID>
316           </input>
317           <input>
318
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</parameterUID>
319           </input>
320           <input>
```

```
321
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MLW</parameterUID>
322         </input>
323         <input>
324
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Gamma</parameterUID>
325         </input>
326         <input>
327
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</parameterUID>
328         </input>
329         <input>
330           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M</parameterUID>
331         </input>
332         <input>
333           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed</parameterUID>
334         </input>
335         <input>
336           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_payload</parameterUID>
337         </input>
338         <input>
339           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M</parameterUID>
340         </input>
341         <input>
342           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H</parameterUID>
343         </input>
344         <input>
345           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H</parameterUID>
346         </input>
347         <input>
348           <parame-
    ↪ terUID>/cpacs/architectureNodes/couplingCopyVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_fuel</
349         </input>
350         <input>
351           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_skin</parameterUID>
352         </input>
353         <input>
354           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M</parameterUID>
355         </input>
356         <input>
357
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Lambda</parameterUID>
358         </input>
359         <input>
360
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</parameterUID>
361         </input>
362         <input>
363           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</parameterUID>
364         </input>
365         <input>
366           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_sys</parameterUID>
367         </input>
368         <input>
369
    ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</parameterUID>
370         </input>
371         <input>
372           <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n</parameterUID>
373         </input>
374         <input>
375           <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</parameterUID>
376         </input>
377         <input>
```

```
<parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/tc</parameterUID>
      </input>
      <input>
        <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n</parameterUID>
      </input>
      <input>
        <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/reference/rho_skin</parameterUID>
      </input>
      <input>

<parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</parameterUID>
      </input>
      <input>

<parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</parameterUID>
      </input>
      <input>
        <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H</parameterUID>
      </input>
    </inputs>
    <outputs>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_ts</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_bs</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_fs</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_L</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_bs</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_i</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_ts</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_f</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_L</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/vehicles/aircraft/model[@uID="model"]/reference/area</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_rs</parameterUID>
      </output>
      <output>

<parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_fs</parameterUID>
      </output>
```

```
433             <output>
434
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_rs</parameterUID>
435             </output>
436             <output>
437
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_ts</parameterUID>
438             </output>
439             <output>
440               <parameterUID>/cpacs/toolspecific/dAEDalus/m_wing</parameterUID>
441             </output>
442             <output>
443
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_i</parameterUID>
444             </output>
445             <output>
446
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_f</parameterUID>
447             </output>
448             <output>
449
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_i</parameterUID>
450             </output>
451             <output>
452
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_f</parameterUID>
453             </output>
454             <output>
455
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_L</parameterUID>
456             </output>
457             <output>
458
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_fs</parameterUID>
459             </output>
460             <output>
461
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_bs</parameterUID>
462             </output>
463             <output>
464
    ↪  <parameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_rs</parameterUID>
465             </output>
466           </outputs>
467           <metadata>
468             <generalInfo>
469               <description>main execution mode</description>
470             </generalInfo>
471           </metadata>
472         </designCompetence>
473       </designCompetences>
474     </executableBlocks>
475     <parameters>
476       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs">
477         <label>t_fs</label>
478       </parameter>
479       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_init">
480         <label>m_fuel_init</label>
481       </parameter>
482       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs">
483         <label>con_sigma_bs</label>
484       </parameter>
485       <parameter uID="/cpacs/toolspecific/dAEDalus/load_collector/sigma_bs">
486         <label>sigma_bs</label>
487       </parameter>
488       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_other">
489         <label>C_D_other</label>
490       </parameter>
491       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_ts">
492         <label>sigma_ts</label>
493       </parameter>
```

```
494     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_res">
495       <label>m_fuel_res</label>
496     </parameter>
497     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs">
498       <label>con_sigma_rs</label>
499     </parameter>
500     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_rs">
501       <label>sigma_rs</label>
502     </parameter>
503     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/sigma_yield">
504       <label>sigma_yield</label>
505     </parameter>
506     <parameter uID="/cpacs/toolspecific/dAEDalus/load_collector/sigma_rs">
507       <label>sigma_rs</label>
508     </parameter>
509     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon">
510       <label>epsilon</label>
511     </parameter>
512     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_L">
513       <label>C_L</label>
514     </parameter>
515     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_bs">
516       <label>sigma_bs</label>
517     </parameter>
518     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_bs">
519       <label>sigma_bs</label>
520     </parameter>
521     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_MLW">
522       <label>m_MLW</label>
523     </parameter>
524     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_ts">
525       <label>sigma_ts</label>
526     </parameter>
527     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_payload">
528       <label>m_payload</label>
529     </parameter>
530     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts">
531       <label>con_sigma_ts</label>
532     </parameter>
533     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_L">
534       <label>C_L</label>
535     </parameter>
536     <parameter uID="/cpacs/vehicles/aircraft/model[@uID=&amp;quot;model&amp;quot;]/reference/area">
537       <label>area</label>
538     </parameter>
539     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_rs">
540       <label>sigma_rs</label>
541     </parameter>
542     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_fs">
543       <label>sigma_fs</label>
544     </parameter>
545     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/t_skin">
546       <label>t_skin</label>
547     </parameter>
548     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs">
549       <label>xsi_fs</label>
550     </parameter>
551     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_i">
552       <label>C_D_i</label>
553     </parameter>
554     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/C_L_buffet">
555       <label>C_L_buffet</label>
556     </parameter>
557     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/tc">
558       <label>tc</label>
559     </parameter>
560     <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_wing_init">
561       <label>m_wing_init</label>
562     </parameter>
563     <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_ts">
564       <label>sigma_ts</label>
```

```
565       </parameter>
566       <parameter uID="/cpacs/toolspecific/fuel_weight_estimator/C_L">
567         <label>C_L</label>
568       </parameter>
569       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_fs">
570         <label>sigma_fs</label>
571       </parameter>
572       <parameter uID="/cpacs/toolspecific/dAEDalus/load_collector/sigma_fs">
573         <label>sigma_fs</label>
574       </parameter>
575       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_wings">
576         <label>f_m_wings</label>
577       </parameter>
578       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel">
579         <label>obj_m_fuel</label>
580       </parameter>
581       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs">
582         <label>t_bs</label>
583       </parameter>
584       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/Gamma">
585         <label>Gamma</label>
586       </parameter>
587       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_f">
588         <label>C_D_f</label>
589       </parameter>
590       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_fus">
591         <label>C_D_fus</label>
592       </parameter>
593       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_MTOW">
594         <label>m_MTOW</label>
595       </parameter>
596       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/SFC">
597         <label>SFC</label>
598       </parameter>
599       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_sys">
600         <label>f_m_sys</label>
601       </parameter>
602       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts">
603         <label>t_ts</label>
604       </parameter>
605       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_i">
606         <label>C_D_i</label>
607       </parameter>
608       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_f">
609         <label>C_D_f</label>
610       </parameter>
611       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_i">
612         <label>C_D_i</label>
613       </parameter>
614       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_f">
615         <label>C_D_f</label>
616       </parameter>
617       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/WS_init">
618         <label>WS_init</label>
619       </parameter>
620       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs">
621         <label>con_sigma_fs</label>
622       </parameter>
623       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS">
624         <label>con_WS</label>
625       </parameter>
626       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/incidence">
627         <label>incidence</label>
628       </parameter>
629       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/R">
630         <label>R</label>
631       </parameter>
632       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs">
633         <label>xsi_rs</label>
634       </parameter>
635       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_L">
```

```
636         <label>C_L</label>
637       </parameter>
638       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_fs">
639         <label>sigma_fs</label>
640       </parameter>
641       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_bs">
642         <label>sigma_bs</label>
643       </parameter>
644       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/Lambda">
645         <label>Lambda</label>
646       </parameter>
647       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs">
648         <label>t_rs</label>
649       </parameter>
650       <parameter uID="/cpacs/toolspecific/dAEDalus/load_collector/sigma_ts">
651         <label>sigma_ts</label>
652       </parameter>
653       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/rho_skin">
654         <label>rho_skin</label>
655       </parameter>
656       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_rs">
657         <label>sigma_rs</label>
658       </parameter>
659       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/c">
660         <label>c</label>
661       </parameter>
662       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/planform/b">
663         <label>b</label>
664       </parameter>
665       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n">
666         <label>n</label>
667       </parameter>
668       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H">
669         <label>H</label>
670       </parameter>
671       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M">
672         <label>M</label>
673       </parameter>
674       <parameter uID="/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed">
675         <label>m_fixed</label>
676       </parameter>
677       <parameter uID="/cpacs/toolspecific/fuel_weight_estimator/m_fuel">
678         <label>m_fuel</label>
679       </parameter>
680       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M">
681         <label>M</label>
682       </parameter>
683       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H">
684         <label>H</label>
685       </parameter>
686       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H">
687         <label>H</label>
688       </parameter>
689       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M">
690         <label>M</label>
691       </parameter>
692       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n">
693         <label>n</label>
694       </parameter>
695       <parameter uID="/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n">
696         <label>n</label>
697       </parameter>
698       <parameter uID="/cpacs/toolspecific/dAEDalus/m_wing">
699         <label>m_wing</label>
700       </parameter>
701     </parameters>
702     <problemDefinition uID="MDFGauss-Seidel">
703       <problemFormulation>
704         <mdaoArchitecture>MDF</mdaoArchitecture>
705         <convergerType>Gauss-Seidel</convergerType>
706         <executableBlocksOrder>
```

```
      <executableBlock position="1">Aeroelastics</executableBlock>
      <executableBlock position="2">FWE</executableBlock>
      <executableBlock position="3">dLC</executableBlock>
      <executableBlock position="4">ConstraintFunctions</executableBlock>
      <executableBlock position="5">ObjectiveFunctions</executableBlock>
    </executableBlocksOrder>
    <allowUnconvergedCouplings>false</allowUnconvergedCouplings>
  </problemFormulation>
  <problemRoles>
    <parameters>
      <designVariables>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs">

↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</parameterUID>
          <nominalValue>
            <nominalvalue>0.00450588</nominalvalue>
            <nominalvalue>0.00458215</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>0.001</minimum>
                <minimum>0.001</minimum>
              </minimum>
              <maximum>
                <maximum>0.03</maximum>
                <maximum>0.03</maximum>
              </maximum>
            </limitRange>
          </validRanges>
        </designVariable>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon">
          <parame-
↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</parameterUID>
          <nominalValue>
            <nominalvalue>-0.1039</nominalvalue>
            <nominalvalue>-0.1826</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>-0.25</minimum>
                <minimum>-0.25</minimum>
              </minimum>
              <maximum>
                <maximum>0.25</maximum>
                <maximum>0.25</maximum>
              </maximum>
            </limitRange>
          </validRanges>
        </designVariable>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs">
          <parame-
↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</parameterUID>
          <nominalValue>
            <nominalvalue>0.1</nominalvalue>
            <nominalvalue>0.1925</nominalvalue>
            <nominalvalue>0.35</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>0.05</minimum>
                <minimum>0.05</minimum>
                <minimum>0.05</minimum>
              </minimum>
              <maximum>
                <maximum>0.4</maximum>
```

```
                <maximum>0.4</maximum>
                <maximum>0.4</maximum>
              </maximum>
            </limitRange>
          </validRanges>
        </designVariable>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs">

↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</parameterUID>
          <nominalValue>
            <nominalvalue>0.02553329</nominalvalue>
            <nominalvalue>0.02237119</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>0.001</minimum>
                <minimum>0.001</minimum>
              </minimum>
              <maximum>
                <maximum>0.03</maximum>
                <maximum>0.03</maximum>
              </maximum>
            </limitRange>
          </validRanges>
        </designVariable>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts">

↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</parameterUID>
          <nominalValue>
            <nominalvalue>0.02553329</nominalvalue>
            <nominalvalue>0.02237119</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>0.001</minimum>
                <minimum>0.001</minimum>
              </minimum>
              <maximum>
                <maximum>0.03</maximum>
                <maximum>0.03</maximum>
              </maximum>
            </limitRange>
          </validRanges>
        </designVariable>
        <designVariable
↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs">
          <parame-
↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</parameterUID>
          <nominalValue>
            <nominalvalue>0.6</nominalvalue>
            <nominalvalue>0.8023</nominalvalue>
            <nominalvalue>0.6</nominalvalue>
          </nominalValue>
          <validRanges>
            <limitRange>
              <minimum>
                <minimum>0.6</minimum>
                <minimum>0.6</minimum>
                <minimum>0.6</minimum>
              </minimum>
              <maximum>
                <maximum>0.9</maximum>
                <maximum>0.9</maximum>
                <maximum>0.9</maximum>
              </maximum>
            </limitRange>
          </validRanges>
```

```
837           </designVariable>
838           <designVariable
      ↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs">
839
      ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</parameterUID>
840             <nominalValue>
841               <nominalvalue>0.00450611</nominalvalue>
842               <nominalvalue>0.00456957</nominalvalue>
843             </nominalValue>
844             <validRanges>
845               <limitRange>
846                 <minimum>
847                   <minimum>0.001</minimum>
848                   <minimum>0.001</minimum>
849                 </minimum>
850                 <maximum>
851                   <maximum>0.03</maximum>
852                   <maximum>0.03</maximum>
853                 </maximum>
854               </limitRange>
855             </validRanges>
856           </designVariable>
857           <designVariable
      ↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/c">
858
      ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</parameterUID>
859             <nominalValue>
860               <nominalvalue>13.7131</nominalvalue>
861               <nominalvalue>7.2595</nominalvalue>
862               <nominalvalue>2.7341</nominalvalue>
863             </nominalValue>
864             <validRanges>
865               <limitRange>
866                 <minimum>
867                   <minimum>1.0</minimum>
868                   <minimum>1.0</minimum>
869                   <minimum>1.0</minimum>
870                 </minimum>
871                 <maximum>
872                   <maximum>15.0</maximum>
873                   <maximum>15.0</maximum>
874                   <maximum>15.0</maximum>
875                 </maximum>
876               </limitRange>
877             </validRanges>
878           </designVariable>
879           <designVariable
      ↪ uID="__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/b">
880
      ↪ <parameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</parameterUID>
881             <nominalValue>
882               <nominalvalue>12.7178</nominalvalue>
883               <nominalvalue>22.7016</nominalvalue>
884             </nominalValue>
885             <validRanges>
886               <limitRange>
887                 <minimum>
888                   <minimum>5.0</minimum>
889                   <minimum>5.0</minimum>
890                 </minimum>
891                 <maximum>
892                   <maximum>25.0</maximum>
893                   <maximum>25.0</maximum>
894                 </maximum>
895               </limitRange>
896             </validRanges>
897           </designVariable>
898         </designVariables>
899         <objectiveVariables>
900           <objectiveVariable
      ↪ uID="__objVar__/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel">
```

```
901             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</parameterUID>
902           </objectiveVariable>
903         </objectiveVariables>
904         <constraintVariables>
905           <constraintVariable
    ↪ uID="__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs">
906             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</parameterUID>
907             <constraintType>inequality</constraintType>
908             <constraintOperator>&lt;=</constraintOperator>
909             <referenceValue>0.0</referenceValue>
910           </constraintVariable>
911           <constraintVariable
    ↪ uID="__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs">
912             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</parameterUID>
913             <constraintType>inequality</constraintType>
914             <constraintOperator>&lt;=</constraintOperator>
915             <referenceValue>0.0</referenceValue>
916           </constraintVariable>
917           <constraintVariable
    ↪ uID="__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts">
918             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</parameterUID>
919             <constraintType>inequality</constraintType>
920             <constraintOperator>&lt;=</constraintOperator>
921             <referenceValue>0.0</referenceValue>
922           </constraintVariable>
923           <constraintVariable
    ↪ uID="__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs">
924             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</parameterUID>
925             <constraintType>inequality</constraintType>
926             <constraintOperator>&lt;=</constraintOperator>
927             <referenceValue>0.0</referenceValue>
928           </constraintVariable>
929           <constraintVariable
    ↪ uID="__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS">
930             <parame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</parameterUID>
931             <constraintType>inequality</constraintType>
932             <constraintOperator>&lt;=</constraintOperator>
933             <referenceValue>0.0</referenceValue>
934           </constraintVariable>
935         </constraintVariables>
936       </parameters>
937       <executableBlocks>
938         <coupledBlocks>
939           <coupledBlock>Aeroelastics</coupledBlock>
940           <coupledBlock>FWE</coupledBlock>
941         </coupledBlocks>
942         <postCouplingBlocks>
943           <postCouplingBlock>dLC</postCouplingBlock>
944           <postCouplingBlock>ConstraintFunctions</postCouplingBlock>
945           <postCouplingBlock>ObjectiveFunctions</postCouplingBlock>
946         </postCouplingBlocks>
947       </executableBlocks>
948     </problemRoles>
949   </problemDefinition>
950   <workflow>
951     <problemDefinitionUID>MDFGauss-Seidel</problemDefinitionUID>
952     <dataGraph>
953       <name>MDG1</name>
954       <edges>
955         <edge>
956           <fromParame-
    ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
957           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
958         </edge>
959         <edge>
```

```
960           <fromParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_init</fromParameterUID>
961           <toExecutableBlockUID>ObjectiveFunctions</toExecutableBlockUID>
962         </edge>
963         <edge>
964           <fromExecutableBlockUID>FWE</fromExecutableBlockUID>
965           <toParameterUID>/cpacs/toolspecific/fuel_weight_estimator/C_L</toParameterUID>
966         </edge>
967         <edge>
968           <fromExecutableBlockUID>FWE</fromExecutableBlockUID>
969           <toParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MTOW</toParameterUID>
970         </edge>
971         <edge>
972           <fromExecutableBlockUID>FWE</fromExecutableBlockUID>
973           <toParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</toParameterUID>
974         </edge>
975         <edge>
976           <fromExecutableBlockUID>FWE</fromExecutableBlockUID>
977           <toParame-
    ↪ terUID>/cpacs/architectureNodes/finalCouplingVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_fuel<
978         </edge>
979         <edge>
980           <fromParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</fromParameterUID>
981           <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
982         </edge>
983         <edge>
984           <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n</fromParameterUID>
985           <toExecutableBlockUID>FWE</toExecutableBlockUID>
986         </edge>
987         <edge>
988           <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n</fromParameterUID>
989           <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
990         </edge>
991         <edge>
992           <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_bs</fromParameterUID>
993           <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
994         </edge>
995         <edge>
996           <fromParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_other</fromParameterUID>
997           <toExecutableBlockUID>FWE</toExecutableBlockUID>
998         </edge>
999         <edge>
1000          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constra
1001          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1002        </edge>
1003        <edge>
1004          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planfor
1005          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1006        </edge>
1007        <edge>
1008          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_ts</fromParameterUID>
1009          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1010        </edge>
1011        <edge>
1012          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H</fromParameterUID>
1013          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1014        </edge>
1015        <edge>
1016          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H</fromParameterUID>
1017          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
```

```
1018          </edge>
1019          <edge>
1020            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M</fromParameterUID>
1021            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1022          </edge>
1023          <edge>
1024            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M</fromParameterUID>
1025            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1026          </edge>
1027          <edge>
1028            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1029            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1030          </edge>
1031          <edge>
1032            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed</fromParameterUID>
1033            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1034          </edge>
1035          <edge>
1036            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed</fromParameterUID>
1037            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1038          </edge>
1039          <edge>
1040            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_res</fromParameterUID>
1041            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1042          </edge>
1043          <edge>
1044            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</fromParameterUID>
1045            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1046          </edge>
1047          <edge>
1048            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_rs</fromParameterUID>
1049            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1050          </edge>
1051          <edge>
1052            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/sigma_yield</fromParameterUID>
1053            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1054          </edge>
1055          <edge>
1056            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1057            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1058          </edge>
1059          <edge>
1060            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_rs</fromParameterUID>
1061            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1062          </edge>
1063          <edge>
1064            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</fromParameterUID>
1065            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1066          </edge>
1067          <edge>
1068            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_L</fromParameterUID>
1069            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1070          </edge>
1071          <edge>
1072            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1073            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1074          </edge>
```

```
1075        <edge>
1076          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1077          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1078        </edge>
1079        <edge>
1080          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_bs</fromParameterUID>
1081          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1082        </edge>
1083        <edge>
1084
    ↪ <fromParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</fromParameterUID>
1085          <toExecutableBlockUID>ObjectiveFunctions</toExecutableBlockUID>
1086        </edge>
1087        <edge>
1088
    ↪ <fromParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</fromParameterUID>
1089          <toExecutableBlockUID>Converger</toExecutableBlockUID>
1090        </edge>
1091        <edge>
1092          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structu
1093          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1094        </edge>
1095        <edge>
1096          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planfor
1097          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1098        </edge>
1099        <edge>
1100          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planfor
1101          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1102        </edge>
1103        <edge>
1104          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structu
1105          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1106        </edge>
1107        <edge>
1108          <fromParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structu
1109          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1110        </edge>
1111        <edge>
1112          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_bs</fromParameterUID>
1113          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1114        </edge>
1115        <edge>
1116          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M</fromParameterUID>
1117          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1118        </edge>
1119        <edge>
1120          <fromParame-
    ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M</fromParameterUID>
1121          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1122        </edge>
1123        <edge>
1124          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1125          <toParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</toParameterUID>
1126        </edge>
1127        <edge>
1128          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1129          <toParame-
    ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planfor
1130        </edge>
1131        <edge>
```

```
1132          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1133          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/p
1134        </edge>
1135        <edge>
1136          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1137          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1138        </edge>
1139        <edge>
1140          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1141          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1142        </edge>
1143        <edge>
1144          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1145          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</toParameterUID>
1146        </edge>
1147        <edge>
1148          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1149          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1150        </edge>
1151        <edge>
1152          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1153          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/p
1154        </edge>
1155        <edge>
1156          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1157          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</toParameterUID>
1158        </edge>
1159        <edge>
1160          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1161
      ↪ <toParameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</toParameterUID>
1162        </edge>
1163        <edge>
1164          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1165          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</toParameterUID>
1166        </edge>
1167        <edge>
1168          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1169          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1170        </edge>
1171        <edge>
1172          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1173          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1174        </edge>
1175        <edge>
1176          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1177
      ↪ <toParameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</toParameterUID>
1178        </edge>
1179        <edge>
1180          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1181          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1182        </edge>
1183        <edge>
1184          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1185          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</toParameterUID>
1186        </edge>
1187        <edge>
1188          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
```

```
1189          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</toParameterUID>
1190        </edge>
1191        <edge>
1192          <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1193          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</toParameterUID>
1194        </edge>
1195        <edge>
1196          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MLW</fromParameterUID>
1197          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1198        </edge>
1199        <edge>
1200          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1201          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1202        </edge>
1203        <edge>
1204          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1205          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_init</toParameterUID>
1206        </edge>
1207        <edge>
1208          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1209
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/n</toParameterUID>
1210        </edge>
1211        <edge>
1212          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1213          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/incidence</toParameterUID>
1214        </edge>
1215        <edge>
1216          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1217          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_other</toParameterUID>
1218        </edge>
1219        <edge>
1220          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1221
      ↪ <toParameterUID>/cpacs/toolspecific/wingOptimizationProblem/reference/R</toParameterUID>
1222        </edge>
1223        <edge>
1224          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1225          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_wings</toParameterUID>
1226        </edge>
1227        <edge>
1228          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1229          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MLW</toParameterUID>
1230        </edge>
1231        <edge>
1232          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1233          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1234        </edge>
1235        <edge>
1236          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1237          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Gamma</toParameterUID>
1238        </edge>
1239        <edge>
1240          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1241
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/H</toParameterUID>
1242        </edge>
1243        <edge>
1244          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
```

```
1245          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_fus</toParameterUID>
1246        </edge>
1247        <edge>
1248          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1249
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/M</toParameterUID>
1250        </edge>
1251        <edge>
1252          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1253          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1254        </edge>
1255        <edge>
1256          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1257          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fixed</toParameterUID>
1258        </edge>
1259        <edge>
1260          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1261          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1262        </edge>
1263        <edge>
1264          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1265          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_payload</toParameterUID>
1266        </edge>
1267        <edge>
1268          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1269          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_fuel_res</toParameterUID>
1270        </edge>
1271        <edge>
1272          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1273          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/SFC</toParameterUID>
1274        </edge>
1275        <edge>
1276          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1277
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/M</toParameterUID>
1278        </edge>
1279        <edge>
1280          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1281          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1282        </edge>
1283        <edge>
1284          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1285          <toParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessCouplingVariables/cpacsCopy/toolspecific/fuel_weight_esti
1286        </edge>
1287        <edge>
1288          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1289
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H</toParameterUID>
1290        </edge>
1291        <edge>
1292          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1293          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_skin</toParameterUID>
1294        </edge>
1295        <edge>
1296          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1297
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H</toParameterUID>
1298        </edge>
1299        <edge>
1300          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
```

```
1301          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/sigma_yield</toParameterUID>
1302        </edge>
1303        <edge>
1304          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1305          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Lambda</toParameterUID>
1306        </edge>
1307        <edge>
1308          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1309          <toParame-
      ↳ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1310        </edge>
1311        <edge>
1312          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1313          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_sys</toParameterUID>
1314        </edge>
1315        <edge>
1316          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1317
      ↳ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M</toParameterUID>
1318        </edge>
1319        <edge>
1320          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1321          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_L_buffet</toParameterUID>
1322        </edge>
1323        <edge>
1324          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1325
      ↳ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n</toParameterUID>
1326        </edge>
1327        <edge>
1328          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1329          <toParame-
      ↳ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1330        </edge>
1331        <edge>
1332          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1333
      ↳ <toParameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/tc</toParameterUID>
1334        </edge>
1335        <edge>
1336          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1337
      ↳ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n</toParameterUID>
1338        </edge>
1339        <edge>
1340          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1341          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/WS_init</toParameterUID>
1342        </edge>
1343        <edge>
1344          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1345          <toParame-
      ↳ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1346        </edge>
1347        <edge>
1348          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1349          <toParame-
      ↳ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1350        </edge>
1351        <edge>
1352          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1353          <toParame-
      ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/rho_skin</toParameterUID>
1354        </edge>
1355        <edge>
1356          <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
```

```
1357          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_wing_init</toParameterUID>
1358        </edge>
1359        <edge>
1360          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_ts</fromParameterUID>
1361          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1362        </edge>
1363        <edge>
1364          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_payload</fromParameterUID>
1365          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1366        </edge>
1367        <edge>
1368          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</fromParameterUID>
1369          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1370        </edge>
1371        <edge>
1372          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_L</fromParameterUID>
1373          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1374        </edge>
1375        <edge>
1376          <fromParame-
      ↪ terUID>/cpacs/vehicles/aircraft/model[@uID="model"]/reference/area</fromParameterUID>
1377          <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1378        </edge>
1379        <edge>
1380          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_rs</fromParameterUID>
1381          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1382        </edge>
1383        <edge>
1384          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_fs</fromParameterUID>
1385          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1386        </edge>
1387        <edge>
1388          <fromExecutableBlockUID>ObjectiveFunctions</fromExecutableBlockUID>
1389          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1390        </edge>
1391        <edge>
1392          <fromExecutableBlockUID>ObjectiveFunctions</fromExecutableBlockUID>
1393          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</toParameterUID>
1394        </edge>
1395        <edge>
1396          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/couplingCopyVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_
1397          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1398        </edge>
1399        <edge>
1400          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1401          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1402        </edge>
1403        <edge>
1404          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H</fromParameterUID>
1405          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1406        </edge>
1407        <edge>
1408          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/H</fromParameterUID>
1409          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1410        </edge>
1411        <edge>
1412          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H</fromParameterUID>
```

