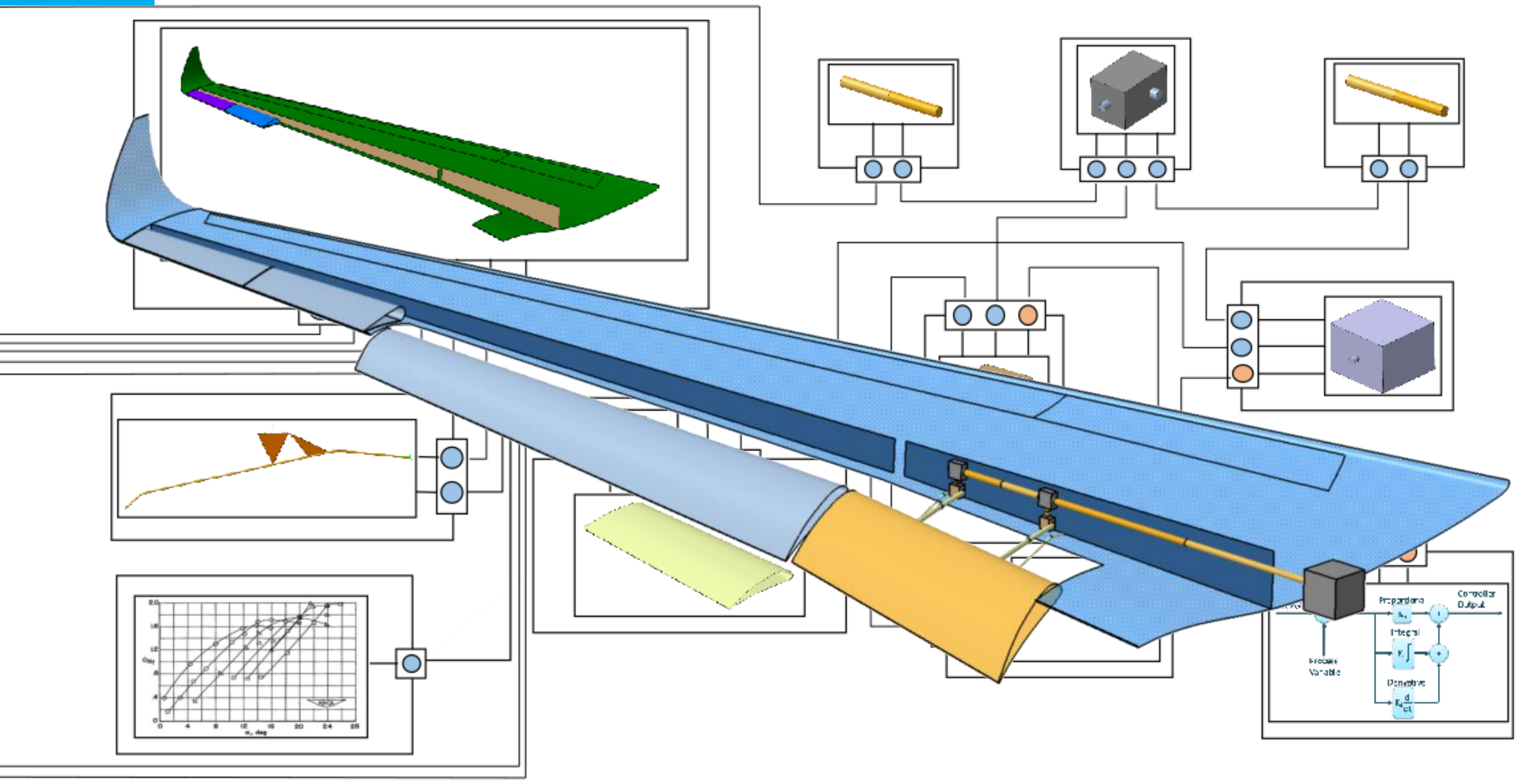
Enabling an architecture- based design approach for multi-body simulation of complex systems

Elias Allegaert



ENABLING AN ARCHITECTURE- BASED DESIGN APPROACH FOR MULTI-BODY SIMULATION OF COMPLEX SYSTEMS

by

Elias Allegaert

in partial fulfillment of the requirements for the degree of

Master of Science

in Aerospace Engineering

at Delft University of Technology,
to be defended publicly on Wednesday August 23, 2017 at 9:30 AM.

Student number:	4047044	
Supervisors:	Dr.ir. Gianfranco La Rocca	TU Delft
	Dr. Yves Lemmens	Siemens Industry Software NV
Thesis committee:	Prof. dr. ing. Georg Eitelberg	TU Delft
	Dr. Jian Guo	TU Delft

ABSTRACT

As the complexity of high-tech systems continuously increases, engineers look for possibilities to reduce time and cost of the development of these systems. Architecture-based design enables a front-loaded design process with knowledge reuse. By enabling the automatic synthesis of simulation models, different configurations of an architecture can be realized and simulated efficiently. Current practices are found in the automotive and aerospace industry where architecture-based design is used for the automatic synthesis of multi-physics simulation models. In this way, different architecture options and simulation model variations can be efficiently investigated early in the development process. Multi-body simulations are also frequently used in the conceptual design of complex mechatronic systems. However a suitable methodology to synthesize multi-body simulation models is lacking.

In this thesis, a methodology is developed that makes three necessary improvements to an existing approach that can synthesize only multi-physics models. Firstly, the existing approach assumed that the order of synthesis of the architecture was irrelevant. However, the use of modular and independent simulation models breaks associative links. Therefore, an additional sorting algorithm needs to be introduced during the synthesis of the architecture. An initial successful attempt is made by adding an implementation of Kahn's algorithm to a prototype code synthesis tool. A second addition of the thesis is the definition of a formalized modelling process for modular multi-body simulation models which are simulation models that can be simulated independently or added to a larger simulation. The formalized process guarantees that the interfaces between simulation models will always work. It also simplifies the modelling approach which leads to the third improvement: clarity of design intent. A procedure was developed and automated that takes away the burden of creating interfaces between simulation models. Consequently, the modeller can focus fully on creating robust models with clear design intent.

The developed methodology is verified with a case study on the conceptual design of a trailing-edge high-lift system. An architecture is defined and parameterized multi-body simulation models are created that realize the components of the architecture. Besides the components that actuate the flap, such as gearboxes, shafts and a motor, four different deployment mechanism types are modelled. The parameterized geometry of the simulation models adapts automatically in order to provide the correct flap trajectory and to form a consistent simulation model. A tool synthesizes automatically all architecture configurations. Interface forces and moments between bodies can be inspected directly and the required actuation torque is found. The sizing of the components is not performed.

The methodology is evaluated for knowledge reuse. For the case study, this leads from 41% up to 92% of reuse of simulation models. However, before these results can be generalized, a trade-off needs to be made from case to case between the time that is saved by the automatic synthesis of simulation models and the time it takes to create the architecture and the compatible simulation models. It can be concluded that the developed methodology enables an architecture-based design approach for complex multi-body simulation models. Furthermore, the methodology is advantageous compared to a traditional design process where all system configurations have to be modelled and analyzed individually.

Preface

This thesis is the result of a year of research at Siemens Industry Software (SISW) in Leuven. I would not have been able to do this without the knowledge and opportunities that the TU Delft has given to me. I remember that in the last year of secondary school, I received an information package from BeNeLair. It contained two editions of the Leonardo Times, the magazine from the aerospace faculty's study association. Back then, I understood little of its content. Now, it keeps me updated about the amazing aerospace research that is being conducted at the faculty and in the world.

To begin with, I want to thank Yves Lemmens for his daily supervision and for offering me the opportunity to do a thesis at SISW. To my supervisor at TU Delft, Gianfranco La Rocca, thank you for your patience, feedback and your inspirational course on advanced design methodologies. Thank you, Dirk, for correcting the grammar of this report. Finally, I am glad Dr. Eitelberg and Dr. Guo want to join my graduation committee and assess my thesis.

All those people have helped me grow intellectually, but where would I be without the people that are closest to me? Thank you, JC Zogezegd, for the many great evenings, weekends and holidays. Thank you, to the Fellowship group, for being the best friends when we had to study as well as when we were hanging out. Thank you Rufus for your enthusiasm. My dearest Heleen, thank you for your endless love, support and patience in a period with too much work and too little holidays. I am deeply indebted to you for the time you spent on improving the structure and grammar of this thesis. And lastly, thank you mother and father for your faith in me. Thank you for letting me study at the TU Delft and for allowing the many intermezzos.

This thesis has forced me to broaden my perspective on engineering and use a more hands-on approach that is favored in the industry. However, I have always tried to get to the core of what I was doing. This thesis has made me think about the fundamentals of engineering and this report is my humble initial effort to master the complexity of it.

Elias Allegaert
Delft, August 2017

CONTENTS

List of Figures

LIST OF TABLES

1 INTRODUCTION

1.1. THE CHALLENGE

The aerospace and defense industry is highly competitive which results in a constant pressure for companies to keep innovating. Modern technological innovations often result in complex systems with interconnected subsystems. This led more than once to schedule delays and cost overruns [1]. The Boeing 777 contains about 100 electronic control units (ECUs) while the new Boeing 787 and Airbus A380 are estimated to have more than 1000 of them [2]. Another example, the F-35 has a number of interfaces between components in the order of 10^5 and 90% of its functionality is managed by software while the older F-16 has order 10^3 interfaces and less than 40% of its functions are managed by software [3].

Although the complexity of systems has increased, current design practices still follow an iterative process that starts from a design point that is based on rough estimates and empirical relations [4]. Starting from this inaccurate information, the designers early on need to make decisions that will strongly affect the final design. At the same time, collaborating teams are geographically dispersed an they often have their own domain specific engineering tools [5].

The pressure to innovate clashes with a traditional design approach. On the one hand, collaborating teams fail to communicate their design decisions. This leads to unforeseen interactions between subsystems later in the design. On the other hand, the use of rough estimates and overly simplified models in the conceptual design phase leads to an infeasible design point. For these two reasons experts have expressed their concerns that traditional design approaches are no longer adequate for the large and complex systems that are being developed these days [1] [2] [3].

1.2. FRONT-LOADING THE DESIGN PROCESS

Industry and research institutions responded to the challenge by developing technologies that bring more design knowledge earlier in the design process because decisions made during the conceptual design have a high impact on the finished product and better informed decisions should lead to a better design [4] [6]. In figure 1.1, the current state of the design process is denoted by the dashed lines. Design knowledge (green) is the product specific knowledge that is accumulated during the design process. A

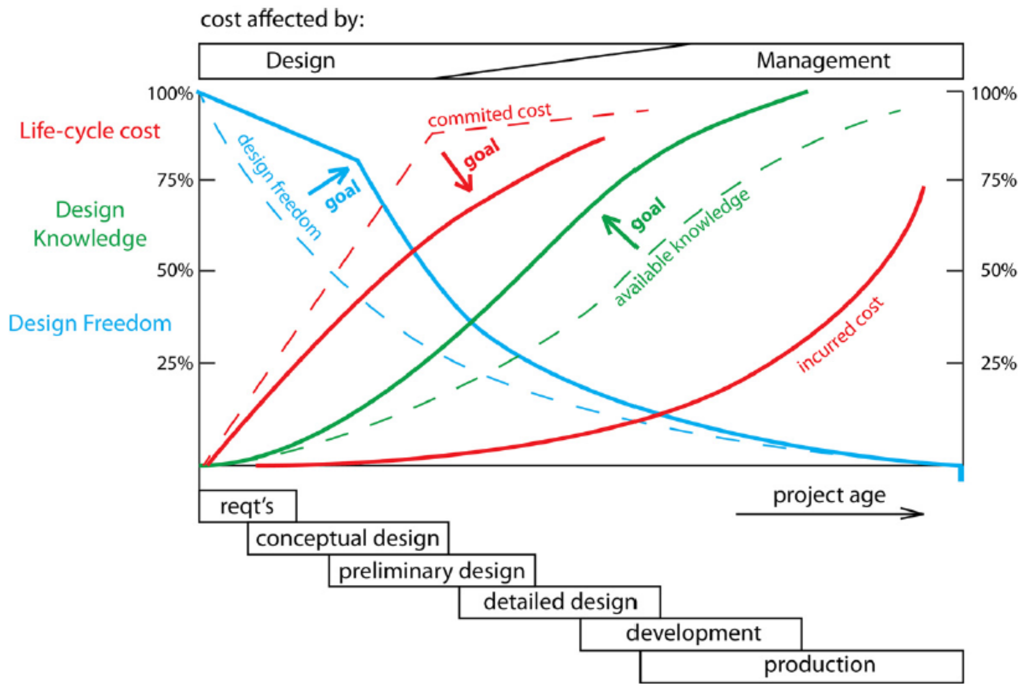


Figure 1.1: Principles of front-loaded design. KBE has the potential ot bring knowledge more forward into the design. Notice also the concurrent design phases. [6]

front-loaded design aims to shift this curve more to the left. Design freedom (blue) is the ease to make design changes. It can be seen that directly from the start there is a sharp decrease because assumptions and decisions need to be made. A front-loaded design aims to shift this curve upwards which means that irreversible design decisions happen later in the design process. As a result to the increase in design freedom, committed cost (red) shifts more towards the end of the design process. Bringing design knowledge earlier into the design while at the same time postponing design decisions would normally lead to a longer design process. However, the advancement of technology on many fronts makes it now possible to greatly increase the efficiency of design teams.

One technology that is becoming well-known is knowledge based engineering (KBE) which makes it possible to efficiently capture and reuse engineering knowledge [7]. The fact that KBE systems store knowledge in a formalized way make them well suited as support for integration between disciplines. KBE lends its basis for knowledge formalization from the object-oriented paradigm and it has a strong foundation in a branch of artificial intelligence called expert systems.

Another technology, also borrowing some principles from the object-oriented paradigm is bond-graph theory. Bond-graph models are a domain-independent graphical description of physical systems. It is based on the fact that many physical concepts from different disciplines are analogous. Simulation tools based on bond-graph theory, like LMS Imagine.Lab Amesim and Modelica have shown to be powerful in the first stages of design when no geometry is available yet. Accordingly, they are often called 1D simulation software[8]. Despite the fact that geometry is greatly simplified or neglected, they can provide accurate performance simulation results on which important design decisions can be made with confidence. These software packages come with a library of components containing the equations that represent the physical behavior. Bond-graph models are perfectly fit for a drag-and-drop modelling approach because adding a component to a bond-graph is similar to adding equations to a system. Before the system is solved, the equations are rewritten, during a compilation phase, depending on how the bond-graph is structured [9].

1.3. ARCHITECTURE-BASED DESIGN

The beginning of a design process usually has a diverging phase when many solutions are identified that potentially can meet the requirements [7]. As described earlier, a front-loaded design process dictates that a quantitative comparison between all the different solutions needs to be made. Very often, the best way is to run a numerical simulation with the help of a simulation tool. A multitude of load cases will lead to a long list of simulation configurations. It is also possible that initially, components are modelled in a more simple and computationally less expensive way. Later, they are replaced with a more accurate version if the first analysis showed good performance. This leads to even more simulation configurations. At this stage, KBE and 1D simulation are indispensable to reduce the time that needs to be spent on modelling and analysis.

Users of 1D simulation software benefits from the disciplinary libraries of components and the component modularity. However, there is still quite some human interaction needed to create a system model. During the diverging phase, this results in a lot of time being spent on modelling and analyzing the many simulations and managing their result files. Therefore, 1D simulation software vendors have added tools that aid the engineer in doing these tasks. First of all, a model that was built from elementary components can serve as a new 'super component'. Secondly, it is possible to define an abstract description of the system whose components are then realized with their corresponding simulation models. Since most of the simulation configurations share a common topology a.k.a. architecture, this can be leveraged to let a tool automatically synthesize simulation models. The synthesis requires a certain 'protocol' on how components and super components can be connected. Since the system architecture is a vital part of the design, this approach is called architecture-based design (ABD).

The ABD process can be illustrated with an example that is visualized in figure 1.2. The aim of the study was to investigate different combinations of electrical and mechanical components such as motors and propellers while simultaneously testing different cooling strategies. First, a base architecture was defined which describes what components interact with each other. Then, more information is added, e.g. the kinds of energy and effort flows between components. Subsequently, the configurations are defined by appointing the component's simulation models which were modelled in LMS Imagine.Lab Amesim. Multiple simulation models of receivers, servos, motors, batteries etc. were tested. Finally, the synthesis tool automatically combined the individual component simulation models into one large simulation model.

ABD lends its principles from the model-based systems engineering (MBSE) paradigm. This is the general term for the notion that all information in the design process needs to be described by a formal model in order to have a streamlined product life-cycle management (PLM). The International Council on Systems Engineering (INCOSE) published a report in 2007 [11] and in 2014 [12] where the use of MBSE was promoted. They referred to the success that companies and institutions had for the design of large software and computer hardware systems.

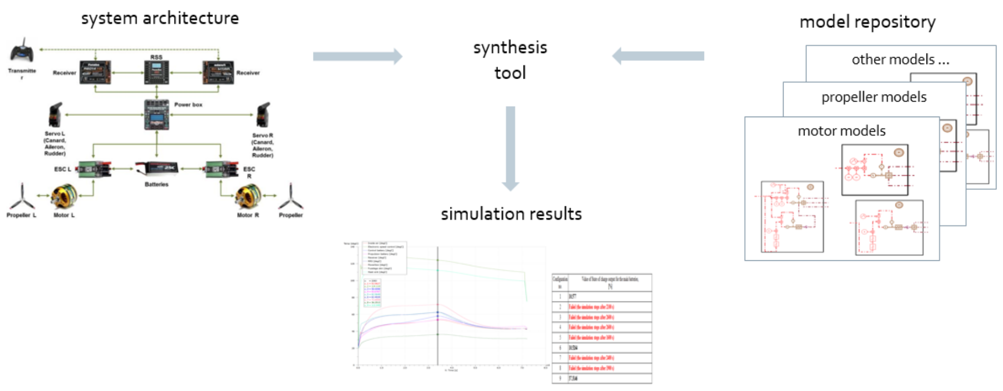


Figure 1.2: Architecture-based design used as an example for design space exploration of the electrical system of an unmanned aerial vehicle (UAV). [based on [10]]

1.4. SCOPE OF THE THESIS

In spite of the success of 1D multi-physics simulation, many systems in aerospace and mechanical engineering also require multi-body mechanics to analyze kinematic and dynamic behavior. For example, the forces obtained during multi-body simulation (MBS) are often the input for the sizing of parts. Therefore, a logical next step is to use ABD for MBS. Compared with bond-graph models, MBS models usually are more complex due to their large amount of geometrical constraints and the many parameters that define their geometry. Therefore, it is not certain if ABD can be used for the automatic synthesis of MBS models. The goal of this thesis is to assess the potential of using ABD for the conceptual design of complex systems that rely on MBS. The design of a trailing-edge high-lift system for a commercial airliner was chosen as a case study because it involves complex kinematics that can be obtained by a wide array of mechanisms as is demonstrated by Rudolph [13]. The thesis aims to answer two research questions:

1. Is it possible to use ABD to automatically assemble MBS models of a trailing-edge high-lift device to perform load analysis?

2. Is the proposed method advantageous compared to traditional modelling?

The following activities were performed to find an answer on these research questions:

- Extend the current ABD methodology to enable the automatic synthesis of multi-body simulation models.

- Verify the developed methodology on the conceptual design of a trailing-edge high-lift system.

- Evaluate time efficiency, benefits and limitations of the developed methodology.

The scope of this research is limited to homogeneous simulation architectures where all the components are represented with multi-body simulation models. It was not the intention to have a heterogeneous architecture realization with, for example, a mix of multi-body simulation models and 1D simulation models because that would involve co-simulation strategies which would complicate the research.

1.5. STRUCTURE OF THE REPORT

This introductory chapter mentioned some systems engineering concepts which are discussed more in depth. In the next chapter, the state-of-the-art of some important systems engineering concepts is discussed. The chapter starts with an introduction to the object-oriented programming paradigm, MBSE and KBE. Thereafter, an elaborate discussion on ABD follows. Finally, a concise overview on the state-of-the-art in high-lift system design is presented. Chapter 3 explains the development of the methodology that leads to a successful synthesis of MBS models. The chapter starts with some requirements after which the methodology (and how it influences the modelling process) is presented. The methodology is applied to the case study and verified in chapter 4. The case study follows the conceptual design process of the high-lift system in a chronological fashion. In the fifth chapter, the results from the case study and the advantages and disadvantages of ABD are discussed. The last chapter contains the conclusion of this research and recommendations for future work.

2
State-of-the-art

This chapter discusses the state-of-the-art of some important concepts that we will encounter in this report. Section 2.1 gives an overview of three important systems engineering paradigms: object-oriented programming (OOP), model-based systems engineering (MBSE) and knowledge based engineering (KBE). Section 2.2 broadly discusses architecture-based design (ABD). Section 2.3 gives an overview of the multi-body simulation software that is used for this research. Section 2.4 reviews the state-of-the-art of high-lift system design.

2.1. Systems engineering paradigms

Three systems engineering paradigms that are important for this thesis are discussed in this section: object-oriented programming, model-based systems engineering and knowledge based engineering.

2.1.1. Object-oriented programming

Computer science is a rather new discipline when compared with other scientific fields. However, it quite rapidly developed very complex structures and concepts. This happened for the computer hardware, which consists basically of numerous aggregations of transistors, and for the software that ran on these computers. It was during the Apollo program that a successful method was found to organize all the lines of code and to let many teams work together on one large system. This method is called object-oriented programming (OOP).

The essence of OOP is to encapsulate the underlying complexity of computer programs into higher-order components and to expose their functionality so they can be easily used by someone who did not write the code [14]. It allows for modular design and component reuse. As the name suggests, the most important element of OOP is the object. It can possess functions (sometimes called methods), attributes (sometimes called variables) and child objects. An object is an instance of a class which can be seen as the general description of that object. Classes can be structured into a hierarchy by using generalization .

There are three large advantages of using OOP for large projects. Firstly, code can be easily found by navigating through the hierarchy because OOP provides a view on the whole project structure. Besides well navigable code, also data can be quickly accessed due to the structuring. The second advantage is the possibility to hide certain variables and data for users that shouldn't have access to it. The third advantage is the easy reuse of code. By instantiating a class, the object inherits all properties, functions and child objects of that class.

2.1.2. MODEL-BASED SYSTEMS ENGINEERING

As software systems continued growing in complexity, failures made clear that the OOP paradigm alone did not guarantee success. It was found that lots of errors could be attributed to miscommunication and outdated or incomplete requirements among other human errors. Just like the OOP helps to structure code, a method was developed to structure the whole development process. The method is based on models because they are a convenient way to bring information together to help a stakeholder retrieve the information that he needs. Ludewig [15] described three criteria that a model must meet:

- mapping criterion: there is an original object or phenomenon that is mapped to the model.
- reduction criterion: not all the properties of the original are mapped on to the model. However, the model must sufficiently reflect the original. One could think about a quote attributed to Albert Einstein: "Everything should be as simple as possible, but not simpler."
- pragmatic criterion: the model can replace the original for some purpose.

The benefits of a model-based design process grow stronger as various aspects of a system are interlinked. Models can be made, among others, for the system architecture, the behavior of the components, the requirements, the parameters and object hierarchy. Theoretically, models can replace the documents in which information about a system is stored. The Object Management Group (OMG) says that we go from a document-centric way of working to a model-centric one [16]. INCOSE uses the following definition for MBSE:

> "Model-based systems engineering is the formalized application of modeling to support system requirements, design, analysis, and verification and validation (V&V) activities beginning in the conceptual design phase and continuing throughout development and later lifecycle phases." [17, p.189]

It is important to point out that the models discussed here are abstract models. They serve as an aid to communicate information about a system to different stakeholders. For different types of information, different models are used and usually a combination of models is needed to represent complexity. For example, a structural model might show how signals are transferred through a vehicle. Together with a geometrical model, behavioral model etc., it can be found out how a signal error can lead to a certain kind of failure. Besides abstract models, there are simulation models that can be physical or numerical computer-based. Simulation models are used to get quantitative information about a system.

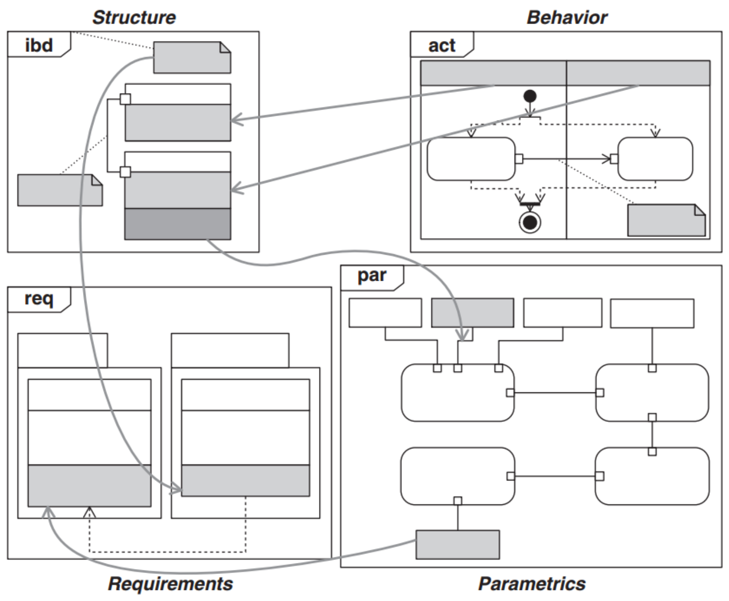


Figure 2.1: Example of four SysML diagrams that are interconnected. The fields are blanked out for generality. [16]

As MBSE grew in popularity there was a proliferation of model types. For this reason the Object Management Group (OMG) wanted to unify all the different models, an initiative that led to the Unified Modeling Language (UML). UML was initially meant for software development but it was later extended with some extra models to also express concepts that are related to ’hardware’ engineering which led to the SysML language. Figure 2.1 illustrates four different SysML models. Their interconnectedness is illustrated with the lines that link different concepts of each model. As a final note, it should be pointed out that drawing diagrams has been an integral part of engineering and scientific insight for a long time. The contribution of MBSE and OMG is a formalization of this process.

2.1.3. KNOWLEDGE BASED ENGINEERING

Practically speaking, KBE is the building of applications that automate repetitive and non-creative tasks and support multi-disciplinary optimization (MDO) through the design process with the aim to lower cost and time during all stages of the design process [7]. There are many examples of KBE applications but they are mainly concentrated in aerospace and automotive engineering [6]. In the early years, KBE focused on detailed design tasks such as meshing [18] [19], detailed design of a turbine stage [20] and electric wire harnass routing in aircraft [21]. However, KBE has also helped in creating the tools that allow variant generation and optimization in the conceptual design [22].

KBE has evolved from Knowledge Based Systems (KBS) a.k.a. expert systems. KBSs are based on the idea that a human expert solves problems by applying his knowledge to a specific situation using computers [23]. The knowledge of the expert can be captured and stored in the long-term memory while the facts of the problem are stored in the short-term memory. The reasoning mechanism is called the inference engine. Sometimes the KBS has explanation facilities to justify its advice. There is a user interface and a developer interface to modify the knowledge base. An expert system also has an external interface which allows communication with external programs and databases. This is analogous to how an expert consults other experts.

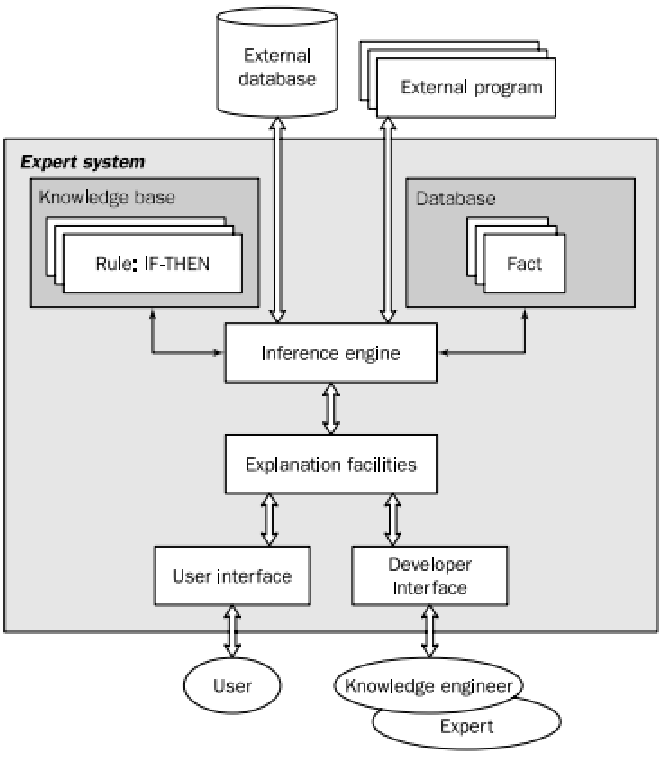


Figure 2.2: Complete architecture of a (rule-based) knowledge-based system. A KBE system has an identical structure. [23]

Although the first KBSs were rule-based, newer KBSs are frame-based systems. Rule-based systems only have IF-THEN relations in their knowledge base which are used, or 'fired', by the inference engine. This can be done in a forward-chaining manner or backwards-chaining manner [23]. The architecture of a rule-based system is visualized in figure 2.2. Frame-based systems have much more flexibility to formalize design knowledge. They are very closely related to OOP. Figure 2.3 illustrates a frame-based description of an aircraft which belongs to the transport vehicles superclass.

Besides using OOP, KBE systems have two extra features: runtime caching and dependency tracking [7]. Caching means that objects and attributes are saved once they have been generated or calculated because they might be requested multiple times which saves time especially for expensive computations. A garbage collector is built in the shell which clears memory from calculations that are not longer valid. Many attributes of objects only need to be computed when the object or its children need them or when other objects ask for their evaluation. By means of dependency tracking, calculation resources are used efficiently.

KBE allows geometry manipulation and it offers a great design framework for MDO. For example, the Multi-Model Generator (MMG) generated and analyzed more than 50 different blended wing body variants within just a few days [24]. Besides time savings, the fact that KBE relies on a programming language makes it excellent for capturing design intent. The use of a language also guarantees consistency during automation and multi-disciplinary collaborative design. These advantages show up in the Design and Engineering Engine (DEE) which is a loosely integrated software system built by Delft University of Technology for conceptual design and MDO. Figure 2.4 shows the high-

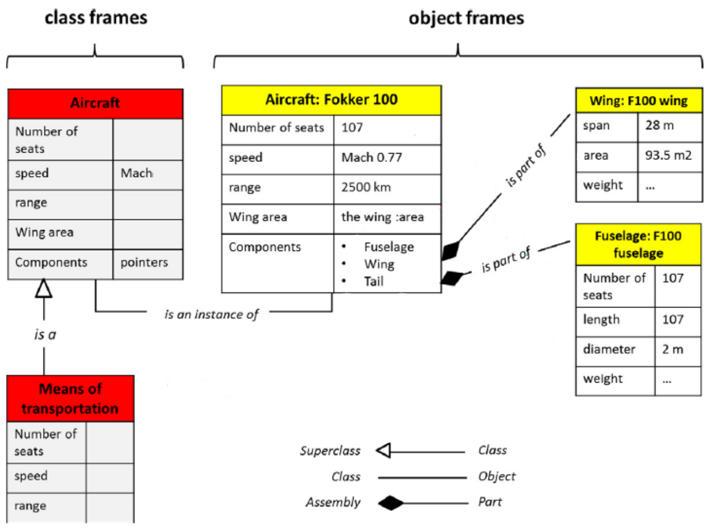


Figure 2.3: Example of class and object frames. A superclass is an abstraction of a class and an object is an instance of a class thereby inheriting all its attributes. [7]

level architecture of the DEE. Two important components are the Initiator and the MMG. The initiator is a MATLAB application which calculates a baseline design of an aircraft for a given set of requirements. Initially, only a parametric aircraft model is produced with limited geometry to iterate the design until a feasible design is found that fulfills all requirements. Furthermore, the baseline design is analyzed and optimized using more detailed models for which knowledge from the MMG is used. The architecture of the Initiator is similar to the high-level architecture of the DEE. This makes sense because the design process should have a fractal nature. Analogous to the mathematical definition of fractals, refinement of the design gives rise to new complexity [25].