```
1413          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1414        </edge>
1415        <edge>
1416          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/H</fromParameterUID>
1417          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1418        </edge>
1419        <edge>
1420          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_skin</fromParameterUID>
1421          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1422        </edge>
1423        <edge>
1424          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M</fromParameterUID>
1425          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1426        </edge>
1427        <edge>
1428          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/M</fromParameterUID>
1429          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1430        </edge>
1431        <edge>
1432          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</fromParameterUID>
1433          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1434        </edge>
1435        <edge>
1436          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</fromParameterUID>
1437          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1438        </edge>
1439        <edge>
1440          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_L_buffet</fromParameterUID>
1441          <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1442        </edge>
1443        <edge>
1444          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n</fromParameterUID>
1445          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1446        </edge>
1447        <edge>
1448          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/n</fromParameterUID>
1449          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1450        </edge>
1451        <edge>
1452          <fromParame-
    ↳ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1453          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1454        </edge>
1455        <edge>
1456          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/tc</fromParameterUID>
1457          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1458        </edge>
1459        <edge>
1460          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n</fromParameterUID>
1461          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1462        </edge>
1463        <edge>
1464          <fromParame-
    ↳ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/n</fromParameterUID>
1465          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1466        </edge>
1467        <edge>
1468          <fromParame-
    ↳ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_wing_init</fromParameterUID>
1469          <toExecutableBlockUID>ObjectiveFunctions</toExecutableBlockUID>
```

```
1470          </edge>
1471          <edge>
1472            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1473            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1474          </edge>
1475          <edge>
1476            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</fromParameterUID>
1477            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1478          </edge>
1479          <edge>
1480            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1481            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1482          </edge>
1483          <edge>
1484            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_ts</fromParameterUID>
1485            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1486          </edge>
1487          <edge>
1488            <fromParameterUID>/cpacs/toolspecific/fuel_weight_estimator/C_L</fromParameterUID>
1489            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1490          </edge>
1491          <edge>
1492            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_fs</fromParameterUID>
1493            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1494          </edge>
1495          <edge>
1496            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_fs</fromParameterUID>
1497            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1498          </edge>
1499          <edge>
1500            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalCouplingVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m
1501            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1502          </edge>
1503          <edge>
1504            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_wings</fromParameterUID>
1505            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1506          </edge>
1507          <edge>
1508            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</fromParameterUID>
1509            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1510          </edge>
1511          <edge>
1512            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_i</fromParameterUID>
1513            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1514          </edge>
1515          <edge>
1516            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Gamma</fromParameterUID>
1517            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1518          </edge>
1519          <edge>
1520            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_f</fromParameterUID>
1521            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1522          </edge>
1523          <edge>
1524            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/C_D_fus</fromParameterUID>
1525            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1526          </edge>
1527          <edge>
```

```
1528          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/m_MTOW</fromParameterUID>
1529          <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1530        </edge>
1531        <edge>
1532          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/SFC</fromParameterUID>
1533          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1534        </edge>
1535        <edge>
1536          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1537          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1538        </edge>
1539        <edge>
1540          <fromParameterUID>/cpacs/toolspecific/dAEDalus/m_wing</fromParameterUID>
1541          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1542        </edge>
1543        <edge>
1544          <fromParameterUID>/cpacs/toolspecific/dAEDalus/m_wing</fromParameterUID>
1545          <toExecutableBlockUID>ObjectiveFunctions</toExecutableBlockUID>
1546        </edge>
1547        <edge>
1548          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/f_m_sys</fromParameterUID>
1549          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1550        </edge>
1551        <edge>
1552          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</fromParameterUID>
1553          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1554        </edge>
1555        <edge>
1556          <fromExecutableBlockUID>dLC</fromExecutableBlockUID>
1557
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_ts</toParameterUID>
1558        </edge>
1559        <edge>
1560          <fromExecutableBlockUID>dLC</fromExecutableBlockUID>
1561
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_bs</toParameterUID>
1562        </edge>
1563        <edge>
1564          <fromExecutableBlockUID>dLC</fromExecutableBlockUID>
1565
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_rs</toParameterUID>
1566        </edge>
1567        <edge>
1568          <fromExecutableBlockUID>dLC</fromExecutableBlockUID>
1569
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_fs</toParameterUID>
1570        </edge>
1571        <edge>
1572          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_i</fromParameterUID>
1573          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1574        </edge>
1575        <edge>
1576          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_f</fromParameterUID>
1577          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1578        </edge>
1579        <edge>
1580          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_i</fromParameterUID>
1581          <toExecutableBlockUID>FWE</toExecutableBlockUID>
1582        </edge>
1583        <edge>
1584          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_f</fromParameterUID>
1585          <toExecutableBlockUID>FWE</toExecutableBlockUID>
```

```
1586          </edge>
1587          <edge>
1588            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/WS_init</fromParameterUID>
1589            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1590          </edge>
1591          <edge>
1592            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</fromParameterUID>
1593            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1594          </edge>
1595          <edge>
1596            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</fromParameterUID>
1597            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1598          </edge>
1599          <edge>
1600            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/s
1601            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1602          </edge>
1603          <edge>
1604            <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
1605            <toParame-
      ↪ terUID>/cpacs/architectureNodes/couplingCopyVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_
1606          </edge>
1607          <edge>
1608            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/incidence</fromParameterUID>
1609            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1610          </edge>
1611          <edge>
1612            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/R</fromParameterUID>
1613            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1614          </edge>
1615          <edge>
1616            <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</fromParameterUID>
1617            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1618          </edge>
1619          <edge>
1620            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_L</fromParameterUID>
1621            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1622          </edge>
1623          <edge>
1624            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_fs</fromParameterUID>
1625            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1626          </edge>
1627          <edge>
1628            <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_bs</fromParameterUID>
1629            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1630          </edge>
1631          <edge>
1632            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1633            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1634          </edge>
1635          <edge>
1636            <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationPr
1637            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1638          </edge>
1639          <edge>
1640            <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1641            <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_ts</toParameterUID>
1642          </edge>
```

```
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_bs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_fs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>

↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_L</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_bs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_i</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_ts</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/C_D_f</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>

↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_L</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/vehicles/aircraft/model[@uID="model"]/reference/area</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_rs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_fs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_rs</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParame-
↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_ts</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
        <toParameterUID>/cpacs/toolspecific/dAEDalus/m_wing</toParameterUID>
      </edge>
      <edge>
        <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
```

```
1701          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_i</toParameterUID>
1702        </edge>
1703        <edge>
1704          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1705          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_f</toParameterUID>
1706        </edge>
1707        <edge>
1708          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1709          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/C_D_i</toParameterUID>
1710        </edge>
1711        <edge>
1712          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1713          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_D_f</toParameterUID>
1714        </edge>
1715        <edge>
1716          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1717
      ↪ <toParameterUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/C_L</toParameterUID>
1718        </edge>
1719        <edge>
1720          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1721          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[2]/sigma_fs</toParameterUID>
1722        </edge>
1723        <edge>
1724          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1725          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[3]/sigma_bs</toParameterUID>
1726        </edge>
1727        <edge>
1728          <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1729          <toParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_rs</toParameterUID>
1730        </edge>
1731        <edge>
1732          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessCouplingVariables/cpacsCopy/toolspecific/fuel_weight_esti
1733          <toExecutableBlockUID>Converger</toExecutableBlockUID>
1734        </edge>
1735        <edge>
1736          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1737          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1738        </edge>
1739        <edge>
1740          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1741          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1742        </edge>
1743        <edge>
1744          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1745          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</toParameterUID>
1746        </edge>
1747        <edge>
1748          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1749          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</toParameterUID>
1750        </edge>
1751        <edge>
1752          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1753          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</toParameterUID>
1754        </edge>
1755        <edge>
1756          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
```

```
1757          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constra
1758        </edge>
1759        <edge>
1760          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1761          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constra
1762        </edge>
1763        <edge>
1764          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1765          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</toParameterUID>
1766        </edge>
1767        <edge>
1768          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1769          <toParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constra
1770        </edge>
1771        <edge>
1772          <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1773          <toParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</toParameterUID>
1774        </edge>
1775        <edge>
1776          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/Lambda</fromParameterUID>
1777          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1778        </edge>
1779        <edge>
1780          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</fromParameterUID>
1781          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1782        </edge>
1783        <edge>
1784          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/
1785          <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1786        </edge>
1787        <edge>
1788          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/load_collector/sigma_ts</fromParameterUID>
1789          <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1790        </edge>
1791        <edge>
1792          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structu
1793          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1794        </edge>
1795        <edge>
1796          <fromParame-
      ↪ terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/objecti
1797          <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1798        </edge>
1799        <edge>
1800          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/reference/rho_skin</fromParameterUID>
1801          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1802        </edge>
1803        <edge>
1804          <fromParame-
      ↪ terUID>/cpacs/toolspecific/dAEDalus/loadCases/loadCase[1]/sigma_rs</fromParameterUID>
1805          <toExecutableBlockUID>dLC</toExecutableBlockUID>
1806        </edge>
1807        <edge>
1808          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</fromParameterUID>
1809          <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1810        </edge>
1811        <edge>
1812          <fromParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</fromParameterUID>
```

```
1813            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1814          </edge>
1815          <edge>
1816            <fromParame-
      ↪  terUID>/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/c
1817            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1818          </edge>
1819        </edges>
1820      </dataGraph>
1821      <processGraph>
1822        <name>MPG1</name>
1823        <edges>
1824          <edge>
1825            <fromExecutableBlockUID>dLC</fromExecutableBlockUID>
1826            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1827            <processStepNumber>7</processStepNumber>
1828          </edge>
1829          <edge>
1830            <fromExecutableBlockUID>ObjectiveFunctions</fromExecutableBlockUID>
1831            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1832            <processStepNumber>7</processStepNumber>
1833          </edge>
1834          <edge>
1835            <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1836            <toExecutableBlockUID>Coordinator</toExecutableBlockUID>
1837            <processStepNumber>8</processStepNumber>
1838          </edge>
1839          <edge>
1840            <fromExecutableBlockUID>Optimizer</fromExecutableBlockUID>
1841            <toExecutableBlockUID>Converger</toExecutableBlockUID>
1842            <processStepNumber>2</processStepNumber>
1843          </edge>
1844          <edge>
1845            <fromExecutableBlockUID>FWE</fromExecutableBlockUID>
1846            <toExecutableBlockUID>Converger</toExecutableBlockUID>
1847            <processStepNumber>5</processStepNumber>
1848          </edge>
1849          <edge>
1850            <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
1851            <toExecutableBlockUID>dLC</toExecutableBlockUID>
1852            <processStepNumber>6</processStepNumber>
1853          </edge>
1854          <edge>
1855            <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
1856            <toExecutableBlockUID>ObjectiveFunctions</toExecutableBlockUID>
1857            <processStepNumber>6</processStepNumber>
1858          </edge>
1859          <edge>
1860            <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
1861            <toExecutableBlockUID>ConstraintFunctions</toExecutableBlockUID>
1862            <processStepNumber>6</processStepNumber>
1863          </edge>
1864          <edge>
1865            <fromExecutableBlockUID>Converger</fromExecutableBlockUID>
1866            <toExecutableBlockUID>Aeroelastics</toExecutableBlockUID>
1867            <processStepNumber>3</processStepNumber>
1868          </edge>
1869          <edge>
1870            <fromExecutableBlockUID>Coordinator</fromExecutableBlockUID>
1871            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1872            <processStepNumber>1</processStepNumber>
1873          </edge>
1874          <edge>
1875            <fromExecutableBlockUID>ConstraintFunctions</fromExecutableBlockUID>
1876            <toExecutableBlockUID>Optimizer</toExecutableBlockUID>
1877            <processStepNumber>7</processStepNumber>
1878          </edge>
1879          <edge>
1880            <fromExecutableBlockUID>Aeroelastics</fromExecutableBlockUID>
1881            <toExecutableBlockUID>FWE</toExecutableBlockUID>
1882            <processStepNumber>4</processStepNumber>
```

```
        </edge>
      </edges>
      <nodes>
        <node>
          <referenceUID>dLC</referenceUID>
          <processStepNumber>6</processStepNumber>
          <diagonalPosition>5</diagonalPosition>
        </node>
        <node>
          <referenceUID>ObjectiveFunctions</referenceUID>
          <processStepNumber>6</processStepNumber>
          <diagonalPosition>7</diagonalPosition>
        </node>
        <node>
          <referenceUID>Optimizer</referenceUID>
          <processStepNumber>1</processStepNumber>
          <convergerStepNumber>7</convergerStepNumber>
          <diagonalPosition>1</diagonalPosition>
        </node>
        <node>
          <referenceUID>FWE</referenceUID>
          <processStepNumber>4</processStepNumber>
          <diagonalPosition>4</diagonalPosition>
        </node>
        <node>
          <referenceUID>Converger</referenceUID>
          <processStepNumber>2</processStepNumber>
          <convergerStepNumber>5</convergerStepNumber>
          <diagonalPosition>2</diagonalPosition>
        </node>
        <node>
          <referenceUID>Coordinator</referenceUID>
          <processStepNumber>0</processStepNumber>
          <convergerStepNumber>8</convergerStepNumber>
          <diagonalPosition>0</diagonalPosition>
        </node>
        <node>
          <referenceUID>ConstraintFunctions</referenceUID>
          <processStepNumber>6</processStepNumber>
          <diagonalPosition>6</diagonalPosition>
        </node>
        <node>
          <referenceUID>Aeroelastics</referenceUID>
          <processStepNumber>3</processStepNumber>
          <diagonalPosition>3</diagonalPosition>
        </node>
      </nodes>
      <metadata>
        <loopNesting>
          <loopElements>
            <loopElement relatedUID="Optimizer">
              <loopElements>
                <loopElement relatedUID="Converger">
                  <functionElements>
                    <functionElement>FWE</functionElement>
                    <functionElement>Aeroelastics</functionElement>
                  </functionElements>
                </loopElement>
              </loopElements>
              <functionElements>
                <functionElement>dLC</functionElement>
                <functionElement>ObjectiveFunctions</functionElement>
                <functionElement>ConstraintFunctions</functionElement>
              </functionElements>
            </loopElement>
          </loopElements>
        </loopNesting>
      </metadata>
    </processGraph>
  </workflow>
  <architectureElements>
```

```
<parameters>
 <initialGuessCouplingVariables>
  <initialGuessCouplingVariable uID="/cpacs/architectureNodes/initialGuessCouplingVariables/cpacsCopy/toolspecific/fuel_weight_estima
   <relatedParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</relatedParameterUID>
   <label>m_fuel^{c0}</label>
  </initialGuessCouplingVariable>
 </initialGuessCouplingVariables>
 <finalCouplingVariables>
  <finalCouplingVariable uID="/cpacs/architectureNodes/finalCouplingVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_f
   <relatedParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</relatedParameterUID>
   <label>m_fuel^*</label>
  </finalCouplingVariable>
 </finalCouplingVariables>
 <couplingCopyVariables>
  <couplingCopyVariable uID="/cpacs/architectureNodes/couplingCopyVariables/cpacsCopy/toolspecific/fuel_weight_estimator/m_fu
   <relatedParameterUID>/cpacs/toolspecific/fuel_weight_estimator/m_fuel</relatedParameterUID>
   <label>m_fuel^c</label>
  </couplingCopyVariable>
 </couplingCopyVariables>
 <initialGuessDesignVariables>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</relatedParameterUID>
   <label>xsi_fs^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</relatedParameterUID>
   <label>t_ts^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</relatedParameterUID>
   <label>c^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</relatedParameterUID>
   <label>b^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</relatedParameterUID>
   <label>t_fs^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</relatedParameterUID>
   <label>xsi_rs^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
   <relatedParameterUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</relatedParameterUID>
   <label>t_rs^0</label>
  </initialGuessDesignVariable>
  <initialGuessDesignVariable uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProb
```

```
2003          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</relatedParameterUID>
2004          <label>epsilon^0</label>
2005        </initialGuessDesignVariable>
2006        <initialGuessDesignVariable
    ↪ uID="/cpacs/architectureNodes/initialGuessDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/st
2007          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</relatedParameterUID>
2008          <label>t_bs^0</label>
2009        </initialGuessDesignVariable>
2010       </initialGuessDesignVariables>
2011       <finalDesignVariables>
2012        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planform/
2013          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</relatedParameterUID>
2014          <label>epsilon^*</label>
2015        </finalDesignVariable>
2016        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2017          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</relatedParameterUID>
2018          <label>t_rs^*</label>
2019        </finalDesignVariable>
2020        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planform/
2021          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/c</relatedParameterUID>
2022          <label>c^*</label>
2023        </finalDesignVariable>
2024        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/planform/
2025          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/planform/b</relatedParameterUID>
2026          <label>b^*</label>
2027        </finalDesignVariable>
2028        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2029          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</relatedParameterUID>
2030          <label>t_bs^*</label>
2031        </finalDesignVariable>
2032        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2033          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</relatedParameterUID>
2034          <label>xsi_rs^*</label>
2035        </finalDesignVariable>
2036        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2037          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</relatedParameterUID>
2038          <label>t_ts^*</label>
2039        </finalDesignVariable>
2040        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2041          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</relatedParameterUID>
2042          <label>t_fs^*</label>
2043        </finalDesignVariable>
2044        <finalDesignVariable
    ↪ uID="/cpacs/architectureNodes/finalDesignVariables/cpacsCopy/toolspecific/wingOptimizationProblem/structure
2045          <relatedParame-
    ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</relatedParameterUID>
2046          <label>xsi_fs^*</label>
2047        </finalDesignVariable>
2048       </finalDesignVariables>
2049       <finalOutputVariables>
2050        <finalOutputVariable
    ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/constrain
```

```
2051          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</relatedParameterUID>
2052          <label>con_WS^*</label>
2053        </finalOutputVariable>
2054        <finalOutputVariable
      ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/con
2055          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</relatedParameterUID>
2056          <label>con_sigma_ts^*</label>
2057        </finalOutputVariable>
2058        <finalOutputVariable
      ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/con
2059          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</relatedParameterUID>
2060          <label>con_sigma_rs^*</label>
2061        </finalOutputVariable>
2062        <finalOutputVariable
      ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/con
2063          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</relatedParameterUID>
2064          <label>con_sigma_bs^*</label>
2065        </finalOutputVariable>
2066        <finalOutputVariable
      ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/obj
2067          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</relatedParameterUID>
2068          <label>obj_m_fuel^*</label>
2069        </finalOutputVariable>
2070        <finalOutputVariable
      ↪ uID="/cpacs/architectureNodes/finalOutputVariables/cpacsCopy/toolspecific/wingOptimizationProblem/con
2071          <relatedParame-
      ↪ terUID>/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</relatedParameterUID>
2072          <label>con_sigma_fs^*</label>
2073        </finalOutputVariable>
2074      </finalOutputVariables>
2075    </parameters>
2076    <executableBlocks>
2077      <coordinators>
2078        <coordinator uID="Coordinator">
2079          <label>COOR</label>
2080        </coordinator>
2081      </coordinators>
2082      <optimizers>
2083        <optimizer uID="Optimizer">
2084          <label>OPT</label>
2085          <designVariables>
2086            <designVariable>
2087              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_bs</designVariableUID>
2088            </designVariable>
2089            <designVariable>
2090              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/epsilon</designVariableUID>
2091            </designVariable>
2092            <designVariable>
2093              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_fs</designVariableUID>
2094            </designVariable>
2095            <designVariable>
2096              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/c</designVariableUID>
2097            </designVariable>
2098            <designVariable>
2099              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_rs</designVariableUID>
2100            </designVariable>
2101            <designVariable>
2102              <designVari-
      ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_fs</designVariableUID>
2103            </designVariable>
2104            <designVariable>
```

```
2105              <designVari-
    ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/planform/b</designVariableUID>
2106            </designVariable>
2107            <designVariable>
2108              <designVari-
    ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/xsi_rs</designVariableUID>
2109            </designVariable>
2110            <designVariable>
2111              <designVari-
    ↪ ableUID>__desVar__/cpacs/toolspecific/wingOptimizationProblem/structure/t_ts</designVariableUID>
2112            </designVariable>
2113          </designVariables>
2114          <objectiveVariables>
2115            <objectiveVariable>
2116              <objectiveVari-
    ↪ ableUID>__objVar__/cpacs/toolspecific/wingOptimizationProblem/objectives/obj_m_fuel</objectiveVariableUID>
2117            </objectiveVariable>
2118          </objectiveVariables>
2119          <constraintVariables>
2120            <constraintVariable>
2121              <constraintVari-
    ↪ ableUID>__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_ts</constraintVariableU
2122            </constraintVariable>
2123            <constraintVariable>
2124              <constraintVari-
    ↪ ableUID>__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_fs</constraintVariableU
2125            </constraintVariable>
2126            <constraintVariable>
2127              <constraintVari-
    ↪ ableUID>__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_WS</constraintVariableUID>
2128            </constraintVariable>
2129            <constraintVariable>
2130              <constraintVari-
    ↪ ableUID>__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_rs</constraintVariableU
2131            </constraintVariable>
2132            <constraintVariable>
2133              <constraintVari-
    ↪ ableUID>__conVar__/cpacs/toolspecific/wingOptimizationProblem/constraints/con_sigma_bs</constraintVariableU
2134            </constraintVariable>
2135          </constraintVariables>
2136        </optimizer>
2137      </optimizers>
2138      <convergers>
2139        <converger uID="Converger">
2140          <label>CONV</label>
2141        </converger>
2142      </convergers>
2143      <coupledAnalyses>
2144        <coupledAnalysis>
2145          <relatedExecutableBlockUID>FWE</relatedExecutableBlockUID>
2146        </coupledAnalysis>
2147        <coupledAnalysis>
2148          <relatedExecutableBlockUID>Aeroelastics</relatedExecutableBlockUID>
2149        </coupledAnalysis>
2150      </coupledAnalyses>
2151      <postCouplingAnalyses>
2152        <postCouplingAnalysis>
2153          <relatedExecutableBlockUID>ObjectiveFunctions</relatedExecutableBlockUID>
2154        </postCouplingAnalysis>
2155        <postCouplingAnalysis>
2156          <relatedExecutableBlockUID>ConstraintFunctions</relatedExecutableBlockUID>
2157        </postCouplingAnalysis>
2158        <postCouplingAnalysis>
2159          <relatedExecutableBlockUID>dLC</relatedExecutableBlockUID>
2160        </postCouplingAnalysis>
2161      </postCouplingAnalyses>
2162    </executableBlocks>
2163  </architectureElements>
2164 </cmdows>
```

Code frament A.27: CMDOWS file for the simplified wing optimization problem.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<cpacs>
  <toolspecific>
    <wingOptimizationProblem>
      <planform>
        <c mapType="vector">13.7131;7.2595;2.7341</c>
        <tc mapType="vector">0.1542;0.1052;0.095</tc>
        <epsilon mapType="vector">-0.1039;-0.1826</epsilon>
        <b mapType="vector">12.7178;22.7016</b>
        <Lambda mapType="vector">0.5435;0.6077</Lambda>
        <Gamma mapType="vector">0.0508;0.1167</Gamma>
        <incidence>0.1172</incidence>
      </planform>
      <structure>
        <xsi_fs mapType="vector">0.1;0.1925;0.35</xsi_fs>
        <xsi_rs mapType="vector">0.6;0.8023;0.6</xsi_rs>
        <t_fs mapType="vector">0.00450588;0.00458215</t_fs>
        <t_rs mapType="vector">0.00450611;0.00456957</t_rs>
        <t_ts mapType="vector">0.02553329;0.02237119</t_ts>
        <t_bs mapType="vector">0.02553329;0.02237119</t_bs>
        <t_skin>0.0015</t_skin>
      </structure>
      <reference>
        <rho_skin>2180.0</rho_skin>
        <m_fixed>107814.0</m_fixed>
        <m_payload>34000.0</m_payload>
        <m_MLW>213180.0</m_MLW>
        <f_m_sys>0.27</f_m_sys>
        <f_m_wings>0.7</f_m_wings>
        <R>14306700.0</R>
        <SFC>1.5e-05</SFC>
        <m_fuel_res>15000.0</m_fuel_res>
        <C_D_fus>0.006</C_D_fus>
        <C_D_other>0.005</C_D_other>
        <m_fuel_init>108508.0</m_fuel_init>
        <m_wing_init>49591.0</m_wing_init>
        <WS_init>538.725956758</WS_init>
        <sigma_yield>276000000.0</sigma_yield>
        <C_L_buffet>0.525</C_L_buffet>
      </reference>
    </wingOptimizationProblem>
    <dAEDalus>
      <loadCases>
        <loadCase>
          <M>0.85</M>
          <H>11277.6</H>
          <n>1.0</n>
        </loadCase>
        <loadCase>
          <M>0.85</M>
          <H>3048.0</H>
          <n>2.5</n>
        </loadCase>
        <loadCase>
          <M>0.6</M>
          <H>0.0</H>
          <n>-1.0</n>
        </loadCase>
      </loadCases>
    </dAEDalus>
  </toolspecific>
</cpacs>
```

Code frament A.28: Input XML file for the simplified wing optimization problem.

## Knowledge Base

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains a reference to the dAEDalus discipline.
"""
from __future__ import absolute_import, division, print_function

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.SimpleAerostructuralAnalysis
  ↪ import \
    SimpleAerostructuralAnalysis


class Aeroelastics(SimpleAerostructuralAnalysis):
    pass


if __name__ == '__main__':
    n_ws = 2
    n_lc = 3

    aeroelastics = Aeroelastics(n_ws, n_lc)
    aeroelastics.deploy()
```

Code frament A.29: Code of the wing optimization `Aeroelastics` Python module.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains a reference to the problem definition discipline.
"""
from __future__ import absolute_import, division, print_function

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.ProblemDefinition import
  ↪ Constraints


class ConstraintFunctions(Constraints):
    pass


if __name__ == '__main__':
    n_ws = 2

    cons = ConstraintFunctions(n_ws)
    cons.deploy()
```

Code frament A.30: Code of the wing optimization `ConstraintFunctions` Python module.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains a reference to the load collector discipline.
"""
from __future__ import absolute_import, division, print_function

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.dAEDalus import LoadCollector


class dLC(LoadCollector):
    pass


if __name__ == '__main__':
    n_ws = 2
    n_lc = 3

    dlc = dLC(n_ws, n_lc)
    dlc.deploy()
```

Code frament A.31: Code of the wing optimization `dLC` (load collector) Python module.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains a reference to the fuel weight estimator discipline.
"""
from __future__ import absolute_import, division, print_function

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.FuelWeightEstimator import
  ↪ FuelWeightEstimator


class FWE(FuelWeightEstimator):
    pass
```

```
if __name__ == '__main__':
    n_ws = 2
    n_lc = 3

    fwe = FWE(n_ws, n_lc)
    fwe.deploy()
```

Code frament A.32: Code of the wing optimization `FWE` (fuel weight estimator) Python module.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains a reference to the objective function discipline.
"""
from __future__ import absolute_import, division, print_function

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.ProblemDefinition import
  ↪ Objectives


class ObjectiveFunctions(Objectives):
    pass


if __name__ == '__main__':
    objs = ObjectiveFunctions()
    objs.deploy()
```

Code frament A.33: Code of the wing optimization `ObjectiveFunctions` Python module.

### Disciplines

#### A.5.2.3.1 dAEDalus

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definitions of all the dAEDalus disciplines along with static variables
  ↪ they use.
"""
from __future__ import absolute_import, division, print_function
```

```
21
22  import abc
23  import os
24  import time
25  import hashlib
26
27  import matlab
28  from matlab.engine import start_matlab, MatlabExecutionError
29  import numpy as np
30  from openlego.test_suite.test_examples.wing_opt.kb.disciplines.WingObjectModel import
      ↪ WingObjectModel
31  from lxml import etree
32
33  from openlego.api import AbstractDiscipline
34  from openlego.test_suite.test_examples.wing_opt.kb.disciplines.xpaths import *
35  from openlego.utils.general_utils import try_hard
36  from openlego.utils.xml_utils import xml_safe_create_element
37
38  dir_path = os.path.dirname(os.path.realpath(__file__))
39  _mles = {}
40
41  n_seg_x = 10
42  n_seg_y = 20
43
44
45  def start_new_matlab_engine():
46      """Ensure the Matlab engine is renewed once the MATLAB_TIMEOUT is expired.
47
48      This function uses the try_hard() function from the framework.util module to ensure Matlab
      ↪  is started
49      successfully when it needs to be.
50      """
51      mle = try_hard(start_matlab, '-nodesktop -noslpash -nojvm')
52      timestamp = time.time()
53      mle.matlab.engine.shareEngine(nargout=0)
54      engine_name = mle.matlab.engine.engineName(nargout=1)
55      engine_id = int(hashlib.md5(engine_name).hexdigest()[0:7], 16)
56
57      return mle, engine_id, engine_name, timestamp
58
59
60  class LoadCaseSpecific(AbstractDiscipline):
61      """Abstract base class storing the number of wing segments load cases as member properties
      ↪  in the constructor.
62
63      Attributes
64      ----------
65          n_wing_segments : int
66              Number of wing segments.
67
68          n_load_cases : int
69              Number of load cases.
70      """
71
72      def __init__(self, n_wing_segments=2, n_load_cases=1):
73          # type: (int, int) -> None
74          """Create an instance of the 'LoadCollector' discipline.
75
76          Parameters
77          ----------
78              n_wing_segments : int(2)
79                  Number of wing segments.
80
81              n_load_cases : int(1)
82                  Number of load cases.
83          """
84          super(LoadCaseSpecific, self).__init__()
85          self.n_wing_segments = n_wing_segments
86          self.n_load_cases = n_load_cases
87
88      @property
```

```
89      def creator(self):
90          return 'D. de Vries'
91
92      @abc.abstractmethod
93      def generate_input_xml(self):
94          super(LoadCaseSpecific, self).generate_input_xml()
95
96      @abc.abstractmethod
97      def generate_output_xml(self):
98          super(LoadCaseSpecific, self).generate_output_xml()
99
100     @staticmethod
101     @abc.abstractmethod
102     def execute(in_file, out_file):
103         super(LoadCaseSpecific, in_file).execute(in_file, out_file)
104
105     @staticmethod
106     def get_n_loadcases(tree):
107         # type: (etree._ElementTree) -> int
108         """ Obtain the number of load cases from the XML tree representing a CPACS file.
109
110         Parameters
111         ----------
112             tree : :obj:`etree._ElementTree`
113                 `etree._ElementTree` corresponding to a CPACS file.
114
115         Returns
116         -------
117             int
118                 Number of load cases defined in the CPACS file.
119         """
120         return len(tree.xpath('/'.join([x_loadcases, x_loadcase.split('/')[-1][:-4]])))
121
122
123 class SteadyAerostructuralLoop(LoadCaseSpecific):
124
125     MATLAB_TIMEOUT = 1800.
126     _mles = []
127     _timestamp = [0.]
128
129     def __init__(self, n_wing_segments=2, n_load_cases=1):
130         super(SteadyAerostructuralLoop, self).__init__(n_wing_segments, n_load_cases)
131
132     @property
133     def description(self):
134         return 'dAEDalus Steady Aerostructural Loop'
135
136     def generate_input_xml(self):
137         wd = WingObjectModel(self.n_wing_segments)
138         s = wd.generate_output_xml()
139
140         parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
141         doc = etree.fromstring(s, parser)
142
143         for i in range(1, self.n_load_cases + 1):
144             xml_safe_create_element(doc, x_M % i, 0.)
145             xml_safe_create_element(doc, x_H % i, 0.)
146             xml_safe_create_element(doc, x_n % i, 0.)
147
148         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
149
150     def generate_output_xml(self):
151         root = etree.Element('cpacs')
152         doc = etree.ElementTree(root)
153
154         xml_safe_create_element(doc, x_m_wing, 0.)
155
156         for i in range(1, self.n_load_cases + 1):
157             xml_safe_create_element(doc, x_CL % i, 0.)
158             xml_safe_create_element(doc, x_CDf % i, 0.)
159             xml_safe_create_element(doc, x_CDi % i, 0.)
```

```
160
161             for x_sigma in x_sigmas_in:
162                 xml_safe_create_element(doc, x_sigma % i, np.zeros(self.n_wing_segments))
163
164         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
165
166     @staticmethod
167     def execute(in_file, out_file):
168         doc_in = etree.parse(in_file)
169
170         root = etree.Element('cpacs')
171         doc_out = etree.ElementTree(root)
172
173         n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
174         if not len(SteadyAerostructuralLoop._mles) or \
175                 (time.time() - SteadyAerostructuralLoop._timestamp[0] >=
    ↪ SteadyAerostructuralLoop.MATLAB_TIMEOUT):
176             SteadyAerostructuralLoop._mles = []
177             SteadyAerostructuralLoop._timestamp[0] = time.time()
178             for _ in range(n_lc):
179                 mle, _, _, _ = start_new_matlab_engine()
180                 SteadyAerostructuralLoop._mles.append(mle)
181
182         futures = n_lc * [None]
183         for i in range(1, n_lc + 1):
184             M = float(doc_in.xpath(x_M % i)[0].text)
185             H = float(doc_in.xpath(x_H % i)[0].text)
186             n = float(doc_in.xpath(x_n % i)[0].text)
187
188             futures[i - 1] = SteadyAerostructuralLoop._mles[i -
    ↪ 1].dAEDalusSteadyAerostructuralLoop(
189                 in_file, matlab.double([n_seg_x]), matlab.double([n_seg_y]), M, H, n,
    ↪ nargout=8, async=True)
190
191         for i, future in enumerate(futures):
192             if future is not None:
193                 try:
194                     m_wing, C_L, C_D_f, C_D_i, sigma_fs, sigma_rs, sigma_ts, sigma_bs =
    ↪ future.result()
195                     sigmas = [np.array(sigma_fs), np.array(sigma_rs), np.array(sigma_ts),
    ↪ np.array(sigma_bs)]
196
197                     xml_safe_create_element(doc_out, x_m_wing, m_wing)
198                     xml_safe_create_element(doc_out, x_CL % (i + 1), C_L)
199                     xml_safe_create_element(doc_out, x_CDf % (i + 1), C_D_f)
200                     xml_safe_create_element(doc_out, x_CDi % (i + 1), C_D_i)
201
202                     for j in range(4):
203                         xml_safe_create_element(doc_out, x_sigmas_in[j] % (i + 1), sigmas[j])
204                 except MatlabExecutionError:
205                     break
206
207         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
208
209
210 class SteadyModelInitializer(LoadCaseSpecific):
211     """Initialization of the geometric and structural dAEDalus models using Matlab.
212
213     This discipline takes a CPACS file with a fully defined wing as input and initializes the
    ↪  geometric and structural
214     models of dAEDalus accordingly. The weight of the wing is calculated and stored in the
    ↪  output file, along with some
215     pseudo-variables to aid linking this dicipline to the other dAEDalus disciplines.
216
217     Behind the scenes, this discipline initializes the geometric and structural models and
    ↪  stores these in the workspace
218     of the Matlab shared engine for each load case. Subsequent dAEDalus discipline calls can
    ↪  use these initialized
219     models if they have the names of the Matlab shared engines.
220     """
221
```

```
222     MATLAB_TIMEOUT = 1800.
223
224     def __init__(self, n_wing_segments=2, n_load_cases=1):
225         super(SteadyModelInitializer, self).__init__(n_wing_segments, n_load_cases)
226
227     @property
228     def description(self):
229         return 'dAEDalus Steady Model Initializer'
230
231     def generate_input_xml(self):
232         # type: () -> str
233         """Input is a CPACS file with at least a fully defined wing.
234
235         It is possible to specify a timeout for Matlab, the name of a Matlab sharedEngine, and
    ↪ the timestamp at which
236         this engine was shared. However, the Matlab engine should normally be left under the
    ↪ control of this
237         discipline.
238         """
239         wd = WingObjectModel(self.n_wing_segments)
240         s = wd.generate_output_xml()
241
242         parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
243         doc = etree.fromstring(s, parser)
244
245         for i in range(1, self.n_load_cases + 1):
246             xml_safe_create_element(doc, x_ml_timeout % i, self.MATLAB_TIMEOUT)
247
248         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
249
250     def generate_output_xml(self):
251         # type: () -> str
252         """Output is a CPACS file containing a predefined number of load cases.
253
254         Each load case will be assigned pseudo-variables allowing for subsequent disciplines
    ↪ to connect to this
255         discipline (geometric and structural model), as well as the name of the shared Matlab
    ↪ engine and its timestamp
256         of creation.
257         """
258         root = etree.Element('cpacs')
259         doc = etree.ElementTree(root)
260
261         xml_safe_create_element(doc, x_m_wing, 0.)
262
263         for i in range(1, self.n_load_cases + 1):
264             xml_safe_create_element(doc, x_ml_id % i, 0)
265             xml_safe_create_element(doc, x_ml_timestamp % i, 0.)
266
267             for j in range(3):
268                 xml_safe_create_element(doc, x_grid_initial[j] % i, np.zeros(2 * (n_seg_x + 1)
    ↪ * (n_seg_y + 1) * self.n_wing_segments))
269
270         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
271
272     @staticmethod
273     def execute(in_file, out_file):
274         """Call the Matlab function dAEDalusSteadyModelInitializer() and store the resulting
    ↪ mass of the wingbox in the
275         output XML file for each load case.
276
277         The name of the Matlab engine is stored in CPACS. In this way it can be shared with
    ↪ all subsequent disciplines
278         that need to use the same instance of Matlab in order to share the workspace.
279         """
280         doc_in = etree.parse(in_file)
281
282         root = etree.Element('cpacs')
283         doc_out = etree.ElementTree(root)
284
285         n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
```

```
286         engine_ids = []
287         futures = n_lc * [None]
288         for i in range(1, n_lc + 1):
289             timeout = SteadyModelInitializer.MATLAB_TIMEOUT
290             elem_timeout = doc_in.xpath(x_ml_timeout % i)
291             if len(elem_timeout):
292                 timeout = float(elem_timeout[0].text)
293 
294             timestamp = 0.
295             elem_timestamp = doc_in.xpath(x_ml_timestamp % i)
296             if len(elem_timestamp):
297                 timestamp = float(elem_timestamp[0].text)
298 
299             # Obtain the current matlab engine if it is still valid and exists
300             engine_id = 0
301             mle = None
302             elem_ml_id = doc_in.xpath(x_ml_id % i)
303             if len(elem_ml_id):
304                 engine_id = int(float(elem_ml_id[0].text))
305                 if engine_id in _mles:
306                     if time.time() - timestamp < timeout:
307                         mle = _mles[engine_id]
308                         mle.cd(dir_path)
309                     else:
310                         _mles.pop(engine_id)
311 
312             # If a matlab engine was not connected to, start a new one and reset the timestamp
313             if mle is None:
314                 mle, engine_id, engine_name, timestamp = start_new_matlab_engine()
315                 mle.cd(dir_path)
316                 _mles.update({engine_id: mle})
317 
318             engine_ids.append(engine_id)
319 
320             xml_safe_create_element(doc_out, x_ml_id % i, engine_id)
321             xml_safe_create_element(doc_out, x_ml_timestamp % i, timestamp)
322 
323             futures[i - 1] = mle.dAEDalusSteadyModelInitializer(
324                 in_file, matlab.double([n_seg_x]), matlab.double([n_seg_y]), nargout=2,
    ↪  async=True)
325 
326         # Remove any instances of the Matlab engine which aren't used anymore to free up memory
327         if len(_mles) > n_lc:
328             _ids = _mles.keys()
329             for _id in _ids:
330                 if _id not in engine_ids:
331                     _mles.pop(_id)
332 
333         for i, future in enumerate(futures):
334             if future is not None:
335                 try:
336                     m_wing, initial_grid = future.result()
337                     initial_grid = np.array(initial_grid)
338                     xml_safe_create_element(doc_out, x_m_wing, m_wing)
339 
340                     for j in range(3):
341                         xml_safe_create_element(doc_out, x_grid_initial[j] % (i + 1),
    ↪  initial_grid[j, :])
342                 except MatlabExecutionError:
343                     break
344 
345         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
346 
347 
348 class SteadyAerodynamicModelInitializer(LoadCaseSpecific):
349     """Initialization of the steady aerodynamic dAEDalus model.
350 
351     This discipline takes a CPACS file with a number of load cases containing a Mach number,
    ↪  altitude, and load factor.
352     Furthermore, the pseudo-variable pointing to the geometric model of each load case, as
    ↪  well as the name of the load
```

```
353       case's shared Matlab engine are required.
354
355       Behind the scenes, the initialized geometric and structural models that were stored in the
          ↪ Matlab shared engine's
356       workspace are used, and the aerodynamic model is stored there too once it is initialized.
357       """
358
359       def __init__(self, n_wing_segments=2, n_load_cases=1):
360           super(SteadyAerodynamicModelInitializer, self).__init__(n_wing_segments, n_load_cases)
361
362       @property
363       def description(self):
364           return 'dAEDalus Steady Aerodynamic Model Initializer'
365
366       def generate_input_xml(self):
367           # type: () -> str
368           """Input is a CPACS file containing the Mach number, altitude, and load factor for
          ↪ each load case.
369
370           The link to the geometric model and name of the Matlab shared engine for each load
          ↪ case is also required.
371           """
372           root = etree.Element('cpacs')
373           doc = etree.ElementTree(root)
374
375           for i in range(1, self.n_load_cases + 1):
376               xml_safe_create_element(doc, x_M % i, 0.)
377               xml_safe_create_element(doc, x_H % i, 0.)
378               xml_safe_create_element(doc, x_n % i, 0.)
379
380               xml_safe_create_element(doc, x_ml_id % i, 0)
381
382           return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
383
384       def generate_output_xml(self):
385           # type: () -> str
386           """Output is a CPACS file containing the lift- and friction drag coefficient for each
          ↪ load case, as well as a
387           link to the aerodynamic model.
388           """
389           root = etree.Element('cpacs')
390           doc = etree.ElementTree(root)
391
392           for i in range(1, self.n_load_cases + 1):
393               xml_safe_create_element(doc, x_CL % i, 0.)
394               xml_safe_create_element(doc, x_CDf % i, 0.)
395
396           return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
397
398       @staticmethod
399       def execute(in_file, out_file):
400           """Call the Matlab function dAEDalusSteadyAerodynamicModelInitializer() and store the
          ↪ resulting values of C_L
401           and C_D_f in the the CPACS output file.
402           """
403           doc_in = etree.parse(in_file)
404
405           root = etree.Element('cpacs')
406           doc_out = etree.ElementTree(root)
407
408           n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
409           futures = n_lc * [None]
410           for i in range(1, n_lc + 1):
411               engine_id = int(float(doc_in.xpath(x_ml_id % i)[0].text))
412               if engine_id not in _mles:
413                   break
414               else:
415                   M = float(doc_in.xpath(x_M % i)[0].text)
416                   H = float(doc_in.xpath(x_H % i)[0].text)
417                   n = float(doc_in.xpath(x_n % i)[0].text)
418
```

```
419                 futures[i - 1] = _mles[engine_id].dAEDalusSteadyAerodynamicModelInitializer(
420                     float(M), float(H), float(n), nargout=2, async=True)
421
422         for i, future in enumerate(futures):
423             if future is None:
424                 break
425             else:
426                 try:
427                     C_L, C_D_f = future.result()
428                     xml_safe_create_element(doc_out, x_CL % (i + 1), C_L)
429                     xml_safe_create_element(doc_out, x_CDf % (i + 1), C_D_f)
430                 except MatlabExecutionError:
431                     break
432
433         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
434
435
436 class SteadyAerodynamicAnalysis(LoadCaseSpecific):
437     """Steady aerodynamic analysis of dAEDalus using Matlab."""
438
439     def __init__(self, n_wing_segments=2, n_load_cases=1):
440         super(SteadyAerodynamicAnalysis, self).__init__(n_wing_segments, n_load_cases)
441         self.previous_grids = n_load_cases * [None]
442
443     @property
444     def description(self):
445         return 'dAEDalus Steady Aerodynamic Analysis'
446
447     def generate_input_xml(self):
448         # type: () -> str
449         """Input is a CPACS file with the lift coefficient and deflected grid of the wing for
    ↪ each load case, as well
450         as links to the geometric and aerodynamic models.
451         """
452         root = etree.Element('cpacs')
453         doc = etree.ElementTree(root)
454
455         for i in range(1, self.n_load_cases + 1):
456             xml_safe_create_element(doc, x_CL % i, 0.)
457             xml_safe_create_element(doc, x_ml_id % i, 0)
458
459             for j in range(3):
460                 xml_safe_create_element(doc, x_grid_initial[j] % i, np.zeros(2 * (n_seg_x + 1)
    ↪ * (n_seg_y + 1) * self.n_wing_segments))
461                 xml_safe_create_element(doc, x_grid[j] % i, np.zeros(2 * (n_seg_x + 1) *
    ↪ (n_seg_y + 1) * self.n_wing_segments))
462
463         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
464
465     def generate_output_xml(self):
466         # type: () -> str
467         """Output is a CPACS file with the induced drag coefficient and link to the
    ↪ aerodynamic forces."""
468         root = etree.Element('cpacs')
469         doc = etree.ElementTree(root)
470
471         for i in range(1, self.n_load_cases + 1):
472             xml_safe_create_element(doc, x_CDi % i, 0.)
473
474             for j in range(3):
475                 xml_safe_create_element(doc, x_grid_guess[j] % i, np.zeros(2 * (n_seg_x + 1) *
    ↪ (n_seg_y + 1) * self.n_wing_segments))
476
477         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
478
479     @staticmethod
480     def execute(in_file, out_file):
481         """Call the Matlab function dAEDalusSteadyAerodynamicAnalysis() and store the
    ↪ resulting value of C_D_i and a
482         link to the aerodynamic forces for each load case.
483         """
```

```
484         doc_in = etree.parse(in_file)
485 
486         root = etree.Element('cpacs')
487         doc_out = etree.ElementTree(root)
488 
489         n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
490         futures = n_lc * [None]
491         for i in range(1, n_lc + 1):
492             engine_id = int(float(doc_in.xpath(x_ml_id % i)[0].text))
493             if engine_id not in _mles:
494                 break
495             else:
496                 grid = 3 * [None]
497                 s = 0.
498                 for j in range(3):
499                     g = np.array(doc_in.xpath(x_grid[j] % i)[0].text.split(';'),
    ↪ dtype=float).tolist()
500                     s += np.sum(np.square(g))
501                     grid[j] = g
502                 if s == 0:
503                     for j in range(3):
504                         grid[j] = np.array(doc_in.xpath(x_grid_initial[j] %
    ↪ i)[0].text.split(';'), dtype=float).tolist()
505 
506                 C_L = float(doc_in.xpath(x_CL % i)[0].text)
507 
508                 futures[i - 1] = _mles[engine_id].dAEDalusSteadyAerodynamicAnalysis(
509                     matlab.double(grid), float(C_L), nargout=1, async=True)
510 
511                 for j in range(3):
512                     xml_safe_create_element(doc_out, x_grid_guess[j] % i, np.array(grid[j]))
513 
514         for i, future in enumerate(futures):
515             if future is None:
516                 break
517             else:
518                 try:
519                     C_D_i = future.result()
520                     xml_safe_create_element(doc_out, x_CDi % (i + 1), C_D_i)
521                 except MatlabExecutionError:
522                     break
523 
524         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
525 
526 
527 class SteadyStructuralAnalysis(LoadCaseSpecific):
528     """Steady structural analysis of dAEDalus using Matlab."""
529 
530     def __init__(self, n_wing_segments=2, n_load_cases=1):
531         super(SteadyStructuralAnalysis, self).__init__(n_wing_segments, n_load_cases)
532 
533     @property
534     def description(self):
535         return 'dAEDalus Steady Structural Analysis'
536 
537     def generate_input_xml(self):
538         # type: () -> str
539         """Input is a CPACS file containing the name of the Matlab shared engine, the links to
    ↪  all three models
540         (geometric, structural, and aerodynamic), and the link the the aerodynamic forces for
    ↪  each load case.
541         """
542         root = etree.Element('cpacs')
543         doc = etree.ElementTree(root)
544 
545         for i in range(1, self.n_load_cases + 1):
546             xml_safe_create_element(doc, x_ml_id % i, 0)
547             for j in range(3):
548                 xml_safe_create_element(doc, x_grid_guess[j] % i, np.zeros(2 * (n_seg_x + 1) *
    ↪  (n_seg_y + 1) * self.n_wing_segments))
549 
```

```
550         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
551
552     def generate_output_xml(self):
553         # type: () -> str
554         """Output is a CPACS file with the stresses in the front/rear spars and in the
    ↪  top/bottom skins for each load
555         case, as well as the deflected grid for each load case.
556         """
557         root = etree.Element('cpacs')
558         doc = etree.ElementTree(root)
559
560         for i in range(1, self.n_load_cases + 1):
561             for x_sigma in x_sigmas_in:
562                 xml_safe_create_element(doc, x_sigma % i, np.zeros(self.n_wing_segments))
563
564             for j in range(3):
565                 xml_safe_create_element(doc, x_grid[j] % i, np.zeros(2 * (n_seg_x + 1) *
    ↪  (n_seg_y + 1) * self.n_wing_segments))
566
567         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
568
569     @staticmethod
570     def execute(in_file, out_file):
571         """Call the Matlab function dAEDalusSteadyStructuralAnalysis() and store the resulting
    ↪  values of the stresses
572         (sigma_*) and the deflected grid in the corresponding unknowns.
573         """
574         doc_in = etree.parse(in_file)
575
576         root = etree.Element('cpacs')
577         doc_out = etree.ElementTree(root)
578
579         n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
580         futures = n_lc * [None]
581         for i in range(1, n_lc + 1):
582             engine_id = int(float(doc_in.xpath(x_ml_id % i)[0].text))
583             if engine_id not in _mles:
584                 break
585             else:
586                 futures[i - 1] = _mles[engine_id].dAEDalusSteadyStructuralAnalysis(nargout=5,
    ↪  async=True)
587
588         for i, future in enumerate(futures):
589             if future is None:
590                 break
591             else:
592                 try:
593                     sigma_fs, sigma_rs, sigma_ts, sigma_bs, deflected_grid = future.result()
594                     sigmas = [np.array(sigma_fs), np.array(sigma_rs), np.array(sigma_ts),
    ↪  np.array(sigma_bs)]
595                     deflected_grid = np.array(deflected_grid)
596
597                     for j in range(4):
598                         xml_safe_create_element(doc_out, x_sigmas_in[j] % (i + 1), sigmas[j])
599
600                     for j in range(3):
601                       xml_safe_create_element(doc_out, x_grid[j] % (i + 1), deflected_grid[j])
602                 except MatlabExecutionError:
603                     break
604
605         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
606
607
608 class LoadCollector(LoadCaseSpecific):
609     """Defines the Load Collector discipline.
610
611     This discipline takes the maximum value of the stresses in the front/rear spars and
    ↪  top/bottom skins for any number
612     of load cases and returns a single set of critical stresses.
613     """
614
```

```
615     def __init__(self, n_wing_segments=2, n_load_cases=1):
616         super(LoadCollector, self).__init__(n_wing_segments, n_load_cases)
617 
618     @property
619     def description(self):
620         return 'Load Collector'
621 
622     def generate_input_xml(self):
623         # type: () -> str
624         """Input is a CPACS file with the stresses in the front/rear spars and in the
    ↪  top/bottom skins for each load
625         case.
626         """
627         root = etree.Element('cpacs')
628         doc = etree.ElementTree(root)
629 
630         for i in range(1, self.n_load_cases + 1):
631             for x_sigma in x_sigmas_in:
632                 xml_safe_create_element(doc, x_sigma % i, np.zeros(self.n_wing_segments))
633 
634         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
635 
636     def generate_output_xml(self):
637         # type: () -> str
638         """Output is a CPACS file containing the maximum stresses in the front/rear spars and
    ↪  in the top/bottom skins
639         across all load cases.
640         """
641         root = etree.Element('cpacs')
642         doc = etree.ElementTree(root)
643 
644         for x_sigma in x_sigmas_out:
645             xml_safe_create_element(doc, x_sigma, np.zeros(self.n_wing_segments))
646 
647         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
648 
649     @staticmethod
650     def execute(in_file, out_file='LC-output-loc.xml'):
651         """Computes the maximum stresses across all load cases."""
652         doc_in = etree.parse(in_file)
653 
654         sigmas = 4 * [np.ndarray((0,))]
655         for i in range(1, LoadCaseSpecific.get_n_loadcases(doc_in) + 1):
656             for j in range(4):
657                 data = np.array(doc_in.xpath(x_sigmas_in[j] % i)[0].text.split(';'),
    ↪  dtype=float)
658                 if i == 1:
659                     sigmas[j] = data
660                 else:
661                     sigmas[j] = np.maximum(sigmas[j], data)
662 
663         # Write results to output XML file
664         root = etree.Element('cpacs')
665         doc_out = etree.ElementTree(root)
666         for i in range(4):
667             xml_safe_create_element(doc_out, x_sigmas_out[i], sigmas[i])
668 
669         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
670 
671 
672 class SteadyLiftDistribution(LoadCaseSpecific):
673     """Calculation of the lift distribution using Matlab."""
674 
675     def __init__(self, n_wing_segments=2, n_load_cases=1):
676         super(SteadyLiftDistribution, self).__init__(n_wing_segments, n_load_cases)
677 
678     @property
679     def description(self):
680         return 'Steady Lift Distribution'
681 
682     def generate_input_xml(self):
```

```
683         # type: () -> str
684         """Input is a CPACS file with the name of the Matlab shared engine and links to the
    ↪ geometric model, aerodynamic
685         model and aerodynamic forces for each load case.
686         """
687         root = etree.Element('cpacs')
688         doc = etree.ElementTree(root)
689 
690         for i in range(1, self.n_load_cases + 1):
691             xml_safe_create_element(doc, x_ml_id % i, 0)
692 
693         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
694 
695     def generate_output_xml(self):
696         # type: () -> str
697         """Output is a CPACS file with the normalized y-coordinates and corresponding
    ↪ normalized section lift forces for
698         each load case.
699         """
700         root = etree.Element('cpacs')
701         doc = etree.ElementTree(root)
702 
703         for i in range(1, self.n_load_cases + 1):
704             xml_safe_create_element(doc, x_y_norm % i, np.zeros(n_seg_y *
    ↪ self.n_wing_segments))
705             xml_safe_create_element(doc, x_l_norm % i, np.zeros(n_seg_y *
    ↪ self.n_wing_segments))
706 
707         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
708 
709     @staticmethod
710     def execute(in_file, out_file):
711         """Call the Matlab function dAEDalusSteadyLiftDistribution() and store the resulting
    ↪ values of y_norm
712         (normalized y-coordinates) and l_norm (normalized lift) in the output XML.
713         """
714         doc_in = etree.parse(in_file)
715 
716         root = etree.Element('cpacs')
717         doc_out = etree.ElementTree(root)
718 
719         n_lc = LoadCaseSpecific.get_n_loadcases(doc_in)
720         futures = n_lc * [None]
721         for i in range(1, n_lc + 1):
722             engine_id = int(float(doc_in.xpath(x_ml_id % i)[0].text))
723             if engine_id not in _mles:
724                 break
725             else:
726                 futures[i - 1] = _mles[engine_id].dAEDalusSteadyLiftDistribution(nargout=2,
    ↪ async=True)
727 
728         for i, future in enumerate(futures):
729             if future is None:
730                 break
731             else:
732                 try:
733                     y_norm, l_norm = future.result()
734                     y_norm = np.array(y_norm)
735                     l_norm = np.array(l_norm)
736 
737                     xml_safe_create_element(doc_out, x_y_norm % (i + 1), y_norm)
738                     xml_safe_create_element(doc_out, x_l_norm % (i + 1), l_norm)
739                 except MatlabExecutionError:
740                     break
741 
742         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
```