2.2. ARCHITECTURE-BASED DESIGN

As explained in the introductory chapter, the modularity of components in 1D simulation software allows engineers to quickly create and analyze simulation models. By front-loading the design process, the performance of different configurations of a system can be analyzed very early in the design process and decisions can be made based on reliable data. There are often many configurations to be investigated. For example, a car manufacturer might want to analyze different engines, transmissions, batteries, tire models etc. Besides these different variants, there are also different load scenarios that are coming from the various certification specifications issued by countries. In order to save time, software was developed that can automatically synthesize and simulate these simulations. Only a topological description, a.k.a. architecture, of the simulation model is needed.

2.2.1. HISTORY

The terms 'architecture' and 'architecting' have been used for long in software development and systems engineering. The term 'architecture' denotes a structure that is used to communicate information between design teams. Despite the recognition of an architecture's importance, the creation of successful software architectures remains something of an art and libraries of books have been written about it [27]. Graaf [28] ex-

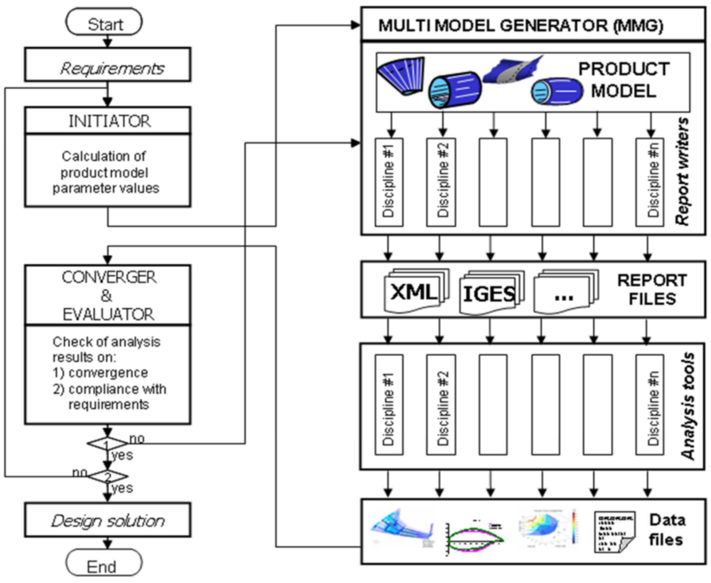


Figure 2.4: The Design and Engineering Engine Architecture [26].

plains that on the one hand, the product architecture has to be rigid enough to provide consistent and clearly defined information about the product. On the other hand, the architecture has to be flexible enough to allow beneficial changes up to a reasonable extent. These two qualities are visualized in figure 2.5.

The definition that INCOSE uses for a system architecture is:

> "Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution." [17, p. 261]

Graaf [28] gives three reasons for using a software architecture. First, the architecture communicates global design decisions to all involved developers and therefore it ensures conceptual integrity. Conceptual integrity can be seen as consistency in the design of the architecture. In software engineering it refers especially to using a consistent way of doing something. Second, software architecture is the first design artifact that allows project assessment. By analyzing the architecture, some aspects of the design, for example, redundancy and failure propagation, can already be evaluated. Thirdly, software architectures allow reuse of existing software solutions.

Not only for computer software, but also for hardware the architecture plays an important part. For the chips of computers the arrangement of logical gates is a very important aspect. Computer systems might always have been one of the most complex machines that humanity has ever built so it is normal that the first attempts to structure the design process happened in this field. Sometimes an architecture is also called a logical organization [29] which might point to the architectures of logical gates of micro processors. Currently, the concept of ABD is applied to the design of mechatronic systems such as aircraft and automobiles.

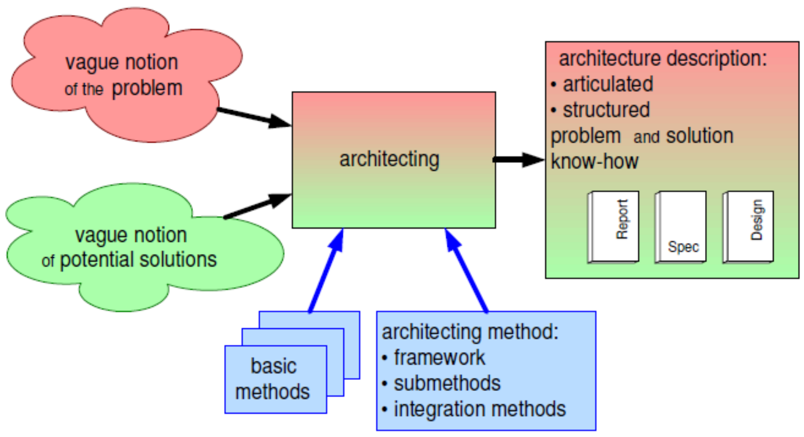


Figure 2.5: the architecting method offers a way to develop from a vague notion of the problem and a vague notion of potential solutions an architecture description which is flexible enough to allow changes while providing the rigidity for consistency throughout the design. [27]

2.2.2. DERIVATION OF THE SYSTEM ARCHITECTURE

One way of obtaining an architecture is with reverse engineering. An existing system can be decomposed and the components can be grouped on the basis of some characteristic they share. This could for example depend on the availability of commercial-off-the-shelf components, the desire to have plug-in extensions afterwards or even the number of manufacturing tools available [30]. However, most often the grouping is done on the basis of the components' shared functionality. The effort to organize knowledge, which does not have to be limited to electromechanical components, is known as creating an ontology. Ontology engineering in computer science tries to formalize concepts and the relations between different concepts [31]. This is the same principle as using abstraction to identify classes and making a functional breakdown diagram (FBD) or a functional flow diagram (FFD).

As early as the 1940s, researchers started to make a list of basic functions that components can perform. In 1999 the National Institute of Standards and Technology (NIST) combined all the lists into one list of elementary functions. Furthermore, three elemental flows were recognized: material, energy and signals. These elementary flows, together with the elementary functions, are called a 'functional basis' [32]. Kurtoglu and Campbell [33] developed a method to convert a functional model into possible system architectures. Such a mapping of function to configuration applies rules that can result in multiple feasible architectures. These efforts can be seen as an attempt to formalize the conceptual design process. They even attempted to automate this process. The formalization also made sure that all possible concepts were found. Another example of research in functional modeling is the function-behavior-state (FBS) method [34]. A function is decomposed into sub-functions until these sub-functions can be linked to physical properties. These physical properties can then be appointed to behavior and states. Behavior can be defined as the evolution of a number of states over time [35].

It can be concluded that there are multiple ways to generate architectures. On one side a system architect creates an architecture based on experience. On the other side, there are formalized processes to obtain feasible architectures from the functional requirements. There are opportunities to integrate automatic function-to-form mapping with the ABD methodology. Alvares-Cabrera [36] points out that a functional description

This figure has been masked due to confidentiality

Figure 2.6: This figure is masked due to confidentiality.

can be made on all levels of detail and that functions are also defined on an abstract level. Umeda describes functions as "the bridge between human intention and physical behavior of artifacts" [37, p.271].

2.2.3. METRICS FOR MODULARITY AND COMPLEXITY

The idea of breaking larger systems up into manageable chunks is encouraged in many disciplines [38]. There is an ongoing effort to formalize concepts like complexity and modularity so they can be used in a scientific approach [30]. However, good metrics for expressing and measuring complexity and modularity are difficult to find. Tamaskar [39] gives a broad literature review on previous attempts to measure complexity. He proposes a graph-based view of a system where each link is attributed a certain weight depending on the number of feedback loops. Tamaskar also evaluates a method to divide a system into modules so that the system becomes more modular. This method is also graph-based and is an extension of the work done by Newman and Girvan [40]. Another way to manage complexity in mechatronic systems is by using the design structure matrix (DSM) [41] [42]. Further, it has been suggested that there exists an optimal level of modularity for a given system [30]. Still, more work is needed in this field to come up with solid methodologies that help creating more modular architectures. The results from this field might contribute to smarter architectural decisions during conceptual design. The decomposition of systems happens at this moment according to various motivations while often architectural complexity is not taken into account [30]. Even so, modularity of the architecture is important because it influences model reuse and enables a concurrent work-flow.

2.2.4. SYNTHESIS OF SIMULATION MODELS

Developers of 1D multi-physics simulation software facilitate the creation of subsystems by providing an application programming interface (API). A synthesis tool uses the API to synthesize components together based on an architecture description. This is what the commercially available LMS Imagine.Lab System Synthesis software does. A prototype version now aims to use the same principles in order to synthesize MBS models. An overview of what the user has to provide and what the tool does, is presented here. Figure 2.6 illustrates this process.

Firstly, the system needs to be described as an architecture. The components in the architecture can be seen as placeholders to which the simulation models are added later. The links act like tubes in which multiple data formats can be housed. Parameters (inputs) and variables (outputs) are appointed to blocks so inherent characteristics of a component can be expressed. For example, it is logical that a gearbox has a parameter defining the gear ratio and that there are output variables of the rotation angles of the shafts. This base architecture description looks somehow like a UML component diagram where also the component attributes from the class diagram are added. In figure 2.6 these are the grey components and their link.

The second step is to define the data formats of the component ports. This tells the synthesis tool what interface type is needed for the simulation model because the subsystems can be modelled in different simulation software. In figure 2.6, this corresponds with the colored ports. The inputs and outputs need to be specified as well. A component can be represented in different simulation software so there can be multiple descriptions for the realization of a component. Furthermore, the user needs to define a file that links the component realization and the component simulation model. This is necessary to link the interfaces correctly in the simulation model and to tag the parameters of the simulation model. As long as the interfaces of the simulation model match with the realization description of the component, they can be synthesized successfully. Finally, a file is written by the user that links the correct realization file for every component in the base architecture.

There are some similarities between the UML diagrams and the architecture description language (ADL) used by the System Synthesis prototype. For example, the components in the base architecture are instances of a general template. This represents generalization and instantiation. There are two other types of relationships in the ADL used by the prototype that correspond with UML. First, there is realization (or implementation) which means that a simulation model realizes (implements or executes) the behavior that the information model specifies. Secondly, the base architecture can be multi-leveled which means that one component can consist of multiple other components. This corresponds with the composition relation in UML.

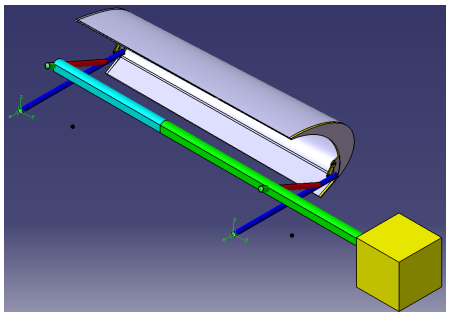
2.2.5. EXAMPLE: SYNTHESIS OF A SLAT MECHANISM

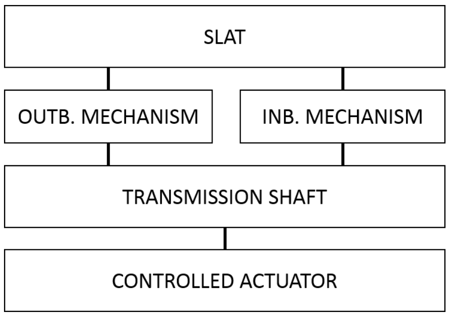
MBS model synthesis using the System Synthesis prototype can best be illustrated with an example: the synthesis of a simple slat mechanism. Figure 2.7 shows a screenshot of the synthesized mechanism and a simplified representation its architecture.

Although the synthesis was successful and the simulation could be solved, there are shortcomings that prohibit a design space exploration (DSE) or optimization of the system.

2.3. MULTI-BODY SIMULATION

Since the subject of this thesis is the automatic synthesis of MBS models, it is important to have a look at the MBS software that will be used. Besides the software architecture, also the concept of submechanisms is explained because it allows the modular composition of simulation models.


(a)



(b)

Figure 2.7: (a) Screenshot of the synthesized slat mechanism in LMS Virtual.Lab Motion (b) Simplified representation of slat mechanism.

2.3.1. MULTI-BODY MECHANICS

The essence of what MBS software does, is solving for the equations of motion that can be derived either from the Newton-Euler method or the Lagrange method. A good references for an introduction into multi-body simulation comes from E.J. Haug [43] or from A.A. Shabana [44]. Each body has its degrees of freedom expressed in **q**. The equation of motion is:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \Phi_{\mathbf{q}}^{\mathbf{T}}(\mathbf{q},t)\Lambda = \mathbf{Q}^{\mathbf{A}}(\mathbf{q}) \tag{2.1}$$

where $\mathbf{M}(\mathbf{q})$ is the mass matrix, $\mathbf{Q}^{\mathbf{A}}$ contains the applied forces and moments and Λ contains the Lagrange multipliers. $\Phi_{\mathbf{q}}^{\mathbf{T}}\Lambda$ represents the forces and moments due to the kinematic constraints. The constraints of a system can be defined as $\Phi(\mathbf{q},t) = \mathbf{0}$ and can be differentiated two times and by applying the chain rule

$$\Phi_{\mathbf{q}}\ddot{\mathbf{q}} = -(\Phi_{\mathbf{q}}\dot{\mathbf{q}}_{\mathbf{q}})\dot{\mathbf{q}} - 2\Phi_{\mathbf{q}t}\dot{\mathbf{q}} - \Phi_{tt} = \Gamma(\dot{\mathbf{q}},\mathbf{q},t) \tag{2.2}$$

where $\Phi_{\mathbf{q}}$ is an important matrix called the Jacobian. The following system of equations is solved in LMS Virtual.Lab Motion

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & \Phi_{\mathbf{q}}{}^{T}(\mathbf{q},t) \\ \Phi_{\mathbf{q}}{}^{T}(\mathbf{q},t) & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_{A}(\mathbf{q}) \\ \Gamma(\dot{\mathbf{q}},\mathbf{q},t) \end{bmatrix} \tag{2.3}$$

The Jacobian needs to have full rank so all redundant constraints have to be removed from the system. These mixed differential-algebraic equations (DAEs) can be solved with multiple strategies [45]. Different MBS tools have different methods.

2.3.2. OVERVIEW OF LMS VIRTUAL.LAB

LMS Virtual.Lab is a package of analysis tools for noise, vibration and harshness (NVH) simulation. NVH depends a lot on loads and accelerations of bodies so MBS is also an important part of this package. MBS can be performed with LMS Virtual.Lab Motion. LMS Virtual.Lab by itself is the collection of validated analysis tools but it uses the CATIA V5 platform for the manipulation of geometrical objects and its graphical user interface (GUI). [1]

Figure 2.8 is a screenshot of LMS Virtual.Lab Motion. The top bar shows that an analysis file is open. The object tree of this file contains a link to the CAD product document that

[1] LMS Virtual.Lab will be migrated to the Siemens PLM Simcenter platform by April 2019.

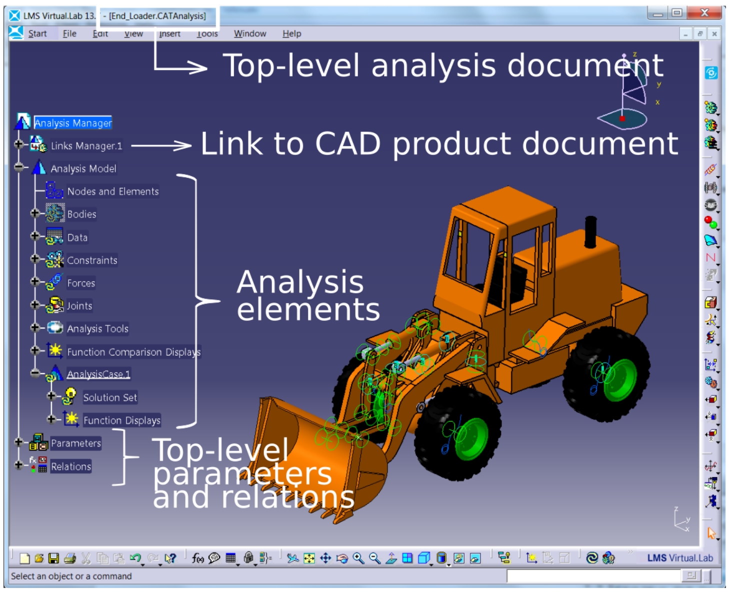


Figure 2.8: Screenshot of LMS Virtual.Lab Motion. The analysis document contains a link to the product document with all the CAD parts. The kinemtic and dynamic analysis elements such as forces and joints are inside the analysis document.

links to the CAD parts. In the object tree are also the kinematic and dynamic analysis elements that are added on top of the CAD geometry. The most important elements in the object tree are:

- Bodies: One body relates to one CAD part. The center of gravity (COG) is automatically calculated from the geometry information in the CAD part. The CAD parts have their own local reference system. However, the bodies are defined w.r.t. the absolute axis system of the analysis file.
- Motion axis systems (MAS): These are axis systems that are not present in the CAD part. All axis systems present in the CAD part are called Part Axis Systems (PAS). MASs can be free, which means that they are not bound to a body. Bound MASs follow the motion of the body during simulation. A MAS can be referred w.r.t. any axis system in the analysis file.
- Hard Points (HP): Similar to MASs, HPs are only present in the analysis document. A HP is a point with three coordinates that is positioned w.r.t. a MAS.
- Joints: They create relations between bodies by adding constraints to the axis systems that belong to that body. These axis systems can be PASs from the CAD part or MAS that are bound to the body.
- Forces: These are elements that apply a force during simulation. These can be user defined expressions or predefined elements. For example, a spring-damper element exerts a force between two bodies. All elements belonging to a body can be used such as MASs, PASs or elements in the CAD part such as vertices and edges.

The most important part of the MBS software is the solver. There are different solver algorithms available depending on the analysis type. LMS Virtual.Lab Motion is based on the DADS solver [46]. After solving the problem, the result files can be used to visualize the dynamics of the bodies. This is a fast way to inspect the solution. However,

This figure has been masked due to confidentiality

Figure 2.9: This figure has been masked due to confidentiality reasons.

for more insight into the results, all states of the bodies and elements can be plotted and exported for post-processing. The states of a body or other element are the variables that can be calculated but are not necessarily used during simulation. The variables that are used for solving are called the independent states. The dependent states can be calculated using only the independent states.

2.3.3. SUBMECHANISMS AND SHARED AXIS SYSTEMS

Submechanisms make it possible to synthesize different LMS Virtual.Lab Motion models in a modular fashion. They can be connected because they can share MASs and signal values. The communication is done via Motion Publications (MP). The process of connecting two different submechanisms is illustrated in figure 2.9. One MAS belonging to a body of submechanism B is being published with a high precedence. This means that the axis system acts as master and will impose its position and orientation to a coupling MAS in submechanism A. The coupling MAS of submechanism A will have to be published with low precedence. For this, a MAS with low precedence needs to be present in submechanism A to intercept the high precedence publication. It is as if an image of submechanism B's MAS is present in submechanism A. Now, a joint can be added between a bound MAS to submechanism A and the coupling MAS. The process described here is a simplified version because a lot of MASs, HPs, publications and formulas need to be added in the background. The steps are as follows.

STEP 1: HIGH PRECEDENCE PUBLICATION

In order to communicate one MAS to another submechanism, three additional MASs need to be created that will communicate the origin, Z-axis and X-axis to the other submechanisms. A fourth MAS (interpret this as the original MAS) communicates the body inertial properties of the first submechanism. The four MASs are then published with high precedence. The three additional axis systems can be called 'DATA_P', 'DATA_-Q' and 'DATA_R', respectively. Their orientation does not matter as only the origin will be used. Their positions are referred w.r.t. the main MAS of the body, whose orientation is important. If the origin of the main shared MAS is (0,0,0), then the coordinates of the origin of the 'DATA_P' MAS is (0,0,0), for 'DATA_Q' this is (0,0,d) and for 'DATA_R' this is (d,0,0). The value d is an arbitrary distance along one of the axes of the main shared MAS of the body. However, it is important that d is not too small because numerical errors might thus change the orientation of the MAS when it is rebuilt in the other submechanism. If the distance d has about the same magnitude as the accuracy

This figure has been masked due to confidentiality

Figure 2.10: This figure has been masked due to confidentiality.

Table 2.1: Maximum $max(\epsilon)$ and minimum $min(\epsilon)$ corresponding angular error for $\vec{P}$ with different length. The accuracy d is constant at 0.001 mm

$\|\|\vec{P}\|\|$	$max(\epsilon)$	$min(\epsilon)$
1 mm	0.08099 deg	0.0006463 deg
10 mm	0.008106 deg	6.472e-5 deg
1000 mm	8.106e-05 deg	8.542e-07 deg

tolerance a, then the orientation of the axis system might deviate significantly. This can be illustrated with figure 2.10. As the distance d grows, the effect of a becomes smaller.

The default accuracy of LMS Virtual.Lab is as a standard set to 1e-3 mm. This accuracy can be transformed into an angular accuracy ϵ by calculating the angle between the correct direction vector, and the vector where the accuracy error is added to every coordinate.

$$\epsilon = \frac{\vec{P} \cdot \vec{P^*}}{||\vec{P}||.||\vec{P^*}||} = \frac{P_x P_x^* + P_y P_y^* + P_z P_z^*}{\sqrt{P_x^2 + P_y^2 + P_z^2}\sqrt{P_x^{*2} + P_y^{*2} + P_z^{*2}}} \tag{2.4}$$

It will not be proven here that the maximum ϵ can be found on one of the corners of the cube for any orientation of Z_L. However, since the edges of the cube are a linear combination of the coordinates of the absolute axis system and the 3D distance formula is a convex function, this is a reasonable assumption. For example, if a length of 1 mm for d (= norm of $\vec{P}$) is chosen, and a is 0.001 mm, then the maximum angular error is 0.08099 degree. This corresponds with $\vec{P}$ being parallel with one of axes of the absolute reference system. Table 2.1 shows the minimum and maximum error for more values of $||\vec{P}||$. To find the minimum and maximum error, ϵ was calculated for each of the eight corner points of the cube that represents the accuracy bounds of $\vec{P}$. This example shows that it is better to take a large value for d in order to have a more accurate representation of the orientation of the MAS.

STEP 2: LOW PRECEDENCE PUBLICATION

Since four MASs are needed to share one MAS, four MASs are also needed to receive the data. The receiving MASs will take the position and orientation if they are published with low precedence under the same name as the MASs with high precedence. Subsequently, for each P, Q and R motion axis system a hard point is created. The value of each coordinate of the HP is linked to the origin of the corresponding MAS using a function that is available in LMS Virtual.Lab Motion. After adding the relations to the HPs, the

This figure has been masked due to confidentiality

Figure 2.11: This figure has been masked due to confidentiality.

three HPs correspond to the origins of the three data MASs. Finally, the main MAS can use the three HPs as origin, Z-axis and X-axis to position and orient. The main MAS is also published with low precedence to receive the inertial properties of the body with high precedence. In this way an image is made of a MAS coming from a different submechanism. These two steps are visualized in figure 2.11.

2.4. HIGH-LIFT SYSTEM DESIGN

This section gives a broad overview of the state-of-the-art of high-lift system design. First, an overview is presented of the purpose and requirements of high-lift devices. Then two different design approaches are presented for high-lift design. One is traditional while the second is a KBE approach. Subsequently, an overview of the most common trailing-edge high-lift devices is presented. Finally, possible actuation architectures are discussed.

2.4.1. REQUIREMENTS

The main function of a high-lift device is to move the flap to influence the aerodynamic properties of the wing. Furthermore, there can be requirements on structural integrity, maintenance, weight etc.

AERODYNAMIC REQUIREMENTS

The top level requirement of the high-lift system is to translate and rotate the flap surface to obtain satisfactory performance of the aircraft during all flight stages [47]. Take-off performance is influenced by the lift-over-drag ratio of the aircraft. To achieve the best performance, the mechanism needs to translate the flap while only deflecting at a small angle. During landing, the drag force is less important and the flap can deflect more than during take-off in order to increase the lift coefficient. The optimal combination of translation and deflection can be found by carefully studying high-lift aerodynamics. A summary of high-lift aerodynamics, based on the famous paper of Smith [48], can be found in appendix A. The optimal performance of the aircraft during take-off and landing can be derived by using aircraft flight mechanics which can be found in appendix B.

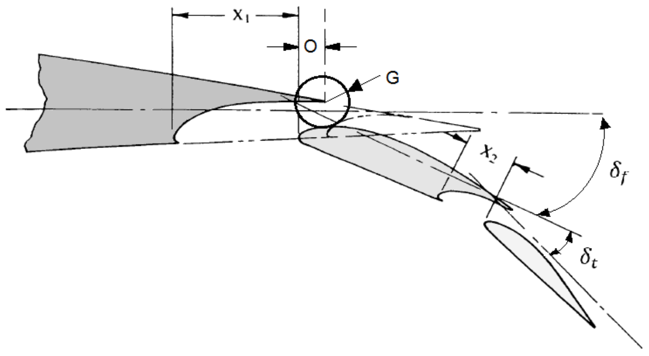


Figure 2.12: Definition of gap G, overlap O, flap deflection δ_f, tab deflection δ_t and Fowler Motion X_1 and X_2. [Based on [13] and [50]]

In order to achieve the desired high-lift performance of the aircraft, a mechanism needs to be designed that can bring the flap to the correct positions. The collection of all possible positions and orientations that the flap can reach is the flap trajectory. It is the goal of the design team to come up with a mechanism that moves the flap along a trajectory that passes through two, three or more prescribed trajectory points. What happens in between the points is usually less clearly defined. However, smooth behavior is always required.

Four important parameters are frequently used for describing high-lift devices: gap, overlap, deflection and Fowler motion. Figure 2.12 illustrates their definition. A trade-off between Fowler motion and flap deflection can best be visualized in a plot such as figure 2.13. During take-off, Fowler motion is more important than deflection so in figure 2.13 the link-track mechanism performs best. A disadvantage is the negative deflection in the initial part of the trajectory. Deflection angle, Fowler motion, gap and overlap can be seen as 2D kinematic characteristics. However, 3D kinematics is also important. The inboard and outboard flap need to stay close together during every stage of deployment. If there is a kink in the wing plan-form, the flap needs to move also in the span-wise direction. This leads to a more complex mechanism compared with flaps that move on a cylinder or a cone. Conical motion is necessary to keep a smooth span-wise lift distribution [47]. Figure 2.14 shows the difference between conical and cylindrical motion. To allow conical motion, one side of the flap needs to move further than the other side. In order to make conical motion possible, the joints between the deployment mechanism and the flap need to allow for two extra degrees of rotation, thereby only fixing pitching rotation. Such joints are called swing links [49].

NOMINAL FLIGHT

Nominal load cases involve the expected loads on structural elements in their normal state. The flight maneuver envelope is specified by EASA CS-25 [52]. Figure 2.15 shows the flight envelope based on design airspeeds and limit loads. The design airspeeds for flaps in stowed position are specified in paragraph CS 25.335. The limit loads for flaps in stowed position can be found in paragraph CS 25.337 and CS 25.341. The flight envelope with limit loads and design airspeeds when flaps are deployed is specified in paragraph CS 25.345.

Besides static loads, also dynamic behavior needs to be taken into account. CS 25.571 specifies requirements about damage tolerance and fatigue. An evaluation of the

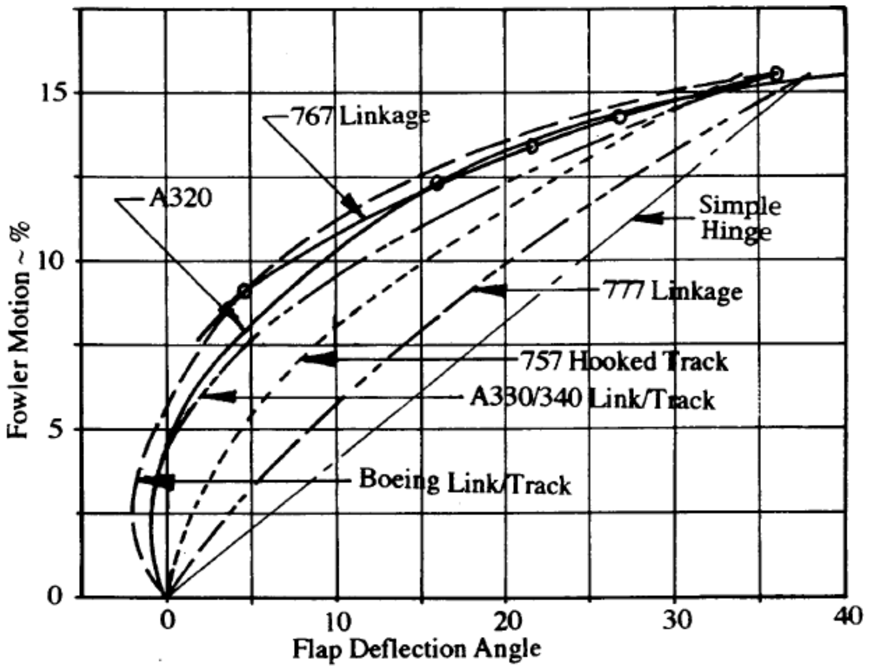


Figure 2.13: Flap deflection angle vs. Fowler translation for some deployment mechanisms. Note that the link-track mechanism first has some negative deflection before deflecting positively. [13]

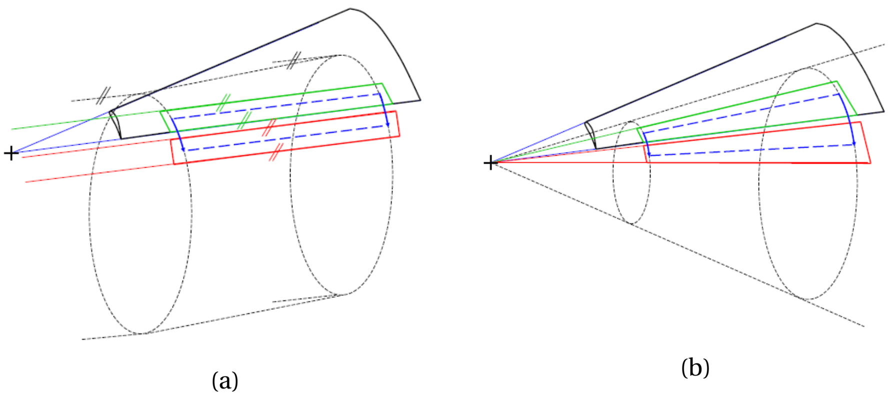


Figure 2.14: (a) Cylindrical motion (b) Conical motion. [51]

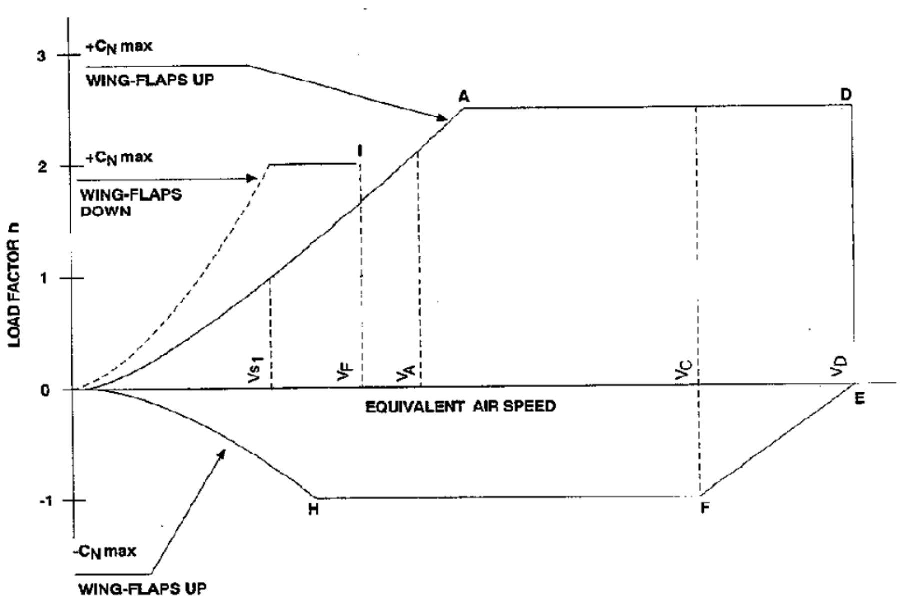


Figure 2.15: The strength requirements must be met at each combination of equivalent airspeed and load factor and within the boundaries of the manoeuvring envelope. (par. CS 25.333) [52]

strength, detail design, and fabrication must show that catastrophic failure due to fatigue, corrosion, or accidental damage will be avoided throughout the operational life of the airplane. The service history of similar airplanes can be studied for that purpose. For damage tolerance, the evaluation must include a determination of the probable location and modes of damage to fatigue, corrosion and accidental damages. The evaluation must incorporate repeated loads and static analysis supported by test. Fatigue analysis must be supported by test evidence.