Code frament A.34: Code of the Python module containing the dAEDalus disciplines.

```
1 function [m_wing, initial_grid] = dAEDalusSteadyModelInitializer(input_cpacs, n_seg_x,
    ↪ n_seg_y)
```

```
    try
        evalin('base', 'clear geometrical_model');
        evalin('base', 'clear structural_model');
        evalin('base', 'clear aerodynamic_model');
    catch e

    end

    % Create Aircraft data structure
    geometric_model = class_aircraft.create_from_cpacs(input_cpacs);
    geometric_model.grid_settings.aerodynamic_fuselage = 0;
    geometric_model.wings = geometric_model.wings(1);
    geometric_model.wings_structural_properties =
    ↳ geometric_model.wings_structural_properties(1);

    % Obtain smallest chord length
    wing = geometric_model.wings(1);
    n_wing_segments = length(wing.wing_segments);
    c_min = 100;
    for i = 1:n_wing_segments
        c_min = min(c_min, wing.wing_segments(i).c_r);
        c_min = min(c_min, wing.wing_segments(i).c_t);

        geometric_model.wings(1).wing_segments(i).n_chord = n_seg_x;
        geometric_model.wings(1).wing_segments(i).n_span = n_seg_y;
    end

    geometric_model.grid_settings.dy_max_struct_grid = geometric_model.reference.b_ref/40;

    % Compute grid and initialize structural model
    geometric_model = geometric_model.compute_grid();
    [geometric_model, structural_model] = create_structural_model(geometric_model);
    structural_model.beam(1) = struc-
    ↳ tural_model.beam(1).f_add_boundary_condition(class_boundary_condition(ceil(length(structural_model.beam.nod
    ↳ 1 1 1 1 1],[0 0 0 0 0 0]));
    geometric_model = geometric_model.compute_force_interpolation_matrix(structural_model);

    % Assemble structural model
    structure_solver_settings = class_wingstructure_solver_settings;
    structure_solver_settings.gravity = 0;
    structural_model = structural_model.f_set_solver_settings(structure_solver_settings);
    structural_model = structural_model.f_assemble(1,0);

    % Make sure the deflected grid of the geometrical model is equal to the
    % initial grid
    geometric_model.grid_deflected = geometric_model.grid;

    % Compute total wingbox mass
    structural_model = structural_model.f_calc_mass(geometric_model.weights);
    m_wing = structural_model.beam(1).m_total - structural_model.beam(1).m_fuel_total;

    initial_grid = geometric_model.grid;

    % Store these objects in the workspace for future use
    assignin('base', 'geometric_model', geometric_model);
    assignin('base', 'structural_model', structural_model);

end
```

Code frament A.35: Code of the Matlab script of the geometric and structural model initializer discipline (`dSMI`).

```
function [C_L, C_D_f] = dAEDalusSteadyAerodynamicModelInitializer( M, H, n )

    geometric_model = evalin('base', 'geometric_model');

    % Create flight state
    ref_state = critical_ref_state(geometric_model, M, H);
    critical_state = critical_g_maneuver_state(ref_state, n);

    % Find the required C_l
```

```
    C_L = critical_state.get_Cl(geometric_model.reference.S_ref);

    % Compute C_D_f
    ac = geometric_model.compute_CD_f(critical_state.aerodynamic_state,
    ↪  geometric_model.reference.S_ref);
    C_D_f = ac.CD_f;

    % Initialize aerodynamic model
    aerodynamic_model = class_VLM_solver(...
        geometric_model.grid, ... % geometric_model.grid_deflected
        geometric_model.te_idx, ...
        geometric_model.panels, ...
        critical_state.aerodynamic_state, ...
        geometric_model.reference);

    % Compute the influence coefficients
    aerodynamic_model.f_calc_coeffs();

    assignin('base', 'aerodynamic_model', aerodynamic_model);

end
```

Code frament A.36: Code of the Matlab script of the aerodynamic model initializer discipline (`dSAMI`).

```
function [C_D_i] = dAEDalusSteadyAerodynamicAnalysis(deflected_grid, C_L)
    % Obtain geometric_model
    geometric_model = evalin('base', 'geometric_model');
    aerodynamic_model = evalin('base', 'aerodynamic_model');

    aerodynamic_model.set_grid(deflected_grid, geometric_model.panels);
    aerodynamic_model = aerodynamic_model.f_solve_for_Cl_fast(C_L);

    % Obtain C_D_i
    C_D_i = aerodynamic_model.Cdi;

    assignin('base', 'aerodynamic_model', aerodynamic_model);
end
```

Code frament A.37: Code of the Matlab script of the aerodynamic analysis discipline (`dSAA`).

```
function [sigma_sp_fr, sigma_sp_re, sigma_sk_up, sigma_sk_lo, deflected_grid] =
  ↪  dAEDalusSteadyStructuralAnalysis()
    % Obtain geometric_model
    geometric_model = evalin('base', 'geometric_model');
    structural_model = evalin('base', 'structural_model');
    aerodynamic_model = evalin('base', 'aerodynamic_model');

    % Transforms aeroloads to structure
    geometric_model = geometric_model.compute_beam_forces(aerodynamic_model.F_body,
    ↪  structural_model);
    for i = 1:length(structural_model.beam)
        if  isa(structural_model.beam(i),'class_wing')
            structural_model.beam(i) =
    ↪  structural_model.beam(i).f_set_aeroloads(geometric_model.wings(i));

        end
    end

    % Solve structural model to get the deflections
    structural_model = structural_model.f_solve();

    % Compute the deflected grid of the geometrical model
    geometric_model =
    ↪  geometric_model.compute_deflected_grid(structural_model.f_get_deflections);
    deflected_grid = geometric_model.grid_deflected;
    %deflected_grid = deflected_grid(:, 1:(size(deflected_grid, 2)/2));

    % Calculate the stresses
    structural_model = structural_model.f_calc_stresses();
```

```

    % Gather all the stresses in arrays for front/rear spars and top/bottom skins
    beam = structural_model.beam;
    n_segments = length(geometric_model.wings.wing_segments);

    sigma_sp_fr = zeros(1, n_segments);
    sigma_sp_re = zeros(1, n_segments);
    sigma_sk_up = zeros(1, n_segments);
    sigma_sk_lo = zeros(1, n_segments);

    for i = 1:length(beam.beamelement)
        cs = beam.beamelement(i).crosssection;
        i_segment = cs.segment_index;

        sigma_sp_fr(i_segment) = max([sigma_sp_fr(i_segment), cs.sigma_sp_fr]);
        sigma_sp_re(i_segment) = max([sigma_sp_re(i_segment), cs.sigma_sp_re]);
        sigma_sk_up(i_segment) = max([sigma_sk_up(i_segment), cs.sigma_sk_up]);
        sigma_sk_lo(i_segment) = max([sigma_sk_lo(i_segment), cs.sigma_sk_lo]);
    end

    assignin('base', 'geometric_model', geometric_model);
    assignin('base', 'structural_model', structural_model);

end
```

Code frament A.38: Code of the Matlab script of the structural analysis discipline (`dSSA`).

```
function [ y_norm, l_norm ] = dAEDalusSteadyLiftDistribution( )
%DAEDALUSSTEADYLIFTDISTRIBUTION Summary of this function goes here
%   Detailed explanation goes here

    geometric_model = evalin('base', 'geometric_model');
    aerodynamic_model = evalin('base', 'aerodynamic_model');

    % Compute the lift distribution
    % Finding chord lengths
    x_p_r_le = geometric_model.grid_deflected(:,aerodynamic_model.panels(1,:));        %
  ↪  Panel root LE locations
    x_p_r_te = geometric_model.grid_deflected(:,aerodynamic_model.panels(4,:));        %
  ↪  Panel root TE locations
    x_p_t_le = geometric_model.grid_deflected(:,aerodynamic_model.panels(2,:));        %
  ↪  Panel tip LE locations
    x_p_t_te = geometric_model.grid_deflected(:,aerodynamic_model.panels(3,:));        %
  ↪  Panel tip TE locations

    idx_te = find(geometric_model.is_te);                  % indices of TE panels
    idx_le = [1,idx_te(1:end-1)+1];                % indices of LE panels
    x_s_r_le = x_p_r_le(:,idx_le);                 % strip root LE positions
    x_s_r_te = x_p_r_te(:,idx_te);                 % strip root TE positions
    x_s_t_le = x_p_t_le(:,idx_le);                 % strip tip LE positions
    x_s_t_te = x_p_t_te(:,idx_te);                 % strip tip TE positions

    x_s_le = x_s_r_le + 0.5*(x_s_t_le - x_s_r_le);  % strip middle LE positions

    n_panels = size(aerodynamic_model.panels,2);
    I1 = 1:n_panels <= idx_te';
    I2 = 1:n_panels >= idx_le';
    I = I1.*I2;

    L_s = (I*aerodynamic_model.F_aero(3,:)')';           % strip lift forces
    b_s = (x_s_t_le(2,:) - x_s_r_le(2,:) + x_s_t_te(2,:) - x_s_r_te(2,:))/2; % strip spans

    l = L_s./b_s;                  % section lift
    l_mean = sum(L_s)/sum(b_s); % mean section lift
    l_norm = l/l_mean; % normalized section lift

    [y,iy] = sort(x_s_le(2,:));                    % sorted y positions of LEs
    y_norm = y./(sum(b_s)/2);
    l_norm = l_norm(iy);

```

```
    y_norm = y_norm(length(y_norm)/2+1:end);
    l_norm = l_norm(length(l_norm)/2+1:end);

end
```

Code frament A.39: Code of the Matlab script of the lift distribution calculation discipline (dSLD).

```
function [m_wing_struct, C_L, C_D_f, C_D_i, sigma_sp_fr, sigma_sp_re, sigma_sk_up,
  ↪  sigma_sk_lo] = dAEDalusSteadyAerostructuralLoop(cpacs, n_x, n_y, M, H, n)
    [m_wing_struct, initial_grid] = dAEDalusSteadyModelInitializer(cpacs, n_x, n_y);

    [C_L, C_D_f] = dAEDalusSteadyAerodynamicModelInitializer(M, H, n);

    deflected_grid_guess = initial_grid;
    tol = 1e-6;
    iter = 0;
    maxiter = 10;
    while 1
        C_D_i = dAEDalusSteadyAerodynamicAnalysis(deflected_grid_guess, C_L);
        [sigma_sp_fr, sigma_sp_re, sigma_sk_up, sigma_sk_lo, deflected_grid] =
  ↪  dAEDalusSteadyStructuralAnalysis();

        r_norm = sqrt(sum((deflected_grid - deflected_grid_guess).^2, 1));
        err = rms(r_norm);
        %fprintf(1, 'iter: %d\t err: %.7f\n', iter, err);
        iter = iter + 1;
        if err < tol
            break
        end
        deflected_grid_guess = deflected_grid;

        if iter > maxiter
            break
        end
    end
%     [y_norm, l_norm] = dAEDalusSteadyLiftDistribution( );
end
```

Code frament A.40: Code of the Matlab script of the full aerostructural loop of dAEDalus.

#### A.5.2.3.2 Fuel Weight Estimator

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definition of the FuelWeightEstimator class along with static variables
  ↪  it uses.
"""
from __future__ import absolute_import, division, print_function

import os

from openlego.test_suite.test_examples.wing_opt.kb.disciplines.dAEDalus import
  ↪  LoadCaseSpecific
```

```
25 from lxml import etree
26 from numpy import sqrt, expm1, exp
27
28 from openlego.test_suite.test_examples.wing_opt.kb.disciplines.xpaths import *
29 from openlego.utils.xml_utils import xml_safe_create_element
30 from openlego.partials.partials import Partials
31
32 dir_path = os.path.dirname(os.path.realpath(__file__))
33
34 g0 = 9.80665        # Standard gravitational acceleration, [m/s^2]
35 rho0 = 1.225        # ISA sea level density, [kg/m^3]
36 R_air = 287.05287   # ISA specific gas constant, [J/kg/K]
37 P0 = 101325         # ISA sea level pressure, [Pa]
38 T0 = 288.15         # ISA sea level temperature, [K]
39 beta = -0.0065      # ISA temperature lapse rate, [K/m]
40 kappa = 1.4         # ISA ratio of specific heats, [-]
41
42
43 class FuelWeightEstimator(LoadCaseSpecific):
44     """Definition of the Fuel Weight Estimator discipline.
45
46     This discipline estimates the weight of the fuel required to fly a given mission based on
   ↪  the Breguet range
47     equation.
48     """
49
50     def __init__(self, n_wing_segments=2, n_load_cases=1):
51         super(FuelWeightEstimator, self).__init__(n_wing_segments, n_load_cases)
52
53     @property
54     def creator(self):
55         return 'D. de Vries'
56
57     @property
58     def description(self):
59         return 'Fuel weight estimator'
60
61     def generate_input_xml(self):
62         # type: () -> str
63         root = etree.Element('cpacs')
64         doc = etree.ElementTree(root)
65
66         for i in range(1, self.n_load_cases + 1):
67             xml_safe_create_element(doc, x_M % i, 0)
68             xml_safe_create_element(doc, x_H % i, 0)
69             xml_safe_create_element(doc, x_n % i, 0)
70             xml_safe_create_element(doc, x_CDf % i, 0)
71             xml_safe_create_element(doc, x_CDi % i, 0)
72             xml_safe_create_element(doc, x_CL % i, 0)
73
74         xml_safe_create_element(doc, x_CDfus, 0)
75         xml_safe_create_element(doc, x_CDother, 0)
76         xml_safe_create_element(doc, x_R, 0)
77         xml_safe_create_element(doc, x_SFC, 0)
78         xml_safe_create_element(doc, x_m_fuel_res, 0)
79         xml_safe_create_element(doc, x_m_fixed, 0)
80         xml_safe_create_element(doc, x_m_wing, 0)
81
82         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
83
84     def generate_output_xml(self):
85         # type: () -> str
86         root = etree.Element('cpacs')
87         doc = etree.ElementTree(root)
88
89         xml_safe_create_element(doc, x_CD, 0)
90         xml_safe_create_element(doc, x_LD, 0)
91         xml_safe_create_element(doc, x_fwe_CL, 0)
92         xml_safe_create_element(doc, x_m_fuel, 0)
93         xml_safe_create_element(doc, x_m_mtow, 0)
94
```

```
95          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
96
97      def generate_partials_xml(self):
98          partials = Partials()
99          partials.declare_partials(x_CD, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother])
100         partials.declare_partials(x_LD, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL % 1])
101         partials.declare_partials(x_fwe_CL, x_CL % 1)
102         partials.declare_partials(x_m_fuel, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL %
    ↪ 1,
103                                              x_M % 1, x_H % 1, x_R, x_SFC, x_m_fuel_res,
    ↪ x_m_fixed, x_m_wing])
104         partials.declare_partials(x_m_mtow, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL %
    ↪ 1,
105                                              x_M % 1, x_H % 1, x_R, x_SFC, x_m_fuel_res,
    ↪ x_m_fixed, x_m_wing])
106         return partials.get_string()
107
108     @property
109     def supplies_partials(self):
110         return False
111
112     @staticmethod
113     def execute(in_file, out_file='FWE-output-loc.xml'):
114         # Obtain data from XML file
115         tree = etree.parse(in_file)
116
117         M = H = C_D_f = C_D_i = C_L = 0
118         for i in range(1, LoadCaseSpecific.get_n_loadcases(tree) + 1):
119             M = float(tree.xpath(x_M % i)[0].text)                          # Mach number, [-]
120             H = float(tree.xpath(x_H % i)[0].text)                          # Altitude, [m]
121             C_D_f = float(tree.xpath(x_CDf % i)[0].text)                    # Friction drag
    ↪ coefficient, [-]
122             C_D_i = float(tree.xpath(x_CDi % i)[0].text)                    # Induced drag
    ↪ coefficient, [-]
123             C_L = float(tree.xpath(x_CL % i)[0].text)                     # Lift coefficient, [-]
124
125             if round(float(tree.xpath(x_n % i)[0].text)) == 1.:
126                 break
127
128       C_D_fus = float(tree.xpath(x_CDfus)[0].text)             # Fuselage drag coefficient, [-]
129         C_D_other = float(tree.xpath(x_CDother)[0].text)          # Remaining drag coefficient,
    ↪ [-]
130         R = float(tree.xpath(x_R)[0].text)                            # Range, [m]
131         SFC = float(tree.xpath(x_SFC)[0].text)                        # Specific fuel consumption,
    ↪ [kg/N/s]
132         m_fuel_res = float(tree.xpath(x_m_fuel_res)[0].text)          # Reserve fuel weight, [kg]
133         m_fixed = float(tree.xpath(x_m_fixed)[0].text)                # Fixed weight, [kg]
134         m_wing = float(tree.xpath(x_m_wing)[0].text)                  # Wing weight, [kg]
135
136         # Perform calculations
137         C_D = C_D_i + C_D_f + C_D_fus + C_D_other                     # Total drag coefficient, [-]
138        L_D = C_L/C_D                                          # Aerodynamic efficiency, L/D, [-]
139         LW = m_fixed + m_wing + m_fuel_res                            # Landing weight, [kg]
140         T = T0 + beta * H                                            # Temperature at altitude, [K]
141        a = sqrt(kappa * R_air * T)                                 # Speed of sound at altitude, [m/s]
142
143         if L_D == 0:
144             m_fuel = 0.
145         else:
146            m_fuel = LW * expm1(R * g0 * SFC / (a * M) / L_D)          # Required fuel weight, [kg]
147
148         m_mtow = m_fuel + LW
149         m_fuel = m_fuel + m_fuel_res
150
151         # Write results to output XML file
152         root = etree.Element('cpacs')
153         doc = etree.ElementTree(root)
154         xml_safe_create_element(doc, x_CD, C_D)
155         xml_safe_create_element(doc, x_LD, L_D)
156         xml_safe_create_element(doc, x_fwe_CL, C_L)
157         xml_safe_create_element(doc, x_m_fuel, m_fuel)
```

```
158         xml_safe_create_element(doc, x_m_mtow, m_mtow)
159         doc.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
160 
161     @staticmethod
162     def linearize(in_file, partials_file):
163         tree = etree.parse(in_file)
164 
165         M = H = C_D_f = C_D_i = C_L = 0
166         for i in range(1, LoadCaseSpecific.get_n_loadcases(tree) + 1):
167             M = float(tree.xpath(x_M % i)[0].text)  # Mach number, [-]
168             H = float(tree.xpath(x_H % i)[0].text)  # Altitude, [m]
169             C_D_f = float(tree.xpath(x_CDf % i)[0].text)  # Friction drag coefficient, [-]
170             C_D_i = float(tree.xpath(x_CDi % i)[0].text)  # Induced drag coefficient, [-]
171             C_L = float(tree.xpath(x_CL % i)[0].text)  # Lift coefficient, [-]
172 
173             if round(float(tree.xpath(x_n % i)[0].text)) == 1.:
174                 break
175 
176         C_D_fus = float(tree.xpath(x_CDfus)[0].text)  # Fuselage drag coefficient, [-]
177         C_D_other = float(tree.xpath(x_CDother)[0].text)  # Remaining drag coefficient, [-]
178         R = float(tree.xpath(x_R)[0].text)  # Range, [m]
179         SFC = float(tree.xpath(x_SFC)[0].text)  # Specific fuel consumption, [kg/N/s]
180         m_fuel_res = float(tree.xpath(x_m_fuel_res)[0].text)  # Reserve fuel weight, [kg]
181         m_fixed = float(tree.xpath(x_m_fixed)[0].text)  # Fixed weight, [kg]
182         m_wing = float(tree.xpath(x_m_wing)[0].text)  # Wing weight, [kg]
183 
184         C_D = C_D_i + C_D_f + C_D_fus + C_D_other               # Total drag coefficient, [-]
185         L_D = C_L / C_D  # Aerodynamic efficiency, L/D, [-]
186         LW = m_fixed + m_wing + m_fuel_res  # Landing weight, [kg]
187         T = T0 + beta * H  # Temperature at altitude, [K]
188         a = sqrt(kappa * R_air * T)  # Speed of sound at altitude, [m/s]
189 
190         if L_D == 0:
191             dLD_dd = 5*[1.]
192             dmfuel_d2 = 12*[1.]
193             dmmtow_dd = 12*[1.]
194         else:
195            m_fuel = LW * expm1(R * g0 * SFC / (a * M) / L_D)       # Required fuel weight, [kg]
196 
197             dmfuel_dd = LW*R*g0*SFC/(a*M*L_D)*exp(R*g0*SFC/(a*M*L_D))
198             da_dH = .5*kappa*R_air/a
199 
200             dLD_dd = 4*[1./C_L] + [-C_D/C_L**2]
201             dmfuel_d2 = 4*[dmfuel_dd/C_D] + [-dmfuel_dd/C_L, -dmfuel_dd/M,
202                                              -dmfuel_dd/a*da_dH, dmfuel_dd/R,
203                                              dmfuel_dd/SFC, 1., m_fuel/m_fuel_res,
204                                              m_fuel/m_fixed, m_fuel/m_wing]
205             dmmtow_dd = 4 * [dmfuel_dd / C_D] + [-dmfuel_dd / C_L, -dmfuel_dd / M,
206                                                  -dmfuel_dd / a * da_dH, dmfuel_dd / R,
207                                              dmfuel_dd / SFC, 1., dmfuel_dd / m_fuel_res + 1.,
208                                                  m_fuel / m_fixed + 1., m_fuel / m_wing + 1.]
209 
210         partials = Partials()
211         partials.declare_partials(x_CD, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother], 4*[1.])
212         partials.declare_partials(x_LD, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL % 1],
    ↪ dLD_dd)
213         partials.declare_partials(x_fwe_CL, x_CL % 1, 1.)
214         partials.declare_partials(x_m_fuel, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL %
    ↪ 1,
215                                              x_M % 1, x_H % 1, x_R, x_SFC, x_m_fuel_res,
    ↪ x_m_fixed, x_m_wing],
216                                 dmfuel_d2)
217         partials.declare_partials(x_m_mtow, [x_CDi % 1, x_CDf % 1, x_CDfus, x_CDother, x_CL %
    ↪ 1,
218                                              x_M % 1, x_H % 1, x_R, x_SFC, x_m_fuel_res,
    ↪ x_m_fixed, x_m_wing],
219                                 dmmtow_dd)
220 
221         partials.write(partials_file)
```

Code frament A.41: Code of the Python module containing the fuel weight estimator discipline.

#### A.5.2.3.3 Problem Definition

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definition of the ProblemDefinition discipline."""
from __future__ import absolute_import, division, print_function

import numpy as np
from lxml import etree

from openlego.api import AbstractDiscipline
from openlego.test_suite.test_examples.wing_opt.kb.disciplines.xpaths import *
from openlego.utils.xml_utils import xml_safe_create_element
from openlego.partials.partials import Partials


class Constraints(AbstractDiscipline):
    """Defines all the constraints for the problem."""

    def __init__(self, n_wing_segments=2):
        # type: (int) -> None
        """Initialize the Constraints discipline for a given number of wing segments.

        Parameters
        ----------
            n_wing_segments : int
                Number of wing segments.
        """
        self.n_wing_segments = n_wing_segments
        super(Constraints, self).__init__()

    @property
    def creator(self):
        return 'D. de Vries'

    @property
    def description(self):
        return 'Calculates the constraint values for the wing optimization problem'

    def generate_input_xml(self):
        # type: () -> str
        root = etree.Element('cpacs')
        doc = etree.ElementTree(root)

        xml_safe_create_element(doc, x_sigma_yield, 1.)
        xml_safe_create_element(doc, x_WS_init, 1.)
        xml_safe_create_element(doc, x_CL_buffet, 1.)

        for x_sigma in x_sigmas_out:
            xml_safe_create_element(doc, x_sigma, np.zeros(self.n_wing_segments))

        xml_safe_create_element(doc, x_ref_area, 0.)
        xml_safe_create_element(doc, x_m_mtow, 0.)
```

```
        xml_safe_create_element(doc, x_fwe_CL, 0.)

        return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)

    def generate_output_xml(self):
        # type: () -> str
        root = etree.Element('cpacs')
        doc = etree.ElementTree(root)

        for x_sigma in x_con_sigmas:
            xml_safe_create_element(doc, x_sigma, np.zeros(self.n_wing_segments))
        # xml_safe_create_element(doc, x_con_ks, 0.)

        xml_safe_create_element(doc, x_con_WS, 0.)
        xml_safe_create_element(doc, x_con_buffet, 0.)

        return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)

    def generate_partials_xml(self):
        partials = Partials()
        for i, x_sigma in enumerate(x_con_sigmas):
            partials.declare_partials(x_sigma, [x_sigma_yield, x_sigmas_out[i]])

        partials.declare_partials(x_con_WS, [x_WS_init, x_ref_area, x_m_mtow])
        partials.declare_partials(x_con_buffet, [x_CL_buffet, x_fwe_CL])
        return partials.get_string()

    @property
    def supplies_partials(self):
        return False

    @staticmethod
    def execute(in_file, out_file):
        doc_in = etree.parse(in_file)

        sigma_yield = float(doc_in.xpath(x_sigma_yield)[0].text)

        root = etree.Element('cpacs')
        doc_out = etree.ElementTree(root)
        for i in range(4):
            xml_safe_create_element(
                doc_out, x_con_sigmas[i],
                np.array(doc_in.xpath(x_sigmas_out[i])[0].text.split(';'),
 ↪  dtype=float)/sigma_yield - 1.)

        MTOW = float(doc_in.xpath(x_m_mtow)[0].text)
        S = float(doc_in.xpath(x_ref_area)[0].text)
        WS_init = float(doc_in.xpath(x_WS_init)[0].text)
        con_ws = MTOW/S/WS_init - 1.

        xml_safe_create_element(doc_out, x_con_WS, con_ws)
        xml_safe_create_element(doc_out, x_con_buffet,

 ↪  float(doc_in.xpath(x_fwe_CL)[0].text)/float(doc_in.xpath(x_CL_buffet)[0].text) - 1.)

        doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)

    @staticmethod
    def linearize(in_file, partials_file):
        doc_in = etree.parse(in_file)

        sigma_yield = float(doc_in.xpath(x_sigma_yield)[0].text)

        n_ws = len(doc_in.xpath(x_sigmas_out[0])[0].text.split(';'))

        MTOW = float(doc_in.xpath(x_m_mtow)[0].text)
        S = float(doc_in.xpath(x_ref_area)[0].text)
        WS_init = float(doc_in.xpath(x_WS_init)[0].text)

        partials = Partials()
        for i in range(4):
```

```
136             partials.declare_partials(x_con_sigmas[i], [x_sigma_yield, x_sigmas_out[i]], [
137                 -np.array(doc_in.xpath(x_sigmas_out[i])[0].text.split(';'), dtype=float) /
    ↪ sigma_yield**2,
138                 n_ws * [1./sigma_yield]])
139
140         partials.declare_partials(x_con_WS, [x_WS_init, x_ref_area, x_m_mtow],
141                                   [-MTOW/S/WS_init**2, -MTOW/S**2/WS_init, 1./S/WS_init])
142         partials.declare_partials(x_con_buffet, [x_CL_buffet, x_fwe_CL],
143
    ↪ [-float(doc_in.xpath(x_fwe_CL)[0].text)/float(doc_in.xpath(x_CL_buffet)[0].text)**2,
144                                   1./float(doc_in.xpath(x_CL_buffet)[0].text)])
145
146         partials.write(partials_file)
147
148
149 class Objectives(AbstractDiscipline):
150     """Defines the objective functions for the problem."""
151
152     def __init__(self):
153         # type: (int) -> None
154         """Initialize the Objectives discipline."""
155         pass
156
157     @property
158     def creator(self):
159         return 'D. de Vries'
160
161     @property
162     def description(self):
163         return 'Calculates the objective values for the wing optimization problem'
164
165     def generate_input_xml(self):
166         # type: () -> str
167         root = etree.Element('cpacs')
168         doc = etree.ElementTree(root)
169
170         xml_safe_create_element(doc, x_m_fuel_init, 1.)
171         xml_safe_create_element(doc, x_m_fuel, 1.)
172         xml_safe_create_element(doc, x_m_wing_init, 1.)
173         xml_safe_create_element(doc, x_m_wing, 1.)
174
175         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
176
177     def generate_output_xml(self):
178         # type: () -> str
179         root = etree.Element('cpacs')
180         doc = etree.ElementTree(root)
181
182         xml_safe_create_element(doc, x_obj_m_fuel, 0.)
183         xml_safe_create_element(doc, x_obj_m_wing, 0.)
184
185         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
186
187     def generate_partials_xml(self):
188         partials = Partials()
189         partials.declare_partials(x_obj_m_fuel, [x_m_fuel_init, x_m_fuel])
190         partials.declare_partials(x_obj_m_wing, [x_m_wing_init, x_m_wing])
191         return partials.get_string()
192
193     @property
194     def supplies_partials(self):
195         return False
196
197     @staticmethod
198     def execute(in_file, out_file):
199         doc_in = etree.parse(in_file)
200
201         root = etree.Element('cpacs')
202         doc_out = etree.ElementTree(root)
203
204         xml_safe_create_element(doc_out, x_obj_m_fuel,
```

```
205                                 float(doc_in.xpath(x_m_fuel)[0].text) /
    ↪ float(doc_in.xpath(x_m_fuel_init)[0].text))
206             xml_safe_create_element(doc_out, x_obj_m_wing,
207                                 float(doc_in.xpath(x_m_wing)[0].text) /
    ↪ float(doc_in.xpath(x_m_wing_init)[0].text))
208
209             doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
210
211         @staticmethod
212         def linearize(in_file, partials_file):
213             doc_in = etree.parse(in_file)
214
215             partials = Partials()
216             partials.declare_partials(x_obj_m_fuel, [x_m_fuel_init, x_m_fuel], [
217                 -float(doc_in.xpath(x_m_fuel)[0].text) /
    ↪ float(doc_in.xpath(x_m_fuel_init)[0].text)**2,
218                 1. / float(doc_in.xpath(x_m_fuel_init)[0].text)
219             ])
220             partials.declare_partials(x_obj_m_wing, [x_m_wing_init, x_m_wing], [
221                 -float(doc_in.xpath(x_m_wing)[0].text) /
    ↪ float(doc_in.xpath(x_m_wing_init)[0].text) ** 2,
222                 1. / float(doc_in.xpath(x_m_wing_init)[0].text)
223             ])
224
225             partials.write(partials_file)
```