OPERATION

Operational requirements are concerned with requirements specifically during operation of the high-lift devices. Paragraph CS 25.697 states that:

- Each lift device control must be designed so that the pilots can place the device in any take-off, en-route approach, or landing position.
- The lift device's control must be designed to retract the surfaces from the fully extended position, during steady flight at maximum continuous engine power at any speed below V_F + 17 km/hr.
- The rate of motion must give satisfactory flight and performance characteristics under steady or changing conditions of airspeed, engine power and airplane attitude.
- Design of each high-lift device must make inadvertent operation impossible.

MANEUVERING

Maneuvering load cases investigate the loads on components during flight maneuvers like pitching and turning. These maneuvers create inertial forces on the high-lift devices and wings which needs to be analyzed. Also the deflection of primary control surfaces might lead to a change of the lift distribution which leads to higher loads.

Failure

An important set of load cases comes from component failure. For example, ice accretions can lead to the jamming of one component which can lead to increased loads on other components. These failure cases can dominate the sizing of components according to Zaccai [53]. According to paragraph CS 25.671 [52, 1-D-6]:

> The airplane must be shown by analysis, test, or both, to be capable of continued safe flight and landing after any of the following failures or jamming in the flight control system and surfaces within the normal flight envelope, without requiring exceptional piloting skill or strength. Probable malfunctions must have only minor effects on control system operation and must be capable of being readily counteracted by the pilot. Any jam in a control position normally encountered during take-off, climb, cruise, normal turns, descent and landing , unless it is extremely improbable, should be alleviated. Also a runaway to an adverse position should be accounted for.

2.4.2. The design approach

Two different design approaches for trailing-edge high-lift devices are presented. First a traditional approach presented by Boeing during a conference [54]. The second is a KBE application that is used to extend the MMG from TU Delft [55].

The traditional approach

The traditional engineering approach for high-lift devices proceeds sequentially [54]. Figure 2.16 illustrates the design process. First, the geometry and airfoil of the wing and flaps is calculated, based on the aerodynamic requirements. Then, the physical layout of the high-lift system is selected based on kinematic analysis. Subsequently, the actuators are added and the required actuation torque is calculated after adding the aerodynamic load that acts on the flaps. Finally, structural analysis is used to determine the stresses. Usually, a few iterations are needed to optimize weight and the exact movement of the flaps. The process flow as described above suggests that it is assumed that the high-lift system can be isolated and designed separately from the other subsystems. Figure 2.16 shows no interaction with the actuation team, aerodynamics team or manufacturing division. Other authors describe similar 'uncoupled' design processes [56][57].

A knowledge based engineering approach

An improvement on the traditional sequential approach is the use of design modules to speed up the design [58]. The goal was to provide reliability, weight, maintainability and cost estimates at the beginning of the design process. This example illustrates the shift towards a more front-loaded design. The KBE application for high-lift system design extends the MMG [55]. The flow-chart of the application is shown in figure 2.17. The application starts with the geometry of a clean wing in cruise flight. The flap planform is then determined based on low-speed requirements. An aerodynamics module determines the aerodynamic loads on the wing and flap. The number, position and type of deployment mechanisms has an effect on the trajectory. Therefore, a trade-off between aerodynamic performance and kinematics has to be performed. Engineering handbooks and basic analytical methods are used to determine the most critical loads

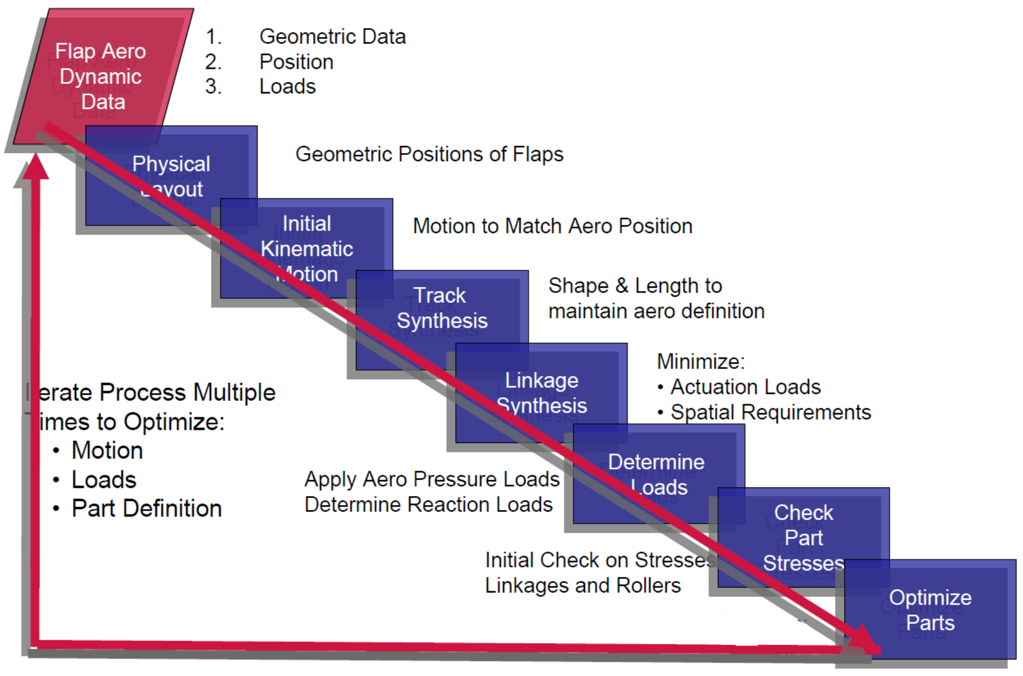


Figure 2.16: Process flow of airplane flap design according to Boeing. [54]

on which sizing of the mechanism is based. Finally, two modules calculate the weight of the mechanism and the power that is required to drive it.

2.4.3. TRENDS IN TRAILING-EDGE HIGH-LIFT DEVICE DESIGN

Rudolph [13] provides an extensive list of the configurations of high-lift devices that are present on modern commercial airliners. Zaccai [53] also extensively describes the configurations on existing aircraft. Figure 2.18 gives an overview of the configurations in a chronological fashion. It is clear that in the 1970s and 1980s triple and double slotted flaps were most preferred. However, recent airliners now fly with double slotted or single slotted flaps. The evolution of high lift devices went from very simple systems such as split and plain flaps in the early days of aviation to very complex triple slotted Fowler flaps and then back to simpler single slotted flaps in the last decades. The reason for this trend is likely the extra weight that triple slotted flaps carry.In addition, they have a higher chance of system failure and a higher maintenance and manufacturing cost [13]. Although a flap consisting of multiple elements can produce a much larger lift increase, drag inducing 3D effects and larger fairings reduce their aerodynamic efficiency [56].

Rudolph [13] points out that it is not evident to use single slotted flaps because they often require the aircraft to fly at a large pitch angle which impedes pilot vision. However, a combination of developments made the single slotted flap possible. Firstly, the replacement of high-speed ailerons with differential flaps (a.k.a. flaperons) and the removal of a thrust gate allows for one continuous flap surface. Secondly, the introduction of drooped spoilers which are linked to the flaps allows for a larger flap deflection without flow separation.

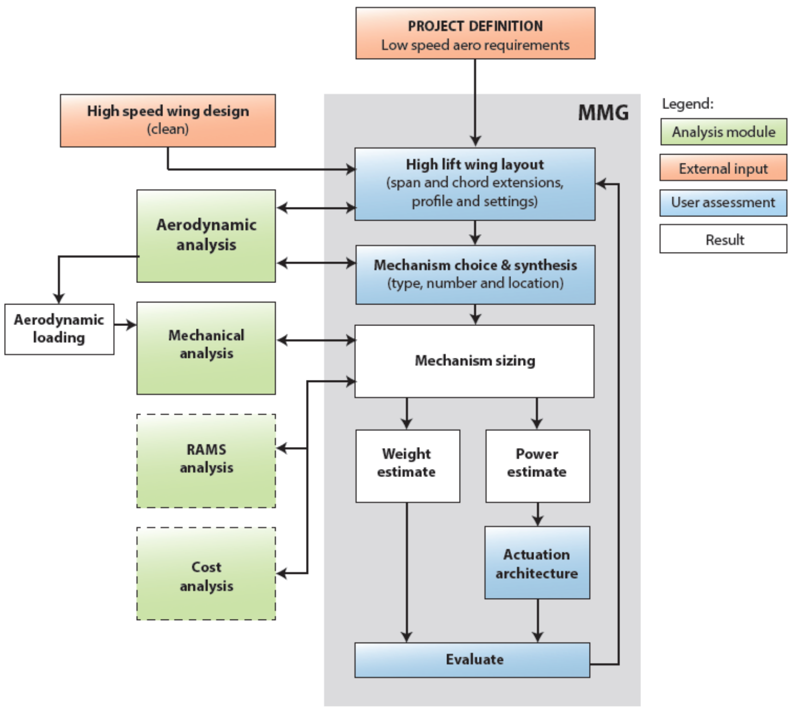


Figure 2.17: Preliminary high-lift design process for a KBE application. The dotted lines indicate possible additional analysis modules. [53]

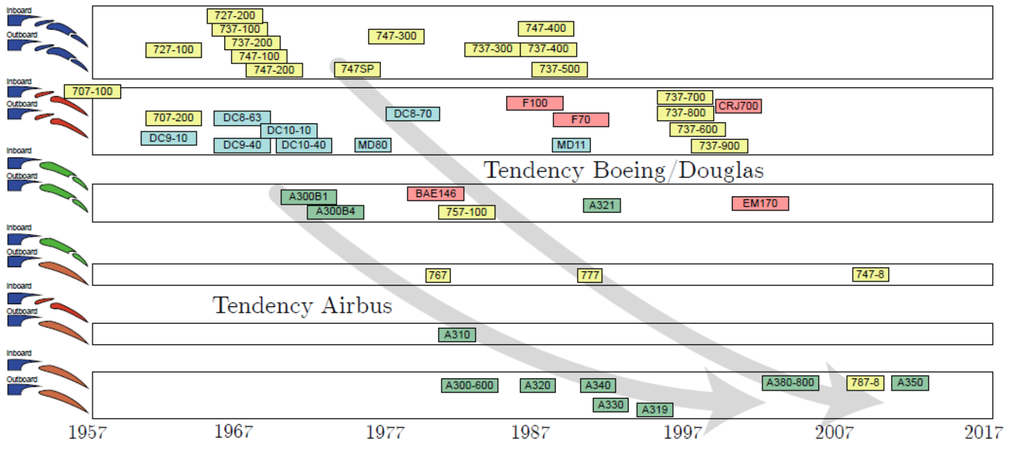


Figure 2.18: There is a trend towards the more simple single slotted flap configuration for trailing-edge high-lift devices. [59]

Table 2.2: Trailing-edge flap specific weights (lb/ft^2). Weight savings for composites for the flap are taken into account. A synchronized shaft with jack screw actuation is used. [13]

	Flap type					
	Single slotted Hooked track	Fixed vane Hooked track	Art. vane Hooked track	Double slotted Hooked track	Triple slotted Hooked track	Single slotted Link-track
Flap panels	2.7	3.0	3.5	4.8	5.5	2.7
Supports	3.0	3.2	3.8	4.7	5.6	1.5
Actuation	2.2	2.2	2.3	2.4	2.5	2.0
Fairing	1.0	1	1.15	1.3	1.4	0.1
TOTAL	8.9	9.4	10.7	13.2	15.0	6.3

There is a less clear trend for the deployment mechanisms of trailing-edge high-lift devices. Most popular mechanisms can be classified into five types: drooped hinge, various 4-bar mechanisms, hooked track variations and a link-track mechanism. There are many factors which influence the choice for an actuation system. Aerodynamics of the fairing, reliability, cost, Fowler motion and weight are important ones. Appendix C gives an overview of the application of deployment mechanisms on commercial airliners. More detailed drawings are available in another report of Rudolph [60].

An important aspect of high-lift devices is the weight. It is possible to do an estimation of the high-lift devices based on formulas which have been constructed using empirical data. Anderson [61] created formulas based on manufacturer data for weight estimation of high-lift devices. Rudolph [13] used those formulas to create the data in table 2.2 which gives an accurate overview of the specific weights of trailing-edge high-lift devices. It becomes clear that the single slotted flap leads to the lightest solution. Torenbeek [62] offers another weight estimation for wing components including high-lift devices.

2.4.4. ACTUATION SYSTEM ARCHITECTURES

Rudolph [13] proposes two architectures for actuating high-lift devices. Firstly, it can be done separately for each panel with a set of actuators that are independent of other control systems. Secondly, it can be done with a central architecture that has a transmission shaft that is powered by a power drive unit (PDU). Modern high-lift actuation systems use the centralized architecture with ball-screw actuators or rotary actuators [59]. According to Rudolph [13] screw jacks, rotary hinges and rack-pinion drives are also used. Figure 2.19 shows the actuation system for the A330/A340 airliner. This system is similar for all Boeing commercial airliners starting from the Boeing 707.

Besides the common centralized actuation architecture, the newest airliners (Boeing B787 and Airbus A350) have varying camber during cruise. Varying the camber allows for a more optimized wing profile as the weight of the aircraft decreases during cruise which leads to a significant reduction in fuel consumption. Next to varying camber, the Airbus A350 has differential flap settings (DFS). This means that the inner and outer flap can move independently and this is done with a motorized gearbox between the inner and outer flap. In this way the center of lift can be altered for load alleviation which allows for a lighter wing structure [59].

Most modern airliners use a hydraulic actuation system. However there is a tendency to make high-lift systems electrical. Initiatives like the European Commision's Hori-

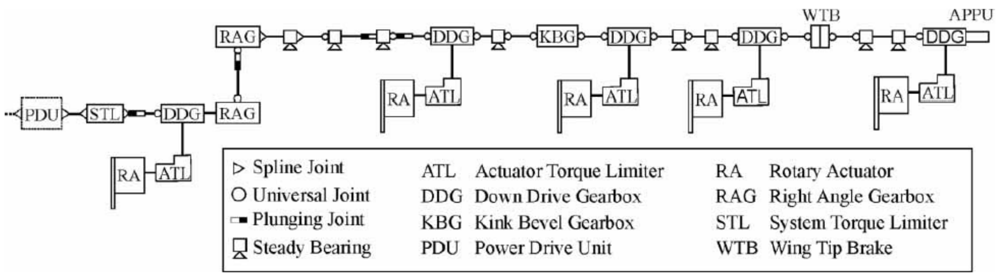


Figure 2.19: Right wing flap actuation system of the A330/A340 airplane. The wing tip is positioned on the right. [63]

zon2020 [64] investigate novel technologies contributing to more electric aircraft (MEA). The break-by-wire system installed in the B787 Dreamliner is a first step towards MEA. Boeing has adopted this system because it reduces weight and because the modular approach of electrical systems leads to faster assembly and testing. Airbus also has set a step towards MEA by replacing one of its two redundant hydraulic systems with electro-hydrostatic actuators (EHAs) [65]. Although a major issue was the cooling, the overall weight reduction was worth it. Jones [66] sums up the work that had been performed up to 2002. Studies showed potential gain on maintenance cost and engine efficiency since less bleed air is needed for MEA. In a more recently published study in 2014 Chakraborty et al. [67] state that it is generally accepted that MEA have lots of benefits. However, it is not clear what the best actuation architectures are. For example, is an EHA to be preferred over an electro-mechanical actuator (EMA)? The use of EHAs is far from new as they are already applied as horizontal stabilizer trim actuators. Figure 2.20 illustrates the working principle of the EHA and three other types. The research concluded that EMAs for non-primary surfaces and EHAs for flight-critical control surfaces could be marginally superior in weight compared with an all EHA configuration (with flaps having EMAs). Another point that is made in this paper is that the actuation subsystems need to be integrated in the design process much earlier. This is where a traditional design approach fails and novel design approaches such as ABD can contribute.

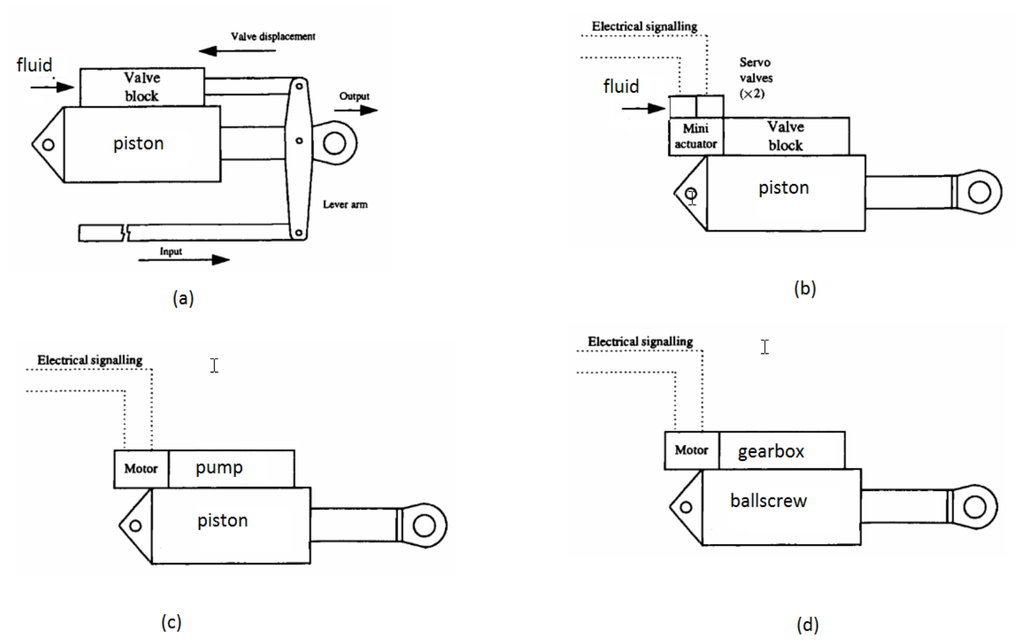


Figure 2.20: Four actuator-control types: (a) mechanical signalling and feedback (b) electrical signalling with electro-hydraulic servo valves (c) electrical signalling with a electro-hydrostatic actuator (d) electric signalling with an electo-mechanical actuator. (Based on [67])

3 Methodology

Section 2.2 discussed architecture-based design in depth and gave an example of how submechanisms could be synthesized into a large multi-body simulation. Section 2.3 explained the practice of synthesizing modular multi-body simulation models, which is similar to what can be done with 1D multi-physics models. It should be clear from the example that the synthesis of submechanisms currently has many limitations. The example also showed that there is not really an underlying methodology for how these multi-body models should be modelled. This chapter describes the development of a methodology that solves the shortcomings. The task was realized by reflecting on the paradigms that were explained in section 2.1: object-oriented programming, model-based systems engineering and knowledge based engineering. Section 3.1 sums up the requirements that a possible methodology needs to satisfy. Section 3.2 solves a problem that was identified during the development of the methodology. The final section goes more into the details of the modelling process of individual components so they can communicate with other components.

3.1. Requirements for the methodology

The main requirement for the methodology is that it should enable the efficient synthesis of parametrically built multi-body simulation models to realize an architecture configuration. Use is to be made of the prototype version of LMS Imagine.Lab System Synthesis and the submechanism functionality in LMS Virtual.Lab Motion. There are some examples of the current capabilities. However, the example slat system in section 2.2 showed many shortcomings, such as unparameterized geometry and badly understandable models. This section defines the areas where improvement is needed in order to employ ABD. Later on, this will help to explain why certain decisions are made.

Only a few parameters can be changed in the slat example and they are not related to geometry. Up till now, there is no example of submechanisms whose geometry is driven by other submechanisms. It can be said that the CAD geometry, such as surfaces and solids, in the submechanisms serves a purely visual function. Therefore, the model behavior is mainly determined by hard points (HPs) and motion axis systems (MASs) which are rather primitive representations of geometry. For simple mechanisms this is not a problem, but more complex mechanisms require the use of CAD to determine the position and orientation of interface points between components. In the slat example, changing the geometry of one submechanism almost always leads to an incorrect mechanism. Therefore, the methodology should enable highly parameterized models.

Design intent is the way of how associative relations between model features were created. It allows the parameters of a model to be changed while the model still correctly represents the original object [68]. A modeller usually has a clear idea about the design intent of his model. However, the design intent should also be clear to other users. Therefore, the model hierarchy and model history should be intuitive, or a standardized modelling procedure should be used. The models of the slat mechanism example in section 2.3 are not easy to understand. First of all, this is because the method of publishing motion axis systems (MASs) is complex. Secondly, the modeller did not use a standard procedure. His design intent might be clear to him, but it is very difficult to understand for others. Therefore, the methodology that will be developed should bring uniformity to the modelling process. In this way, the cumbersome process of creating the shared MASs will also be easier to automate later.

3.2. ASSOCIATIVE MODELLING OF SUBMECHANISMS

A methodology is developed that fulfills the requirements that were expressed in section 3.1. This section explains that the direction of model dependencies needs to be taken into account during synthesis. A closer view is provided on submechanism associativity and how this leads to the need for an additional sorting algorithm. Although the procedural modelling approach will not be used, the techniques of KBE can still be useful to come up with a method to model robust submechanisms.

3.2.1. GEOMETRICAL DEPENDENCIES

Multi-body simulation (MBS) models consist of geometry and elements such as joints, forces, drivers, contact forces etc. The geometry can have various degrees of detail. For example, it might only consist of points and axis systems. The set of these basic elements is called the 'skeleton model'. In other cases, a complete CAD model with solid and wire-frame geometry can be associated with it. However, detailed geometry can mostly be condensed into a skeleton model. Elements such as joints and forces depend on the skeleton model. The points and axis systems of the skeleton model that are shared with other submechanisms are called interfaces. The interfaces exchange geometrical information so submechanisms can connect to each other. This is an important difference between 1D multi-physics models and MBS models because 1D models can always be connected (as long as their interface type matches) while MBS models need to adjust their geometry. This is illustrated in figure 3.1. The details of how interface axis systems are shared were presented in section 2.3. However, a choice needs to be made on how the location of the interface axis systems is determined. There are two options to do this.

The first option is to create a 'parent' model where all important interface points are specified. This parent model acts as a skeleton from which the individual subsystems receive their geometrical input data to position themselves and adapt their geometry. Figure 3.2a illustrates this approach, which is elegant but has two shortcomings. Firstly, it defies the principle of having a modular approach because the designer would have to recreate the skeleton for each specific system. The idea behind ABD is that all knowledge should be contained inside the submechanisms themselves (and partially

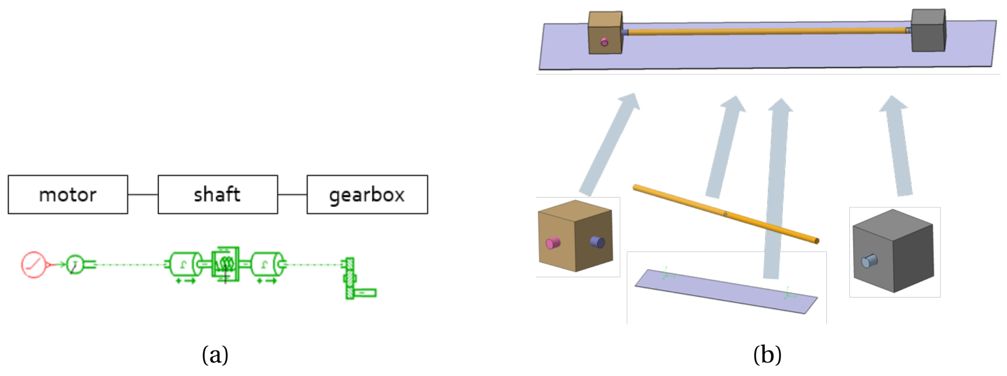


Figure 3.1: (a) Representation in LMS Imagine.Lab Amesim of a bond-graph model of a shaft between a motor and a gearbox. (b) Assembly of the multi-body models of the shaft, motor and gearbox. In this case, the length of the shaft adapts to the position of the gearbox and motor.

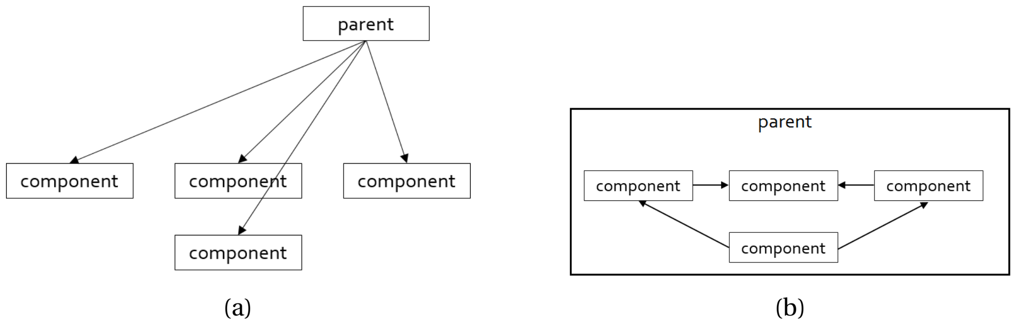


Figure 3.2: (a) Option1: a centralized approach where each component receives design information from a parent (b) Option 2: a decentralized approach where each component receives design information from the component it depends on.

in the architecture definition). Secondly, the user needs to open the tool where this skeleton is made. This creates extra work every time a new system needs to be assembled even if it is just a geometric change. To demonstrate this, take as an example the MBS model in figure 3.1b of a motor and gearbox connected by a shaft and both mounted on the support plate. A skeleton model would contain four axis systems that specify the location of the interface point between the motor and the support plate, the gearbox and the support plate, the motor and the shaft and the interface point between the gearbox and the shaft. During synthesis, each of the four submechanisms would position and shape themselves w.r.t the interface axis systems in the skeleton model. In other words, all interfaces of the skeleton model are outputs and the interfaces of the submechanisms are inputs.

The second option is to not have a model where all interfaces are defined and to let submechanisms provide the master interfaces. This option is illustrated in figure 3.2b. Because of the shared axis systems, a parameter change in one component can propagate through the system. In the case of the MBS model example in figure 3.1b, it could be that the position of the support plate is fixed and provides the locations of its connections to the motor and the gearbox. In turn, the latter two components move to the right position and provide the connection with the shaft. The shaft will adapt in order to connect to both the the gearbox and motor. Effectively, the length will be influenced by the support plate and the gearbox and motor. As this approach does not require a specific skeleton model, it more closely resembles the modular approach that

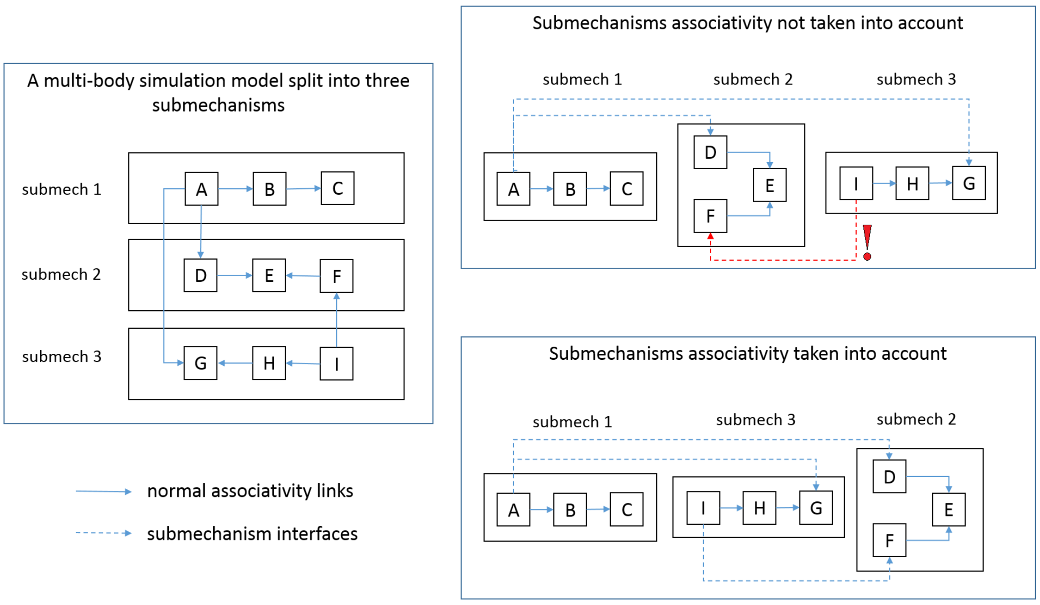

A multi-body simulation model split into three submechanisms
submech 1
submech 2
submech 3
A
B
C
D
E
F
G
H
I
normal associativity links
submechanism interfaces
Submechanisms associativity not taken into account
submech 1
submech 2
submech 3
Submechanisms associativity taken into account
submech 1
submech 3
submech 2

Figure 3.3: Synthesis sequence for a system simulation model split into three submechanisms. The top right sequence does not take dependencies between submechanisms into account. The bottom right sequence is correct.

is the required for ABD. Therefore, this option is preferred and will the basis for the methodology in this thesis.

3.2.2. SYNTHESIS CAUSALITY

We allow the submechanism to adapt to other submechanisms via shared axis systems. As a consequence there is no explicit rule how the master and slave interfaces should be divided between two submechanisms. This means that multiple model versions can be possible for the same system component and that the right version with the correct inputs and outputs needs to be selected for every component. In case of the example (see figure 3.1b), the information was provided from the support to the motor and gearbox and from the latter two towards the shaft which had to adapt itself. Alternatively, a solution could be that the length of the shaft is specified, and that this provides location information to the motor and gearbox and consequently the length of the support plate is adapted.

By splitting up the system into independent submechanisms, the CAD software can only determine the correct hierarchy inside every submechanism. Therefore, extra work needs to be performed to determine the correct hierarchy of the submechanisms. This problem is illustrated in figure 3.3 which shows a sample system that consists of 9 components called A to I. It is split in three submechanisms. The top right sequence pays no attention to the dependencies between these submechanisms; accordingly, they are ordered in a standard fashion from 1 to 3. In the bottom right sequence, the submechanisms are ordered correctly so that all components get their information from up-to-date components. The direction in which information is exchanged between submechanisms is called the synthesis causality because it needs to be taken into account during synthesis.

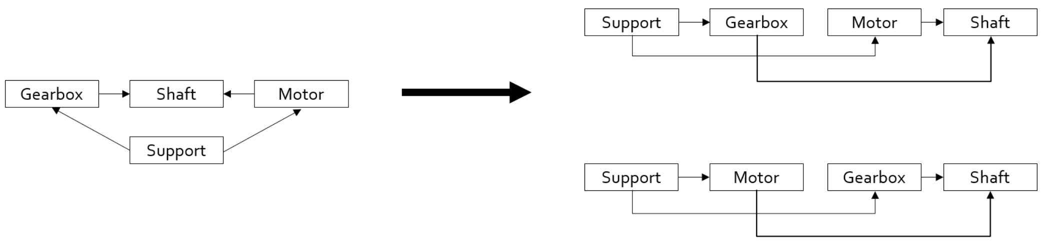


Figure 3.4: Example of topological sorting for a simple architecture with the two solutions on the right.

3.2.3. THE CORRECT SYNTHESIS SEQUENCE

The consequence of the synthesis causality is that not every synthesis order is acceptable because if a submechanism with an interface point that needs to receive information is added, the submechanism that should provide the information needs to be already assembled. As a result, before synthesis of the system, the number of inputs and outputs of every selected submechanism model needs to be analyzed and a correct synthesis order needs to be identified. In case of the example in figure 3.1b, the support plate needs to be included first. Then, the motor and the gearbox have to be synthesized. Lastly, the shaft is added which will adapt itself to the positions of the motor and gearbox.

Finding the correct order of an architecture with causality can be translated into a mathematical problem, viz. the topological sorting of a directed acyclic graph (DAG). This is a well-studied problem and there are many algorithms available in literature that solve this problem. The problem can be defined in the following way. Let $G = (V, E)$ be a DAG where V is the set of vertices and $E \subset V \times V$ is the set of edges. A topological sort is a total order of V such that for every edge $(u, v) \in E$, vertex u precedes v [69]. Fig. 3.4 shows an example of topological sorting applied to a simple architecture. It also shows that multiple solutions may be possible.

The topological sorting algorithm that was implemented in this methodology is Kahn's algorithm [70]. This rather simple algorithm is not the most efficient way of sorting DAGs, although it scales linearly $\mathcal{O}(\#(edges) + \#(vertices))$, but it was the fastest to implement and it provides enough rigidity for this research. The pseudo-code is provided in algorithm 1. The in-degree is the number of incoming edges that a vertex has.

3.3. THE MODELLING PROCESS

The rough lines of the methodology that enables the synthesis of MBS models have been drawn. However, a clearly defined modelling process that allows the creation of modular parametric MBS models is also important. The developed modelling process consists of six steps which are illustrated in figure 3.5 with an example. Firstly, the inputs of the model are determined. These are the shared MASs, as discussed in section 2.3, and the design parameters. The second step is to build the model geometry based on the inputs that were defined in step one. Subsequently, elements such as joints, constraints, forces etc. are added on top of the geometry. In the fourth step, the model is tested as a stand-alone simulation. In the fifth step, output axis systems are defined on top of the geometry that will link with the input axis systems of other submechanisms. The final step consists of saving and documenting the simulation model. The steps are explained in greater detail.

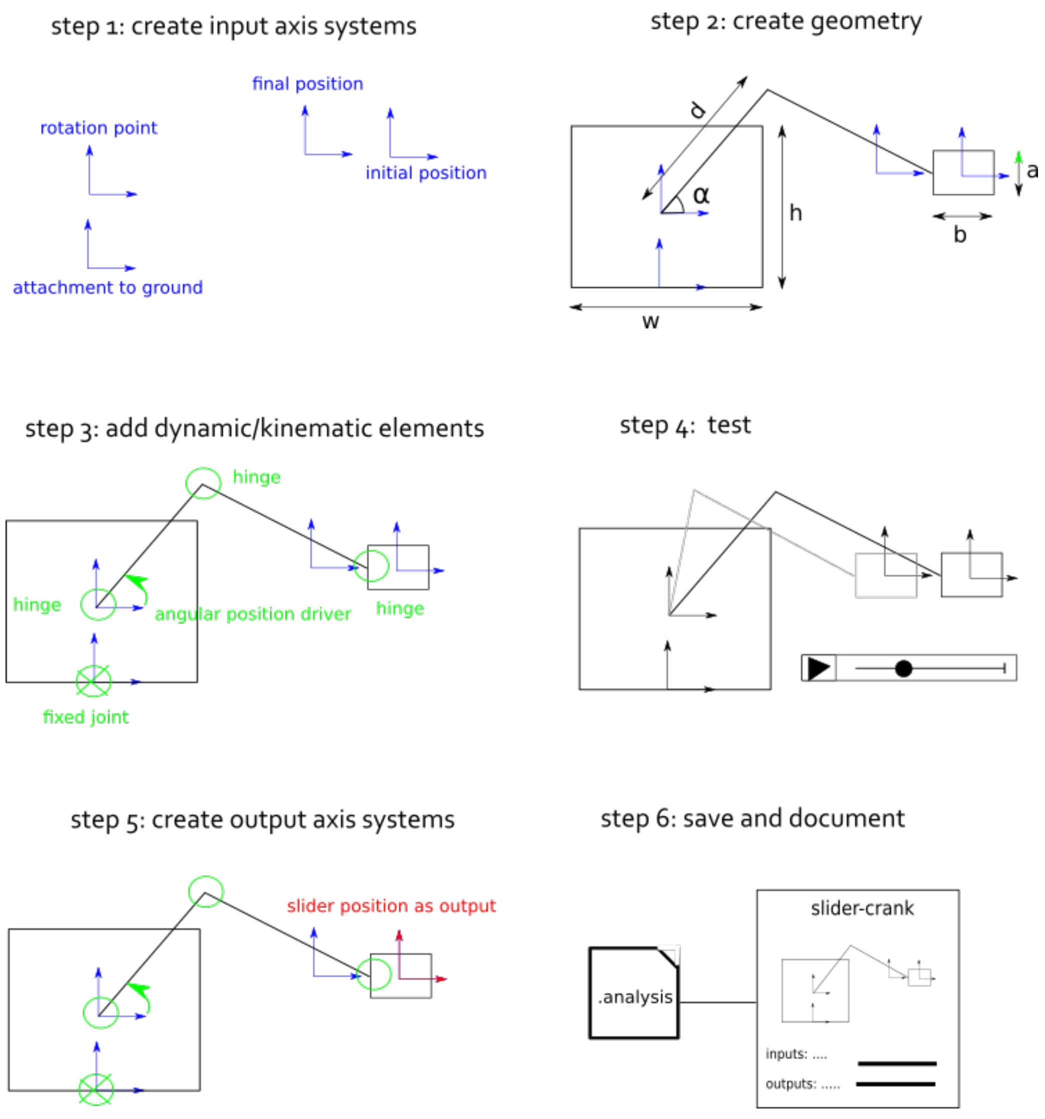


Figure 3.5: The six steps of the modelling process applied to a simple slider-crank mechanism.

Algorithm 1 Kahn's algorithm

```
1:  L = empty list that will contain correct ordering
2:  B = dictionary containing neighbors of every vertex
3:  N = dictionary containing in-degree of every vertex
4:  S = list of all vertices with in-degree 0
5:  count = 0
6:  while S not empty do
7:      increase count by 1
8:      remove a vertex v from S
9:      add v to end of L
10:     for each neighbor n of v do
11:         decrease in-degree by 1
12:         if in-degree is 0 then
13:             add n to S
14:         end if
15:     end for
16: end while
17: if count not equal to length of N then
18:     raise error because of a cycle
19: end if
```

STEP 1: INPUT AXIS SYSTEMS

Modelling a submechanism that represents a rigid (or flexible) body starts with the identification of the interaction between other submechanisms. First of all, these can be locations where the submechanism connects to other submechanisms. Submechanisms can also interact without having a direct connection. For example, a force between two bodies can have a magnitude and direction that depends on distance. It is also possible that a submechanism shares an axis system without the intention that other submechanisms connect (for example, to communicate a certain direction or trajectory). Besides axis systems, submechanisms can also interact via a quantity flow, for example an electrical current representing a control signal.

When the modeller starts from an empty document, the first thing he does is to add the interface input axis systems. However, it is also possible that the model does not rely on other submechanisms so there is no need to create input axis systems. This is usually the case when a submechanism is a boundary of the system that is being investigated. For example, the airframe of an aircraft can be fixed to the ground when analyzing the dynamics of control surfaces. Sometimes there is an existing document or geometry, such as when the geometry of a submechanism is fixed because the design is frozen. If the geometry is not parametric, it is more difficult for the body to fit in an architecture as it cannot change anymore. However, its position and orientation can still depend on other submechanisms.

As was explained in section 2.3, the CAD document in LMS Virtual.Lab cannot readily make use of MASs and other elements in the analysis document. A copy needs to be created of the motion axis system into the CAD document. A part axis system (PAS) can be positioned and oriented if provided with an origin point and two directions. Use is made of the three hard points (HPs). Section 2.3 explained that these hard points represent the origin, a point on the z-axis and a point on the x-axis. Figure 3.6 illustrates

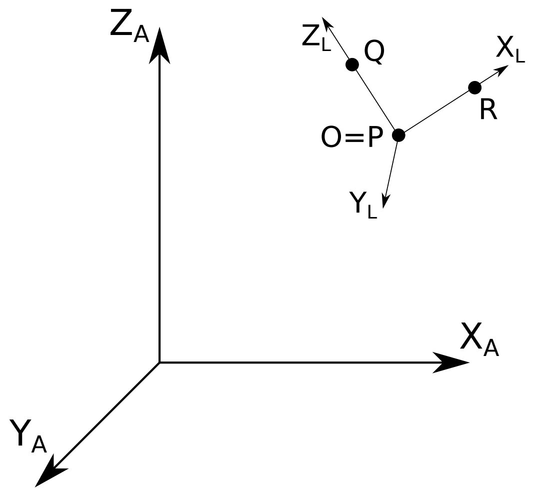


Figure 3.6: The P, Q and R hard point with respect to local and absolute axis system.

this. Equations 3.1 construct the part axis system with origin vector O, x-axis vector X and z-axis vector Z. In this way, an axis system defined by three points is converted into a definition consisting of a point and two directions. A macro was made which automates the process of creating the MASs and HPs. Additionally, it also creates automatically a PAS in the CAD document. The code of this macro can be found in appendix E.

$$\begin{aligned}
O_x &= P_x \\
O_y &= P_y \\
O_z &= P_z \\
X_x &= R_x - P_x \\
X_y &= R_y - P_y \\
X_z &= R_z - P_z \\
Z_x &= Q_x - P_x \\
Z_y &= Q_y - P_y \\
Z_z &= Q_z - P_z
\end{aligned} \tag{3.1}$$

When a MAS in Virtual.Lab is created it is by default positioned and oriented on the absolute reference frame. The modeller can change the orientation and position values that are expected when the model would be positioned in an assembly. This is similar to setting a default position of the MAS. During synthesis the default values are overwritten by the corresponding publication with high precedence. During modelling, the shared MASs can be translated and rotated to check if the model is robust enough. Since the PASs are linked to the MASs, it suffices to move the MASs. This can be done by specifying one of the following parameter sets:

1. Bryant angles
2. Euler angles
3. Euler parameters a.k.a. quaternions
4. PQR points (i.e. origin and two directions)

A macro was made which automates the process of transforming the shared MAS. Since the shared MAS is defined by three points, a transformation means that the new positions for the three points need to be found. Accordingly, this macro saves a lot of work. The documentation can be found in Appendix G.

STEP 2: CAD MODELLING

After the input axis systems have been defined, the modeller adds the geometry that allows the submechanism to fulfill its functions. In figure 3.5 for example, the modeller needs to find a slider-crank mechanism with a crack that is allowed to rotate along a certain axis and a slider that should move along a trajectory that is given as two PASs. The level of detail can vary greatly. For some studies a skeleton model is enough to evaluate kinematics and dynamics. In other cases, inertia of a component influences the dynamics too much, so a more detailed model is necessary that takes solid geometry into account. It is normal that a submechanisms contains multiple bodies. Because the shared MASs are now mapped into the CAD product as PASs, no attention needs to be paid to the complexity of all the publications etc.

STEP 3: KINEMATIC AND DYNAMIC SIMULATION ELEMENTS

Elements such as joints and forces depend on the geometry of the submechanism. The location of a joint, for example, is fully determined by a point or axis system in that submechanism. Since the shape and position of submechanisms changes depending of the context, the joint definition also changes. If no attention is paid, a combination of submechanisms can easily lead to a geometric over-constrained system. The mathematical meaning of adding joints and dynamical elements was explained in section 2.3. When two submechanisms share a MAS, the receiving submechanism has information of the two axis systems. As is illustrated in figure 2.9, a joint (or force) can be added between the coupling MAS and the MAS that is bound to the submechanism.

STEP 4: TEST

After adding the kinematic and dynamic elements, it is possible to test the behavior of the MBS model. As there are no other submechanisms involved, the bodies all have their default location and shape. LMS Virtual.Lab Motion allows to simulate the submechanism alone because it can be seen as a complete simulation model. If the position of the submechanism is constrained in the complete architecture, it might be necessary to fix a body of the submechanism to the 'ground' because otherwise all bodies will drop due to gravity.

STEP 5: OUTPUT AXIS SYSTEMS

In this step, the interface axis systems are added that will serve as the master for the input axis systems of other submechanisms. By associating the axis system to a body, it will stay on the same position relative to that body. The method on how to add an output axis system to a body is explained in section 2.3. Again, a macro was made to automate the tedious work of creating all the axis systems and formulas. The code and documentation can be found in appendix F.

STEP 6: SAVING AND DOCUMENTING

At this stage, a properly functioning submechanism exists. The final action is saving and documenting it. This should be done in a repository. Depending on how the repository works, it might be saved into a hierarchy of folders to keep an overview of the available component models.

4 Case study

The developed methodology that was explained in the previous chapter was applied to the design of a trailing-edge high-lift device. Firstly, the details of the case study are described followed by the software setup. Thereafter, all six steps of the design process follow. The penultimate section shows where and how the topological sorting algorithm was used. Finally, the simulation models are synthesized and their results are discussed.

4.1. Description of the case study

The design of a trailing-edge high-lift system is a good case study because it involves complex kinematics that can be obtained by a variety of deployment mechanisms. It is also important that the design problem of the case study has a complexity that is similar to problems found in industry. There is also some in-house experience regarding the design of high-lift devices for commercial aircraft[1].

The case study emulates a typical design process in the conceptual phase for a flap mechanism. Imagine the following scenario: an aircraft manufacturer has developed a conceptual design for a regional jet aircraft and has written a request for proposal (RFP) to develop and build the trailing-edge high-lift system. A potential subcontractor wants to come up with a competitive design. At this stage, a very large number of concepts are possible. They should all be analyzed because competition is fierce and the company really wants to win the contract. Therefore, ABD will be used to speed up a design space exploration (DSE). The aerodynamic design of the wing and the flaps has already been performed and the CAD models are available. Figure 4.1 shows the wing surface, the wing spar and the position of the inboard flap surface. It is the aim to design and analyze the high-lift devices that can be found in literature, see section 2.4. This case study will analyze the power required to drive the system.

The design process, which is illustrated in figure 4.2, follows the engineering 'V' model [71]. The sections in this chapter follow the steps in the design process. First, the requirements for the positions of the flap follow from the aircraft and operations requirements. Then, trailing-edge high-lift system architectures are defined. Subsequently, simulation

[1]Siemens Industry Software NV participates in the Cleansky and Cleansky 2 project, funded by the European Commission.

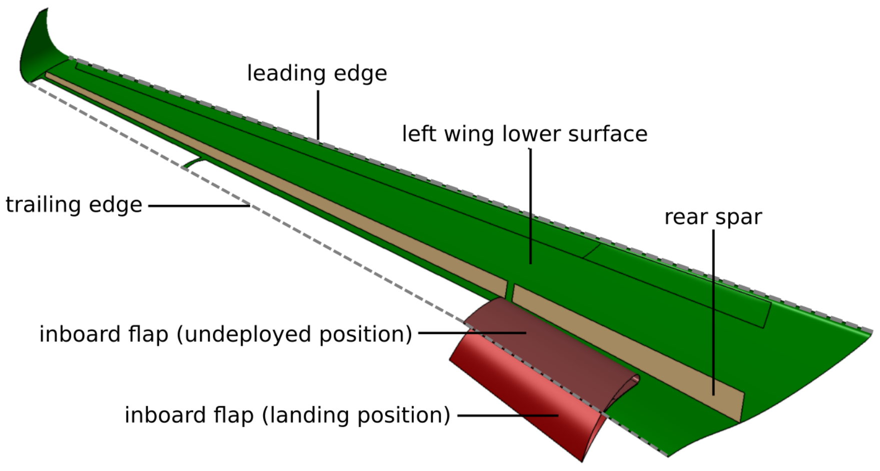

leading edge
left wing lower surface
trailing edge
rear spar
inboard flap (undeployed position)
inboard flap (landing position)

Figure 4.1: Wing and inboard flap CAD model

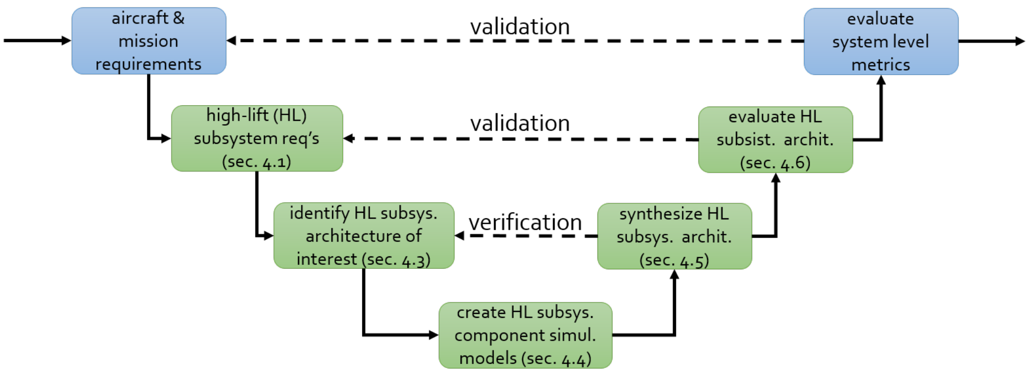

aircraft & mission requirements
validation
evaluate system level metrics
high-lift (HL) subsystem req's (sec. 4.1)
validation
evaluate HL subsist. archit. (sec. 4.6)
identify HL subsys. architecture of interest (sec. 4.3)
verification
synthesize HL subsys. archit. (sec. 4.5)
create HL subsys. component simul. models (sec. 4.4)

Figure 4.2: The design process for the high-lift system that was used for the case study. The process follows the 'V' model. The green steps are performed in the case study.

models are created that can realize the components in the architecture. In order to evaluate the high-lift subsystem, it needs to be synthesized and simulated.

4.2. SOFTWARE SETUP

In the case study, two software programs are used that are developed at Siemens PLM Software: a version of LMS Imagine.Lab System Synthesis which is in a prototype stage and the MBS tool named LMS Virtual.Lab Motion. Figure 4.3 visualizes the software setup that was used for the case study. Section 2.2 explained how the System Synthesis tool works. Section 2.3 explained the use of submechanisms in LMS Virtual.Lab Motion. The next sections in this chapter follow the activities that are presented in figure 4.3. First, the case study is looked at from the perspective of the system architect who creates the architecture definition. Then, the process continues but from the perspective of the modeller who creates the MBS models and stores them in a location that the synthesis tool has access to. The last step of the design process is performed by the system analyst who uses the synthesis tool to create the multi-body simulations for the concepts that he wants to evaluate.

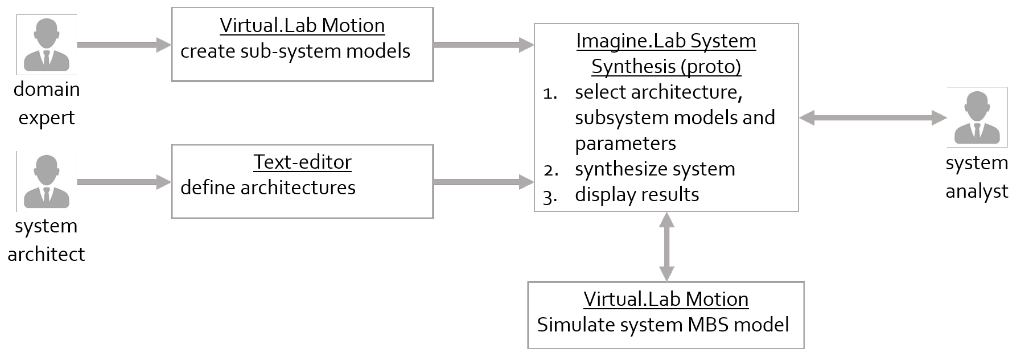


Figure 4.3: Software setup showing the interaction between the simulation software LMS Virtual.Lab and the LMS Imagine.Lab System Synthesis prototype.

It is possible to manually connect all subsystems but that would take a lot of time. Therefore, the system architect uses the architecture description language (ADL) of the synthesis tool to describe the architecture. The tool has three important functionalities:

1. It reads and interprets the architecture description files and, if possible, supports the user with the creation and modification of the architecture.
2. It interprets which simulation models are available in the model library that can realize a component from the architecture. Furthermore, the tool checks in which software the simulation models are created. A model can for example be defined in a general programming language like Python or it can be a MBS model.
3. It synthesizes the system.

4.3. ARCHITECTURE DEFINITION

The logical starting point to create system architectures is the list of functional requirements which leads directly to the need for certain components. The architecture for the case study reflects the functional engineering principle. For example, it was decided to break up the high-lift system in a load carrying part and an actuating part. Figure 4.4 gives an overview of all components that are part of the high-lift system. The figure uses UML to represent the hierarchical relations between the components. Composition and generalization are important concepts that can be easily understood. However, the ’realization’ is a UML concept that is less famous. The realization expresses the relationship between the domain specific simulation model and the abstract information model. Namely, the realization represents some characteristics or behavior of the realized component that cannot be expressed by the realized component. For example, the abstract model of a wheel specifies its attributes and functions but for the evaluation of some attributes a simulation model is needed. The simulation model is the realization of the abstract wheel model. Let’s go back to figure 4.4. The system that we want to analyze consists of a high-lift system that moves the flap relative to the wing. The flap is influenced by the surrounding air which is realized by a simulation model that contains a force element to represent the aerodynamic normal force acting on the flap. The inboard flap is realized by a model called ’IBFlap_VLM_Rez’. ’HighLiftSystem’ consists of ’Deployment Mechanism’ and ’Actuation Mechanism’. There are four kinds of deployment mechanisms which each have a model that realizes their behavior. For example, ’HookedTrackMechanism’ is realized by ’HookedTrackMechanism_VLM_Rez’.

Table 4.1: Functions of the components in the high-lift system architecture.

Component	**Function**
Air	Interacts with any moving object.
Flap	Increases lifting force of objects in front of it.
Main wing	Provides a lifting force.
Deployment Mechanism	Constrains flap on a certain trajectory relative to a wing.
Actuation Mechanism	Moves the flap on a certain trajectory relative to the wing.
Branched Gearbox	Branches a shaft at a certain angle with a certain gear ratio (GR).
Bevel Gearbox	Changes the angular velocity of a shaft or changes the rotation axis of a shaft or does both.
Motor	Converts energy (electric, hydraulic, ...) into rotational mechanical energy.
Shaft	Transfers rotational mechanical energy.
Linkage	Converts rotational motion into a combination of rotational and translational motion.
Controller	Transforms the value of an inputs signal to an output signal.

There are different types of Actuation Mechanisms but for this case study only a mechanical actuation mechanism is analyzed. The functions of each component in figure 4.4 are provided in table 4.1.

The architecture of the trailing-edge high-lift system is visualized in figure 4.5. The architecture is inspired by the A340 flap actuation architecture that was visualized in figure 2.19. The case study only takes the inboard flap into account. The outboard flap system can be modelled and added similarly. In fact, the ABD methodology makes it possible for the user to simply define the architecture of the outboard flap system and automatically synthesize the simulation model because the components are the same as the inboard flap system. Figure 4.5 shows that a centralized actuation system was chosen where the actuators are driven by one power drive unit (PDU). This is a common architecture for the flight controls system of a commercial airliner as was shown in section 2.4.

4.4. SIMULATION MODELS

This section gives an overview of the MBS models that were made for the high-lift device. All models were created in LMS Virtual.Lab Motion. The modelling process as presented in section 3.3 will be followed for all components.

4.4.1. WING

The geometry of the wing and flap models is fixed. When the high-lift device is designed, usually the wing geometry is already frozen [56]. Therefore, steps 1 and 2 (the creation of input axis systems and CAD geometry) of the modelling process can be skipped because the wing is not built parametrically. The geometry was imported as an .IGES-file into the CAD document. The wing consists of only one body and it is fixed to the ground. Therefore, no joints, constraints, forces etc. need to be added to the model. There is no point in testing the dynamics or kinematics of the wing because it is static. Continuing with the process steps, output axis systems are built on top of the wing

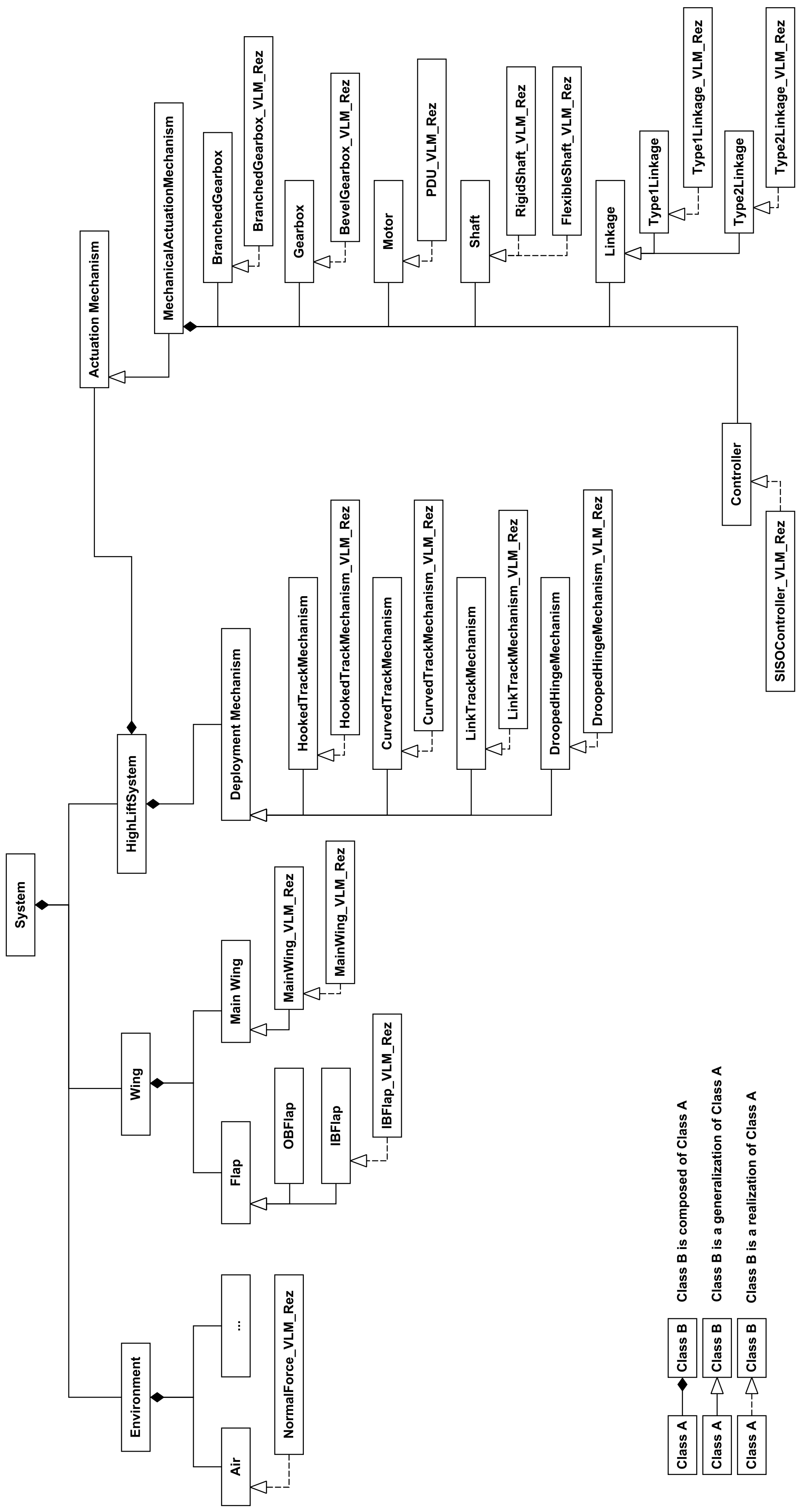


Figure 4.4: UML class diagram of the system that consists of an environment, a wing with flap and a high-lift system. Realization relations show that each component is represented by a simulation model.