Code frament A.42: Code of the Python module containing the constraint and objective function disciplines.

#### A.5.2.3.4 Simple Aerostructural Analysis

```
1  from __future__ import absolute_import
2  from __future__ import division
3  from __future__ import print_function
4
5  import numpy as np
6  import matlab
7  import time
8  import os
9
10 from lxml import etree
11 from matlab.engine import MatlabExecutionError
12
13 from openlego.test_suite.test_examples.wing_opt.kb.disciplines.dAEDalus import
     ↪ LoadCaseSpecific,\
14     start_new_matlab_engine, n_seg_x, n_seg_y
15 from openlego.test_suite.test_examples.wing_opt.kb.disciplines.WingObjectModel import
     ↪ WingObjectModel
16 from openlego.test_suite.test_examples.wing_opt.kb.disciplines.xpaths import *
17 from openlego.utils.xml_utils import xml_safe_create_element, xml_merge
18
19
20 dir_path = os.path.dirname(os.path.realpath(__file__))
21
22
23 class SimpleAerostructuralAnalysis(LoadCaseSpecific):
24
25     MATLAB_TIMEOUT = 1800.
26     _mles = []
27     _timestamp = [0.]
28
29     data = []
30     count = 0
31
32     def __init__(self, n_wing_segments=2, n_load_cases=1):
33         super(SimpleAerostructuralAnalysis, self).__init__(n_wing_segments, n_load_cases)
34
35     @property
36     def description(self):
37         return 'dAEDalus Steady Aerostructural Loop'
```

```
38
39      def generate_input_xml(self):
40          wd = WingObjectModel(self.n_wing_segments)
41          s = wd.generate_input_xml()
42          parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
43          doc = etree.fromstring(s, parser)
44
45          elem = doc.xpath(x_m_wing)
46          elem[0].getparent().remove(elem[0])
47
48          for i in range(1, self.n_load_cases + 1):
49              xml_safe_create_element(doc, x_M % i, 0.)
50              xml_safe_create_element(doc, x_H % i, 0.)
51              xml_safe_create_element(doc, x_n % i, 0.)
52
53          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
54
55      def generate_output_xml(self):
56          wd = WingObjectModel(self.n_wing_segments)
57          s = wd.generate_output_xml()
58
59          parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
60          doc = etree.fromstring(s, parser)
61
62          xml_safe_create_element(doc, x_m_wing, 0.)
63
64          for i in range(1, self.n_load_cases + 1):
65              xml_safe_create_element(doc, x_CL % i, 0.)
66              xml_safe_create_element(doc, x_CDf % i, 0.)
67              xml_safe_create_element(doc, x_CDi % i, 0.)
68
69              for x_sigma in x_sigmas_in:
70                  xml_safe_create_element(doc, x_sigma % i, np.zeros(self.n_wing_segments))
71
72          return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
73
74      @property
75      def supplies_partials(self):
76          return False
77
78      @staticmethod
79      def execute(in_file, out_file):
80          in_file = os.path.abspath(in_file)
81          out_file = os.path.abspath(out_file)
82
83          parser = etree.XMLParser(remove_blank_text=True, encoding='utf-8')
84
85          doc_in = etree.parse(in_file, parser)
86          n_lc = SimpleAerostructuralAnalysis.get_n_loadcases(doc_in)
87
88          # Prepare matlab engines
89          if not len(SimpleAerostructuralAnalysis._mles) or \
90                  (time.time() - SimpleAerostructuralAnalysis._timestamp[
91                      0] >= SimpleAerostructuralAnalysis.MATLAB_TIMEOUT):
92              SimpleAerostructuralAnalysis._mles = []
93              SimpleAerostructuralAnalysis._timestamp[0] = time.time()
94              for _ in range(n_lc):
95                  mle, _, _, _ = start_new_matlab_engine()
96                  mle.cd(dir_path)
97                  SimpleAerostructuralAnalysis._mles.append(mle)
98
99          fail = False
100
101         # Converge wing weight first
102         err = 1.
103         tol = 1.e-2
104         imax = 10
105         i = 1
106         m_wing_prev = 1.
107         m_wing = 1.e10
108         while err > tol and i < imax:
```

```
109             WingObjectModel.execute(in_file, out_file)
110             try:
111                 m_wing = SimpleAerostructuralAnalysis._mles[0].dAEDalusSteadyModelInitializer(
112                     out_file, matlab.double([n_seg_x]), matlab.double([n_seg_y]), nargout=1)
113             except MatlabExecutionError:
114                 fail = True
115                 print('fail')
116                 break
117 
118             err = abs(m_wing - m_wing_prev) / m_wing_prev
119             m_wing_prev = m_wing
120 
121             xml_safe_create_element(doc_in, x_m_wing, m_wing)
122             doc_in.write(in_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
123 
124         doc_out = etree.parse(out_file, parser)
125         xml_safe_create_element(doc_out, x_m_wing, m_wing)
126         for i in range(n_lc):
127             xml_safe_create_element(doc_out, x_CL % (i + 1), 1.)
128             xml_safe_create_element(doc_out, x_CDf % (i + 1), 1.)
129             xml_safe_create_element(doc_out, x_CDi % (i + 1), 1.)
130 
131             for j in range(4):
132               xml_safe_create_element(doc_out, x_sigmas_in[j] % (i + 1), np.ones(n_lc) * 1e10)
133 
134         if not fail:
135             # Gather load case flight states
136             M = n_lc * [0.]
137             H = n_lc * [0.]
138             n = n_lc * [0.]
139             for i in range(1, n_lc + 1):
140                 M[i - 1] = float(doc_in.xpath(x_M % i)[0].text)
141                 H[i - 1] = float(doc_in.xpath(x_H % i)[0].text)
142                 n[i - 1] = float(doc_in.xpath(x_n % i)[0].text)
143 
144             # Perform full analysis on each loadcase
145             futures = n_lc * [None]
146             for i in range(n_lc):
147                 futures[i] =
    ↪ SimpleAerostructuralAnalysis._mles[i].dAEDalusSteadyAerostructuralLoop(
148                     out_file, matlab.double([n_seg_x]), matlab.double([n_seg_y]), M[i], H[i],
    ↪ n[i],
149                     nargout=8, async=True)
150 
151             # Obtain results and write to file
152             for i, future in enumerate(futures):
153                 if future is not None:
154                     try:
155                         _, C_L, C_D_f, C_D_i, sigma_fs, sigma_rs, sigma_ts, sigma_bs =
    ↪ future.result()
156                         sigmas = [np.array(sigma_fs), np.array(sigma_rs), np.array(sigma_ts),
    ↪ np.array(sigma_bs)]
157 
158                         xml_safe_create_element(doc_out, x_CL % (i + 1), C_L)
159                         xml_safe_create_element(doc_out, x_CDf % (i + 1), C_D_f)
160                         xml_safe_create_element(doc_out, x_CDi % (i + 1), C_D_i)
161 
162                         for j in range(4):
163                          xml_safe_create_element(doc_out, x_sigmas_in[j] % (i + 1), sigmas[j])
164                     except MatlabExecutionError:
165                         print('double fail')
166                         break
167 
168         doc_out.write(out_file, encoding='utf-8', pretty_print=True, xml_declaration=True)
```

Code frament A.43: Code of the Python module containing the collapsed, integrated aerostructural analysis discipline.

#### A.5.2.3.5 Wing Object Model

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains the definition of the WingObjectModel class along with a collection of static variables and methods
it uses.
"""
from __future__ import absolute_import, division, print_function

import os

import numpy as np
from lxml import etree
from typing import Optional, Union, Tuple, Sized

from openlego.api import AbstractDiscipline
from openlego.test_suite.test_examples.wing_opt.kb.disciplines.xpaths import *
from openlego.utils.xml_utils import xml_safe_create_element
from openlego.partials.partials import Partials

dir_path = os.path.dirname(os.path.realpath(__file__))

af_x = (1.000000, 0.997840, 0.995410, 0.992720, 0.989770, 0.986550, 0.983080, 0.979350, 0.975360, 0.971120,
        0.966630, 0.961890, 0.956900, 0.951680, 0.946210, 0.940500, 0.934550, 0.928380, 0.921980, 0.915350,
        0.908500, 0.901440, 0.894160, 0.886670, 0.878980, 0.871080, 0.862990, 0.854700, 0.846230, 0.837570,
        0.828730, 0.819720, 0.810540, 0.801190, 0.791680, 0.782020, 0.772210, 0.762260, 0.752160, 0.741940,
        0.731580, 0.721100, 0.710510, 0.699800, 0.688990, 0.678080, 0.667080, 0.655990, 0.644810, 0.633560,
        0.622240, 0.610860, 0.599420, 0.587930, 0.576390, 0.564820, 0.553210, 0.541580, 0.529920, 0.518250,
        0.506580, 0.494900, 0.483220, 0.471560, 0.459920, 0.448290, 0.436700, 0.425150, 0.413630, 0.402170,
        0.390760, 0.379410, 0.368130, 0.356920, 0.345790, 0.334740, 0.323780, 0.312930, 0.302170, 0.291520,
        0.280990, 0.270580, 0.260290, 0.250130, 0.240110, 0.230230, 0.220500, 0.210920, 0.201490, 0.192230,
        0.183140, 0.174220, 0.165480, 0.156920, 0.148540, 0.140360, 0.132370, 0.124580, 0.116990, 0.109610,
        0.102450, 0.095500, 0.088770, 0.082260, 0.075980, 0.069920, 0.064110, 0.058520, 0.053180, 0.048080,
        0.043220, 0.038610, 0.034260, 0.030150, 0.026300, 0.022700, 0.019370, 0.016290, 0.013480, 0.010920,
        0.008640, 0.006620, 0.004870, 0.003380, 0.002160, 0.001220, 0.000540, 0.000140, 0.000000, 0.000000,
        0.000140, 0.000540, 0.001220, 0.002160, 0.003380, 0.004870, 0.006620, 0.008640, 0.010920, 0.013480,
        0.016290, 0.019370, 0.022700, 0.026300, 0.030150, 0.034260, 0.038610, 0.043220, 0.048080, 0.053180,
        0.058520, 0.064110, 0.069920, 0.075980, 0.082260, 0.088770, 0.095500, 0.102450, 0.109610, 0.116990,
        0.124580, 0.132370, 0.140360, 0.148540, 0.156920, 0.165480, 0.174220, 0.183140, 0.192230, 0.201490,
```

```
53          0.210920, 0.220500, 0.230230, 0.240110, 0.250130, 0.260290, 0.270580, 0.280990,
      ↪  0.291520, 0.302170,
54          0.312930, 0.323780, 0.334740, 0.345790, 0.356920, 0.368130, 0.379410, 0.390760,
      ↪  0.402170, 0.413630,
55          0.425150, 0.436700, 0.448290, 0.459920, 0.471560, 0.483220, 0.494900, 0.506580,
      ↪  0.518250, 0.529920,
56          0.541580, 0.553210, 0.564820, 0.576390, 0.587930, 0.599420, 0.610860, 0.622240,
      ↪  0.633560, 0.644810,
57          0.655990, 0.667080, 0.678080, 0.688990, 0.699800, 0.710510, 0.721100, 0.731580,
      ↪  0.741940, 0.752160,
58          0.762260, 0.772210, 0.782020, 0.791680, 0.801190, 0.810540, 0.819720, 0.828730,
      ↪  0.837570, 0.846230,
59          0.854700, 0.862990, 0.871080, 0.878980, 0.886670, 0.894160, 0.901440, 0.908500,
      ↪  0.915350, 0.921980,
60          0.928380, 0.934550, 0.940500, 0.946210, 0.951680, 0.956900, 0.961890, 0.966630,
      ↪  0.971120, 0.975360,
61          0.979350, 0.983080, 0.986550, 0.989770, 0.992720, 0.995410, 0.997840, 1.000000)
62  af_z = (0.000000, 0.001150, 0.002320, 0.003500, 0.004690, 0.005900, 0.007110, 0.008340,
      ↪  0.009560, 0.010810,
63          0.012040, 0.013270, 0.014490, 0.015680, 0.016880, 0.018070, 0.019240, 0.020410,
      ↪  0.021570, 0.022730,
64          0.023890, 0.025070, 0.026270, 0.027480, 0.028720, 0.029980, 0.031270, 0.032590,
      ↪  0.033930, 0.035310,
65          0.036690, 0.038090, 0.039510, 0.040920, 0.042320, 0.043720, 0.045100, 0.046460,
      ↪  0.047790, 0.049090,
66          0.050340, 0.051550, 0.052720, 0.053830, 0.054890, 0.055900, 0.056850, 0.057750,
      ↪  0.058580, 0.059360,
67          0.060080, 0.060760, 0.061380, 0.061950, 0.062470, 0.062940, 0.063360, 0.063720,
      ↪  0.064070, 0.064350,
68          0.064590, 0.064790, 0.064940, 0.065060, 0.065120, 0.065140, 0.065130, 0.065060,
      ↪  0.064960, 0.064810,
69          0.064610, 0.064370, 0.064090, 0.063750, 0.063380, 0.062970, 0.062510, 0.062010,
      ↪  0.061460, 0.060890,
70          0.060280, 0.059620, 0.058930, 0.058200, 0.057460, 0.056670, 0.055870, 0.055030,
      ↪  0.054160, 0.053270,
71          0.052350, 0.051400, 0.050450, 0.049450, 0.048440, 0.047390, 0.046340, 0.045250,
      ↪  0.044150, 0.043020,
72          0.041870, 0.040710, 0.039510, 0.038320, 0.037080, 0.035840, 0.034570, 0.033290,
      ↪  0.031980, 0.030650,
73          0.029290, 0.027920, 0.026510, 0.025090, 0.023630, 0.022130, 0.020610, 0.019050,
      ↪  0.017450, 0.015840,
74          0.014170, 0.012480, 0.010770, 0.009020, 0.007240, 0.005460, 0.003660, 0.001820,
      ↪  0.000000, 0.000000,
75          -0.001550, -0.003060, -0.004570, -0.006050, -0.007490, -0.008930, -0.010320,
      ↪  -0.011680, -0.013020,
76          -0.014310, -0.015580, -0.016810, -0.018010, -0.019190, -0.020340, -0.021460,
      ↪  -0.022570, -0.023650,
77          -0.024730, -0.025770, -0.026820, -0.027850, -0.028860, -0.029860, -0.030860,
      ↪  -0.031830, -0.032810,
78          -0.033760, -0.034700, -0.035620, -0.036540, -0.037430, -0.038290, -0.039130,
      ↪  -0.039960, -0.040770,
79          -0.041540, -0.042290, -0.043010, -0.043710, -0.044370, -0.045000, -0.045600,
      ↪  -0.046160, -0.046670,
80          -0.047160, -0.047600, -0.048000, -0.048350, -0.048640, -0.048900, -0.049100,
      ↪  -0.049250, -0.049340,
81          -0.049380, -0.049360, -0.049270, -0.049140, -0.048940, -0.048670, -0.048350,
      ↪  -0.047970, -0.047520,
82          -0.046990, -0.046410, -0.045760, -0.045050, -0.044270, -0.043420, -0.042500,
      ↪  -0.041520, -0.040470,
83          -0.039350, -0.038160, -0.036920, -0.035610, -0.034240, -0.032800, -0.031320,
      ↪  -0.029790, -0.028210,
84          -0.026600, -0.024940, -0.023270, -0.021560, -0.019840, -0.018110, -0.016390,
      ↪  -0.014660, -0.012960,
85          -0.011270, -0.009600, -0.007970, -0.006380, -0.004840, -0.003350, -0.001910,
      ↪  -0.000550, 0.000760, 0.001980,
86          0.003120, 0.004190, 0.005150, 0.006030, 0.006800, 0.007480, 0.008040, 0.008490,
      ↪  0.008850, 0.009090,
87          0.009220, 0.009250, 0.009180, 0.009010, 0.008750, 0.008400, 0.007970, 0.007470,
      ↪  0.006900, 0.006280,
88          0.005600, 0.004890, 0.004130, 0.003340, 0.002540, 0.001710, 0.000880, 0.000000)
89
```

```
90
91  def add_cleared_child(parent, child_name, attrib=None):
92      # type: (etree._Element, str, Optional[dict]) -> etree._Element
93      """Clears the child with the given name of the given parent with the given attributes and
        ↪ returns it.
94
95      If a child doesn't exist yet with the given name and attributes, it is created.
96
97      Parameters
98      ----------
99          parent : :obj:'etree._Element'
100             Parent element.
101
102         child_name : str
103             Name of the child element.
104
105         attrib : dict, optional
106             Dictionary of attributes.
107
108     Returns
109     -------
110         :obj:'etree._Element'
111             The cleared child element.
112     """
113     for child in parent.findall(child_name):
114         if attrib is None:
115             child.clear()
116             return child
117         elif all([value == child.attrib[key] for (key, value) in attrib.items() if key in
        ↪ child.attrib]):
118             child.clear()
119             return child
120
121     return etree.SubElement(parent, child_name, attrib)
122
123
124 def add_point(tree, parent, point_name, *args):
125     # type: (etree._ElementTree, Union[str, etree._Element], str, *Union[float,
        ↪ Tuple[float]]) -> etree._Element
126     """Safely creates a new point in the XML tree.
127
128     Parameters
129     ----------
130         tree : :obj:'etree._ElementTree'
131             Tree in which to add the point.
132
133         parent : str or :obj:'etree._Element'
134             XPath of the element or 'etree._Element' under which to add the point.
135
136         point_name : str
137             Name of the point.
138
139         *args
140             Either three floats (x, y, z) or a length 3 tuple correspondingly, which describes
        ↪ the point.
141
142     Returns
143     -------
144         :obj:'etree._Element'
145             The 'etree._Element' corresponding to the newly created point.
146     """
147     if len(args) == 1 and len(args[0]) == 3:
148         x, y, z = args[0]
149     elif len(args) == 3:
150         x = args[0]
151         y = args[1]
152         z = args[2]
153     else:
154         raise ValueError('*args should be three floats or a tuple or length 3')
155
156     if isinstance(parent, str):
```

```
157         parent = xml_safe_create_element(tree, parent)
158 
159     point = add_cleared_child(parent, point_name)
160     etree.SubElement(point, 'x').text = str(x)
161     etree.SubElement(point, 'y').text = str(y)
162     etree.SubElement(point, 'z').text = str(z)
163 
164     return point
165 
166 
167 def add_transform(tree, parent, scaling=(1, 1, 1), rotation=(0, 0, 0), translation=(0, 0, 0)):
168     # type: (etree._ElementTree, Union[str, etree._Element], tuple, tuple, tuple) ->
      ↪  etree._Element
169     """Safely creates a new CPACS transformation within the XML tree at the given XPath.
170 
171     Parameters
172     ----------
173         tree : :obj:'etree._ElementTree'
174             Tree in which to add the transformation.
175 
176         parent : str or :obj:'etree._Element'
177             XPath of an element or 'etree._Element' under which to add the transformation.
178 
179         scaling, rotation, translation : tuple of float
180             Tuples of length 3 corresponding to the x-, y-, and z-components of the scaling,
      ↪  rotation, and translation.
181 
182     Returns
183     -------
184         :obj:'etree._Element'
185             'etree._Element' corresponding to the newly added transformation.
186     """
187     if len(scaling) != len(rotation) != len(translation) != 3:
188         raise ValueError('scaling, rotation, and translation should be tuples of length 3')
189 
190     if isinstance(parent, str):
191         parent = xml_safe_create_element(tree, parent)
192 
193     transform = add_cleared_child(parent, 'transformation')
194     add_point(tree, transform, 'scaling', scaling)
195     add_point(tree, transform, 'rotation', rotation)
196     add_point(tree, transform, 'translation', translation)
197 
198     return transform
199 
200 
201 def add_spar_position(tree, parent, uid, element_uid, xsi):
202     # type: (etree._ElementTree, Union[str, etree._Element], str, str, float) ->
      ↪  etree._Element
203     """Safely creates a new CPACS spar position with the given XML tree.
204 
205     Parameters
206     ----------
207         tree : :obj:'etree._ElementTree'
208             Tree in which to add the spar position.
209 
210         parent : str or :obj:'etree._Element'
211             XPath of an element or 'etree._Element' under which to add the spar position.
212 
213         uid, element_uid : str
214             Unique identifiers of the spar position and the corresponding element.
215 
216         xsi : float
217             Chordwise location of the spar position.
218 
219     Returns
220     -------
221         :obj:'etree._Element'
222             'etree._Element' corresponding to the newly added spar position.
223     """
224     if isinstance(parent, str):
```

```
225         parent = xml_safe_create_element(tree, parent)
226
227     spar_pos = add_cleared_child(parent, 'sparPosition', {'uID': uid})
228     etree.SubElement(spar_pos, 'name').text = uid
229     etree.SubElement(spar_pos, 'elementUID').text = element_uid
230     etree.SubElement(spar_pos, 'xsi').text = str(xsi)
231
232     return spar_pos
233
234
235 def add_spar_segment(tree, parent, uid, start_pos_uid, end_pos_uid, mat_uid, t_web, t_top,
    ↪  t_bottom):
236     # type: (etree._ElementTree, Union[str, etree._Element], str, str, str, str, float, float,
    ↪  float) -> etree._Element
237     """Safely creates a new CPACS spar segment within the XML tree.
238
239     Parameters
240     ----------
241         tree : :obj:`etree._ElementTree`
242             Tree in which to add the spar segment.
243
244         parent : str or :obj:`etree._Element`
245             XPath of an element or `etree._Element` under which to add the spar segment.
246
247         uid, start_pos_uid, end_pos_uid, mat_uid : str
248             Unique identifiers of the spar segment, its start and end positions, and its
    ↪  material.
249
250         t_web, t_top, t_bottom : float
251             Thicknesses of the web, top, and bottom of the spar segment.
252
253     Returns
254     -------
255         :obj:`etree._Element`
256             `etree._Element` corresponding to the newly added spar segment.
257     """
258     if isinstance(parent, str):
259         parent = xml_safe_create_element(tree, parent)
260
261     spar_seg = add_cleared_child(parent, 'sparSegment', {'uID': uid})
262     etree.SubElement(spar_seg, 'name').text = uid
263     etree.SubElement(spar_seg, 'description').text = uid
264
265     x_sparseg = tree.getpath(spar_seg)
266     xml_safe_create_element(tree, '/'.join([x_sparseg, 'sparCrossSection/rotation']), 90)
267     xml_safe_create_element(tree, '/'.join([x_sparseg, 'sparCrossSection/web1/relPos']), 0.5)
268     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/web1/material/materialUID']), mat_uid)
269     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/web1/material/thickness']), t_web)
270
271     xml_safe_create_element(tree, '/'.join([x_sparseg, 'sparCrossSection/upperCap/area']), 0)
272     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/upperCap/material/materialUID']), mat_uid)
273     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/upperCap/material/thickness']), t_top)
274
275     xml_safe_create_element(tree, '/'.join([x_sparseg, 'sparCrossSection/lowerCap/area']), 0)
276     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/lowerCap/material/materialUID']), mat_uid)
277     xml_safe_create_element(tree, '/'.join([x_sparseg,
    ↪  'sparCrossSection/lowerCap/material/thickness']), t_bottom)
278
279     uids = etree.SubElement(spar_seg, 'sparPositionUIDs')
280     etree.SubElement(uids, 'sparPositionUID').text = start_pos_uid
281     etree.SubElement(uids, 'sparPositionUID').text = end_pos_uid
282
283     return spar_seg
284
285
286 def add_mass_description(tree, parent, element_name, mass, uid=None):
```

```
    # type: (etree._ElementTree, Union[str, etree._Element], str, float, Optional[str]) ->
      etree._Element
    """Safely adds a CPACS mass description to the given XML tree at the given parent.