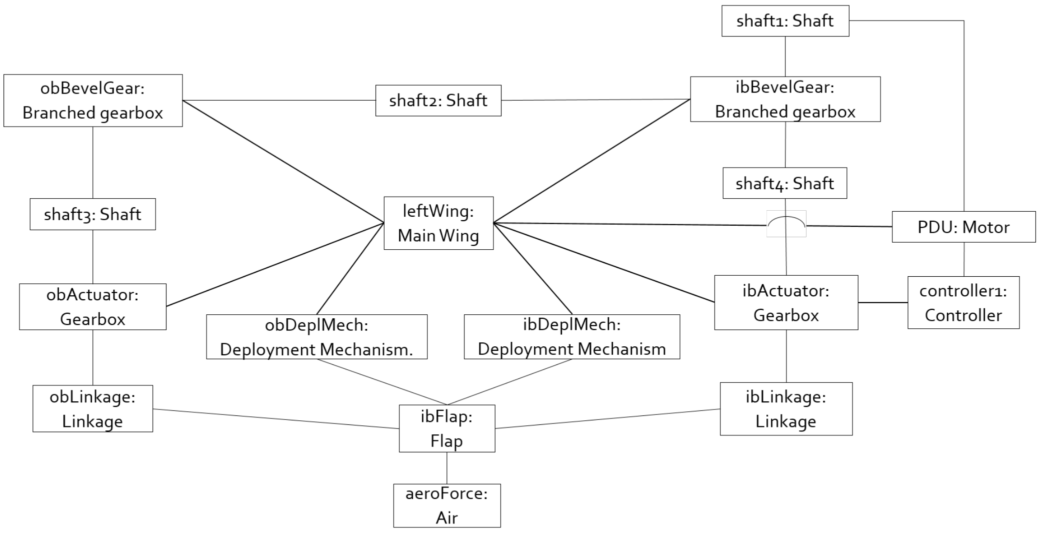


Figure 4.5: Architecture of the high-lift system

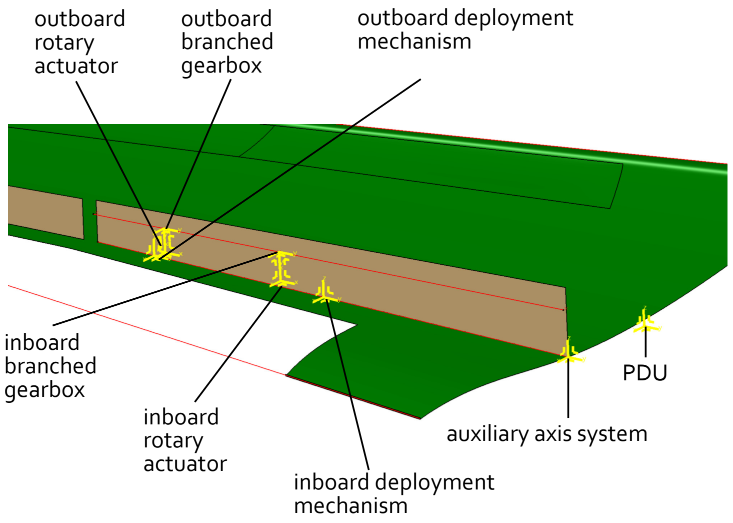


Figure 4.6: Geometry of the wing (green) and spar (brown) with the output axis systems (yellow) that provide the interface with other submechanisms.

geometry. Figure 4.6 shows the output axis systems that provide the location to which other submechanisms can be attached. The output axis systems are positioned with the help of parameterized points and lines that were built on top of the wing surface.

4.4.2. FLAP

Similarly to the wing model, the flap geometry is fixed and defined in an .IGES-file. Figure 4.7 visualizes the flap geometry. Auxiliary geometry is added to define the location of the attachment points of the linkages and the deployment mechanisms. The attachment points of the linkages are positioned at a distance of 10% ($= 0.1c_f$) of the leading edge of the flap, with c_f the chord length of the flap. The aerodynamic center is positioned at $0.25c_f$ and the locations where the deployment mechanisms attach are positioned at $0.3c_f$. All axis systems are positioned span-wise symmetrically w.r.t. the leading edge of the flap. As mentioned before, all these locations are defined paramet-

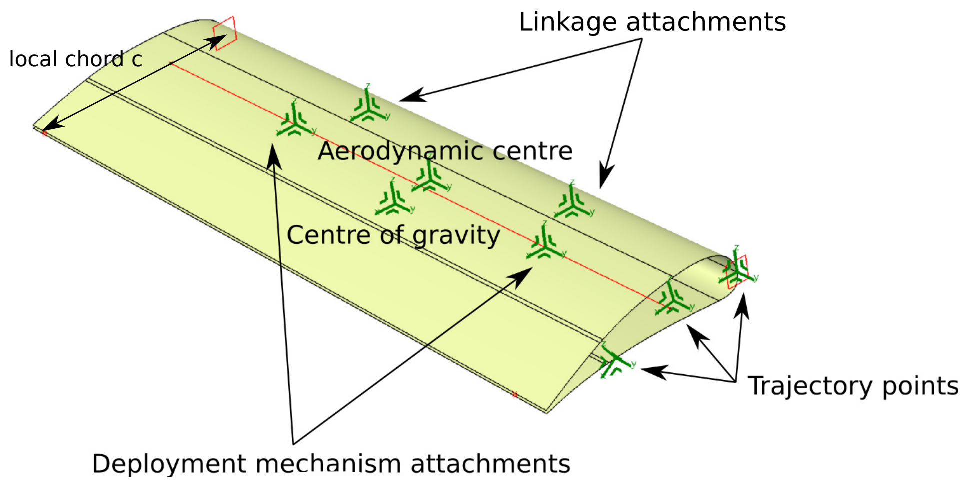


Figure 4.7: Flap model with output axis systems(green) indicating the location where linkages and deployment mechanisms attach. Also the aerodynamic center and centre of gravity is added. The trajectory points indicate the undeployed position, take-off position and landing position of the flap.

rically so they can be easily changed. The trajectory points indicate the undeployed position, the take-off position and the landing position of the flap. Since the flap is one rigid body, the position and orientation of one axis systems w.r.t. the undeployed axis system contains all information to position the flap accordingly. In other words, the trajectory axis systems represent where the leading edge of the flap should be positioned during undeployed, take-off and landing position.

4.4.3. AERODYNAMIC NORMAL FORCE

The simulation model of the aerodynamic force uses the orientation of an input axis system to determine the magnitude of a force. The force element is applied to a dummy body that is fixed rigidly to the input axis system. The dummy body can be seen as part of the flap because it follows exactly the motion of the input axis system which is the aerodynamic axis system of the flap that is being shared.

The magnitude of the normal force is based on semi-empirical data provided by ESDU [72] which give an estimate for the normal force coefficient C_{N_f} for different wing/flap chord ratios c_f/c_w and flap deflection angles δ_f. The measurement points were placed in an external data table that was linked to the force model. During simulation, the submechanism uses an interpolation of the data to find the corresponding force coefficient from which the normal force can be calculated. The data is visualized in figure 4.8. Furthermore, the air density ρ, air velocity V and flap length l need to be provided as parameters to the force submechanism. The equation is:

$$N = C_{N_f}\frac{1}{2}\rho V^2 c_f l \tag{4.1}$$

4.4.4. DEPLOYMENT MECHANISMS

Literature shows five common types of deployment mechanisms: link-track, curved track, hooked track, four-bar and drooped hinge [13]. Drawings of these mechanisms

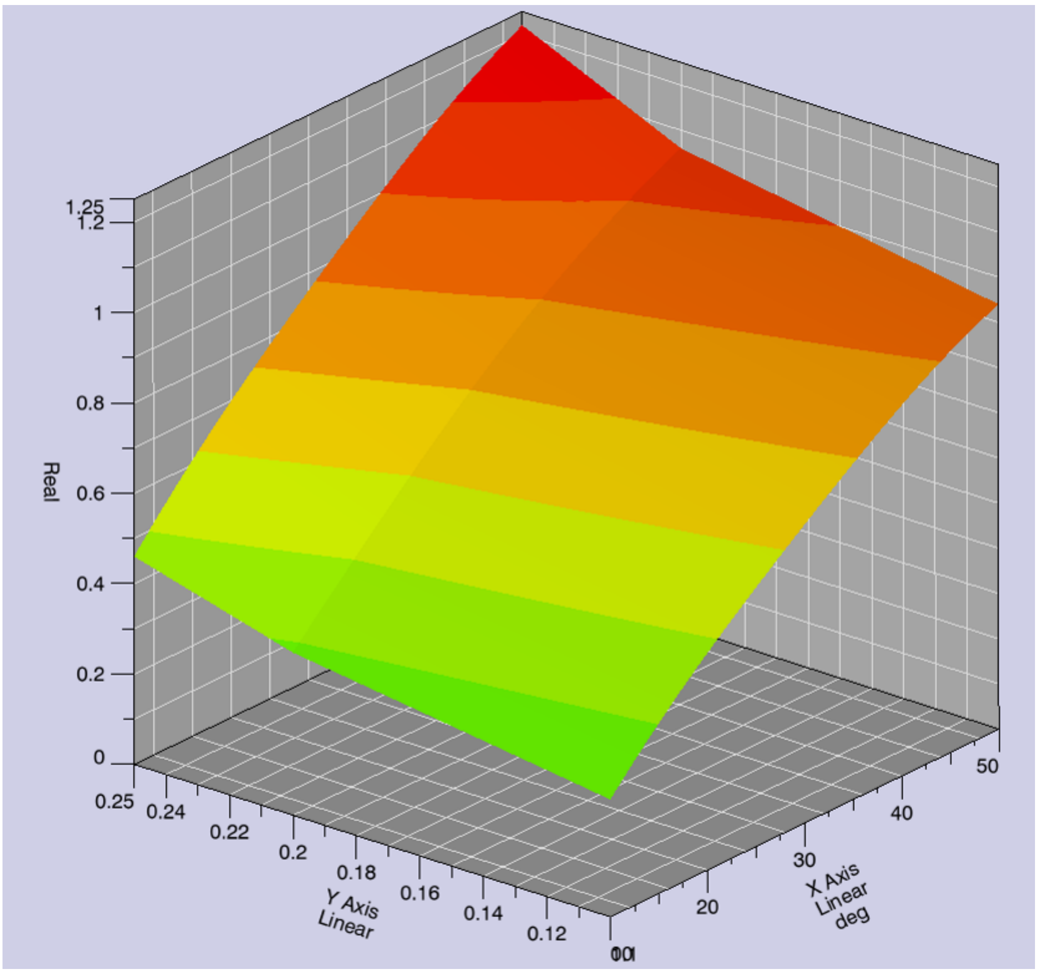


Figure 4.8: Interpolation of values for the aerodynamic normal force of a single-slotted flap. The normal force coefficient C_{N_f} is plotted on the Z-axis. The wing/flap chord ratios c_f/c_w is plotted on the Y-axis and the flap deflection (deg) is plotted on the X-axis. (based on [72])

can be found in chapter 2. The construction starts from two (or three when possible) input axis systems which represent the desired trajectory of the flap. One other axis system is provided by the wing and indicates where the deployment mechanism connects to the wing. The link-track, curved track and four-bar mechanism can take three positions into account while the hooked track and drooped hinge mechanism can only satisfy two position requirements.

HOOKED TRACK MECHANISM

Compared to modelling the wing and flap, the designer has more freedom when modelling deployment mechanisms because there is no fixed geometry. Therefore, the first four steps of the modelling process can be demonstrated in figure 4.9. The first step is to create the input axis systems (blue) that will receive their position and orientation from the corresponding shared axis systems of the flap and the wing. In the second step, the trajectory axis systems are used to determine the landing position of the carriage's input axis system that connects to the flap. Based on the initial and final position of the carriage, it is now possible to construct the complete hooked-track mechanism according to the method provided by Zaccai et al [55]. In the third step, the joints and constraints are added between the bodies. There is a revolute joint between the two carriage (triangular) parts. There are three points of the carriages that are connected with a point-curve constraint to the spline that represents the track. There is one bushing that (loosly) constrains the carriage to the flap. Now, the model can be tested by running a simulation. First, the track is fixed in space because there is no wing present to attach it to. During simulation, a gravitational force makes the carriage parts, that have been given a test weight of 1 kg each, roll down the track. Since there are no other submechanisms that depend on the deployment mechanism, there is no need to create output axis systems. The final step is saving and documenting the model.

CURVED TRACK MECHANISM

The curved track mechanism is a variation between the hooked track mechanism and a carriage-track mechanism that has a circular track (used for the Boeing 707 [13]). It is not found directly in literature. It has a lot of design freedom and basically any trajectory can be realized with this design. The design process for this MBS model started by creating the input axis systems (red in figure 4.10). Then the geometry was created that defines the carriage body and track body, based on parametric relations. The figure only shows the CAD geometry and not the dynamic elements. However, they are similar to the joints and constraints that were present in the hooked track simulation model. The flap is attached (semi)-rigidly to the carriage with a bushing. This means that the flap's orientation depends on the position of the carriage on the track. This is different compared with the hooked track mechanism where the orientation of the flap depends on how the rear carriage hinges around the front carriage.

LINK-TRACK MECHANISM

The link-track mechanism is another track-carriage mechanism and it is very similar to the curved track mechanism. The difference is that the flap is attached to the carriage with a hinge so one rotational degree of freedom is unconstrained. The flap's deflection is controlled by the linkage which makes the design of this submechanism more complex. It is repeated that all geometry is built using only the three red input axis systems

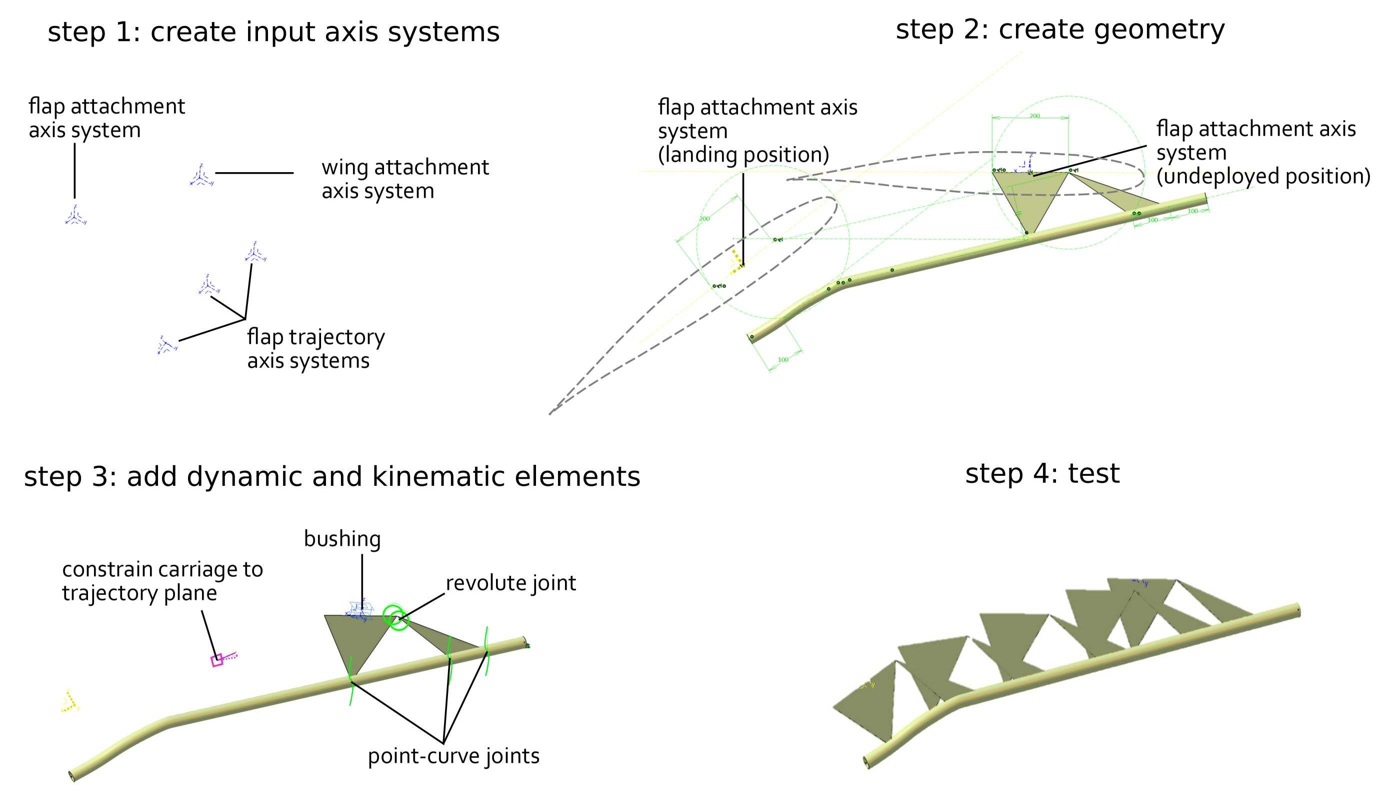

step 1: create input axis systems
flap attachment axis system
wing attachment axis system
flap trajectory axis systems
step 2: create geometry
flap attachment axis system (landing position)
flap attachment axis system (undeployed position)
step 3: add dynamic and kinematic elements
bushing
constrain carriage to trajectory plane
revolute joint
point-curve joints
step 4: test

Figure 4.9: Detail of how the hooked-track mechanisms was designed.

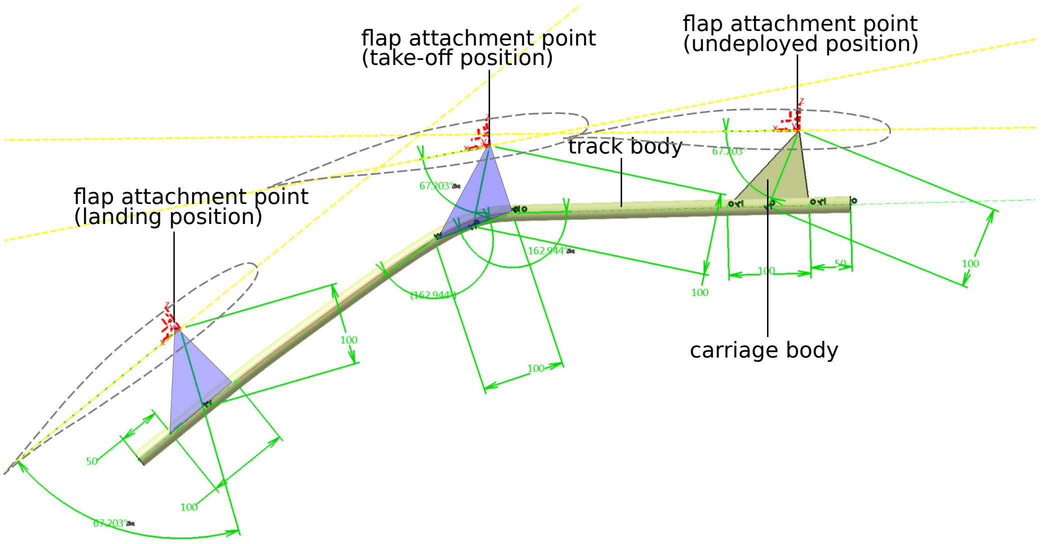


Figure 4.10: Geometry of the curved track mechanism. Input axis systems (red), sketch drawings (green). The blue triangles represent the position of the carriage at take-off and landing. They help to create the track trajectory.

that are shown in figure 4.11. The geometry is built parametrically so the dimensions of the carriages can be changed easily. Different carriage geometry will lead to a different track.

DROOPED HINGE MECHANISM

The drooped hinge mechanism is fairly simple when considering trajectory and the number of parts. It consists of a fixed part that is connected to the wing frame and leads to a point where a moving part is attached that is rigidly connected with the flap. The flap is constrained to follow a circular trajectory around the hinge point. Since a circle is fully determined by two tangents, only an undeployed position and a landing position can be accepted as input. Figure 4.12 illustrates the drooped hinge mechanism. The wing attachment point and flap attachment point are input axis systems. The position of the flap during landing is constructed by using the three trajectory input axis systems that are shared by the flap submechanism.

4.4.5. LINKAGE

While the deployment mechanism has a structural supporting function, the linkage has an actuation function. Figure 4.5 shows that the linkage connects between the actuator and the flap. Its converts the rotary motion of the rotary actuator into a translational motion (and sometimes a combination of translation and rotation). The linkage consists of two links which lengths are just long enough so the flap can reach the landing positions. By minimizing the length of the links, an optimal starting point is provided if the linkage would be structurally sized later. Two models were made because the drooped hinge, hooked track and curved track mechanism leave only one degree of freedom unconstrained while the link-track mechanism leaves two degrees of freedom unconstrained, namely one translational and one rotational. The free degrees of freedom of the deployment mechanism need to be constrained by the linkage.

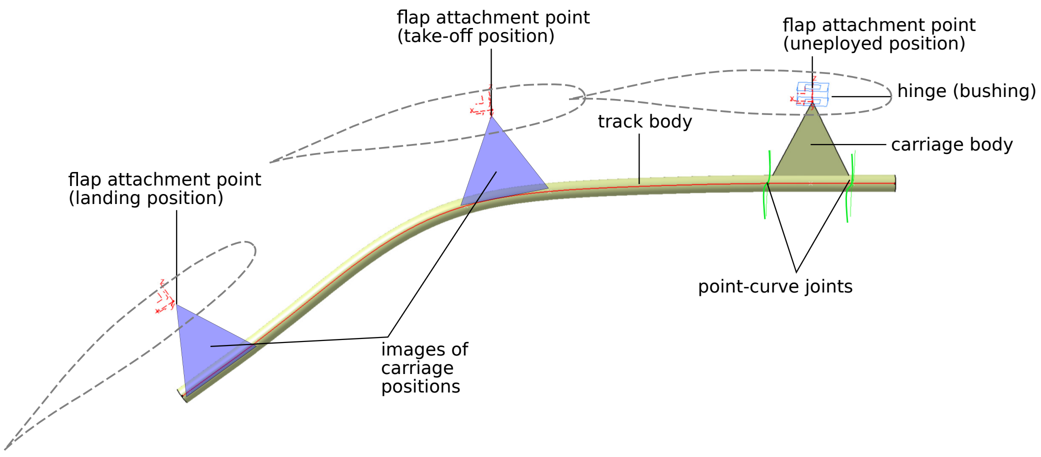


Figure 4.11: Link-track mechanism geometry. The carriage is attached to the flap with a bushing where a rotational degree is free to represent a hinge. The carriage is attached to the track body by two point-curve joints.

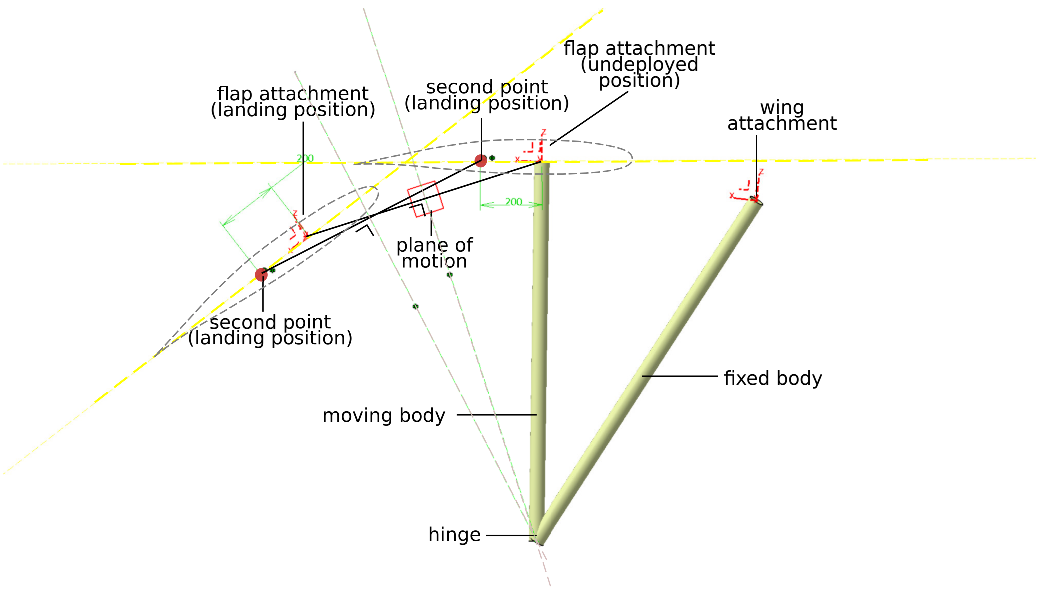


Figure 4.12: The drooped hinge mechanism. The mechanism is built from three axis systems (red).

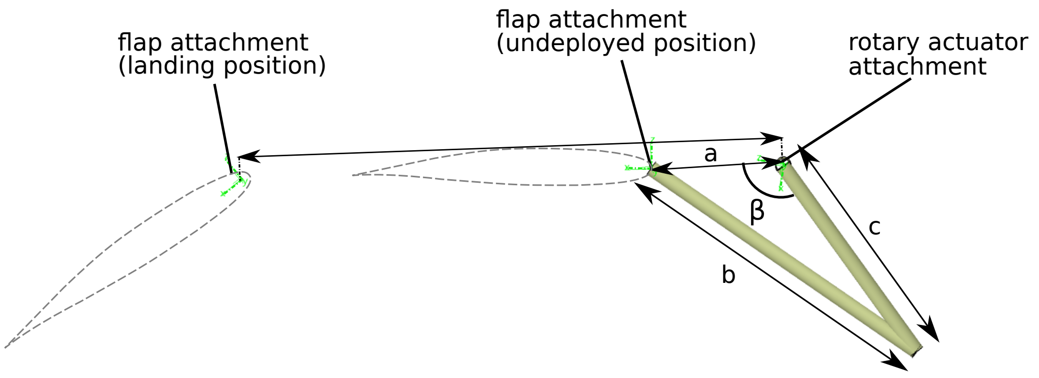


Figure 4.13: Illustration of linkage with one degree of freedom

TYPE 1: LINKAGE CONSTRAINING ONE DEGREE OF FREEDOM

Figure 4.13 shows the input elements (green) from which the linkage is constructed. The single design variable of the linkage is the initial angle β of the arm connected to the actuator. It is assumed that the actuated arm will rotate to its maximum reachable position. The sum of the lengths of the two links b and c is then equal to the distance d between the input axis system representing the fully extended position and the attachment with the actuator, see equation 4.2a. The cosine rule provides a relation between the lengths of the two arms given the distance between the initial position of the flap and the actuator axis system, see equation 4.2b. Solving the system of two equations with two unknowns leads to equation 4.3 which determines the length c of the actuated arm. An important assumption here is that the mechanism moves in one plane. This plane is defined by the actuator attachment point and the landing position. However, the take-off position does not necessarily lie on this plane. Therefore, the link with length c is allowed to rotate slightly out of plane by using a bushing with a high stiffness.

$$\begin{cases} d = c + b & \text{(4.2a)} \\ b^2 = a^2 + c^2 - 2ac\cos(\beta) & \text{(4.2b)} \end{cases}$$

$$c = \frac{a^2 - d^2}{2a\cos\beta - 2d} \tag{4.3}$$

TYPE 2: LINKAGE CONSTRAINING TWO DEGREES OF FREEDOM

The linkage mechanism of the second type constrains rotation and translation. Because of complexity it cannot be easily captured in a simple system of equations anymore. Therefore, geometrical relationships are defined in a sketch of the CAD product to obtain the correct length of the actuator arm b. The angle γ is a design parameter. The construction of the linkage mechanism is illustrated in figure 4.14. Also here is the link

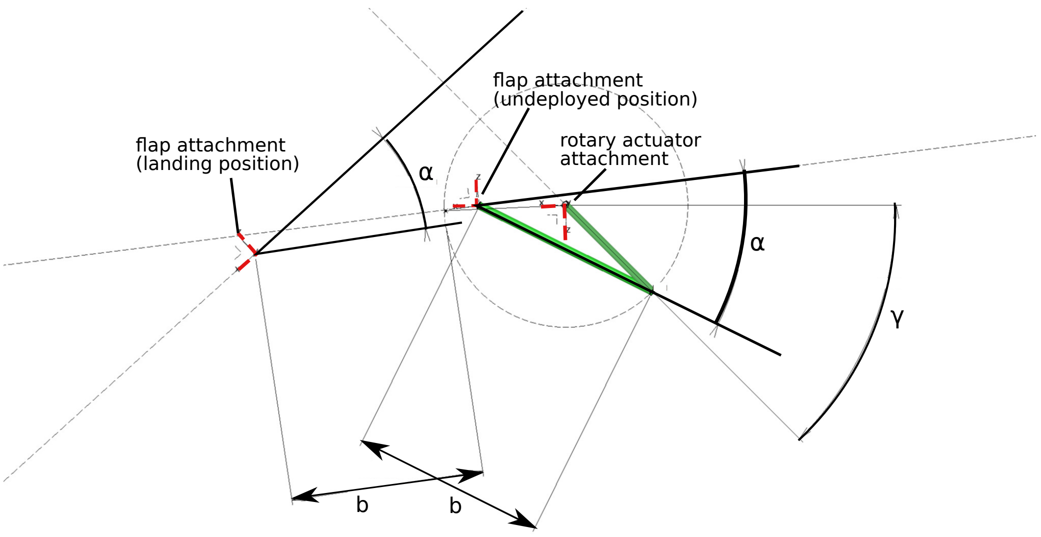


Figure 4.14: Linkage mechanism constraining translation and rotation

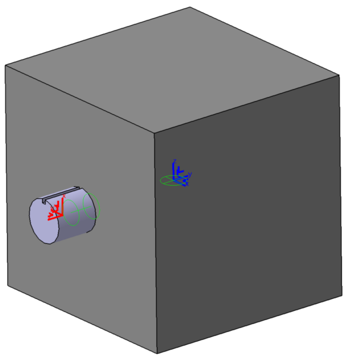


Figure 4.15: Screenshot of the PDU submechanism. The input axis system (blue) determines where the casing body is positioned and fixates it with a bracket joint (green). The casing determines the location of the shaft. A shaft body is attached to the casing using a revolute joint (green). The output axis system (red) allows other submechanisms to connect to the shaft body.

that is attached to the actuator allowed to rotate slightly out of plane with the use of a stiff bushing.

4.4.6. POWER DRIVE UNIT

The actuation system provides rotational energy to the linkage and receives rotational energy via a series of shafts that are driven by a PDU. Gearboxes between shafts allow changes of direction of the transmission shafts. It should be noted that the PDU is not connected to a power source. This simulation models drives the shaft as a position driver. It is also possible to specify a torque between the casing and the shaft.

4.4.7. SHAFT

The shaft's function is to transmit rotational energy from the PDU to the actuator. Therefore, a shaft only receives the axis systems of the subsystems that it has to connect. The shafts are connected to the gearboxes through a universal joint. This allows the

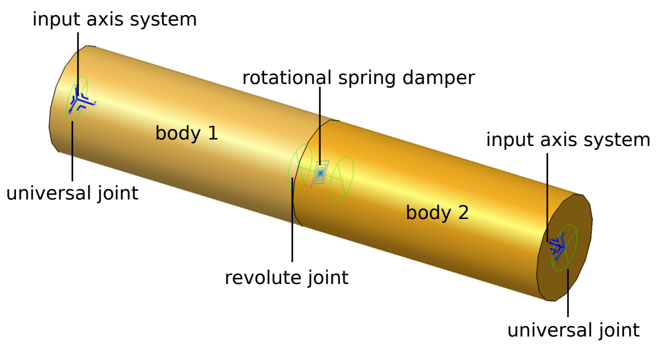
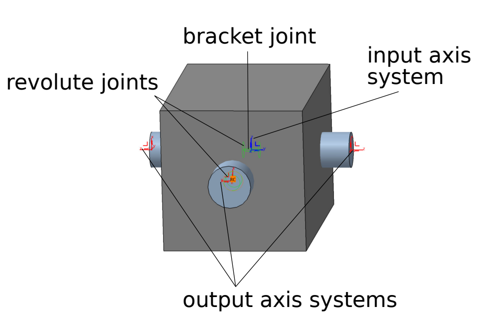
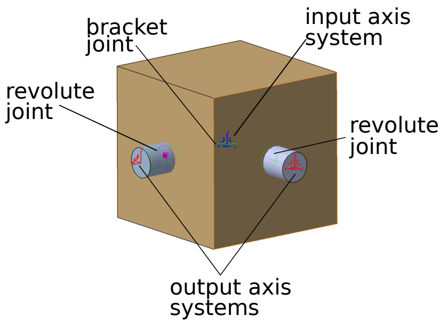


Figure 4.16: Screenshot of the flexible shaft submechanism



(a)



(b)

Figure 4.17: (a) A screenshot of the branched gearbox submechanism. (b) A screenshot of the bevel gearbox submechanism.

gearboxes to be slightly unaligned with respect to their output axes. Two models were created with different levels of fidelity. One model consists of a single rigid body. The other model can represent torsional flexibility. The submechanism with flexibility is visualized in figure 4.15. The following equation is present inside the flexible model that determines the angular stiffness:

$$k = \frac{M}{\theta} = \frac{GJ_T}{l} = \frac{\pi G r^4}{2l} \tag{4.4}$$

The angular stiffness k can be seen as the ratio of how much moment M is necessary to result in a certain angular deflection δ. Engineering knowledge determines that this is the product between the modulus of rigidity and the torsional constant for the geometry divided by the length of the shaft [73]. The cross-sectional geometry of the shaft can be changed by making a new model which has different geometry and different relations. Another possibility is to use a boolean parameter that activates sketches and relations.