    Parameters
    ----------
        tree : :obj:'etree._ElementTree'
            Tree in which to add the mass description.

        parent : str or :obj:'etree._Element'
            XPath of an element or 'etree._Element' under which to add the mass description.

        element_name : str
            Name of the element.

        mass : float
            Value of the mass.

        uid : str, optional
            Unique identifier of the mass description.

    Returns
    -------
        :obj:'etree._Element'
            'etree._Element' corresponding to the newly added mass description.
    """
    if uid is None:
        uid = element_name

    if isinstance(parent, str):
        parent = xml_safe_create_element(tree, parent)

    elem = etree.SubElement(parent, element_name, {'uID': uid})
    etree.SubElement(elem, 'mass').text = str(mass)
    return elem


def add_mass(tree, parent, mass_name, mass, uid=None):
    # type: (etree._ElementTree, Union[str, etree._Element], str, float, Optional[str]) ->
      etree._Element
    """Safely creates a CPACS mass within the given XML tree at the given parent.

    Parameters
    ----------
        tree : :obj:'etree._ElementTree'
            Tree in which to add the mass.

        parent : str or :obj:'etree._Element'
            XPath of an element or 'etree._Element' under which to add the mass.

        mass_name : str
            Name of the mass.

        uid : str, optional
            Unique identifier of the mass.

    Returns
    -------
        :obj:'etree._Element'
            'etree._Element' corresponding to the newly added mass.
    """
    if uid is None:
        uid = mass_name

    if isinstance(parent, etree._Element):
        parent = tree.getpath(parent)

    _mass = xml_safe_create_element(tree, '/'.join([parent, mass_name]))
    add_mass_description(tree, _mass, 'massDescription', mass, uid)
    return _mass
```

```
356
357 def add_cpacs_header(tree, name, creator, version, description, cpacs_version):
358     # type: (etree._ElementTree, str, str, str, str, str) -> etree._Element
359     """Add a CPACS header to the given XML tree root.
360
361     Parameters
362     ----------
363         tree : :obj:'etree._ElementTree'
364             Tree in which to add the header.
365
366         name, creator, version, description, cpacs_version : str
367             Name, creator, version identifier, and version of CPACS to put in the CPACS header
    ↪  information.
368
369     Returns
370     -------
371         :obj:'etree._Element'
372             'etree._Element' corresponding to the newly added header.
373     """
374     root = tree.getroot()
375     header = etree.SubElement(root, 'header')
376     etree.SubElement(header, 'name').text = name
377     etree.SubElement(header, 'creator').text = creator
378     etree.SubElement(header, 'version').text = version
379     etree.SubElement(header, 'description').text = description
380     etree.SubElement(header, 'cpacsVersion').text = cpacs_version
381     return header
382
383
384 class WingObjectModel(AbstractDiscipline):
385     """This discipline transforms a reduced aero-structural model of a wing into a CPACS file.
386
387     The wing is modeled with a number of wing segments, n_wing_segments. Each wing segment has
    ↪  a span, b, sweep angle,
388     Lambda, dihedral angle, Gamma, as well as four thicknesses, t_fs, t_rs, t_ts, and t_bs for
    ↪  the front spar,
389     rear spar, top skin, and bottom skin associated with it. The sections joining two wing
    ↪  segments each have a chord
390     length, c, thickness over chord ratio, t/c, twist angle, epsilon, from spar position,
    ↪  xsi_fs, and rear spar
391     position, xsi_rs, associated with it. The most inboard section does not have a twist
    ↪  angle, but an incidence angle
392     associated with it. Furthermore, a single thickness and material density are defined for
    ↪  the skin of the wing
393     outside of the wingbox.
394
395     Upon executing this discipline, a CPACS file containing the parameters of this reduced
    ↪  model is translated to a
396     regular CPACS file. The result should be a wing that has the same geometry as was
    ↪  described by the reduced model.
397     """
398
399     def __init__(self, n_wing_segments=2):
400         super(WingObjectModel, self).__init__()
401         self.n_wing_segments = n_wing_segments
402
403     @property
404     def creator(self):
405         return 'D. de Vries'
406
407     @property
408     def description(self):
409         return 'Wing object model'
410
411     @property
412     def supplies_partials(self):
413         return True
414
415     def generate_input_xml(self):
416         # type: () -> str
417         root = etree.Element('cpacs')
```

```
418         doc = etree.ElementTree(root)
419
420         xml_safe_create_element(doc, x_c, np.zeros(self.n_wing_segments + 1))
421         xml_safe_create_element(doc, x_tc, np.zeros(self.n_wing_segments + 1))
422         xml_safe_create_element(doc, x_epsilon, np.zeros(self.n_wing_segments))
423         xml_safe_create_element(doc, x_b, np.zeros(self.n_wing_segments))
424         xml_safe_create_element(doc, x_Lambda, np.zeros(self.n_wing_segments))
425         xml_safe_create_element(doc, x_Gamma, np.zeros(self.n_wing_segments))
426         xml_safe_create_element(doc, x_xsi_fs, np.zeros(self.n_wing_segments + 1))
427         xml_safe_create_element(doc, x_xsi_rs, np.zeros(self.n_wing_segments + 1))
428         xml_safe_create_element(doc, x_t_fs, np.zeros(self.n_wing_segments))
429         xml_safe_create_element(doc, x_t_rs, np.zeros(self.n_wing_segments))
430         xml_safe_create_element(doc, x_t_ts, np.zeros(self.n_wing_segments))
431         xml_safe_create_element(doc, x_t_bs, np.zeros(self.n_wing_segments))
432         xml_safe_create_element(doc, x_incidence, 0.)
433         xml_safe_create_element(doc, x_t_skin, 0.)
434         xml_safe_create_element(doc, x_rho_skin, 0.)
435
436         xml_safe_create_element(doc, x_m_fixed, 0.)
437         xml_safe_create_element(doc, x_m_payload, 0.)
438         xml_safe_create_element(doc, x_m_wing, 0.)
439         xml_safe_create_element(doc, x_m_fuel, 0.)
440         xml_safe_create_element(doc, x_m_mlw, 0.)
441         xml_safe_create_element(doc, x_f_m_sys, 0.)
442         xml_safe_create_element(doc, x_f_m_wings, 0.)
443
444         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
445
446     def generate_output_xml(self):
447         # type: () -> str
448         _1 = np.zeros(self.n_wing_segments + 1)
449         _2 = np.zeros(self.n_wing_segments)
450         _3 = np.zeros((3, self.n_wing_segments + 1))
451
452         return WingObjectModel.write_data(0, 0, _1, _1, _1, _3, _1, _1, _2, _2, _2, _2, _2, 0,
    ↪  0, 0, 0, 0, 0, 0, _2)
453
454     def generate_partials_xml(self):
455         partials = Partials()
456
457         partials.declare_partials(x_ref_area, x_c)
458         partials.declare_partials(x_ref_area, x_b)
459         partials.declare_partials(x_ref_length, x_c)
460
461         for i in range(self.n_wing_segments + 1):
462             _x_sec = x_sec % i
463             x_trans = '/'.join([_x_sec, 'transformation'])
464
465             x_scaling = '/'.join([x_trans, 'scaling'])
466             partials.declare_partials('/'.join([x_scaling, 'x']), x_c)
467             partials.declare_partials('/'.join([x_scaling, 'y']), x_c)
468             partials.declare_partials('/'.join([x_scaling, 'z']), x_c)
469             partials.declare_partials('/'.join([x_scaling, 'z']), x_tc)
470
471             x_rotation = '/'.join([x_trans, 'rotation'])
472             if i != 0:
473                 partials.declare_partials('/'.join([x_rotation, 'y']), x_epsilon)
474             partials.declare_partials('/'.join([x_rotation, 'y']), x_incidence)
475
476             x_translation = '/'.join([x_trans, 'translation'])
477             partials.declare_partials('/'.join([x_translation, 'x']), x_c)
478             partials.declare_partials('/'.join([x_translation, 'y']), x_c)
479             partials.declare_partials('/'.join([x_translation, 'z']), x_c)
480
481             partials.declare_partials('/'.join([x_translation, 'x']), x_epsilon)
482             partials.declare_partials('/'.join([x_translation, 'y']), x_epsilon)
483             partials.declare_partials('/'.join([x_translation, 'z']), x_epsilon)
484
485             partials.declare_partials('/'.join([x_translation, 'x']), x_incidence)
486             partials.declare_partials('/'.join([x_translation, 'y']), x_incidence)
487             partials.declare_partials('/'.join([x_translation, 'z']), x_incidence)
```

```
488
489             if i != 1:
490                 partials.declare_partials('/'.join([x_translation, 'x']), x_Gamma)
491                 partials.declare_partials('/'.join([x_translation, 'y']), x_Gamma)
492                 partials.declare_partials('/'.join([x_translation, 'z']), x_Gamma)
493
494                 partials.declare_partials('/'.join([x_translation, 'x']), x_Lambda)
495                 partials.declare_partials('/'.join([x_translation, 'y']), x_Lambda)
496                 partials.declare_partials('/'.join([x_translation, 'z']), x_Lambda)
497
498                 partials.declare_partials('/'.join([x_translation, 'x']), x_b)
499                 partials.declare_partials('/'.join([x_translation, 'y']), x_b)
500                 partials.declare_partials('/'.join([x_translation, 'z']), x_b)
501
502         for i in range(self.n_wing_segments):
503             partials.declare_partials(x_fs_web_t % (i, i), x_t_fs)
504             partials.declare_partials(x_fs_lowerCap_t % (i, i), x_t_bs)
505             partials.declare_partials(x_fs_upperCap_t % (i, i), x_t_ts)
506             partials.declare_partials(x_rs_web_t % (i, i), x_t_rs)
507             partials.declare_partials(x_rs_lowerCap_t % (i, i), x_t_bs)
508             partials.declare_partials(x_rs_upperCap_t % (i, i), x_t_ts)
509
510             partials.declare_partials(x_fs_r_xsi % (i, i), x_xsi_fs)
511             partials.declare_partials(x_rs_r_xsi % (i, i), x_xsi_rs)
512             partials.declare_partials(x_fs_t_xsi % (i, i), x_xsi_fs)
513             partials.declare_partials(x_rs_t_xsi % (i, i), x_xsi_rs)
514
515         x_md = '{0}[@uID="{1}"]/mass'
516         x_m = '/'.join(['{0}', x_md.format('massDescription', '{0}')])
517         x_mass = '/'.join([x_mbd, x_m])
518         x_moem = x_mass.format('mOEM')
519         x_mem = '/'.join([x_moem[:-34], x_m.format('mEM')])
520         x_sys = '/'.join([x_mem[:-33], x_m.format('mSystems')])
521
522         x_m_struct = '/'.join([x_mem[:-33], x_m.format('mStructure')])
523         x_m_wings = '/'.join([x_m_struct[:-40], x_m.format('mWingsStructure')])
524         _x_m_wing = '/'.join([x_m_wings[:-45], x_m.format('mWingStructure')])
525
526         partials.declare_partials('/'.join([x_mbd,
    ↪ 'fuel/massDescription[@uID="mFuel"]/mass']), x_m_fuel)
527         partials.declare_partials('/'.join([x_mbd,
    ↪ 'payload/massDescription[@uID="mPayload"]/mass']), x_m_fuel)
528         partials.declare_partials(x_moem, [x_m_wing, x_m_fixed])
529         partials.declare_partials(x_mem, [x_m_wing, x_m_fixed])
530         partials.declare_partials(x_sys, [x_m_wing, x_m_fixed, x_f_m_sys])
531
532         partials.declare_partials(x_m_struct, [x_m_wing, x_m_fixed, x_f_m_sys])
533         partials.declare_partials(x_m_wings, [x_m_wing, x_f_m_wings])
534         partials.declare_partials(_x_m_wing, x_m_wing)
535
536         for i in range(self.n_wing_segments):
537             x_m_compseg = '/'.join([_x_m_wing[:-44], 'mComponentSegment[{:d}]'.format(i + 1)])
538
539             partials.declare_partials(
540                 '/'.join([x_m_compseg, x_md.format('massDescription',
    ↪ 'mWing_{:d}'.format(i))]),
541                 [x_c, x_xsi_fs, x_xsi_rs, x_b, x_t_skin, x_rho_skin, x_m_wing])
542
543             partials.declare_partials(
544                 '/'.join([x_m_compseg, 'mWingBox', x_md.format('massDescription',
    ↪ 'mWingbox_{:d}'.format(i))]),
545                 [x_c, x_xsi_fs, x_xsi_rs, x_b, x_m_wing])
546
547             partials.declare_partials(
548                 '/'.join([x_mSkins % (i + 1), 'mSkins', x_md.format('massDescription',
    ↪ 'mSkins_{:d}'.format(i))]),
549                 [x_c, x_xsi_fs, x_xsi_rs, x_b, x_t_skin, x_rho_skin])
550
551         x_des = '/'.join([x_mbd, 'designMasses'])
552         partials.declare_partials('/'.join([x_des, x_md.format('mMLM', 'mMLM')]), x_m_mlw)
553         partials.declare_partials('/'.join([x_des, x_md.format('mMRM', 'mMRM')]),
```

```
554                                 [x_m_fuel, x_m_wing, x_m_fixed, x_m_payload])
555             partials.declare_partials('/'.join([x_des, x_md.format('mTOM', 'mTOM')]),
556                                 [x_m_fuel, x_m_wing, x_m_fixed, x_m_payload])
557             partials.declare_partials('/'.join([x_des, x_md.format('mZFM', 'mZFM')]),
558                                 [x_m_wing, x_m_fixed, x_m_payload])
559
560             return partials.get_string()
561
562         @staticmethod
563         def read_data(file):
564             tree = etree.parse(file)
565
566             c = np.array(tree.xpath(x_c)[0].text.split(';'), dtype=float)
567             tc = np.array(tree.xpath(x_tc)[0].text.split(';'), dtype=float)
568             epsilon = np.array(tree.xpath(x_epsilon)[0].text.split(';'), dtype=float)
569             b = np.array(tree.xpath(x_b)[0].text.split(';'), dtype=float)
570             sweep = np.array(tree.xpath(x_Lambda)[0].text.split(';'), dtype=float)
571             dihed = np.array(tree.xpath(x_Gamma)[0].text.split(';'), dtype=float)
572             xsi_fs = np.array(tree.xpath(x_xsi_fs)[0].text.split(';'), dtype=float)
573             xsi_rs = np.array(tree.xpath(x_xsi_rs)[0].text.split(';'), dtype=float)
574             t_fs = np.array(tree.xpath(x_t_fs)[0].text.split(';'), dtype=float)
575             t_rs = np.array(tree.xpath(x_t_rs)[0].text.split(';'), dtype=float)
576             t_ts = np.array(tree.xpath(x_t_ts)[0].text.split(';'), dtype=float)
577             t_bs = np.array(tree.xpath(x_t_bs)[0].text.split(';'), dtype=float)
578             incidence = np.array(tree.xpath(x_incidence)[0].text.split(';'), dtype=float)
579             t_skin = np.array(tree.xpath(x_t_skin)[0].text.split(';'), dtype=float)
580             rho_skin = np.array(tree.xpath(x_rho_skin)[0].text.split(';'), dtype=float)
581
582             m_fuel = tree.xpath(x_m_fuel)
583             if not m_fuel:
584                 m_fuel = 0.  # float(tree.xpath(x_m_fuel_init)[0].text)
585             else:
586                 m_fuel = float(m_fuel[0].text)
587
588             m_wing = tree.xpath(x_m_wing)
589             if not m_wing:
590                 m_wing = 0.  # float(tree.xpath(x_m_wing_init)[0].text)
591             else:
592                 m_wing = float(m_wing[0].text)
593
594             m_fixed = float(tree.xpath(x_m_fixed)[0].text)
595             m_payload = float(tree.xpath(x_m_payload)[0].text)
596             m_mlw = float(tree.xpath(x_m_mlw)[0].text)
597             f_m_sys = float(tree.xpath(x_f_m_sys)[0].text)
598             f_m_wings = float(tree.xpath(x_f_m_wings)[0].text)
599
600             return c, tc, epsilon, b, sweep, dihed, \
601                    xsi_fs, xsi_rs, t_fs, t_rs, t_ts, t_bs, \
602                    incidence, t_skin, rho_skin, \
603                    m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys, f_m_wings
604
605         @staticmethod
606         def write_data(*args):
607             # type: (*(str, str, Sized)) -> str
608             """Utility method that writes all output variables to the given XML file.
609
610             Parameters
611             ----------
612                 *args
613                     (S_ref, c_ref, c, t/c, twists, x_LE,
614                     xsi_fs, xsi_rs, t_fs, t_rs, t_ts, t_bs,
615                     m_skin, m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys, f_m_wings,
    ↪ m_wingbox)
616
617             Returns
618             -------
619                 str
620                     String representing the output XML file.
621             """
622             s_ref, c_ref, c, tc, twists, x_le, xsi_fs, xsi_rs, t_fs, t_rs, t_ts, t_bs, m_skin,
    ↪ m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys, f_m_wings, m_wingbox = args
```

```
623
624         root = etree.Element('cpacs')
625         doc = etree.ElementTree(root)
626
627         add_cpacs_header(doc, 'WingObjectModel', 'Automatically generated from python', '1.0',
    ↪ 'Wing object model',
628                          '2.3')
629
630         xml_safe_create_element(doc, '/'.join([x_model, 'name']), 'Wing Definition')
631
632         xml_safe_create_element(doc, x_ref_area, s_ref)
633         xml_safe_create_element(doc, x_ref_length, c_ref)
634         add_point(doc, x_ref, 'point', 0, 0, 0)
635
636         xml_safe_create_element(doc, x_wing)
637         add_transform(doc, x_wing)
638
639         for i in range(len(c)):
640             _x_sec = x_sec % i
641             elem_sec = xml_safe_create_element(doc, _x_sec)
642             etree.SubElement(elem_sec, 'name').text = 'Section %d' % i
643             add_transform(doc, elem_sec, (c[i], 1, tc[i] * c[i]), (0, np.rad2deg(twists[i]),
    ↪ 0), tuple(x_le[:, i]))
644
645             elem_elems = etree.SubElement(elem_sec, 'elements')
646             elem_elem = etree.SubElement(elem_elems, 'element', {'uID': 'elem_%d' % i})
647             etree.SubElement(elem_elem, 'name').text = 'Element %d' % i
648             etree.SubElement(elem_elem, 'airfoilUID').text = 'af'
649             add_transform(doc, elem_elem)
650
651             x_pos = '/'.join([x_wing, r"positionings/positioning"])
652             if i != 0:
653                 x_pos += '[%d]' % (i + 1)
654             pos = xml_safe_create_element(doc, x_pos)
655             etree.SubElement(pos, 'length').text = str(0)
656             etree.SubElement(pos, 'sweepAngle').text = str(0)
657             etree.SubElement(pos, 'dihedralAngle').text = str(0)
658             if i != 0:
659                 etree.SubElement(pos, 'fromSectionUID').text = 'sec_%d' % (i - 1)
660             etree.SubElement(pos, 'toSectionUID').text = 'sec_%d' % i
661
662         for i in range(len(c) - 1):
663             x_seg = '/'.join([x_wing, r"segments/segment[@uID='seg_%d']" % i])
664             seg = xml_safe_create_element(doc, x_seg)
665             etree.SubElement(seg, 'name').text = 'Segment %d' % i
666             etree.SubElement(seg, 'fromElementUID').text = 'elem_%d' % i
667             etree.SubElement(seg, 'toElementUID').text = 'elem_%d' % (i + 1)
668
669             compseg = xml_safe_create_element(doc, x_compseg % i)
670             etree.SubElement(compseg, 'name').text = 'ComponentSegment %d' % i
671             etree.SubElement(compseg, 'fromElementUID').text = 'elem_%d' % i
672             etree.SubElement(compseg, 'toElementUID').text = 'elem_%d' % (i + 1)
673
674             add_spar_position(doc, x_sparposs % i, 'fs_%d_r' % i, 'elem_%d' % i, xsi_fs[i])
675             add_spar_position(doc, x_sparposs % i, 'fs_%d_t' % i, 'elem_%d' % (i + 1),
    ↪ xsi_fs[i + 1])
676             add_spar_position(doc, x_sparposs % i, 'rs_%d_r' % i, 'elem_%d' % i, xsi_rs[i])
677             add_spar_position(doc, x_sparposs % i, 'rs_%d_t' % i, 'elem_%d' % (i + 1),
    ↪ xsi_rs[i + 1])
678
679             add_spar_segment(doc, x_sparsegs % i, 'fs_%d' % i, 'fs_%d_r' % i, 'fs_%d_t' % i,
680                              'mat_al', t_fs[i], t_ts[i], t_bs[i])
681             add_spar_segment(doc, x_sparsegs % i, 'rs_%d' % i, 'rs_%d_r' % i, 'rs_%d_t' % i,
682                              'mat_al', t_rs[i], t_ts[i], t_bs[i])
683
684         # x_mbd = '/'.join([x_model, 'analyses/massBreakdown'])
685
686         m_empty = m_wing + m_fixed
687         m_sys = m_empty * f_m_sys
688         m_struct = m_empty - m_sys
689         m_wings = m_struct * f_m_wings
```

```
690         m_wing = m_wing * (1. - f_m_sys)
691
692         m_mzf = m_empty + m_payload
693         m_mto = m_mzf + m_fuel
694         m_mrm = m_mto * 1.01
695
696         add_mass(doc, x_mbd, 'fuel', m_fuel, 'mFuel')
697         add_mass(doc, x_mbd, 'payload', m_payload, 'mPayload')
698         m_oem = add_mass(doc, x_mbd, 'mOEM', m_empty)
699         m_em = add_mass(doc, m_oem, 'mEM', m_empty)
700         add_mass(doc, m_em, 'mSystems', m_sys)
701
702         m_structure = add_mass(doc, m_em, 'mStructure', m_struct)
703         m_wings = add_mass(doc, m_structure, 'mWingsStructure', m_wings)
704         m_wing = add_mass(doc, m_wings, 'mWingStructure', m_wing)
705
706         for i in range(0, len(c) - 1):
707             x_m_compseg = '/'.join([doc.getpath(m_wing), 'mComponentSegment[%d]' % (i + 1)])
708             m_compseg = xml_safe_create_element(doc, x_m_compseg)
709             add_mass_description(doc, m_compseg, 'massDescription', m_skin[i] + m_wingbox[i],
    ↪ 'mWing_%d' % i)
710             add_mass(doc, m_compseg, 'mWingBox', m_wingbox[i], 'mWingbox_%d' % i)
711             add_mass(doc, x_mSkins % (i + 1), 'mSkins', m_skin[i], 'mSkins_%d' % i)
712
713         m_des = etree.SubElement(m_oem.getparent(), 'designMasses')
714         add_mass_description(doc, m_des, 'mMLM', m_mlw)
715         add_mass_description(doc, m_des, 'mMRM', m_mrm)
716         add_mass_description(doc, m_des, 'mTOM', m_mto)
717         add_mass_description(doc, m_des, 'mZFM', m_mzf)
718
719         x_af = r"/cpacs/vehicles/profiles/wingAirfoils/wingAirfoil[@uID='af']"
720         af = xml_safe_create_element(doc, x_af)
721         etree.SubElement(af, 'name').text = 'Airfoil'
722         point_list = etree.SubElement(af, 'pointList')
723
724         etree.SubElement(point_list, 'x', {'mapType': 'vector'}).text = ';'.join([str(_) for _
    ↪ in af_x])
725         etree.SubElement(point_list, 'y', {'mapType': 'vector'}).text = ';'.join([str(_) for _
    ↪ in len(af_x) * [0.0]])
726         etree.SubElement(point_list, 'z', {'mapType': 'vector'}).text = ';'.join([str(_) for _
    ↪ in af_z])
727
728         mat = xml_safe_create_element(doc, '/'.join([x_vehicles,
    ↪ r"materials/material[@uID='mat_al']"]))
729         etree.SubElement(mat, 'name').text = 'Al7075A'
730         etree.SubElement(mat, 'rho').text = str(2180)
731         etree.SubElement(mat, 'k11').text = str(71.7E9)
732         etree.SubElement(mat, 'k12').text = str(26.9E9)
733         etree.SubElement(mat, 'sig11').text = str(572E6)
734         etree.SubElement(mat, 'sig12').text = str(331E6)
735
736         return etree.tostring(doc, encoding='utf-8', pretty_print=True, xml_declaration=True)
737
738     @staticmethod
739     def execute(in_file, out_file='WOM-output-loc.xml'):
740         """Translate the reduced model parameters to CPACS format."""
741         c, tc, epsilon, b, sweep, dihed, \
742             xsi_fs, xsi_rs, t_fs, t_rs, t_ts, t_bs, \
743             incidence, t_skin, rho_skin, \
744             m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys, f_m_wings =
    ↪ WingObjectModel.read_data(in_file)
745
746         n_wing_segments = len(b)
747
748         s_ref = sum(b * (c[:-1] + c[1:]))
749
750         c_ref = c[:]
751         for i in range(0, n_wing_segments):
752             c_ref = 2. / 3. * (c_ref[:-1] ** 2 + c_ref[:-1] * c_ref[1:] + c_ref[1:] ** 2) /
    ↪ (c_ref[1:] + c_ref[:-1])
753         c_ref = c_ref[0]
```