4.4.8. GEARBOXES

The are two gearbox types in the architecture: a branched gearbox that allows a shaft to branch off the main transmission shaft and a gearbox that contains a bevel gear to make an angle of 90 degrees with the incoming shaft. Figure 4.17 shows the two types.

BRANCHED GEARBOX

The branched gearbox consists of a casing that is positioned by an input axis system. The corresponding output axis system. There is a main shaft that runs through the casing and that is fixed with a revolute joint. Another shaft is the branch that is driven by the main shaft as if it is connected with a bevel gear. Also this branch shaft is constrained w.r.t. the casing by a revolute joint. A constraint relates the rotation angles of the branched and main shaft by a certain constant. The constant represents the gear ratio (GR). On the three ends of the shafts is a output axis system so other submechanisms can connect to them. For example, a shaft can be connected.

BEVEL GEARBOX

The bevel gearbox is very similar to the branched gearbox, except that there are only two outgoing shafts and two connection points. There is also a constraint that relates the rotation angle of the two shafts. In the high-lift architecture, the bevel gearbox acts as an actuator because it drives the linkages that moves the flap.

4.4.9. CONTROLLER

One controller was made that can switch between two modes. The first mode is a feed-forward controller that sends a predetermined signal as output. The signal is an interpolation between two input parameters: the `START_VALUE` and `TARGET_VALUE`. The first mode is activated when it's input parameter `TEST_MODE` is switched to 1. The second mode is a closed-loop PID scheme. The PID controller acts on the error between the input signal and a target value. The target value at a certain time is read from an interpolation between two input parameters: the `START_VALUE` and `TARGET_VALUE`. The availability of two controllers demonstrates the flexibility in reusing the architecture for different purposes. The first controller mode helps to analyze kinematic behaviour of the system. The second mode gives a more realistic analysis because the first mode uses a driver that can lead to high force peaks due to having almost a hyper-static system.

4.4.10. SAVING AND DOCUMENTING

Figure 4.18 shows the folder structure of the repository. The folder of Virtual.Lab Motion (VLM) models is subdivided into component types depending on their functionality: actuators, gearboxes, deployment mechanisms etc. Each component can be realized by multiple submechanisms. The submechanism consists .CATPart files (that contain the CAD geometry), one .CATProduct (that bundles the CAD geometry) and one .CAT-Analysis file (that contains the multi-body mechanics). It is very important to keep a documentation file. In figure 4.18 it is called "curvedTrack_doc.txt", to log versions, version changes and an explanation of design intent.

4.5. THE SYNTHESIS

Before the system can be simulated, it has to be synthesized. Figure 4.5 shows the architecture of the high-lift system without the causality (i.e. the direction) of the dependencies between components. The causality of the high-lift system architecture is visualized in figure 4.19. The figure shows that almost all components exchange

This figure has been masked due to confidentiality

Figure 4.18: This figures has been masked due to confidentiality.

axis systems except for the controller which receives and sends a signal. The high-lift architecture can be converted into a directed acyclic graph (DAG) so the synthesis tool can calculate the correct synthesis sequence. Figure 4.19 is a hand-made visualization that shows the correct sequence in the upper right corner of the components.

First, the wing is added which provides the attachment points for the gearboxes, branched gearboxes, deployment mechanisms and the PDU. Then, the flap is added which does not require any input from other submechanisms. However, the user can specify input parameters to change the default flap trajectory. The flap provides the interface points for the deployment mechanisms and the linkages. Subsequently, the bevel gearboxes and branched gearboxes are added and positioned on the axis system received from the wing. In turn, they provide the attachment points for the shafts and linkages. The aerodynamic force is the next submechanism that is added. It receives the position and orientation of the flap as a shared axis system from the flap's aerodynamic center. Furthermore, the two deployment mechanisms are added. The deployment mechanisms will position and adapt their geometry to constrain the flap on the required take-off and landing trajectory. In the next step, the controller is added. It receives an input signal from the gearbox. The signal is the gearbox' angle of the outgoing shaft. An output signal is sent to the PDU that represents the torque (or rotation angle depending on the mode) that needs to be applied. Subsequently, the linkages are added based on the input axis systems from the gearboxes and the flap. Similar to the deployment mechanisms, the linkages will adapt to reach the required positions of the flap. Finally, the shafts and the PDU are added. It is important to point out that this sequence is not a unique solution but it was one possible solution that was found by the algorithm that was presented in section 3.2. The code of the algorithm is provided in appendix D.

The System Synthesis tool also changes the input parameters of the components if that is necessary. Table 4.2 shows the exposed parameters of every component. These parameters have default values which are also specified in the table. The description of what design aspect every parameter controls can be found in appendix H

4.6. SIMULATION RESULTS

The high-lift system architecture was realized with the components as presented in section 4.4. There are four different simulation models for the deployment mechanism: the drooped hinge, hooked track, curved track and link-track. The shaft can also be

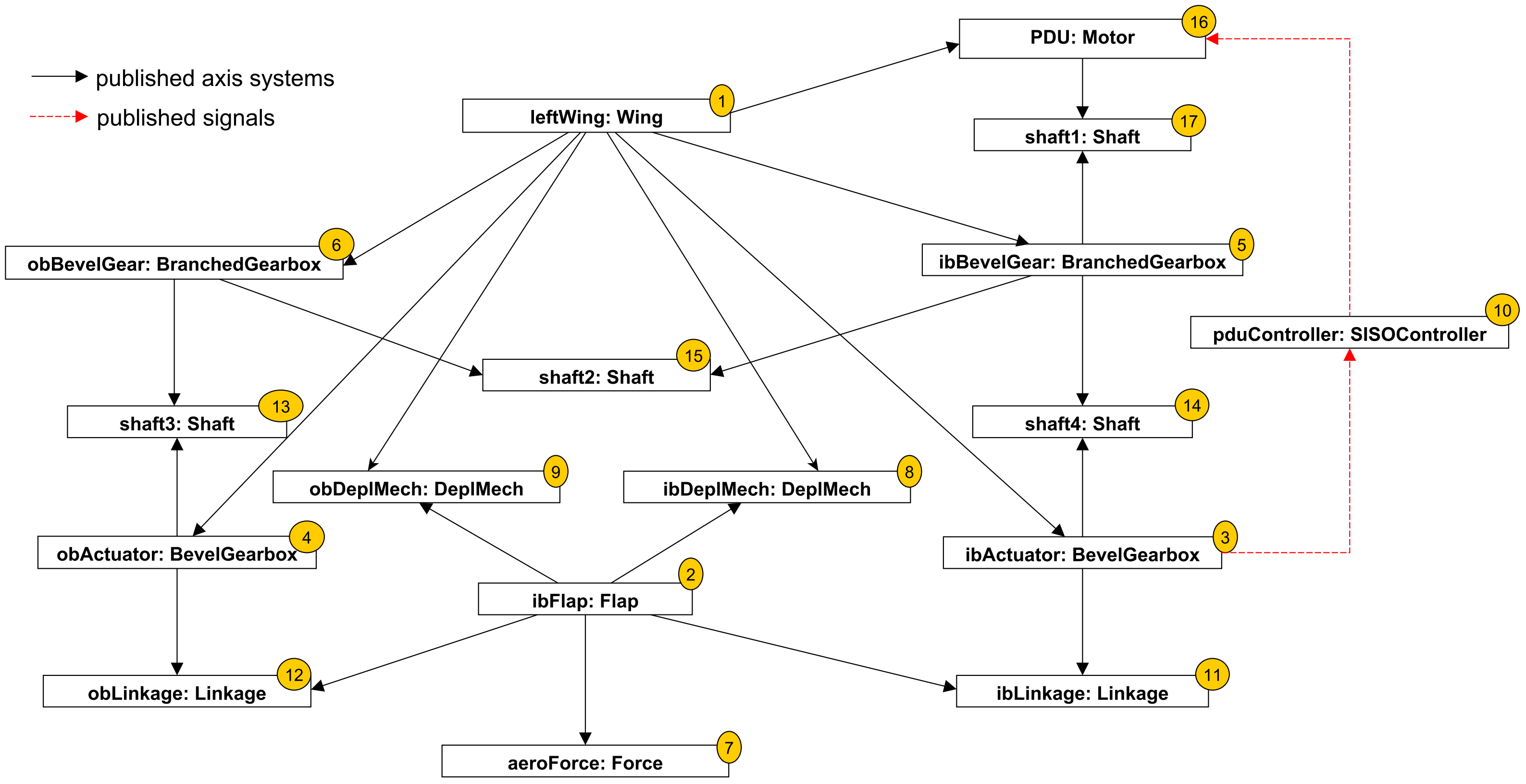


Figure 4.19: Causality of the high-lift system architecture and the types of data that are exchanged. Some attributes and functions of the components are provided. The synthesis sequence is denoted in the top right corner of every component.

Table 4.2: The exposed input parameters of the components and their default values.

Component	Input parameter	Default Value	Unit
Wing	OB_DeplMech_Position	0.12	-
	IB_DeplMech_Position	0.48	-
	TransmissionLine_VPosition	0.3	-
	TransmissionLine_HPosition	50	mm
	OB_GB_Position	0.85	-
	IB_GB_Position	0.605	-
	OB_ActuatorVPosition	250	mm
	IB_ActuatorVPosition	300	mm
Flap	Undeployed_Pos(X; Y; Z)	(23779.5 ; -2216.7 ; 3012.0)	mm
	Undeployed_Orient($\psi;\theta;\phi$)	(-163.6; 4.0; 168.0)	degree
	TakeOff_Pos(X; Y; Z)	(24192.3; -2157.4; 3063.3)	mm
	TakeOff_Orient($\psi;\theta;\phi$)	(115.8; 9.8; -113.8)	degree
	Landing_Pos(X; Y; Z)	(24648.3; -2031.9; 3004.2)	mm
	Landing_Orient($\psi;\theta;\phi$)	(97.0; 36.0; -97.4)	degree
	DeplMech_LongPosition	0.3	-
	DeplMech_Spacing	0.5	-
	Linkage_LongPosition	0.05	-
	Linkage_Spacing	0.5	-
Deployment Mechanism	JammingMode	0	-
	JammingLocation	0.8	-
Linkage	Initial_Angle	45	degree
Branched Gearbox	Width	100	mm
	Length	150	mm
	Height	100	mm
	Gear_Ratio	1	-
	FailureMode	0	-
	Radius_MainShaft	20	mm
	Radius_BranchedShaft	20	mm
Bevel Gearbox	Width	100	mm
	Length	100	mm
	Height	100	mm
	Gear_Ratio	1	-
	FailureMode	0	-
	Radius_Shaft1	20	mm
	Radius_Shaft2	20	mm
Aerodynamic Force	Chord_Ratio	0.2	-
	Flap_Chord	1200	mm
	Air_Density	1.225	$\frac{kg}{m^3}$
	Air_Velocity	85	$\frac{m}{s}$
	Flap_Span	3000	mm
Controller	P_Gain	1000	-
	I_Gain	10	-
	D_Gain	10	-
	Target_Time	10	s
	Target_Value	2	-
	TestMode	0	-
	Output_LimitMode	0	-
	Output_Lower_Limit	-10000	-
	Output_Upper_Limit	10000	-
Flexible Shaft	Radius	20	mm
	Material	1	-
	Damping ratio	0.05	-
Rigid Shaft	Radius	20	mm
Motor	Width	100	mm
	Length	100	mm
	Height	100	mm
	Shaft_Radius	20	mm
	TestMode	0	-

Table 4.3: One configuration with drooped hinge deployment mechanisms, flexible shafts and a PID controller scheme. The settings can be written in a configuration file for the System Synthesis tool or they can be set using the GUI of the tool.

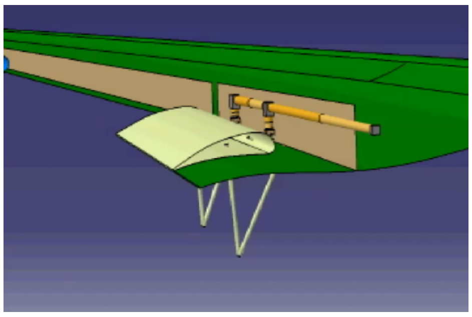
Constrain configuration with:	**Component**	**Value**
asset	ibDeplMech	DroopedHingeMechanism_VLM_Rez
	obDeplMech.	DroopedHingeMechanism_VLM_Rez
	ibLinkage	Type1Linkage_VLM_Rez
	obLinkage	Type1Linkage_VLM_Rez
	shaft1	FlexibleShaft_VLM_Rez
	shaft2	FlexibleShaft_VLM_Rez
	shaft3	FlexibleShaft_VLM_Rez
	shaft4	FlexibleShaft_VLM_Rez
	PDU	PDU_VLM_Rez
	pduController	pidController_VLM_Rez
parameter	pduController.Target_Time	10
	pduController.Target_Value	2.25
	pduController.P_Gain	10000
	pduController.I_Gain	1000
	pduController.D_Gain	1000

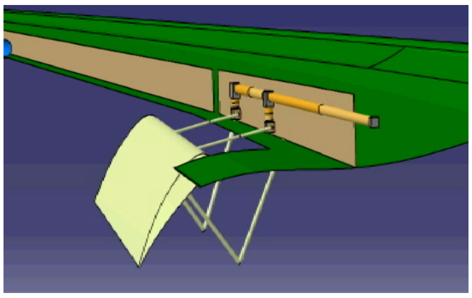
This figure has been masked due to confidentiality

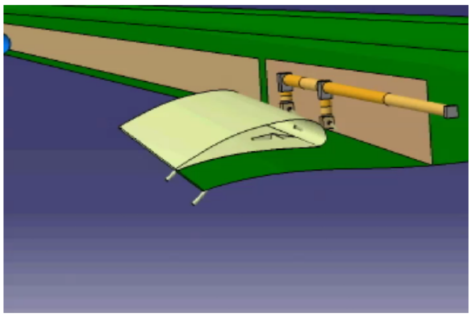
Figure 4.20: This figure has been masked due to confidentiality.

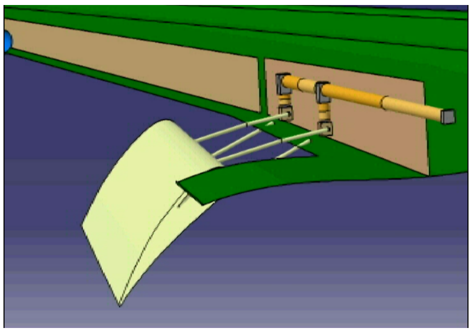
flexible or rigid. Furthermore, the controller can be feed-forward or it can have a PID feed-back scheme. This results in 16 different simulation models. The synthesis tool automatically synthesizes the complete simulation models using the synthesis order from the previous section. Table 4.3 shows the settings for a single realization of a high-lift system with drooped hinge mechanisms, flexible shafts and a PID controller. The parameters that are available correspond with the parameters presented in table 4.2. The explanation of their value can be found in appendix H. Figure 4.21 shows four different realizations of the created architecture. In the four realizations, the shafts are flexible and a PID controller scheme is used. The System Synthesis tool determines all 16 configurations by creating all permutations that are defined by the three variations in figure 4.20: the controller variation, the shaft variation and the deployment mechanism variation.

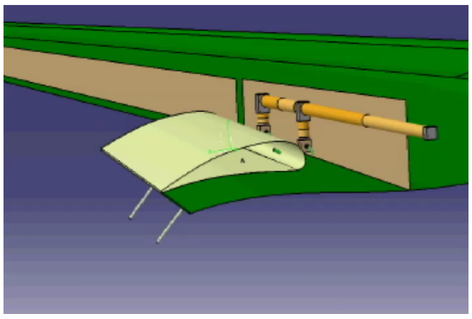
First, the four deployment mechanisms with rigid shafts and a position driver are simulated to verify that the landing position is reached with correct flap deflection. Figure 4.22 shows the flap translational motion along the lengthwise axis of the aircraft versus the flap deflection angle. It can be seen that all deployment mechanisms have the correct final position while having different deflection-translation trajectories. None of the configurations go through the take-off position point. Normally, the link-track and the curved track should be able to do this. However, the deflection of the link-track mechanism depends mainly on the linkage (type 2) which cannot take the angle of the

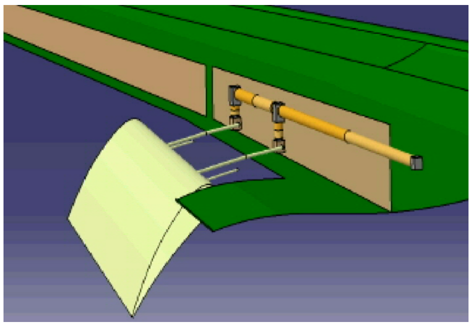

(a) Drooped hinge undeployed

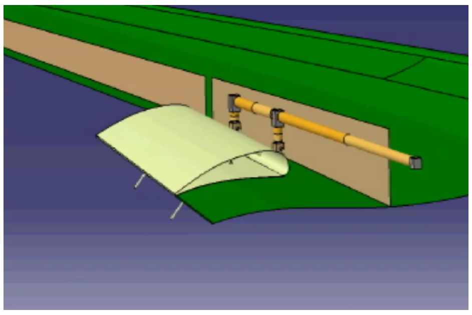

(b) Drooped hinge landing position


(c) Hooked track undeployed


(d) Hooked track landing position


(e) Curved track undeployed


(f) Curved track landing position


(g) Link-track undeployed

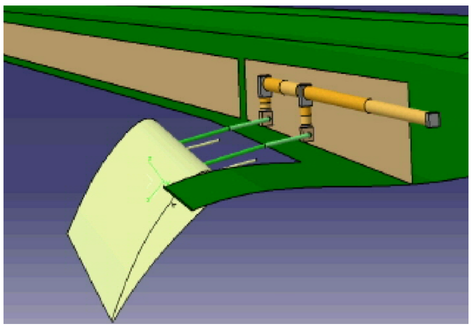

(h) Link-track landing position

Figure 4.21: Screenshots of 4 different realizations of the deployment mechanism.

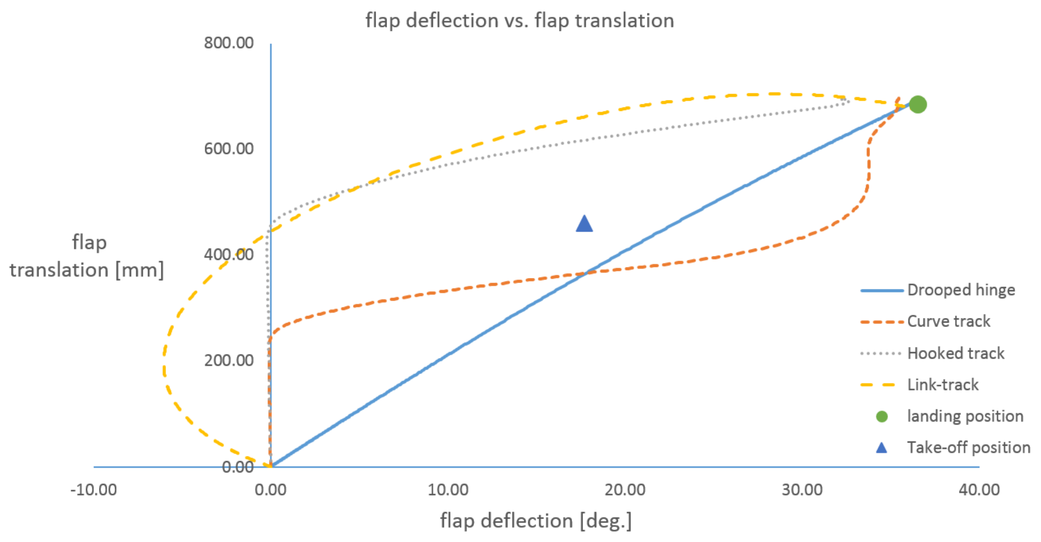


Figure 4.22: Flap deflection vs. Fowler motion of four realizations. Take-off and landing position are indicated.

take-off position into account. If the position of the rotary actuator could be controlled by the linkage component, then it would be possible to let the flap have the correct deflection angle during take-off. The curved track does not obtain the correct deflection angle at take-off because a sudden change in the curvature of the track causes the linkage to apply a large force on the flap. The bushing between the flap and the carriage allows the flap to rotate due to the moment that the linkage produces. In the plot this looks like the flap deflects too early. The problem could be solved by attaching the linkage to the carriage instead of the flap. The plot in figure 4.22 also shows large negative deflection angles for the link-track mechanism. This is because an initial angle of 45 degrees for the linkage mechanism was used. The initial negative deflection angles are smaller for smaller initial angles of the linkage. However, the loads increase for smaller initial linkage angles because the force component that is normal to the track increases. From an aerodynamic point of view, these negative deflection angles require careful attention. Not only can they lead to unpredictable forces on the flap, but they can change the whole flow field around the wing if the flap does not provide the 'right' circulation. An abrupt drop in lift force during the climb phase is to be avoided at all cost.

Figure 4.23 shows the translation of the flap in the x (longitudinal) and z (vertical) direction. It is clear that all deployment mechanisms reach the landing position but not the take-off position. However, as expected, the configurations with link-track and curved track deployment mechanisms satisfy the take-off position too.

Figure 4.24 shows the torque that the PDU delivers when the shafts of the actuation system are flexible and when a PID controller controls the torque. The results from the MBS tell what PDU torque is necessary to actuate the high-lift system and how much power it will use. A negative torque corresponds with a force on the flap that deploys it. The gains of the PID controller were the same for all the deployment mechanisms. However, it was found that a PID scheme leads already to lower maximum torque requirements for the link-track and hooked track mechanism, compared to the controller with only the position driver (see figure 4.25). The torque limiter, which is set at 2000 Nm, reduced the maximum torque requirement for the curved track. Possibilities

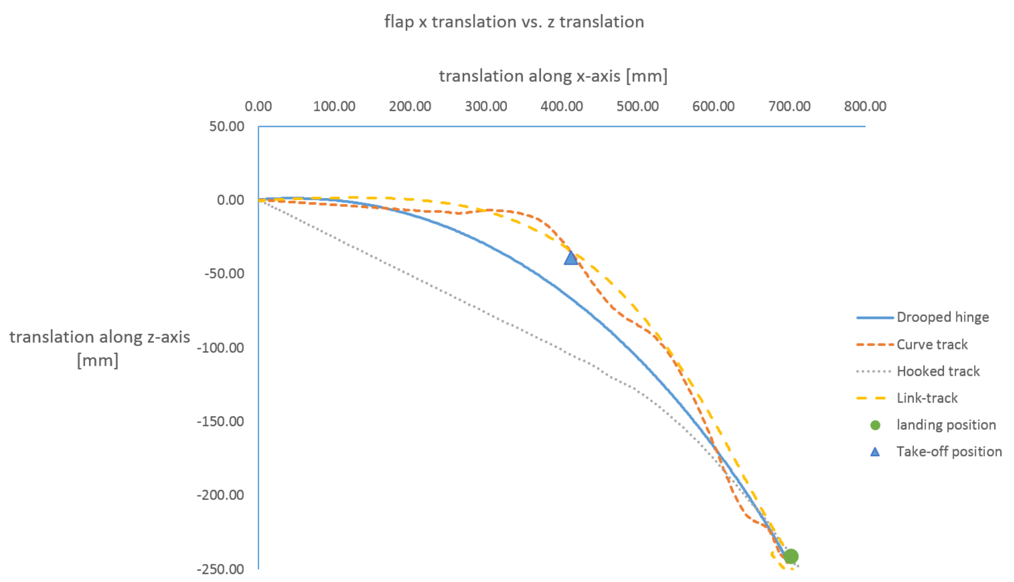


Figure 4.23: Longitudinal vs. vertical translation of the flap. Take-off and landing position are indicated.

are open to optimize the gains, torque limits etc. These ‘what if’ studies are easier to perform now that a working architecture is available.

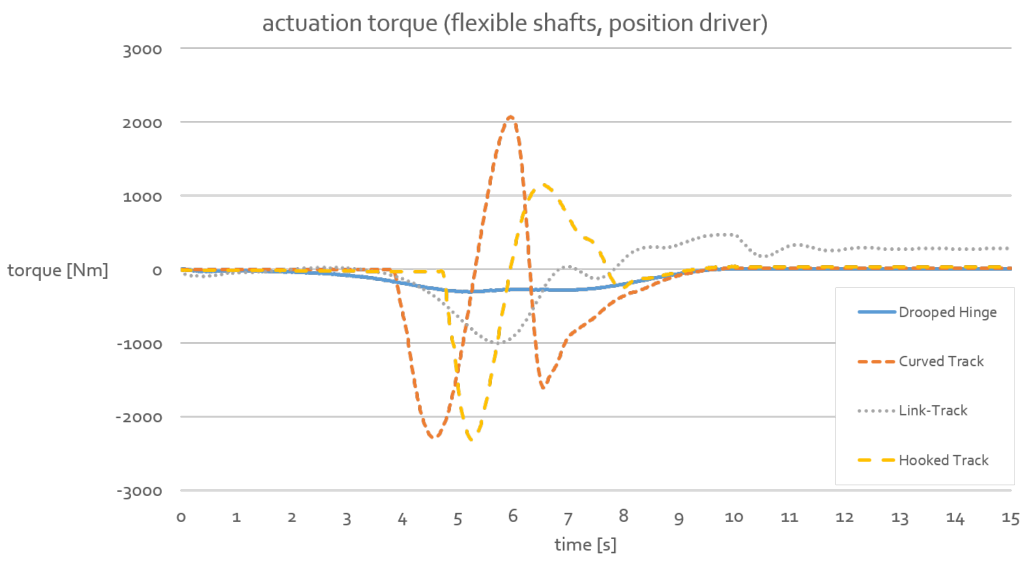


Figure 4.24: Required torque delivered by the PDU for four different deployment mechanisms.

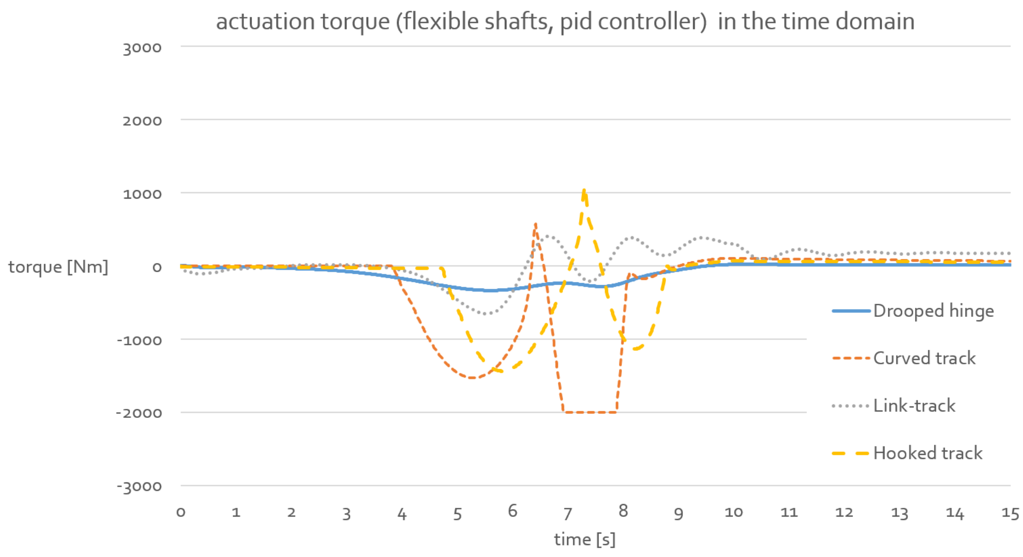


Figure 4.25: Required torque to be delivered by the PDU that is modelled as a position driver. Four configurations with different deployment mechanisms were simulated.

5
Discussion

The case study led to new insights into ABD. In this section the degree of knowledge reuse is discussed. Furthermore, ABD is reflected upon from a KBE point of view.

5.1. Reusing the Architecture

The degree of knowledge reuse of the architecture can be quantified by different metrics. One possibility is to compare the number of simulation models that were reused N_r to the total number of simulation models that were used N. If one particular configuration is considered, 10 submechanism models were manually created: 1 flap, 1 wing, 1 shaft, 2 gearboxes, 1 controller, 1 deployment mechanism, 1 PDU, 1 linkage and 1 aerodynamic force model. The complete high-lift system contains a total of 17 submechanisms. Therefore, the reuse ratio for 1 configuration in the case study is:

$$\frac{N_r}{N} = \frac{17-10}{17} = 0.41 \tag{5.1}$$

Another way is to consider all simulation models that have been created. In this case 16 unique submechanism models were created manually: 1 flap, 1 wing, 2 shafts, 2 gearbox, 2 controllers, 4 deployment mechs, 1 PDU, 2 linkages and 1 aerodynamic force model. In total 17x12 = 204 submechanisms were needed for all high-lift system variations. This results in the following reuse ratio:

$$\frac{N_r}{N} = \frac{204-16}{204} = 0.92 \tag{5.2}$$

There is also the possibility to reuse the models in different projects. Models of actuators, shafts, gearboxes etc. are intensely used in many mechatronic systems.

The benefit of model reuse is reduced by the extra time that is spent on creating the architecture. Therefore, a trade-off needs to be made between the time that can be gained by the automatic generation of simulations and the time that is invested in creating an architecture in the System Synthesis tool. There is still a lot of room for improvement in order to make the System Synthesis tool more time efficient and to provide the tool with the capabilities to do more advanced design. Recommendations can be found in section 6.2. Also the extra effort to make modular MBS models needs to be taken into account. However, the case study showed that the effort is rather small because this thesis has defined an efficient workflow and tools were made that help the modeller to create the interfaces between submechanisms.

The case study shows that ABD is most useful when different variants of a system need to be investigated that cannot be obtained by only changing parameters. Therefore, ABD is very fit for conceptual studies. Later in the design process, ABD still can provide benefits compared to a traditional design approach: the simpler simulation models that are used during conceptual design can be interchanged with more detailed models with little effort.

5.2. THE RELATIONS BETWEEN ABD AND KBE

In this thesis some similarities were pointed out between ABD and KBE. For example, both approaches use an abstract representation of a system to which specific data is added to create a particular configuration. The developed ABD framework allows to create simulation models as 'templates'. Subsequently, the templates are instantiated to represent a component in a simulation architecture. The synthesis tool provides a few functionalities that a KBE system also provides: it compiles the system simulation, it provides a useful interface to set configuration parameters and it retrieves desired data from the result files. Compared with the generative models in KBE, it is not possible to make big changes in the simulation model. This could be a disadvantage as new abstract models and simulation models need to be created if for example the number of interfaces changes. A PLM system that manages the large amount of models is important for ABD.