```
754
755         dx_c4 = np.zeros((3, n_wing_segments))
756         dx_c4[1, :] = b[:]
757
758         x_c4 = np.zeros((3, n_wing_segments + 1))
759         c_sweep, s_sweep = np.cos(sweep), np.sin(sweep)
760         c_dihed, s_dihed = np.cos(dihed), np.sin(dihed)
761
762         for i in range(n_wing_segments):
763             rot_sweep = np.matrix([(c_sweep[i], s_sweep[i], 0), (-s_sweep[i], c_sweep[i], 0),
    ↪ (0, 0, 1)])
764             rot_dihed = np.matrix([(1, 0, 0), (0, c_dihed[i], -s_dihed[i]), (0, s_dihed[i],
    ↪ c_dihed[i])])
765             x_c4[:, i + 1] = np.matmul(rot_dihed * rot_sweep, dx_c4[:, i]) + x_c4[:, i]
766
767         dx_le = np.zeros((3, n_wing_segments + 1))
768         dx_le[0, :] = -.25 * c
769
770         x_le = np.zeros((3, n_wing_segments + 1))
771         twists = np.concatenate(([0.], epsilon)) + incidence
772         c_twist, s_twist = np.cos(twists), np.sin(twists)
773         for i in range(n_wing_segments + 1):
774             rot_twist = np.matrix([(c_twist[i], 0, s_twist[i]), (0, 1, 0), (-s_twist[i], 0,
    ↪ c_twist[i])])
775             x_le[:, i] = np.matmul(rot_twist, dx_le[:, i]) + x_c4[:, i]
776
777         length_out = c * (1. - xsi_rs + xsi_fs)
778         area_out = 0.5 * (length_out[:-1] + length_out[1:]) * b
779         m_skin = 2. * area_out * t_skin * rho_skin
780
781         m_wingbox = m_wing * area_out / sum(area_out)
782
783         xml = WingObjectModel.write_data(s_ref, c_ref,
784                                          c, tc, twists,
785                                          x_le, xsi_fs, xsi_rs,
786                                          t_fs, t_rs, t_ts,
787                                          t_bs, m_skin,
788                                          m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys,
    ↪ f_m_wings, m_wingbox)
789         with open(out_file, 'w') as f:
790             f.write(xml)
791
792     @staticmethod
793     def linearize(in_file, partials_file):
794         c, tc, epsilon, b, sweep, dihed, \
795             xsi_fs, xsi_rs, t_fs, t_rs, t_ts, t_bs, \
796             incidence, t_skin, rho_skin, \
797             m_fuel, m_wing, m_fixed, m_payload, m_mlw, f_m_sys, f_m_wings =
    ↪ WingObjectModel.read_data(in_file)
798
799         n_ws = len(b)
800         partials = Partials()
801
802         # Reference values
803         dSref_dc = np.zeros(len(c))
804         dSref_dc[:-1] += b
805         dSref_dc[1:] += b
806         partials.declare_partials(x_ref_area, x_c, dSref_dc)
807
808         dSref_db = c[:-1] + c[1:]
809         partials.declare_partials(x_ref_area, x_b, dSref_db)
810
811         dcref_dc = np.eye(n_ws + 1, n_ws + 1)
812         c_ref = c[:]
813         for i in range(0, n_ws):
814             dcref_dc = 2. / 3. * (
815                 (c_ref[:-1] + c_ref[1:])
816                 * (2. * c_ref[:-1] * dcref_dc[:, :-1] + c_ref[:-1] * dcref_dc[:, :-1]
817                    + c_ref[1:] * dcref_dc[:, 1:] + 2 * c_ref[1:] * dcref_dc[:, 1:])
818                 - (c_ref[:-1] ** 2 + c_ref[:-1] * c_ref[1:] + c_ref[1:] ** 2) * (dcref_dc[:,
    ↪ :-1] + dcref_dc[:, 1:])
```

```
            ) / (c_ref[:-1] + c_ref[1:]) ** 2
            c_ref = 2. / 3. * (c_ref[:-1] ** 2 + c_ref[:-1] * c_ref[1:] + c_ref[1:] ** 2) /
    (c_ref[1:] + c_ref[:-1])
        partials.declare_partials(x_ref_length, x_c, dcref_dc.flatten())

        # Geometry
        de = np.concatenate(([0.], np.ones(len(epsilon))))
        twist = np.concatenate(([0.], epsilon)) + incidence

        c_twist, s_twist = np.cos(twist), np.sin(twist)

        dx_le = np.zeros((3, n_ws + 1))
        dx_le[0, :] = -.25 * c

        dct_di, dst_di = -np.sin(twist), np.cos(twist)
        dct_de, dst_de = dct_di * de, dst_di * de

        c_sweep, s_sweep = np.cos(sweep), np.sin(sweep)
        c_dihed, s_dihed = np.cos(dihed), np.sin(dihed)

        dcs_ds, dss_ds = -np.sin(sweep), np.cos(sweep)
        dcd_dd, dsd_dd = -np.sin(dihed), np.cos(dihed)

        dx_c4 = np.zeros((3, n_ws))
        dx_c4[1, :] = b[:]

        dxle_dd = np.zeros(3)
        dxle_ds = np.zeros(3)
        dxle_db = np.zeros(3)

        for i in range(n_ws + 1):
            _x_sec = x_sec % i
            x_trans = '/'.join([_x_sec, 'transformation'])

            x_scaling = '/'.join([x_trans, 'scaling'])
            dc = np.zeros(n_ws + 1)
            dc[i] = 1.
            partials.declare_partials('/'.join([x_scaling, 'x']), x_c, dc)
            partials.declare_partials('/'.join([x_scaling, 'z']), x_c, dc * tc)
            partials.declare_partials('/'.join([x_scaling, 'z']), x_tc, dc * c)

            x_rotation = '/'.join([x_trans, 'rotation'])
            if i != 0:
                de = np.zeros(2)
                de[i - 1] = 180./np.pi
                partials.declare_partials('/'.join([x_rotation, 'y']), x_epsilon, de)
            partials.declare_partials('/'.join([x_rotation, 'y']), x_incidence, 180./np.pi)

            rot_twist = np.matrix([(c_twist[i], 0, s_twist[i]), (0, 1, 0), (-s_twist[i], 0,
    c_twist[i])])
            drt_di = np.matrix([(dct_di[i], 0, dst_di[i]), (0, 1, 0), (-dst_di[i], 0,
    dct_di[i])])

            dxle_dc = np.asarray(np.matmul(rot_twist, dx_le[:, i] / c[i])).flatten()
            dxle_di = np.asarray(np.matmul(drt_di, dx_le[:, i])).flatten()

            d = np.zeros(n_ws + 1)
            d[i] = 1.

            x_translation = '/'.join([x_trans, 'translation'])
            partials.declare_partials('/'.join([x_translation, 'x']), x_c, dxle_dc[0] * d)
            partials.declare_partials('/'.join([x_translation, 'y']), x_c, dxle_dc[1] * d)
            partials.declare_partials('/'.join([x_translation, 'z']), x_c, dxle_dc[2] * d)

            if i != 0:
                drt_de = np.matrix([(dct_de[i], 0, dst_de[i]), (0, 1, 0), (-dst_de[i], 0,
    dct_de[i])])
                dxle_de = np.asarray(np.matmul(drt_de, dx_le[:, i])).flatten()

                d = np.zeros(n_ws)
                d[i - 1] = 1.
```

```
886
887                 partials.declare_partials('/'.join([x_translation, 'x']), x_epsilon,
    ↪ dxle_de[0] * d)
888                 partials.declare_partials('/'.join([x_translation, 'y']), x_epsilon,
    ↪ dxle_de[1] * d)
889                 partials.declare_partials('/'.join([x_translation, 'z']), x_epsilon,
    ↪ dxle_de[2] * d)
890
891             partials.declare_partials('/'.join([x_translation, 'x']), x_incidence, dxle_di[0])
892             partials.declare_partials('/'.join([x_translation, 'y']), x_incidence, dxle_di[1])
893             partials.declare_partials('/'.join([x_translation, 'z']), x_incidence, dxle_di[2])
894
895             if i < n_ws:
896                 d = np.zeros(n_ws)
897                 d[i] = 1.
898
899                 rot_sweep = np.matrix([(c_sweep[i], s_sweep[i], 0), (-s_sweep[i], c_sweep[i],
    ↪ 0), (0, 0, 1)])
900                 rot_dihed = np.matrix([(1, 0, 0), (0, c_dihed[i], -s_dihed[i]), (0,
    ↪ s_dihed[i], c_dihed[i])])
901
902                 drs_ds = np.matrix([(dcs_ds[i], dss_ds[i], 0), (-dss_ds[i], dcs_ds[i], 0), (0,
    ↪ 0, 1)])
903                 drd_dd = np.matrix([(1, 0, 0), (0, dcd_dd[i], -dsd_dd[i]), (0, dsd_dd[i],
    ↪ dcd_dd[i])])
904
905                 dxle_dd += np.asarray(np.matmul(drd_dd * rot_sweep, dx_c4[:, i])).flatten()
906                 dxle_ds += np.asarray(np.matmul(rot_dihed * drs_ds, dx_c4[:, i])).flatten()
907                 dxle_db += np.asarray(np.matmul(rot_dihed * rot_sweep, dx_c4[:, i] /
    ↪ b[i])).flatten()
908
909                 partials.declare_partials('/'.join([x_translation, 'x']), x_Gamma, dxle_dd[0]
    ↪ * d)
910                 partials.declare_partials('/'.join([x_translation, 'y']), x_Gamma, dxle_dd[1]
    ↪ * d)
911                 partials.declare_partials('/'.join([x_translation, 'z']), x_Gamma, dxle_dd[2]
    ↪ * d)
912
913                 partials.declare_partials('/'.join([x_translation, 'x']), x_Lambda, dxle_ds[0]
    ↪ * d)
914                 partials.declare_partials('/'.join([x_translation, 'y']), x_Lambda, dxle_ds[1]
    ↪ * d)
915                 partials.declare_partials('/'.join([x_translation, 'z']), x_Lambda, dxle_ds[2]
    ↪ * d)
916
917                 partials.declare_partials('/'.join([x_translation, 'x']), x_b, dxle_db[0] * d)
918                 partials.declare_partials('/'.join([x_translation, 'y']), x_b, dxle_db[1] * d)
919                 partials.declare_partials('/'.join([x_translation, 'z']), x_b, dxle_db[2] * d)
920
921         # Structures
922         for i in range(n_ws):
923             d = np.zeros(2)
924             d[i] = 1.
925
926             partials.declare_partials(x_fs_web_t % (i, i), x_t_fs, d)
927             partials.declare_partials(x_fs_lowerCap_t % (i, i), x_t_bs, d)
928             partials.declare_partials(x_fs_upperCap_t % (i, i), x_t_ts, d)
929             partials.declare_partials(x_rs_web_t % (i, i), x_t_rs, d)
930             partials.declare_partials(x_rs_lowerCap_t % (i, i), x_t_bs, d)
931             partials.declare_partials(x_rs_upperCap_t % (i, i), x_t_ts, d)
932
933             d = np.zeros(3)
934             d[i] = 1.
935
936             partials.declare_partials(x_fs_r_xsi % (i, i), x_xsi_fs, d)
937             partials.declare_partials(x_rs_r_xsi % (i, i), x_xsi_rs, d)
938
939             d[i] = 0.
940             d[i + 1] = 1.
941
942             partials.declare_partials(x_fs_t_xsi % (i, i), x_xsi_fs, d)
```

```
943             partials.declare_partials(x_rs_t_xsi % (i, i), x_xsi_rs, d)
944
945         # Weights
946         length_out = c * (1. - xsi_rs + xsi_fs)
947         area_out = 0.5 * (length_out[:-1] + length_out[1:]) * b
948         m_skin = 2. * area_out * t_skin * rho_skin
949
950         m_wingbox = m_wing * area_out / sum(area_out)
951
952         m_empty = m_wing + m_fixed
953         m_sys = m_empty * f_m_sys
954         m_struct = m_empty - m_sys
955         m_wing = m_wing * (1. - f_m_sys)
956
957         x_md = '{0}[@uID="{1}"]/mass'
958         x_m = '/'.join(['{0}', x_md.format('massDescription', '{0}')])
959         x_mass = '/'.join([x_mbd, x_m])
960         x_moem = x_mass.format('mOEM')
961         x_mem = '/'.join([x_moem[:-34], x_m.format('mEM')])
962         x_sys = '/'.join([x_mem[:-33], x_m.format('mSystems')])
963
964         x_m_struct = '/'.join([x_mem[:-33], x_m.format('mStructure')])
965         x_m_wings = '/'.join([x_m_struct[:-40], x_m.format('mWingsStructure')])
966         _x_m_wing = '/'.join([x_m_wings[:-45], x_m.format('mWingStructure')])
967
968         partials.declare_partials('/'.join([x_mbd,
    ↪ 'fuel/massDescription[@uID="mFuel"]/mass']), x_m_fuel, 1.)
969         partials.declare_partials('/'.join([x_mbd,
    ↪ 'payload/massDescription[@uID="mPayload"]/mass']), x_m_fuel, 1.)
970         partials.declare_partials(x_moem, [x_m_wing, x_m_fixed], [1., 1.])
971         partials.declare_partials(x_mem, [x_m_wing, x_m_fixed], [1., 1.])
972         partials.declare_partials(x_sys, [x_m_wing, x_m_fixed, x_f_m_sys], [f_m_sys, f_m_sys,
    ↪ m_empty])
973
974         partials.declare_partials(x_m_struct, [x_m_wing, x_m_fixed, x_f_m_sys], [1. - f_m_sys,
    ↪ 1. - f_m_sys, -m_empty])
975         partials.declare_partials(x_m_wings, [x_m_wing, x_f_m_wings], [f_m_wings, m_wing])
976         partials.declare_partials(_x_m_wing, x_m_wing, 1.)
977
978         dlout_dc = 1. - xsi_fs + xsi_rs
979         dlout_dxsifs = -c
980         dlout_dxsirs = c
981
982         dAout_db = area_out / b
983         dmskin_db = 2 * dAout_db * t_skin * rho_skin
984
985         dmskin_dtskin = m_skin / t_skin
986         dmskin_drhoskin = m_skin / rho_skin
987
988         dmwingbox_db = m_wing * (sum(area_out) * dAout_db - area_out * sum(dAout_db)) /
    ↪ sum(area_out) ** 2
989         dmwingbox_dmwing = m_wingbox / m_wing
990
991         dmcs_db = dmskin_db + dmwingbox_db
992         dmcs_dtskin = dmskin_dtskin
993         dmcs_drhoskin = dmskin_drhoskin
994         dmcs_dmwing = dmwingbox_dmwing
995
996         dAout_dc = np.zeros((n_ws, n_ws + 1))
997         dAout_dxsifs = np.zeros((n_ws, n_ws + 1))
998         dAout_dxsirs = np.zeros((n_ws, n_ws + 1))
999
1000        dmskin_dc = np.zeros((n_ws, n_ws + 1))
1001        dmskin_dxsifs = np.zeros((n_ws, n_ws + 1))
1002        dmskin_dxsirs = np.zeros((n_ws, n_ws + 1))
1003
1004        dmwingbox_dc = np.zeros((n_ws, n_ws + 1))
1005        dmwingbox_dxsifs = np.zeros((n_ws, n_ws + 1))
1006        dmwingbox_dxsirs = np.zeros((n_ws, n_ws + 1))
1007
1008        for i in range(n_ws):
```

```
1009            d = np.zeros(n_ws + 1)
1010            d[i] = 1.
1011            d[i + 1] = 1.
1012
1013            dAout_dc[i] = .5 * b[i] * dlout_dc * d
1014            dAout_dxsifs[i] = .5 * b[i] * dlout_dxsifs * d
1015            dAout_dxsirs[i] = .5 * b[i] * dlout_dxsirs * d
1016
1017            dmskin_dc[i] = 2 * dAout_dc[i] * t_skin * rho_skin
1018            dmskin_dxsifs[i] = 2 * dAout_dxsifs[i] * t_skin * rho_skin
1019            dmskin_dxsirs[i] = 2 * dAout_dxsirs[i] * t_skin * rho_skin
1020
1021            dmwingbox_dc[i] = m_wing * (sum(area_out) * dAout_dc[i] - area_out[i] *
    ↪ np.sum(dAout_dc, 0)) / sum(area_out) ** 2
1022            dmwingbox_dxsifs[i] = m_wing * (sum(area_out) * dAout_dxsifs[i] - area_out[i] *
    ↪ np.sum(dAout_dxsifs, 0)) / sum(area_out) ** 2
1023            dmwingbox_dxsirs[i] = m_wing * (sum(area_out) * dAout_dxsirs[i] - area_out[i] *
    ↪ np.sum(dAout_dxsirs, 0)) / sum(area_out) ** 2
1024
1025            dmcs_dc = dmskin_dc[i] + dmwingbox_dc[i]
1026            dmcs_dxsifs = dmskin_dxsifs[i] + dmwingbox_dxsifs[i]
1027            dmcs_dxsirs = dmskin_dxsirs[i] + dmwingbox_dxsirs[i]
1028
1029           x_m_compseg = '/'.join([_x_m_wing[:-44], 'mComponentSegment[{:d}]'.format(i + 1)])
1030
1031            partials.declare_partials(
1032                '/'.join([x_m_compseg, x_md.format('massDescription',
    ↪ 'mWing_{:d}'.format(i))]),
1033                [x_c, x_xsi_fs, x_xsi_rs, x_b, x_t_skin, x_rho_skin, x_m_wing],
1034                [dmcs_dc, dmcs_dxsifs, dmcs_dxsirs, dmcs_db[i], dmcs_dtskin[i],
    ↪ dmcs_drhoskin[i], dmcs_dmwing[i]]
1035            )
1036
1037            partials.declare_partials(
1038                '/'.join([x_m_compseg, 'mWingBox', x_md.format('massDescription',
    ↪ 'mWingbox_{:d}'.format(i))]),
1039                [x_c, x_xsi_fs, x_xsi_rs, x_b, x_m_wing],
1040                [dmwingbox_dc[i], dmwingbox_dxsifs[i], dmwingbox_dxsirs[i], dmwingbox_db[i],
    ↪ dmwingbox_dmwing[i]]
1041            )
1042
1043            partials.declare_partials(
1044                '/'.join([x_mSkins % (i + 1), 'mSkins', x_md.format('massDescription',
    ↪ 'mSkins_{:d}'.format(i))]),
1045                [x_c, x_xsi_fs, x_xsi_rs, x_b, x_t_skin, x_rho_skin],
1046                [dmskin_dc[i], dmskin_dxsifs[i], dmskin_dxsirs[i], dmskin_db[i],
    ↪ dmskin_dtskin[i], dmskin_drhoskin[i]]
1047            )
1048
1049        x_des = '/'.join([x_mbd, 'designMasses'])
1050        partials.declare_partials('/'.join([x_des, x_md.format('mMLM', 'mMLM')]), x_m_mlw, 1.)
1051        partials.declare_partials('/'.join([x_des, x_md.format('mMRM', 'mMRM')]),
1052                                  [x_m_fuel, x_m_wing, x_m_fixed, x_m_payload],
1053                                  [1.01, 1.01, 1.01, 1.01])
1054        partials.declare_partials('/'.join([x_des, x_md.format('mTOM', 'mTOM')]),
1055                                  [x_m_fuel, x_m_wing, x_m_fixed, x_m_payload],
1056                                  [1., 1., 1., 1.])
1057        partials.declare_partials('/'.join([x_des, x_md.format('mZFM', 'mZFM')]),
1058                                  [x_m_wing, x_m_fixed, x_m_payload],
1059                                  [1., 1., 1.])
1060
1061        partials.write(partials_file)
```

Code frament A.44: Code of the Python module containing the wing object model discipline.

#### A.5.2.3.6 XPaths

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
```

```
"""
Copyright 2017 D. de Vries

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

This file contains all the XPaths and utility string constants used by the dAEDalus disciplines.
"""
from __future__ import absolute_import, division, print_function

sigma_names = ['sigma_fs', 'sigma_rs', 'sigma_ts', 'sigma_bs']

""" CPACS """
x_vehicles = '/cpacs/vehicles'
x_model = x_vehicles + '/aircraft/model[@uID="model"]'
x_ref = '/'.join([x_model, 'reference'])
x_wing = '/'.join([x_model, 'wings/wing[@symmetry="x-z-plane"][@uID="wing"]'])
x_sec = '/'.join([x_wing, 'sections/section[@uID="sec_%d"]'])
x_elem = '/'.join([x_sec, 'elements/element[@uID="elem_%d"]'])
x_mbd = '/'.join([x_model, 'analyses/massBreakdown'])
x_global = '/'.join([x_model, 'global'])
x_perf = '/'.join([x_global, 'performanceTargets'])

x_ref_area = '/'.join([x_ref, 'area'])
x_ref_length = '/'.join([x_ref, 'length'])
x_compseg = '/'.join([x_wing, 'componentSegments/componentSegment[@uID="compSeg_%d"]'])
x_struct = '/'.join([x_compseg, 'structure'])
x_sparposs = '/'.join([x_struct, 'spars/sparPositions'])
x_fs_r_xsi = '/'.join([x_sparposs, 'sparPosition[@uID="fs_%d_r"]/xsi'])
x_fs_t_xsi = '/'.join([x_sparposs, 'sparPosition[@uID="fs_%d_t"]/xsi'])
x_rs_r_xsi = '/'.join([x_sparposs, 'sparPosition[@uID="rs_%d_r"]/xsi'])
x_rs_t_xsi = '/'.join([x_sparposs, 'sparPosition[@uID="rs_%d_t"]/xsi'])
x_sparsegs = '/'.join([x_struct, 'spars/sparSegments'])
x_fs_web_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="fs_%d"]/sparCrossSection/web1/material/thickness'])
x_fs_lowerCap_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="fs_%d"]/sparCrossSection/lowerCap/material/thickness'])
x_fs_upperCap_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="fs_%d"]/sparCrossSection/upperCap/material/thickness'])
x_rs_web_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="rs_%d"]/sparCrossSection/web1/material/thickness'])
x_rs_lowerCap_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="rs_%d"]/sparCrossSection/lowerCap/material/thickness'])
x_rs_upperCap_t = '/'.join([x_sparsegs,
  'sparSegment[@uID="rs_%d"]/sparCrossSection/upperCap/material/thickness'])
x_mSkins = '/'.join([x_model, 'analyses/massBreakdown/mOEM/mEM/mStructure/mWingsStructure/'
                              'mWingStructure/mComponentSegment[%d]/mWingBox'])

""" Wing optimization problem """
x_opt = '/cpacs/toolspecific/wingOptimizationProblem'
x_planform = '/'.join([x_opt, 'planform'])
x_structure = '/'.join([x_opt, 'structure'])
x_reference = '/'.join([x_opt, 'reference'])

x_const = '/'.join([x_opt, 'constants'])
x_con = '/'.join([x_opt, 'constraints'])
x_obj = '/'.join([x_opt, 'objectives'])

x_c = '/'.join([x_planform, 'c'])
x_tc = '/'.join([x_planform, 'tc'])
x_epsilon = '/'.join([x_planform, 'epsilon'])
```

```
x_b = '/'.join([x_planform, 'b'])
x_Lambda = '/'.join([x_planform, 'Lambda'])
x_Gamma = '/'.join([x_planform, 'Gamma'])
x_incidence = '/'.join([x_planform, 'incidence'])

x_xsi_fs = '/'.join([x_structure, 'xsi_fs'])
x_xsi_rs = '/'.join([x_structure, 'xsi_rs'])
x_t_fs = '/'.join([x_structure, 't_fs'])
x_t_rs = '/'.join([x_structure, 't_rs'])
x_t_ts = '/'.join([x_structure, 't_ts'])
x_t_bs = '/'.join([x_structure, 't_bs'])
x_t_skin = '/'.join([x_structure, 't_skin'])

x_m_fixed = '/'.join([x_reference, 'm_fixed'])
x_m_payload = '/'.join([x_reference, 'm_payload'])
x_f_m_sys = '/'.join([x_reference, 'f_m_sys'])
x_f_m_wings = '/'.join([x_reference, 'f_m_wings'])
x_m_mlw = '/'.join([x_reference, 'm_MLW'])
x_m_mtow = '/'.join([x_reference, 'm_MTOW'])

x_SFC = '/'.join([x_reference, 'SFC'])
x_m_fuel_res = '/'.join([x_reference, 'm_fuel_res'])
x_CDfus = '/'.join([x_reference, 'C_D_fus'])
x_CDother = '/'.join([x_reference, 'C_D_other'])
x_R = '/'.join([x_reference, 'R'])

x_rho_skin = '/'.join([x_reference, 'rho_skin'])
x_sigma_yield = '/'.join([x_reference, 'sigma_yield'])
x_WS_init = '/'.join([x_reference, 'WS_init'])
x_CL_buffet = '/'.join([x_reference, 'C_L_buffet'])
x_m_wing_init = '/'.join([x_reference, 'm_wing_init'])
x_m_fuel_init = '/'.join([x_reference, 'm_fuel_init'])

x_con_sigmas = ['/'.join([x_con, 'con_' + sigma]) for sigma in sigma_names]
x_con_WS = '/'.join([x_con, 'con_WS'])
x_con_buffet = '/'.join([x_con, 'con_buffet'])

x_obj_m_fuel = '/'.join([x_obj, 'obj_m_fuel'])
x_obj_m_wing = '/'.join([x_obj, 'obj_m_wing'])


""" dAEDalus """
x_dAE = '/cpacs/toolspecific/dAEDalus'
x_m_wing = '/'.join([x_dAE, 'm_wing'])

x_loadcases = '/'.join([x_dAE, 'loadCases'])
x_loadcase = '/'.join([x_loadcases, 'loadCase[%d]'])

x_M = '/'.join([x_loadcase, 'M'])
x_H = '/'.join([x_loadcase, 'H'])
x_n = '/'.join([x_loadcase, 'n'])
x_CL = '/'.join([x_loadcase, 'C_L'])
x_CDf = '/'.join([x_loadcase, 'C_D_f'])
x_CDi = '/'.join([x_loadcase, 'C_D_i'])

x_grid_initial = ['/'.join([x_loadcase, 'initial_grid/' + component]) for component in ['x',
  ↪  'y', 'z']]
x_grid = ['/'.join([x_loadcase, 'deflected_grid/' + component]) for component in ['x', 'y',
  ↪  'z']]
x_grid_guess = ['/'.join([x_loadcase, 'guess_grid/' + component]) for component in ['x', 'y',
  ↪  'z']]

x_sigmas_in = ['/'.join([x_loadcase, sigma]) for sigma in sigma_names]
x_load_collector = '/'.join([x_dAE, 'load_collector'])
x_sigmas_out = ['/'.join([x_load_collector, sigma]) for sigma in sigma_names]

x_y_norm = '/'.join([x_loadcase, 'y_norm'])
x_l_norm = '/'.join([x_loadcase, 'l_norm'])

x_geom = '/'.join([x_loadcase, 'geometric_model'])
x_stru = '/'.join([x_loadcase, 'structural_model'])
```

```
135 x_aero = '/'.join([x_loadcase, 'aerodynamic_model'])
136
137 x_mle = '/'.join([x_loadcase, 'matlab_engine'])
138 x_ml_timeout = '/'.join([x_mle, 'timeout'])
139 x_ml_id = '/'.join([x_mle, 'id'])
140 x_ml_timestamp = '/'.join([x_mle, 'timestamp'])
141
142 """ FWE """
143 x_fwe = '/cpacs/toolspecific/fuel_weight_estimator'
144
145 x_CD = '/'.join([x_fwe, 'C_D'])
146 x_LD = '/'.join([x_fwe, 'L_D'])
147 x_m_fuel = '/'.join([x_fwe, 'm_fuel'])
148 x_fwe_CL = '/'.join([x_fwe, 'C_L'])
```

Code frament A.45: Code of the Python module containing all the XPaths of the disciplines.