A difference between the ABD approach and KBE is the absence of a functionality to perform calculations with the system synthesis tool. Furthermore, it does not allow the sharing of component attributes between components. If that would be the case, more knowledge could be captured by the architecture. This would require the tool to be upgraded with two 'extra gears' that are typical for KBE Systems: dependency tracking and runtime cashing [7]. A kind of replacement for the ability to evaluate equations and for dependency tracking was found in the simulation software. Firstly, knowledge was captured in the simulation models as formulas. Secondly, the CAD program tracks the dependencies of the CAD parts (and its features). As explained in section 3.2, it is when there are associative relations between the components that the CAD program cannot incorporate dependencies. Therefore, the sorting algorithm was added to the System Synthesis tool to solve this problem.

6 Conclusions and recommendations

6.1. Conclusions

As the complexity of high-tech systems continuously increases, engineers look for possibilities to reduce time and cost of the development of these systems. By front-loading the design process, more design knowledge is available earlier on. This should lead to better informed decisions and better designs. Architecture-based design enables a front-loaded design process with knowledge reuse. By enabling the automatic synthesis of simulation models, different configurations of an architecture can be realized and simulated efficiently. Current practices are found in automotive and aerospace industry where architecture-based design is used for the automatic synthesis of multi-physics simulation models. In this way, different architecture options and simulation model variations can be rapidly analyzed early in the development. Those practices do not yet integrate multi-body simulations, although multi-body simulations are crucial in the conceptual design of complex mechanical systems. Therefore, the question was raised if architecture-based design can be applied to the automatic synthesis of complex multi-body simulation models.

In this thesis, a methodology was developed that made three important improvements to an existing approach.

- The existing methodology assumed that the order of synthesis of the architecture was irrelevant. Normally, this would be valid because the CAD platform solves the hierarchy for parameterized CAD models automatically. However, the use of submechanisms breaks the associative links between submechanisms. Therefore, an additional sorting algorithm was added to the synthesis tool. An initial successful attempt was made by adding an implementation of Kahn's algorithm to the synthesis tool.
- Changing the CAD geometry of the existing approach resulted in inconsistent models because the CAD geometry was not built parametrically. The developed methodology defines a formalized process that guarantees that the interfaces between modular simulation models will always work. Modular multi-body simulation models can be simulated independently or added to a larger simulation. The modelling process has the following steps:

1. Create parameterized input axis systems
2. Create CAD geometry based on parameterized input axis systems
3. Add joints, forces etc. to CAD geometry
4. Test the model. This is possible because submechanims can be simulated independently.
5. Create parameterized output axis systems
6. Save and document the simulation model

- A procedure was developed and automated that takes away the burden of creating interfaces between simulation models. Consequently, the modeller can focus fully on creating robust models with clear design intent.

The developed methodology was verified with a case study on the conceptual design of a trailing-edge high-lift system. An architecture was defined that is inspired by the A340 flap actuation system. Subsequently, parameterized multi-body simulation models were created according to the formalized process. The simulation-models realize the components of the architecture. Besides the components that actuate the flap, such as gearboxes, shafts and a motor, four simulation models were created representing different deployment mechanisms. The geometry of the simulation models adapted automatically in order to provide the correct flap trajectory and to form a consistent simulation model. A tool synthesized automatically all architecture configurations. Interface forces and moments between bodies could be inspected directly and the required actuation torque was found. The sizing of the components was not performed.

The methodology was evaluated for knowledge reuse. For the case study, this led from 41% up to 92% of reuse of simulation models. This shows that the modelling effort of the whole system and its variants can be significantly reduced by using an architecture-based design approach for multi-body simulations. However, before these results can be generalized, a trade-off needs to be made from case to case between the time that is saved by the automatic synthesis of simulation models and the time it takes to create the architecture and the compatible simulation models. The formalized modelling process is a contribution of this thesis to reduces the extra effort to set up the simulation model interfaces. Consequently, architecture-based design does not result by itself to a longer modelling time of the individual components. It is hard to draw more general conclusions about the efficiency of architecture-based design since the synthesis tool is a prototype. The case study shows that there is certainly room for improvement to increase the efficiency and user-friendliness of the tool.

It can be concluded that the developed methodology enables an architecture-based design approach for complex multi-body simulation models. This was demonstrated by the successful synthesis and simulation of multiple high-lift system configurations. The demonstration, i.e. the case study, showed that many simulation models could be reused, thereby reducing the modelling time. Furthermore, the process to make the simulation models reusable does not lead to a significant increase in modelling time. Therefore, the proposed method is advantageous compared to a modelling approach where all configurations are modelled by hand.

6.2. RECOMMENDATIONS

With the experience of modelling and simulating a representative high-lift system, some recommendations can be made on four areas. First, the architecture and some models of the high-lift system could be improved. Second, recommendations for the synthesis tool are provided in order to make it more user-friendly and more efficient. Third, functionality extensions are proposed for the multi-body simulation software. Finally, some remarks and suggestions of improvement for architecture-based design in general are provided.

RECOMMENDATIONS FOR THE HIGH-LIFT SYSTEM DESIGN

- Currently, the position and orientation of the flap is defined relative to the absolute axis system. However, it would make more sense that the flap position and orientation is specified w.r.t. an axis system of the wing. A more radical change would be to merge the wing and the flap component together. From many perspective these components belong together: they have fixed geometry, they have many mutual dependencies, they are defined by the aerodynamics team etc. The disadvantage of merging the wing and flap is that it becomes more difficult to change the flap geometry.
- The flap could publish the attachment points of the linkages and deployment mechanisms to the wing so the gearboxes can be automatically positioned in the right plane. That plane is defined by the straight motion from the undeployed position of the flap to the landing position and another line which line could be vertical w.r.t. the absolute axis system or perpendicular to the wing surface. This is a choice that influences the aerodynamics.
- For the link-track mechanism, the deflection angle requirement during take-off could be met if the linkage could dictate the position of the rotary actuator. The curved-track mechanism would improve if the linkage would drive the carriage instead of the flap. For these two reasons, it is more beneficial to merge the deployment mechanism and the linkage.

RECOMMENDATIONS FOR THE SYNTHESIS TOOL

This part has been removed due to confidentiality.

RECOMMENDATIONS FOR LMS VIRTUAL.LAB MOTION

This part has been removed due to confidentiality.

RECOMMENDATIONS FOR ARCHITECTURE-BASED DESIGN

- The framework can be extended with a capability that can combine multi-physics and multi-body simulations. This leads to the need for co-simulation strategies.
- In order to create simulation models (or purely CAD models) that can span a larger design space, generative modelling can be used. Instead of changing the parameters of 'dynamic objects', the design is expressed as a list of procedures to generate the design in the simulation (or CAD) software (see MMG [74]). The

current approach of using template models is complementary to the generative modelling approach and both could be used to realize components of an architecture.

- It could be an interesting exercise to perform the design of the same system in two different ways. On one hand, by synthesizing 'template' simulation models, i.e. architecture-based design. On the other hand, by using KBE techniques.

As a final note, this thesis represents research that tries to bring academic work and industrial practice together. I am convinced that KBE is a great technology to apply our ever increasing knowledge of the world to engineer the systems that humanity needs. As a student at TU Delft, I value the theoretical soundness of KBE systems. However, I also came to understand that industry progresses in increments and it takes time to adopt new approaches. I hope that this thesis contributes to bridge the gap between academic research and industrial research.

A
High-lift aerodynamics

This appendix gives an overview of the aerodynamic effects that apply to high-lift devices.

A.1. 2D effects

Leading-edge devices cause an extension of the lift curve while trailing-edge devices shift the lift curve upwards (see figure A.1). This is a rather qualitative understanding of high-lift aerodynamics which was there already since the beginning of aerodynamics studies. However, a quantitative understanding of high-lift aerodynamics was poorly understood until Smith's paper [48] in 1975. This difficulty of understanding the aerodynamics of high-lift devices can be illustrated by how the development of the slats for the F-27 and F-28 MK 6000 happened. Obert writes that the development of slats did not happen according to theory but based on previous experience [75]. In the paper of Smith, the aerodynamics of multi-section airfoils is divided into five effects which together explain to a large extend the aerodynamic behavior observed of high-lift devices. The five effects are presented here.

- The **Slat Effect** can be illustrated by considering a normal airfoil in a flow where a vortex is added to the leading edge representing the slat vorticity. As can be seen in figure A.2a the vortex reduces the velocity ratio at the leading edge. This makes it easier for the boundary layer to overcome the adverse pressure gradient. The main airfoil's lift coefficient dropped but combined with the vortex the total lift coefficient increased.

- The **Circulation Effect** can be explained by again considering the airfoil in a flow but with a vortex at the trailing edge representing an obstruction such as a flap. It can be seen in figure A.2b that the vortex has an increasing effect on the trailing edge velocity ratio and the lift coefficient of the airfoil. Smith explains that the vortex places the trailing edge at a high angle of attack which means that the total vorticity of the airfoil needs to increase in order to maintain the Kutta condition. Therefore, any obstacle at the trailing edge can increase the cross flow and have the same lift increasing effect. This was tested by placing a two dimensional cylinder at the trailing edge which proved to be working [48].

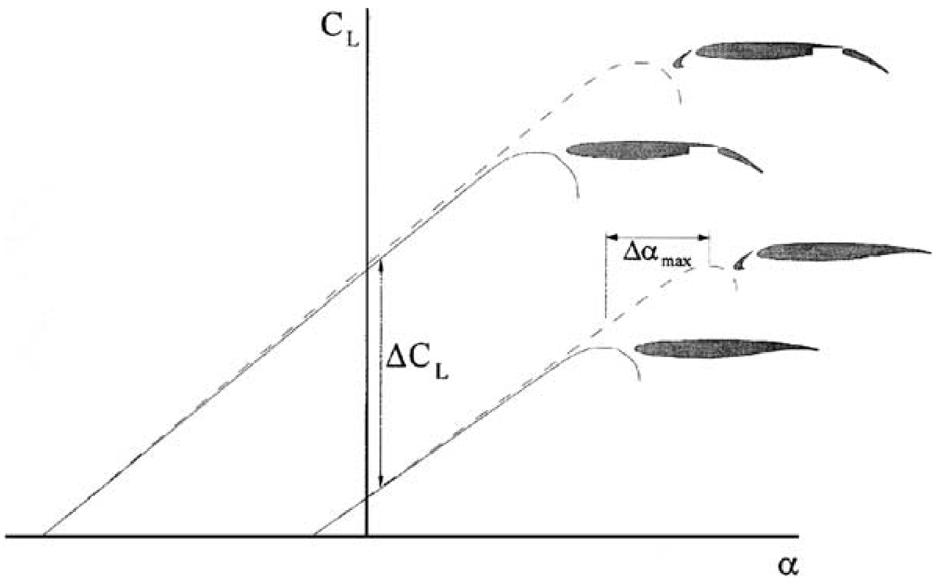


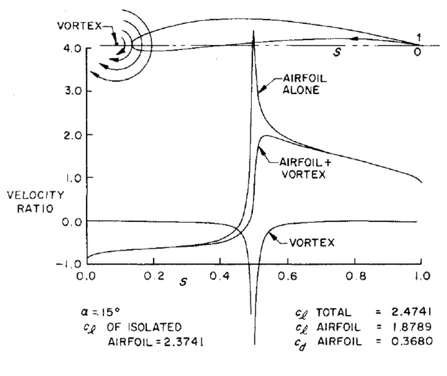
Figure A.1: The effect of leading-edge and trailing-edge devices on the lift curve of the wing [56]

- The **Dumping Effect** is closely related to the circulation effect. A downstream element also increases tangential velocity so that the flow of a forward element is discharged into high velocity flow. This makes it easier to meet pressure-recovery demands. Figure A.3 shows a main airfoil with slotted slat and flap. The canonical velocity profile shows that the dumping velocity u_e / u_{inf} decreases while going downstream because the circulation decreases. As is often done in airfoil design, Smith uses the canonical pressure $C_p = (p - p_{\text{inf}})/(l/2\rho u^2_{\text{inf}})$ in order to compare airfoils with different chords.

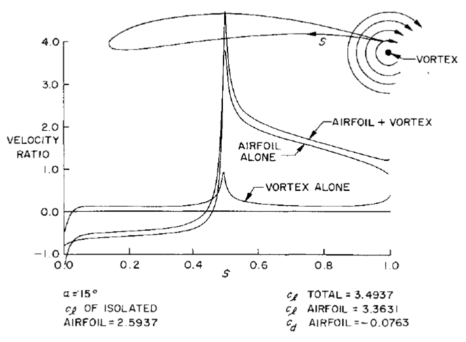
- **Off-the-Surface Pressure Recovery** is an important effect because otherwise the boundary would not be able to drop in velocity so fast. Experiments show that there is quite a drop from the velocity at the trailing edge of the airfoil sections compared to the leading edge of the next element.

- The **Fresh-Boundary-Layer Effect** means that on every surface a new boundary layer develops. The thickness of the boundary layer determines the capability to overcome an adverse pressure gradient. From Stratford's formula with making quasi constant terms constant it can be seen that if the distance is halved, the pressure gradient can be doubled to obtain the same risk of separation. Many people including aerospace engineers seem to think that high-lift devices 're-energize' the flow. However, preceding boundary layers generally do not collide with airfoil elements. Figure A.4 shows this.

These effects explain why a slat leads to a postponement of stall and hence an extension of the lift curve. Figure A.5 shows the effect of multiple airfoils on the pressure distribution. The pressure peak of the main airfoil is reduced which makes it easier to overcome the pressure gradient. The dumping effect , off-the-surface recovery, and a thin new boundary layer all help to postpone separation. Figure A.6 shows that 'strakes' are placed at the inboard part of the engine cowling because that small part of the wing without slats tends to separate. A stabilizing vortex postpones stall. A leading edge device increases the effective camber but this leads to a decrease in lift because of leading-edge droop. Also there is a small increase in effective chord which increases



(a)



(b)

Figure A.2: (a) Explanation of the slat effect (b) explantion of the circulation effect. Comparing the lift distribution with and without a vortex. [48]

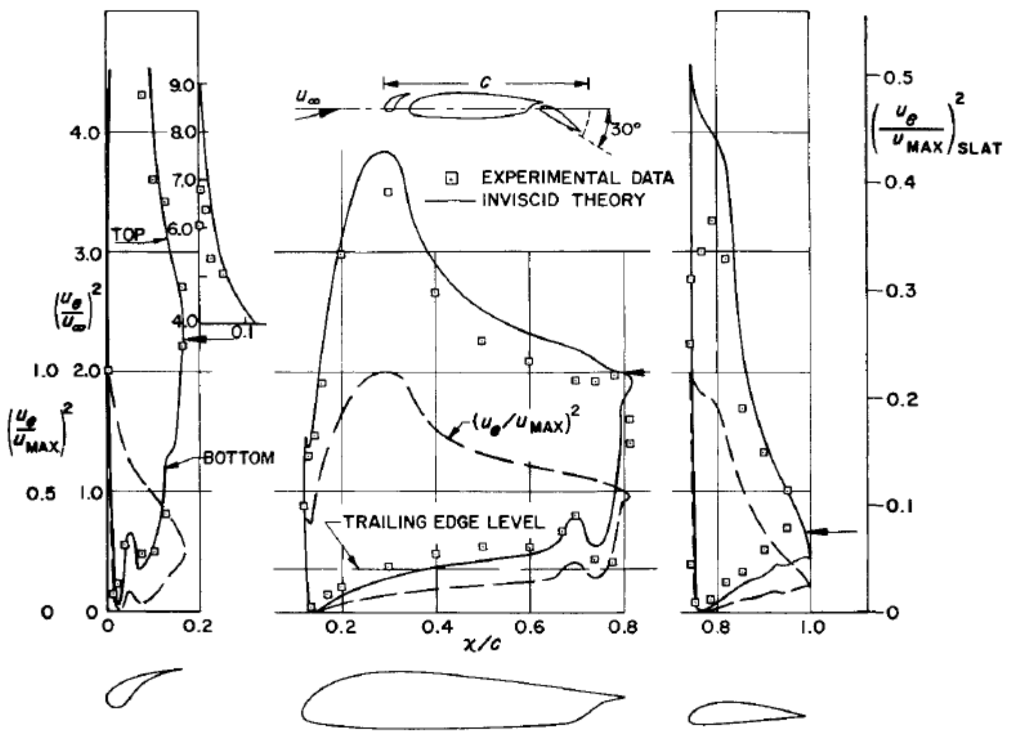


Figure A.3: Three-element airfoil showing the dumping velocity effect. Arrows denote the dumping velocity for the canonical pressure distribution. Starting with flap values are: 0.67, 2.0 and 2.28. The dotted line shows u_e/u_{max}. [48]

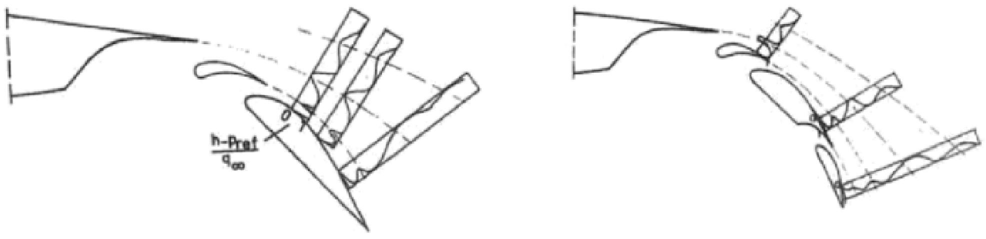


Figure A.4: Wake profiles on an airfoil section with a slat and a double- and triple-slotted flap. [75]

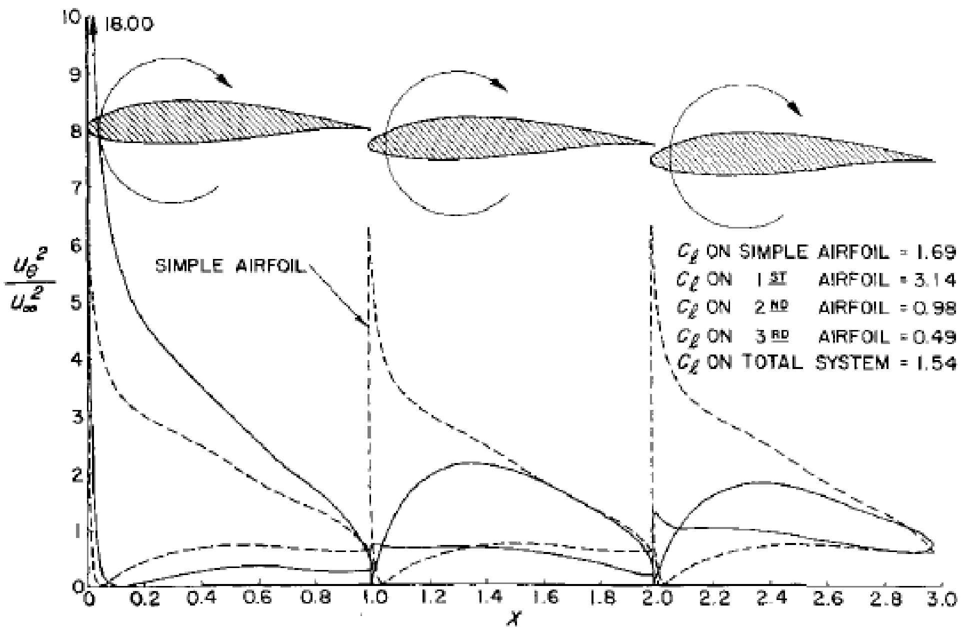


Figure A.5: Pressure distribution on a three-element airfoil from three NACA 632-615 sections. All are at the same angle of attack, being 10°. The dotted line is the pressure distribution on the basic simple airfoil at 10° of attack. [48]

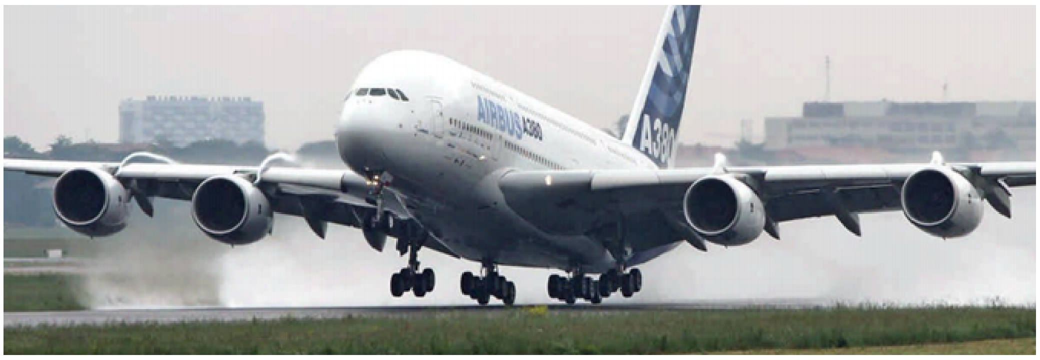


Figure A.6: Stabilizing vortices from strakes at the engine cowling postpone stall locally. [76]

the lift curve slope slightly. [47] The deflection of flaps on the other hand leads to a shift upwards of the lift curve. This can be explained with the circulation effect. Due to the interfering vorticity of the flap at the main airfoil, the vorticity of the main element needs to increase in order to satisfy the Kutta condition.

A.2. 3D EFFECTS

Three-dimensional aerodynamic effects unfortunately decrease the performance of the two-dimensional wing. A first detrimental effect is the fact that the ends of the surfaces create tip vortices just like finite wings do. This leads to a reduction of lift and an increase in drag. A second effect is an increase in drag because the ends of the high-lift surfaces are not always aligned with the flow. This is because of sweep or taper of the wing. Rudolph [13] uses these two reasons as possible explanation why Airbus is creating better high-lift systems than Boeing. A trend of new aircraft is that they don't have an inboard high-speed aileron anymore but a flap which also acts as an aileron which is called a ’flaperon’. The flaperon does not cause an interruption in the flaps so induced drag is minimal. A break in the continuous flap surface is more severe for multiple slotted flaps than for single slotted flaps. Next to these two effects related to the finiteness of the surfaces, there is also the fairing around the deployment mechanism which generates quite some drag.

B

Take-off and landing mechanics

The main purpose of high-lift devices is to give the aircraft acceptable take-off and landing performance. This appendix explains the different qualities of performance that are required during take-off and landing.

B.1. Take-off

The FAR part 25 or EASA CS-25 define specific regulations for the take-off maneuver for large commercial airliners. Figure B.1 summarizes the CS-25 take-off specifications. The aircraft at MTOW starts from standstill and accelerates to V_R at which the aircraft starts rotating. If this velocity is to low then the ground run will endure longer because of the extra drag from it's attitude. If V_R is selected too high then the take-off run is also not minimal. At lift-off the lift force is larger than the weight of the aircraft. Regulations dictate that V_{LOF} has to be at least 1.1 times the minimum unstick speed V_{MU}, or 1.05 times the unstick speed when an engine is inoperative. V_{MU} speeds must be selected by the manufacturer throughout the range of thrust-to-weight ratios to be certificated [52]. The next speed is V_2 which is determined when the aircraft has climbed 35 feet. This speed must be at least 1.13 times the stall speed in steady level flight V_{S1g} and also be at least 1.1 times the minimum control speed V_{MC}. If an engine failure occurs before the minimum control speed the take-off has to be aborted.

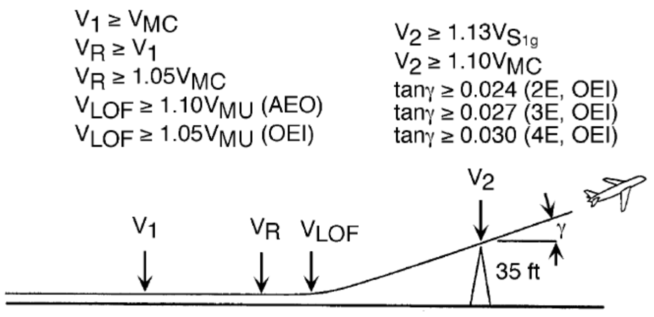

$V_1 \geq V_{MC}$
$V_R \geq V_1$
$V_R \geq 1.05 V_{MC}$
$V_{LOF} \geq 1.10 V_{MU}$ (AEO)
$V_{LOF} \geq 1.05 V_{MU}$ (OEI)
$V_2 \geq 1.13 V_{S_{1g}}$
$V_2 \geq 1.10 V_{MC}$
$\tan\gamma \geq 0.024$ (2E, OEI)
$\tan\gamma \geq 0.027$ (3E, OEI)
$\tan\gamma \geq 0.030$ (4E, OEI)
V_2
V_1
V_R
V_{LOF}
γ
35 ft

Figure B.1: Take-off specifications according to EASA CS-25 [52]

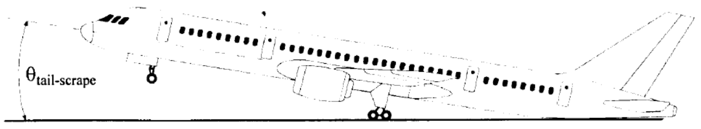


Figure B.2: Tail-scrape angle of a typical civilian transport aircraft [78]

A derivation for the ground run distance can be found in Elements of Aircraft performance [77].

$$s_{ground} = \frac{W V_{LOF}^2}{2g(\bar{T} - \bar{D} - \bar{D}_g)} \tag{B.1}$$

$\bar{T}, \bar{D}$ and $\bar{D}_g$ are values that occur at $V = V_{LOF}/\sqrt{2}$. After writing out the aerodynamic drag and tire friction, it can be seen that the distance is a function of thrust-to-weight, surface-to-weight, C_D and friction of the tires μ and V_{LOF}. It is difficult to determine V_{LOF} since besides the maximum wing loading it also depends on the tail-scrape angle. Reckzeh [59] takes $C_{L_{max}}$ into account so:

$$s_{ground} = f\left(\frac{T}{W}, \frac{W}{S}, C_D, \mu, C_{L_{max}}\right) \tag{B.2}$$

The climb-out is the part between lift-off and V_2. Ruijgrok [77] provides the following formula to calculate the distance:

$$\frac{V_{scr}^2 - V_{LOF}^2}{2g} = sin\gamma_{scr} s_{climb} - h_{scr} \tag{B.3}$$

V_{scr} and s_{scr} are the velocity and the distance traveled when the airplane crosses the screen. The height of the screen h_{scr} is constant and the steady climb angle at screen height γ_{scr} is $(T - D)/W$ when assuming constant $T - D$. After writing the thrust and drag term in full, it becomes clear that the thrust-to-weight and lift-to-drag ratio is important.

$$s_{climb} = f\left(\frac{L}{D}, \frac{T}{W}\right) \tag{B.4}$$

This is also true for the phase after the airplane crosses the screen and continues a steady climb. Equation B.3 shows that a low lift-to-drag ratio is important but also a high maximum lift coefficient to some extent. Equation B.4 shows that during the climb phase the maximum lift coefficient is not important. This explains why airplanes have a high-lift device setting with little deflection during take-off.

B.2. Landing

The EASA part 25 requirements are summarized in figure B.3. The airplane approaches with a glide-path angle of 3 degrees. When it reaches an altitude of 50 feet above ground it should have an approach speed of at least 1.23 times the stall speed when in landing

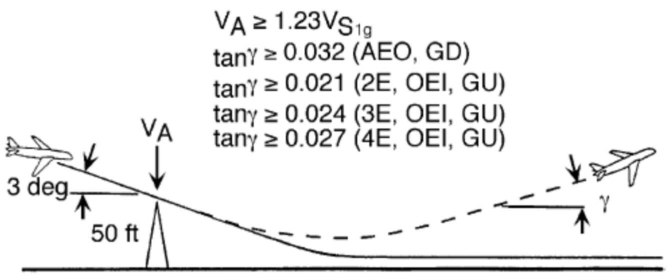

$V_A \geq 1.23V_{S_{1g}}$
$\tan\gamma \geq 0.032$ (AEO, GD)
$\tan\gamma \geq 0.021$ (2E, OEI, GU)
$\tan\gamma \geq 0.024$ (3E, OEI, GU)
$\tan\gamma \geq 0.027$ (4E, OEI, GU)
V_A
3 deg
50 ft
γ

Figure B.3: Landing specifications according to EASA CS-25 [52]

configuration. There is an airborne phase, a ground phase and the possibility of a climb phase when landing is aborted. Ruijgrok [77] gives the following equation which expresses the distance of the airborne phase during landing:

$$s_{approach} = \frac{\frac{V_{Appr}^2}{2g} - \frac{V_T^2}{2g} + h_{scr}}{\frac{1}{2}\left[sin\bar{\gamma}_{Appr} + \left(\frac{C_D}{C_L}\right)_T\right]} \tag{B.5}$$

V_{Appr} is the approach speed and V_T is the touch-down speed. When the approach flight-path angle is substituted for $(T_{Appr} - D_{Appr})/W$, it can be seen that the distance depends on the surface-to-weight ratio, lift-to-drag ratio and especially the maximum lift coefficient. Since the flight-path angle is specified there is no possibility to make this distance after this angle is achieved. However, the deceleration distance on the ground depends heavily on the touch down velocity. From Ruijgrok [77]:

$$s_{deceleration} = \frac{2W^2 1.15^2}{2gS\rho C_{L_{max}}\left(T_{rev} + \bar{D} + \bar{D_g}\right)} \tag{B.6}$$

T_{rev} is the thrust from thrust reversers and $\bar{D_g}$ is the friction force on the tires averaged over the complete ground run. The distance of the ground run then is a function of:

$$s_{deceleration} = f\left(\left(\frac{W}{S}\right)^2, C_{L_{max}}, C_D, \mu, T_{rev}\right) \tag{B.7}$$

μ is the friction coefficient of the tires. It should be noted that the surface-to-weight ratio is to the second power and while the lift force does not have a large influence on the ground run anymore, the surface-to-weight ratio and the maximum lift coefficient hugely define the touch down speed as was mentioned before.

The requirements during a go around is dictate a minimum climb rate. The equation is the same as for the climb-out of the take-off phase. This clashes with the ideal situation for the approach phase which shows that optimal performance of the high lift devices for take off and landing is a difficult exercise between having a high lift-to-drag ratio and a high maximum lift coefficient usually accompanied with a higher drag coefficient in order to keep the descending flight-path angle without accelerating. Dillner and May [79] show a relation between carrier landing accidents and the approach speed. This does not mean that this is exactly the case for landings at airports with commercial airliners. Besides shorter airfield requirements there are also safety benefits from a lower approach speed.

C

Types of high-lift devices

In this section an overview is presented of possible high-lift configurations and deployment mechanisms. At the end, an overview of trailing-edge high-lift devices on current airliners is presented and trends are derived.

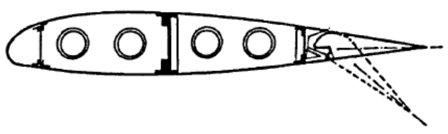
C.1. Differences in Gap and Fowler Motion

No Gap

In the early days of aviation, there was not a real need to equip commercial airliners with high-lift devices. This was because the wing loading of such aircraft was relatively low and the ratio between take-off and landing speed and cruise speed was only around 1:2. However, simple high-lift devices were used for other functions such as glide-slope control and to improve pilot vision by reducing the pitch-up angle during low-speed flight. [13] Two popular high-lift devices back then were the split flap and the plain flap.

The plain flap is the rear part of the wing hinging around a point within the contour. It is basically an aileron. Figure C.1a provides a sketch of a plain flap. Plain flaps were installed, for example, on the Breguet 14 reconnaissance/bomber aircraft. The plain flap increases the effective camber of the airfoil which shifts the lift curve upward but the stall angle of attack is reduced. Also the drag coefficient is reduced together with the lift-to-drag ratio. The flap deflection angle can be increased to about 15 degrees without flow separation. However, the lift coefficient keeps increasing until deflection angles of 70 degrees (for flap lengths up to 30 percent of the chord). Roskam and Lan [80] warn that a gap between flap and main element can seriously reduce the lift coefficient due to leakage between the high pressure on the lower side and the low pressure on the upper side.