# Bibliography


[1] AGILE. Agile - aircraft 3rd generation mdo for innovative collaboration of heterogeneous teams of experts, 2017. URL https://www.agile-project.eu/. Accessed: 20-11-2017.

[2] Jeremy Agte, de Weck, Olivier, Jaroslaw Sobieszczanski-Sobieski, Arendsen, Paul, Alan Morris, and Martin Spieck. MDO: assessment and direction for advancement—an opinion of one international group. *Structural and Multidisciplinary Optimization*, 40(1):17, 2009. ISSN 1615-1488. doi: 10.1007/s00158-009-0381-5. URL https://doi.org/10.1007/s00158-009-0381-5.

[3] B. Aigner, I. van Gent, G. La Rocca, and E. Stumpf. Using graph-based algorithms and data-driven documents for formulation and visualization of large mdo systems. In *6th CEAS Air & Space Conference Aerospace Europe*, volume Paper 173, Bucharest, Romania, 2017. URL https://www.agile-project.eu/cloud/index.php/s/mbdufb7vnnMRb3Y.

[4] E. Allison, I. Kroo, P. Sturdza, Y. Suzuki, and H. Martins-Rivas. Aircraft conceptual design with natural laminar flow. *27th Congress of the International Council of the Aeronautical Sciences 2010, ICAS 2010*, Vol. 1:pp. 428–436, 2010. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84878478319{&}partnerID=tZOtx3y1.

[5] American Airlines. Final approach/our fleet. *American Way*, 50(11):128–129, nov 2017. URL http://americanway.ink-live.com/html5/reader/production/default.aspx?pubname=&edid=81767840-2346-411d-83cc-6988af881d99.

[6] J. D. Anderson. Ludwig prandtl's boundary layer. *Physics Today*, Vol. 58(No. 12):pp. 42–48, dec 2005. doi: 10.1063/1.2169443. URL https://doi.org/10.1063%2F1.2169443.

[7] Flight Operations Support & Line Assistance. Getting to grips with aircraft performance. Internal manual, Customer Services, Airbus, jan 2002. URL https://www.skybrary.aero/bookshelf/books/2263.pdf. Accessed 22-11-2017.

[8] E. Baalbergen, A. Kanakis, and W. Vankan. A practical approach for coordination of multi-partner engineering jobs in the design of small aircraft. *CESAR Special Issue of Journal Czech Aerospace Proceedings / Letechký zpravodaj, Journal for Czech Aerospace Research*, Vol. 3, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.931.4524&rep=rep1&type=pdf.

[9] Erik Baalbergen, Johan Kos, Clément Louriou, Cédric Campguilhem, and James Barron. Streamlining cross-organisation product design in aeronautics. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 231(12):2192–2202, jul 2017. doi: 10.1177/0954410017716480. URL https://doi.org/10.1177/0954410017716480.

[10] N. Bartoli, M. A. Bouhlel, I. Kurek, R. Lafage, T. Lefebvre, J. Morlier, R. Priem, 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*, number June in AIAA AVIATION Forum, pages pp. 1–28. American Institute of Aeronautics and Astronautics, jun 2016. doi: doi:10.2514/6.2016-4001. URL http://dx.doi.org/10.2514/6.2016-4001.

[11] Gary Belie. Non-Technical Barriers to Multidisciplinary Optimization in the Aerospace Industry. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Multidisciplinary Analysis Optimization Conferences. American Institute of Aeronautics and Astronautics, sep 2002. doi: doi:10.2514/6.2002-5439. URL https://doi.org/10.2514/6.2002-5439.

[12] Simon Binder and K. Seywald. dAEDalus - Git repository, 2017. URL https://github.com/sbind/dAEDalusNXT.

[13] Chunmei Chen, Boris Zakharin, and Israel Wygnanski. On the parameters governing fluidic control of separation and circulaton. In *46th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, jan 2008. doi: 10.2514/6.2008-629. URL https://doi.org/10.2514/6.2008-629.

[14] P.D. Ciampa and B. Nagel. Towards the 3rd generation mdo collaborative environment. In *30th Congress of the International Council of the Aeronautical Sciences*, Daejeon, Korea, 2016. 30th ICAS. URL https://www.agile-project.eu/cloud/index.php/s/scilViiMRGzHpbm.

[15] Pier Davide Ciampa and Björn Nagel. The AGILE paradigm: the next generation of collaborative MDO. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, jun 2017. doi: 10.2514/6.2017-4137. URL https://doi.org/10.2514/6.2017-4137.

[16] Pier Davide Ciampa, Erwin Moerland, Doreen Seider, Erik Baalbergen, Riccardo Lombardi, and Roberto D'Ippolito. A collaborative architecture supporting AGILE design of complex aeronautics products. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, jun 2017. doi: 10.2514/6.2017-4138. URL https://doi.org/10.2514/6.2017-4138.

[17] P. Coleman. Innovations in collaborative modeling and simulation to deliver the behavioral digital aicraft. *PDT Europe*, 2012.

[18] CPACS. Cpacs – a common language for aircraft design, 2016. URL https://software.dlr.de/p/cpacs/home/. Accessed 28-03-2017.

[19] T.D. Crouch. *Wings: A History of Aviation from Kites to the Space Age*. Smithsonian National Air and Space Museum, 2004. ISBN 9780393326208. URL https://books.google.de/books?id=cXQmUf-JtNkC.

[20] Sebastian M. Deinert. *Fakultät für Maschinenwesen Lehrstuhl für Leichtbau Sebastian Moritz Deinert*. Phd thesis, Technischen Universität München, 2016. URL https://mediatum.ub.tum.de/doc/1297578/1297578.pdf.

[21] Deutsches Zentrum für Luft- und Raumfahrt e.V. □ Simulation and Software Technology. Remote component environment (rce), 2017. URL http://rcenvironment.de/. Accessed 28-03-2017.

[22] German Aerospace Center DLR. European high lift programme ii: Final publishable report. Technical report, DLR, 2004. URL http://www.transport-research.info/sites/default/files/project/documents/20120822_121607_86015_124729341EN6.pdf.

[23] European Aviation Safety Agency. Certification specifications for large aeroplanes, cs-25, 2013. URL https://www.easa.europa.eu/document-library/certification-specifications/cs-25-initial-issue. Accessed 20-03-2017.

[24] Federal Aviation Administration (FAA). Airworthiness standards: Transport category airplanes, 14 c.f.r. §25, 1964. URL http://www.ecfr.gov/cgi-bin/text-idx?SID=bf54dd6dc736f0acf9a0bc280c45e2cb&mc=true&node=pt14.1.25&rgn=div5#sp14.1.25.a. Accessed 20-03-2017.

[25] DOT Federal Aviation Administration (FAA). Nprm docket no. f aa-2014-0926 (directorate identifier 2014-nm-085-ad): 747-8 vibration during high g maneuvers, airplane model 747-8, 2015. URL https://www.federalregister.gov/d/2015-17023. Accessed 10-04-2017.

[26] F. Flager and J. Haymaker. A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries. In *Conference: 24th International Conference on Information Technology in Construction*, pages pp. 625–630, 2007. URL https://www.researchgate.net/publication/265873970_A_Comparison_of_Multidisciplinary_Design_Analysis_and_Optimization_Processes_in_the_Building_Construction_and_Aerospace_Industries.

[27] Flexop. Flexop project, 2015. URL https://www.flexop.eu/. Accessed 20-04-2017.

[28] M. Fujino. Design and development of the hondajet. *Journal of Aircraft*, Vol. 42(No. 3):pp. 755–764, may 2005. ISSN 0021-8669. doi: 10.2514/1.12268. URL http://dx.doi.org/10.2514/1.12268.

[29] Michimasa Fujino. Natural-laminar-flow airfoil development for the honda jet. In *20th AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics, jun 2002. doi: 10.2514/6.2002-2932. URL https://doi.org/10.2514/6.2002-2932.

[30] E. Gamma, Helm, R., R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Pearson Education, 1994. ISBN 9780321700698. URL https://books.google.nl/books?id=6oHuKQe3TjQC.

[31] Justin Gray, Kenneth Moore, and Bret Naylor. OpenMDAO: An open source framework for multidisciplinary analysis and optimization. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. American Institute of Aeronautics and Astronautics, sep 2010. doi: 10.2514/6.2010-9101. URL https://doi.org/10.2514/6.2010-9101.

[32] Justin Gray, Kenneth Moore, and Bret Naylor. OpenMDAO: An open source framework for multidisciplinary analysis and optimization. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. American Institute of Aeronautics and Astronautics, sep 2010. doi: 10.2514/6.2010-9101. URL https://doi.org/10.2514/6.2010-9101.

[33] Justin Gray, Kenneth Moore, Tristan Hearn, and Bret Naylor. A standard platform for testing and comparison of MDAO architectures. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference<BR>20th AIAA/ASME/AHS Adaptive Structures Conference<BR>14th AIAA*. American Institute of Aeronautics and Astronautics, apr 2012. doi: 10.2514/6.2012-1586. URL https://doi.org/10.2514/6.2012-1586.

[34] D. Grose. Reengineering the aircraft design process. In *5th Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, sep 1994. doi: 10.2514/6.1994-4323. URL https://doi.org/10.2514/6.1994-4323.

[35] Marin D. Guenov and Ernst Kesseler. *Advances in Collaborative Civil Aeronautical Multidisciplinary Design Optimization*. American Institute of Aeronautics and Astronautics, jan 2010. doi: 10.2514/4.867279. URL https://doi.org/10.2514/4.867279.

[36] R. T. Haftka. Automated procedure for design of wing structures to satisfy strength and flutter requirements. Technical report, TN D-7264, NASA Langley Research Center, Hampton, VA, 1973.

[37] R. T. Haftka. Optimization of flexible wing structures subject to strength and induced drag constraints. *AIAA Journal*, Vol. 15(No. 8):pp. 1101–1106, aug 1977. ISSN 0001-1452. doi: 10.2514/3.7400. URL http://dx.doi.org/10.2514/3.7400.

[38] R. T. Haftka and C. P. Shore. Approximate methods for combined thermal/structural design. Technical report, TP-1428, NASA, 1979. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19790017256.pdf.

[39] R. T. Haftka, D. Villanueva, and A. Chaudhuri. Parallel surrogate-assisted global optimization with expensive functions – a survey. *Structural and Multidisciplinary Optimization*, Vol. 54(No. 1):pp. 3–13, apr 2016. doi: 10.1007/s00158-016-1432-3. URL https://doi.org/10.1007%2Fs00158-016-1432-3.

[40] Raphael T. Haftka, James H. Starnes Jr., Furman W. Barton, and Sidney C. Dixon. Comparison of two types of structural optimization procedures for flutter requirements. *AIAA Journal*, 13(10): 1333–1339, oct 1975. doi: 10.2514/3.60545. URL https://doi.org/10.2514/3.60545.

[41] Christopher Heath and Justin Gray. OpenMDAO: Framework for flexible multidisciplinary design, analysis and optimization methods. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference<BR>20th AIAA/ASME/AHS Adaptive Structures Conference<BR>14th AIAA*. American Institute of Aeronautics and Astronautics, apr 2012. doi: 10.2514/6.2012-1673. URL https://doi.org/10.2514/6.2012-1673.

[42] E. H. Hirschel, H. Prem, and G. Madelung. *Aeronautical Research in Germany: From Lilienthal until Today*. Engineering online library. Springer Berlin Heidelberg, 2012. ISBN 9783642184840. URL https://books.google.nl/books/about/Aeronautical_Research_in_Germany.html?id=OoFcHOLpCskC&redir_esc=y.

[43] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Matrix Analysis. Cambridge University Press, 2012. ISBN 9780521839402. URL https://books.google.nl/books?id=5I5AYeeh0JUC.

[44] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/mcse.2007.55. URL https://doi.org/10.1109/mcse.2007.55.

[45] John T. Hwang. *A modular approach to large-scale design optimization of aerospace systems*. PhD thesis, University of Michigan, 2015. URL http://mdolab.engin.umich.edu/sites/default/files/Hwang_dissertation.pdf.

[46] L. R. Jenkinson, P. Simpkin, and D. Rhodes. *Civil Jet Aircraft Design*. AIAA education series. American Institute of Aeronautics and Astronautics, 1999. ISBN 9780340741528. URL https://books.google.nl/books?isbn=034074152X.

[47] Eric Jones, Oliphant, Travis, and Pearu Peterson. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/.

[48] Gaetan Kenway, Graeme Kennedy, and Joaquim Martins. Aerostructural optimization of the common research model configuration. In *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, jun 2014. doi: 10.2514/6.2014-3274. URL https://doi.org/10.2514/6.2014-3274.

[49] I. Kroo and R. S. Shevell. Aircraft design: Synthesis and analysis, 2001. URL http://adg.stanford.edu/aa241/AircraftDesign.html. Accessed 20-03-2017.

[50] D. Küchemann. *The Aerodynamic Design of Aircraft: A Detailed Introduction to the Current Aerodynamic Knowledge and Practical Guide to the Solution of Aircraft Design Problems*. Pergamon International Library of Science, Technology, Engineering, and Social Studies. Pergamon Press, 1978. ISBN 9780080205151.

[51] Timo Kühn, Vlad Ciobaca, Ralf Rudnik, Burkhard Gölling, and Wiebke Breitenstein. Active flow separation control on a high-lift wing-body configuration part 1: Baseline flow and constant blowing. In *29th AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics, jun 2011. doi: 10.2514/6.2011-3168. URL https://doi.org/10.2514/6.2011-3168.

[52] A. B Lambe and J. R. R. A Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural & Multidisciplinary Optimization*, Vol. 46(No. 2):pp. 273–284, 2012. ISSN 1615-147X. doi: 10.1007/s00158-012-0763-y.

[53] R. Lano. The n2 chart. Internal report, TRW Software Series, Redondo Beach, CA, 1977.

[54] M. Lengers, U. Scholz, and M. Bauer. Test and industrial assessment of active flow control integration in order to increase high - lift performance. *Deutscher Luft- und Raumfahrtkongress 2011*, pages pp. 1145–1153, 2011.

[55] O. Lilienthal. *Der Vogelflug als Grundlage der Fliegekunst: Ein Beitrag zur Systematik der Flugtechnik*. Heyfelder, 1889.

[56] E. Livne. The active flutter suppression (afs) technology evaluation project. In *JAMS Meeting, Seattle, WA, March*, 2014. URL https://depts.washington.edu/amtas/events/jams_14/presentations/17.Livne.pdf. Accessed 10-04-2017.

[57] Lufthansa. Unsere flotte, our fleet. *Magazin*, page 83, oct 2017. URL http://www.lhm-lounge.de/downloads/standardbeitrag/625263/lh_1710_inter.pdf.

[58] W. Machunze, A. Gessler, T. Fabel, P. Horst, and Rädel, M. and Wolf, K. and Ulbricht, A. and Münter, S. and Hufenbach, W. Active flow control system integration into a cfrp flap. *CEAS Aeronautical Journal*, Vol. 7(No. 1):pp. 69–81, 2016. ISSN 18695590. doi: 10.1007/s13272-015-0171-2.

[59] J. R. R. A. Martins. Multidisciplinary design optimization laboratory, 2017. URL http://mdolab.engin.umich.edu/. Accessed 22-03-2017.

[60] Joaquim R. R. A. Martins and Andrew B. Lambe. Multidisciplinary design optimization: A survey of architectures. *AIAA Journal*, 51(9):2049–2075, sep 2013. doi: 10.2514/1.j051895. URL https://doi.org/10.2514/1.j051895.

[61] Joaquim R. R. A. Martins, Christopher Marriage, and Nathan Tedford. pyMDO: An object-oriented framework for multidisciplinary design optimization. *ACM Transactions on Mathematical Software*, 36(4):1–25, aug 2009. doi: 10.1145/1555386.1555389. URL https://doi.org/10.1145/1555386.1555389.

[62] Erwin Moerland, Richard-Gregor Becker, and Björn Nagel. Collaborative understanding of disciplinary correlations using a low-fidelity physics-based aerospace toolkit. *CEAS Aeronautical Journal*, 6(3):441–454, apr 2015. doi: 10.1007/s13272-015-0153-4. URL https://doi.org/10.1007/s13272-015-0153-4.

[63] Kenneth Moore, Bret Naylor, and Justin Gray. The development of an open source framework for multidisciplinary analysis and optimization. In *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, sep 2008. doi: 10.2514/6.2008-6069. URL https://doi.org/10.2514/6.2008-6069.

[64] B. Nagel and Böhnke, D. and Gollnick, V. and Schmollgruber, P. and Rizzi, A. and La Rocca, G. and Alonso, J. J. Communication in aircraft design: Can we establish a common language? *28th International Congress of the Aeronautical Sciences*, 2012. ISSN 978-162276754-0. URL http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A654699&dswid=-5597.

[65] NOESIS. Optimus: The industry-leading pido software platform, 2017. URL https://www.noesissolutions.com/our-products/optimus.

[66] OpenMDAO. Openmdao 1.0 alpha is coming, 2015. URL http://openmdao.org/openmdao-1-0-alpha-is-coming/. Accessed 03-04-2017.

[67] OpenMDAO. The alpha has landed. welcome to openmdao 1.0.0a, 2015. URL http://openmdao.org/the-alpha-has-landed-welcome-to-openmdao-1-0a/. Accessed 03-04-2017.

[68] OpenMDAO. Sellar - A Simple Two-Discipline Problem – OpenMDAO Documentation, 2017. URL http://blue-kgm.readthedocs.io/en/latest/basic_guide/sellar.html. Accessed 10-10-2017.

[69] OpenMDAO. Openmdao.org | an open-source mdao framework written in python., 2017. URL http://openmdao.org/. Accessed 22-11-2017.

[70] OpenMDAO. Github - openmdao/openmdao, 2017. URL https://github.com/OpenMDAO/OpenMDAO. Accessed 03-04-2017.

[71] L. Prandtl. Über flüssigkeitsbewegung bei sehr kleiner reibung. In *Verhandlungen des Dritten Internationalen Mathematiker-Kongresses in Heidelberg : vom 8. bis 13. August 1904*, pages pp. 484–491. A. Krazer, ed., Taubner, Leipzig, Germany, 1905.

[72] D. P. Raymer. *Aircraft design: a conceptual approach*. Educ Series. American Institute of Aeronautics and Astronautics, 1989. ISBN 9780930403515.

[73] Robert Reams. Hadamard inverses, square roots and products of almost semidefinite matrices. *Linear Algebra and its Applications*, 288:35–43, feb 1999. doi: 10.1016/s0024-3795(98)10162-3. URL https://doi.org/10.1016/s0024-3795(98)10162-3.

[74] J. Roskam. *Part I: Preliminary Sizing of Airplanes*. Number pt. 1 in Airplane Design. DARcorporation, 1985. ISBN 9781884885426.

[75] J. Roskam. *Part II: Preliminary Configuration Design and Integration of the Propulsion System*. Number pt. 2 in Airplane Design. DARcorporation, 1985. ISBN 9781884885433.

[76] G. J. J. Ruijgrok. *Elements of airplane performance*. VSSD, 2009. ISBN 9789065622037.

[77] P. Schlösser. *Design of an Active Flow Control System at the Pylon/Wing Junction*. PhD thesis, TU München, 2015.

[78] L. A. Schmit. Structural synthesis - its genesis and development. *AIAA Journal*, Vol. 19(No. 10): 1249–1263, oct 1981. ISSN 0001-1452. doi: 10.2514/3.7859. URL http://dx.doi.org/10.2514/3.7859.

[79] L. A. Schmit and W. A. Thornton. Synthesis of an airfoil at supersonic mach number. Technical report, CR 144, NASA, 1965.

[80] Lucien A Schmit. Structural design by systematic synthesis. In *Proceedings of the 2nd conference on electronic computation, ASCE, New York*, pages 105–122, 1960.

[81] L. A. Schmit Jr. Structural synthesis — precursor and catalyst. recent experiences in multidisciplinary analysis and optimization. Technical report, CP-2337, NASA, 1984.

[82] R. Sellar, S. Batill, and J. Renaud. Response surface based, concurrent subspace optimization for multidisciplinary system design. In *34th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, jan 1996. doi: 10.2514/6.1996-714. URL https://doi.org/10.2514/6.1996-714.

[83] K. Seywald. *Wingbox Mass Prediction considering Quasi-Static Nonlinear Aeroelasticity*. PhD thesis, Technischen Universität München, 2011. URL https://www.diva-portal.org/smash/get/diva2:474633/FULLTEXT01.pdf.

[84] K Seywald, N Gommes de Paule, A Wildschek, F Holzapfel, C Breitsamter, and M Förster. Validation of an aeroelastic analysis and simulation tool for the assessment of innovative, highly elastic aircraft configurations. In *Deutscher Luft- und Raumfahrtkongress [DGLRK2014-0232]*, pages pp. 1–10, Augsburg, 2014.

[85] S Shahpar. Challenges to overcome for routine usage of automatic optimisation in the propulsion industry. *The Aeronautical Journal*, 115(1172):615–625, 2011. ISSN 0001-9240. doi: DOI: 10.1017/S0001924000006308. URL https://www.cambridge.org/core/article/challenges-to-overcome-for-routine-usage-of-automatic-optimisation-in-the-propu
0ECEB120D713F631B5271191DDFA88A4.

[86] John W. Shipman. Tkinter - python wiki, 2017. URL https://wiki.python.org/moin/TkInter.

[87] Timothy W. Simpson and Joaquim R. R. A. Martins. Multidisciplinary design optimization for complex engineered systems: Report from a national science foundation workshop. *Journal of Mechanical Design*, 133(10):101002, 2011. doi: 10.1115/1.4004465. URL https://doi.org/10.1115/1.4004465.

[88] J. Sobieszczanski-Sobieski, A. Morris, and M. van Tooren. *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. Wiley, 2015. ISBN 9781118897089.

[89] D. V. Steward. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28(No. 3):pp. 71–74, Aug 1981. ISSN 0018-9391. doi: 10.1109/TEM.1981.6448589.

[90] Nathan P. Tedford and Joaquim R. R. A. Martins. Benchmarking multidisciplinary design optimization algorithms. *Optimization and Engineering*, 11(1):159–183, mar 2009. doi: 10.1007/s11081-009-9082-6. URL https://doi.org/10.1007/s11081-009-9082-6.

[91] TOCIA Consortium. Eu fp7 tocia project public web page, 2016. URL http://www.toica-fp7.eu/. Accessed 10-03-2017.

[92] E. Torenbeek. *Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance*. Delft University Press, 1976. ISBN 9789029825054. URL https://books.google.de/books?id=ZBwIAQAAIAAJ.

[93] E. Torenbeek. *Advanced Aircraft Design: Conceptual Design, Technology and Optimization of Subsonic Civil Airplanes*. Aerospace Series. Wiley, 2013. ISBN 9781118568095.

[94] Imco van Gent and Lukas Müller. Cmdows - git repository, 2017. URL https://bitbucket.org/imcovangent/cmdows. Accessed: 20-11-2017.

[95] Imco van Gent and Lukas Müller. Kadmos - git repository, 2017. URL https://bitbucket.org/imcovangent/kadmos. Accessed 22-11-2017.

[96] Imco van Gent, Pier Davide Ciampa, Benedikt Aigner, Jonas Jepsen, Gianfranco La Rocca, and Joost Schut. Knowledge architecture supporting collaborative MDO in the AGILE paradigm. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, jun 2017. doi: 10.2514/6.2017-4139. URL https://doi.org/10.2514/6.2017-4139.

[97] Imco van Gent, Gianfranco La Rocca, and Leo 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, jun 2017. doi: 10.2514/6.2017-3663. URL https://doi.org/10.2514/6.2017-3663.

[98] John Vassberg, Mark Dehaan, Melissa Rivers, and Richard Wahls. Development of a common research model for applied CFD validation studies. In *26th AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics, aug 2008. doi: 10.2514/6.2008-6919. URL https://doi.org/10.2514/6.2008-6919.

[99] L. E. Wallace. The whitcomb area rule: Naga aerodynamics research and innovation. In *FROM ENGINEERING SCIENCE TO BIG SCIENCE: The NACA and NASA Collier Trophy Research Project Winners*, The NASA History Series. NASA History Office, 1998.

[100] W. L. Wesley and P. Chan-gi. Aeroelastic optimization study based on x-56a model. In *AIAA Atmospheric Flight Mechanics Conference*. American Institute of Aeronautics and Astronautics (AIAA), jun 2014. doi: 10.2514/6.2014-2052. URL https://doi.org/10.2514%2F6.2014-2052.

[101] Andreas Wildschek. Concurrent optimization of a feed-forward gust loads controller and minimization of wing box structural mass on an aircraft with active winglets. In *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, jun 2015. doi: 10.2514/6.2015-2490. URL https://doi.org/10.2514/6.2015-2490.

[102] J.R. Wright and J.E. Cooper. *Introduction to Aircraft Aeroelasticity and Loads*. Aerospace Series. Wiley, 2014. ISBN 9781118700433.

[103] O. Wright. Telegram from orville wright in kitty hawk, n.c., to his father announcing four successful flights, 1903 dec. 17. Photograph, Retrieved from the Library of Congress, 1903. URL https://www.loc.gov/item/2003680165/. Accessed 21-03-2017.

[104] W. Wright. Wilbur wright [henry a. wise wood, aero club of america bulletin, july 1912]. Manuscript/Mixed Material, Retrieved from the Library of Congress, 1912. URL https://www.loc.gov/item/wright003306/. Accessed 21-03-2017.

[105] S. X. Ying. Report from icas workshop on complex systems integration in aeronautics. *30th Congress of the International Council of the Aeronautical Sciences*, 2016.