The split flap only uses the lower part of the rear section of the surface to hinge, see figure C.1b. It produces a higher lift increase compared to the plain flap and since the upper surface is uncambered it is less likely to separate under high angles of attack. However, the drag increases a lot compared with the plain flap because of the large wake. Split flaps taking 20 to 25 percent of the chord together with deflection angles of 60 to 70 degrees lead to high increases in lift coefficient [80]. It is also a good speed


(a)

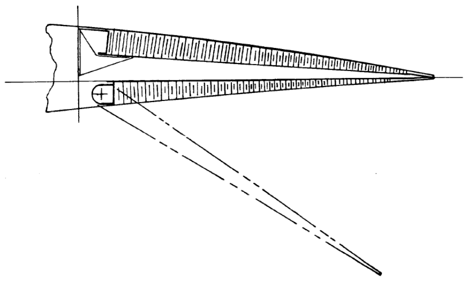

(b)

Figure C.1: (a) Plain flap [81] and (b) split flap [13]

brake because it produces drag without losing lift, contrary to a spoiler [13]. The famous Douglas DC-3 was equipped with split flaps.

Slotted Flaps

Commercial airplanes became faster, their wing-loading increased and a real need for high-lift devices emerged in order to obtain reasonable take-off and landing speeds. Jet engines required even higher speeds in order to be efficient which further increased wing loading. Therefore, wing sweep was added to decrease cruise drag. This led to the typical commercial aircraft we know today. For economical reasons the airfield length cannot be too large. For safety reasons the approach speed cannot be too high and also tires wear faster with high velocities on the ground. This required better high-lift devices [13]. Also leading-edge high-lift devices saw a lot of development.

The simple slotted flap is like a plain flap but with the hinge line outside (and usually below) the airfoil contour. In this way it has a fully developed aerodynamic leading edge. The simple slotted flap only provides a little Fowler motion. Rudolph [13] states that the lower cove panel needs to round upwards in order to allow a good entry for the flow through the slot. The dumping effect, off-the-surface pressure recovery and a fresh boundary layer (aerodynamic effects described in Appendix A) lead to a much larger lift increase and a smaller drag increase compared with the plain flap. Roskam and Lan [80] state that the aerodynamic characteristics are very sensitive to the geometry of the area around the gap. In the years after WWII the simple slotted flap was popular. For the Fokker F27 simple slotted configurations with one or two slots were considered and also a Fowler flap. The simple flap was preferred over the Fowler flap because a simple hinge mechanism could be used and it offered a reasonable section pitch coefficient [75].

Fowler Flap

The Fowler flap is very similar to a simple slotted flap but it also moves backwards (Fowler motion) while deflecting. This increases effective wing area which increases lift without having a large drag increase. There is quite some overlap between flap and upper surface when undeployed. According to Rudolph [13], deployment angles of 40 degrees can be reached without separation. The single-slotted Fowler flap was popular for early jet airplanes. Later double and triple-slotted flaps became more popular.

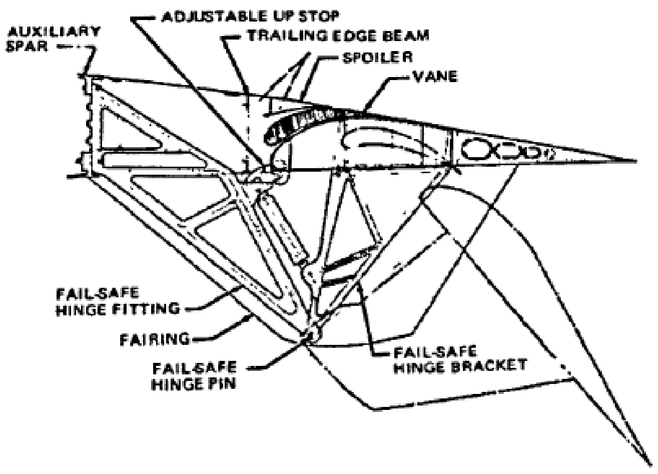


Figure C.2: Double-Slotted Flap with Articulating Vane as on the Douglas DC-10 Aircraft [13]

C.2. DIFFERENCE IN NUMBER OF SURFACES

SINGLE-SLOTTED FLAPS

The slotted and Fowler flaps presented in the previous section are single slotted flaps. They are called like this because there is only one gap between the main element and the flap. They can be applied with the simplest of deployment mechanisms. Figure C.5 shows a simple slotted flap in combination with a drooped hinge mechanism.

DOUBLE-SLOTTED FLAPS

Double slotted flaps consist of two parts. There can be a smaller front or aft element and a main element. The front element, often called vane, can be fixed or articulate (movable). The vane enables deflection angles up to 55 degrees which means that a slightly higher maximum lift coefficient can be obtained while the mechanism can remain simple. The fixed vane was present on some commercial airliners like the Douglas DC-9 and MD-80. Due to the presence of gap also for small deflection angles, there is a decrease in lift-to-drag ratio. Therefore, the vane can be made articulate so it is in contact to the main element during small deflection angles and then moves forward at higher deflection angles, see figure C.2. The vane is usually not actively actuated but spring loaded. It stays retracted because it is pushed against a stow stop. The articulating vane was used on the Douglas DC-10 and the MD-11. [13]

A double-slotted flap with an aft element requires a more complex mechanism than with a vane. However, a total deflection of 60 to 65 degrees can be achieved. Thereby, also extending the effective chord even more because of Fowler motion. Hence, it produces slightly more lift than the vane. The Airbus A300 uses double-slotted flaps with an aft element, see figure C.3

TRIPLE-SLOTTED FLAPS

The triple-slotted flap is a combination of a vane, main element and aft element. The addition of the extra element requires a complex and heavy mechanism. However, it can have the highest maximum lift coefficient because the deflection angle can go up

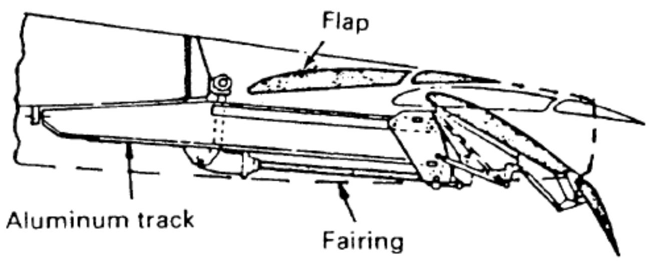


Figure C.3: Double-Slotted Flap with Aft Element as used on the Airbus A300 aircraft [13]

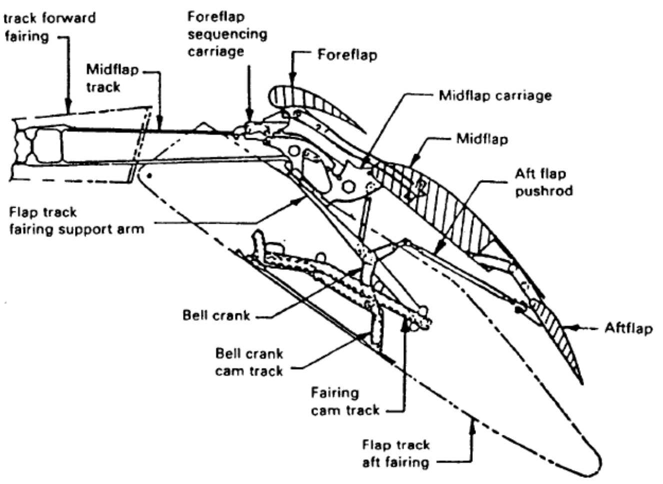


Figure C.4: Triple-Slotted Flaps on the Boeing B737 [13]

to 80 degrees and it provides a lot of Fowler motion. On the other hand it produces quite some induced drag and a substantial downward pitching moment. Nonetheless, triple-slotted flaps have been very popular in the past as they are used on the Boeing B727, B737 (see figure C.4) and B747.

C.3. Difference in Deployment Mechanism

Drooped Hinge

The hinge needs to be fairly deep and far from the main wing element. It could be possible that an extra mechanism is needed to support the side loads on the flaps or fairings due to the length of the hinge fitting. This leads to another problem with simple hinges: for swept wings the rear part of the fairing rotates into the flow producing drag.

4-bar linkage

There are four types discussed by Rudolph [13]: upright, upside-down, upside-down/upright and complex. The naming comes from the orientation of the driver and follower link. Upright means that the link is hinged at its bottom end while upside-down means that the link hinges at its top end. Figure C.6 shows the four different types of 4-bar mechanisms.

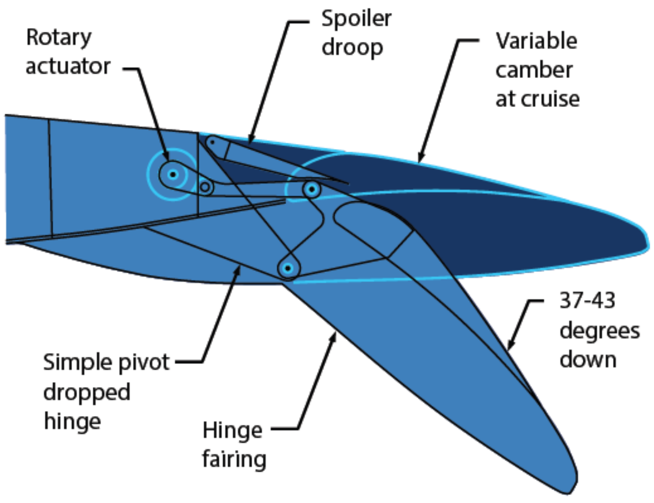
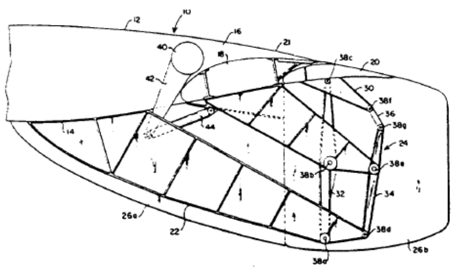
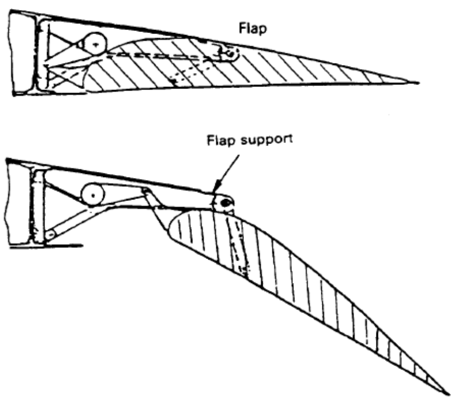


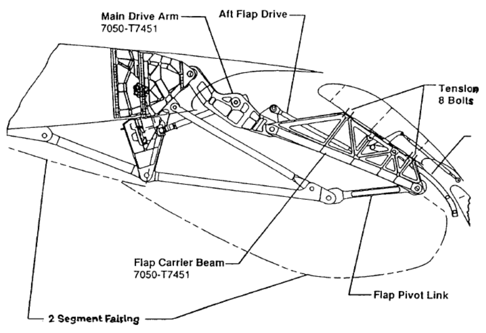
Figure C.5: Sketch of the Drooped Hinge Mechanism on the Boeing B787 aircraft [82]



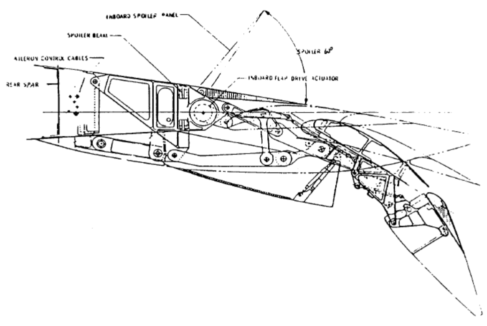
(a) 4-bar Upright



(b) 4-bar Upside-Down



(c) 4-bar Upside-Down/Upright



(d) 4-bar Complex

Figure C.6: Four 4-bar Mechanism Types [13]

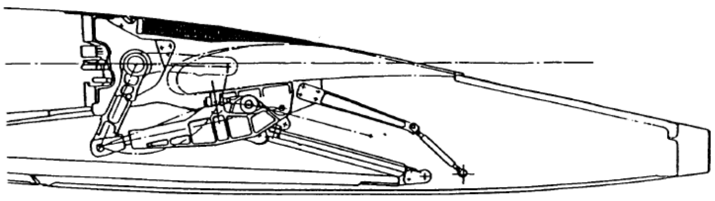


Figure C.7: Sketch of the Link-Track Mechanism used on the Airbus A320 aircraft [13]

The upright 4-bar mechanism reduces fairing depth by 30 to 35 percent compared with a simple hinge. The upside-down variant has more advantages though. The hinges are fixed to the main wing element structure and therefore there is almost no fairing necessary as can be seen in figure C.6b. The main disadvantage for this second type is that there is some counter rotation during in the initial part of the motion. This makes an automatically moving vane problematic since there's no suction at that point. The lift-over-drag ratio is quite good because of a high Fowler motion at small flap angles. Also an adaption can be made to allow streamwise conical motion. Actuation power requirements can be quite high, though.

The upside-down/upright mechanism is a variant of the upside-down variant. There are two combinations possible for the upside-down variant: upside-down in front and the aft link upright or the front link upright and the aft link upside-down. Rudolph states that upside-down link forward seems to have the most promise. It requires a larger fairing compared with the pure upside-down type but it has the same high actuation power requirements.

The idea behind the complex four-bar mechanism is to obtain another few percents of Fowler motion but this makes the mechanism much more complex. There are also a lot of joints placed in series which increases the probability of failure. A successful implementation is used on the Boeing B767. It creates a lot of Fowler motion at low flap angles but it does not allow good conical motion [13]

Link-Track

This type provides a large amount of Fowler motion when a carriage rides over a straight track. The carriage hinges at the end of the track in order to increase the deflection angle. Figure C.7 shows a sketch of this deployment mechanism type. This setup decreases loads on the roller because the flap is free to rotate around the hinge which means that only forces and no couple is experienced by the roller and track flanges. This makes the design less prone to jamming. With some changes the mechanism can be used for streamwise conical motion.

Hooked Track

The hooked track is a downwards sloped track which provides the Fowler motion at low deployment angles and a a hooked down part which provides the flap rotation. Figure C.8 shows a sketch of the hooked-track mechanism. Advantages are that it is good for conical streamwise flap deployment. The major disadvantage is that rollers and tracks experience high loads. This type of deployment mechanism is more prone to jamming and roller wear. Also fatigue limits the life of its components a lot. The circular track as on the B707 is a variant of the hooked track. Actually, there can be a lot of design

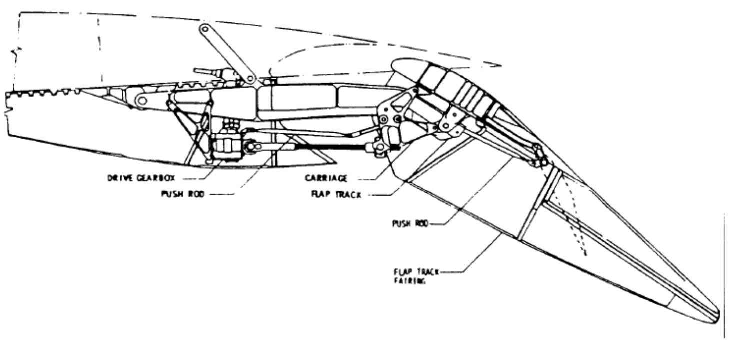


Figure C.8: Sketch of the Hooked-Track Deployment Mechanism as used on the Boeing B757 [13]

freedom for track similar to the hooked track but with a specific shape so the flap can be positioned accurately during every moment of the deployment. It is even possible to completely integrate the track inside the flap so there is no drag of a fairing [83].

Trends

A look at the table shows that the two main manufacturers were converging towards their own preferred deployment mechanism. Airbus uses the link-track while Boeing prefers the hooked-track. However, most recent aircraft, such as the B787, A350 and Mitsubishi Regional Jet (MRJ) used the drooped hinge mechanism.

Table C.1: Application of deployment mechanisms on commercial airliners. [based on [13] and [84]]

Boeing		McDonnell Douglas		Airbus		Other	
707	Circular Track	DC-8	Four-bar	A300B	Hooked-track	E170	Link-track
727	Hooked-track	DC-9	Drooped hinge	A310	Hooked-track	MRJ70	Drooped hinge
737	Hooked-track	DC-10	Drooped hinge	A320	Link-track	CRJ 700	Drooped hinge
747	Hooked-track	MD-80	Drooped hinge	A330	Track-link	F70/F100	Hooked-track
747SP	Four-bar	MD-11	Drooped hinge	A340	Track-link	BAe 146	Curved track
757	Hooked-track			A350	Drooped hinge		
767	Six-bar			A380	Link-track		
777	Six-bar inboard						
	Four-bar outboard						
787	Drooped hinge						

D

KAHN'S ALGORITHM IMPLEMENTED CODE

This code is an implementation of Kahn's algorithm for DAG sorting. The system architecture that was defined in the System Synthesis prototype contains the data to construct the DAG. The habit was formed to always add a prefix to the ports of a connector between two components. One port is prefixed with 'ip' which stands for input port, the other port is prefixed with 'op', being output port. The addition of this prefix has two advantages. Firstly, it makes it easier to create the architecture. Debugging an architecture description is sometimes cumbersome because the definition is spread over multiple text files. The prefix helps to identify two components that are connected. The second advantage makes it easier to recreate the DAG of the architecture. The prefixes give the direction of a connector. The code could be improved by using the realization description of the architecture where a more concrete direction of the connectors is defined. Namely, it is in the realization component file that the interfaces are defined. An example of this interface can be found below. Consequently, the code does not rely anymore on the assumption that the ports have the 'ip' or 'op' prefix. The code corresponds with the pseudo-code presented in section 3.2.

```
import Style VLMotion;

Interface Type MAS_IF expressed_in VLMotion =
{
Participant1 = "Master";
Participant2 = "Slave";

Connect Port MAS (VLMotion_AxisPort(in), VLMotion_AxisPort(out));
Connect Port P(VLMotion_AxisPort(in), VLMotion_AxisPort(out));
Connect Port Q(VLMotion_AxisPort(in), VLMotion_AxisPort(out));
Connect Port R(VLMotion_AxisPort(in), VLMotion_AxisPort(out));
}
```

This code block has been masked due to confidentiality

E

Code of macro for the creation of input axis systems

The steps of the macro are visualized in figure E.1. The macro starts by asking for a name to label the axis systems. The three axis systems that communicate the origin, Z-direction and X-direction get '_DATA_P', _DATA_Q' and _DATA_R' as suffix after the given name. The main MAS is given '_MAS' as suffix. Subsequently, the macro creates the relations between the hard points and the motion axis systems. Furthermore, the publications are added. Then, the macro asks the user to click on the CAD part where the corresponding part axis system should be located. Finally, it automatically adds the relations, see 3.1, that let the PAS correspond with the MAS.

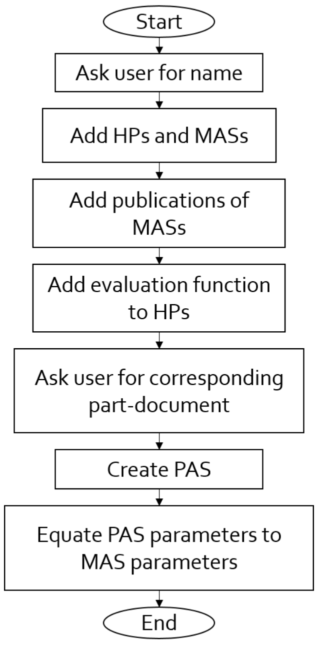

Start
Ask user for name
Add HPs and MASs
Add publications of MASs
Add evaluation function to HPs
Ask user for corresponding part-document
Create PAS
Equate PAS parameters to MAS parameters
End

Figure E.1: Flowchart of the input MAS macro

This code block has been masked due to confidentiality

F

Code of macro for the creation of output axis systems

Practically speaking, this macro publishes a part axis system (PAS). In order to do this, four motion axis systems (MASs) need to be defined in the analysis document. Similar to the input axis systems, there is one main MAS and three auxiliary MASs: P, Q and R. The four MASs are published with high precedence so four other MASs can intercept their position if they are published with low precedence.

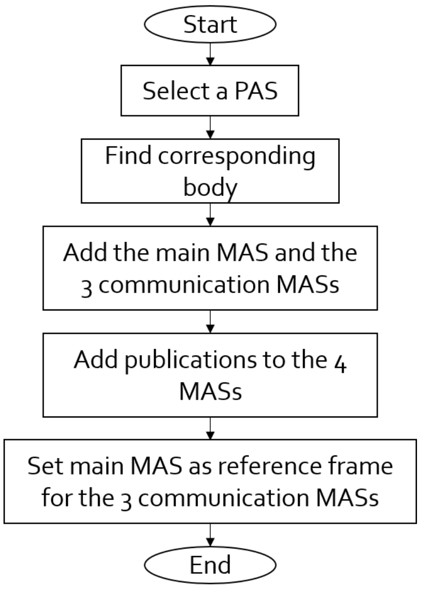

Start
Select a PAS
Find corresponding body
Add the main MAS and the 3 communication MASs
Add publications to the 4 MASs
Set main MAS as reference frame for the 3 communication MASs
End

Figure F.1: Flowchart of output MAS genertor macro

This code block has been masked due to confidentiality

G
Code of macro for axis system transformations

This macro uses functions in order to avoid repetition of code. First, the modeller is asked to select a motion axis system (MAS). Then, a VBA 'user form' appears where the modeller can specify the new position and orientation of the MAS. The angles can be specified with three different definitions: Euler angles (ZX'Z"), Tait-Bryan angles (XY'Z") and another form of Tait-Bryan angles (ZY'X"). The use form is visualized in figure G.1. Using quaternion rotations, the two points defining the Z-axis and X-axis are rotated and then the origin point P together with the new Q and R points are translated.

A rotation in 3D space can be represented as a unit quaternion [85]. The unit quaternion representing a rotation with ZX'Z" Euler angles is the following:

$$\mathbf{q}_{313}(\phi,\theta,\psi) = \begin{bmatrix} c_{\phi/2}c_{\theta/2}c_{\psi/2} - s_{\phi/2}c_{\theta/2}s_{\psi/2} \\ c_{\phi/2}c_{\psi/2}s_{\theta/2} + c_{\phi/2}s_{\theta/2}s_{\psi/2} \\ c_{\phi/2}s_{\psi/2}s_{\theta/2} - s_{\phi/2}c_{\psi/2}s_{\theta/2} \\ c_{\phi/2}c_{\theta/2}s_{\psi/2} + c_{\theta/2}c_{\psi/2}s_{\phi/2} \end{bmatrix} \tag{G.1}$$

Cosines are abbreviated as c and sines are abbreviated as s. A rotation of a point represented as vector $\vec{v}$ to a new vector $\vec{v^*}$ using the unit quaternion $\mathbf{q} = (q_0, q_1, q_2, q_3)$ can be found with the following relation [85]:

$$\vec{v^*} = \vec{v} + 2\vec{r} \times (\vec{r} \times \vec{v} + q_0\vec{v}) \tag{G.2}$$

$\vec{r} = (q_1, q_2, q_3)$

This code block has been masked due to confidentiality

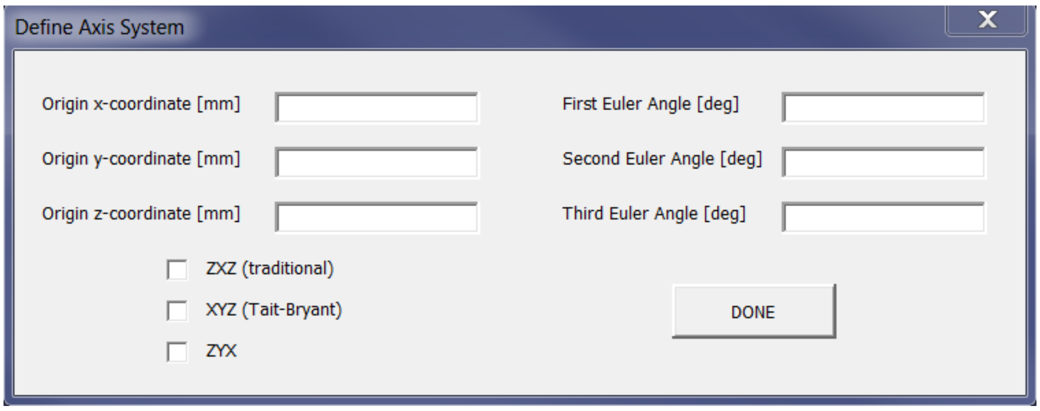


Figure G.1: User form window asking for the new position and orientation of the motion axis system.

H

Information on the exposed component input parameters

This appendix gives a detailed explanation of each exposed input parameter of the components that were presented in chapter 4.

Wing

The outboard deployment mechanism position along the spar is expressed by parameter 'OB_DeplMech_Postion'. It is a distance on the line of the spar section between the wing kink and the fuselage c/a. Similar for the inboard deployment mechanism this is b/a. The transmission line (where the transmission shaft is located) is a double translation of the spar line. Actually, always the beginning and end point of the line are translated. The transmission line is defined by those two points. In this way, proportionality of the wing is kept as the wing thickness decreases along the wing tip direction. First, a vertical translation happened along a distance d that signifies a ratio w.r.t. the spar height. Then a translation of e is performed horizontally. This is not a ratio as the distance is mostly determined by the radius of the transmission shaft so it fits between the spar and the flap. The outboard branched gearbox position 'OB_GB_Position' is a ratio along the transmission line f, starting from the fuselage side. Similarly, the inboard gearbox is positioned at a distance g. The inboard and outboard actuator positions are a negative vertical translation that runs parallel to the wing spar, i and h respectively.

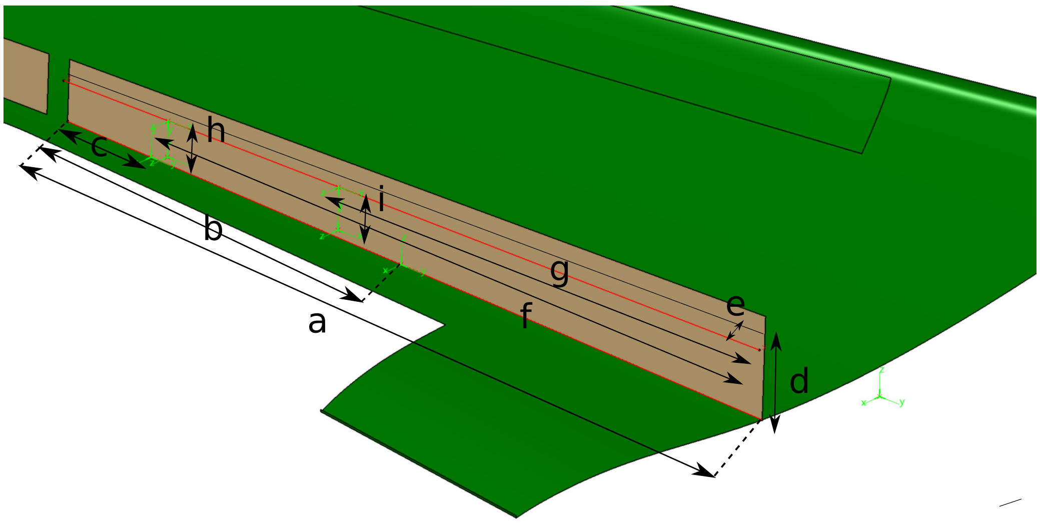


Figure H.1: Wing parameters

FLAP

The position and orientation of the flap trajectory is determined by the trajectory points that are referenced w.r.t. the global axis system using Tait-Bryan angles. The rotation sequence is XY'Z". The first rotation around the x axis corresponds with the roll angle ψ. The second angle is θ and the third angle represent roll ϕ. In figure H.2 e is the flap span. The parameters DeplMech_Spacing and Linkage_Spacing are ratios w.r.t. the flap span e. DeplMech_Spacing = a/e and Linkage_Spacing = b/e. The longitudinal position is the ratio with respect to the flap chord f. DeplMech_LongPosition = d/f and Linkage_LongPosition = c/f.

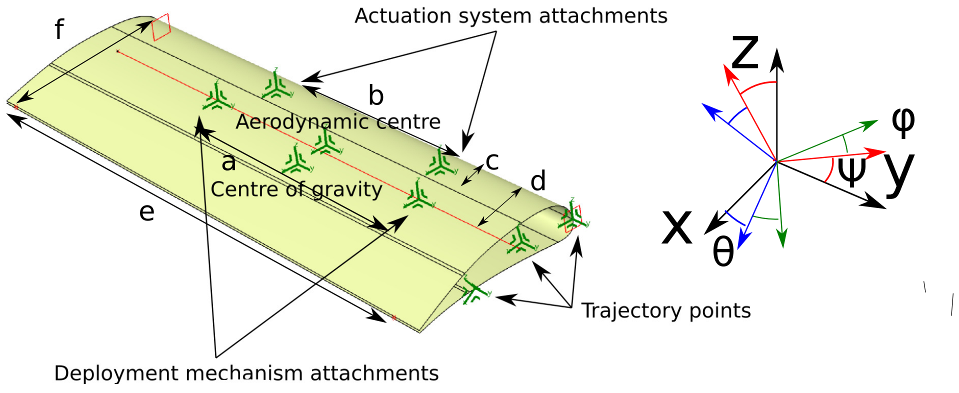


Figure H.2: Flap parameters

DEPLOYMENT MECHANISM

The jamming location is expressed as the ratio where the jamming occurs w.r.t. the maximum range of the movement. For the track-carriage mechanisms this is position on the spline of the track: JammingLocation = a/b

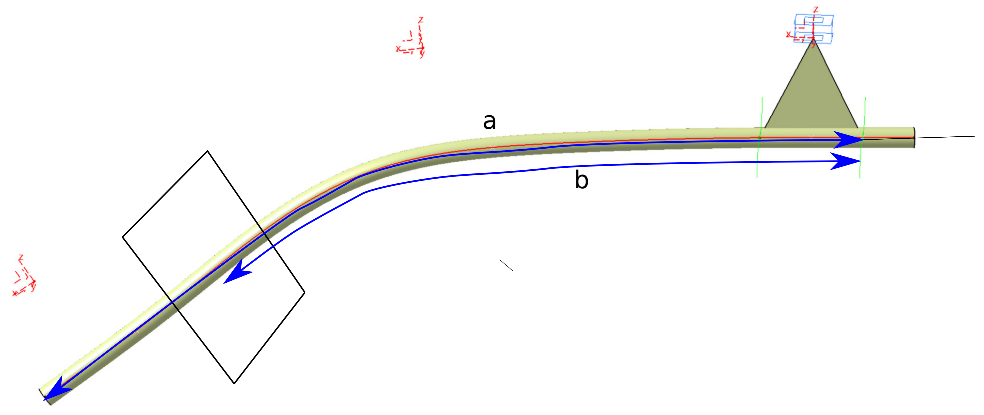


Figure H.3: Parameters of deployment mechanism with track and carriage

For the drooped hinge mechanism this is the ratio between the angle at which the mechanism jamms and the maximal deflection angle: JammingLocation = α/β

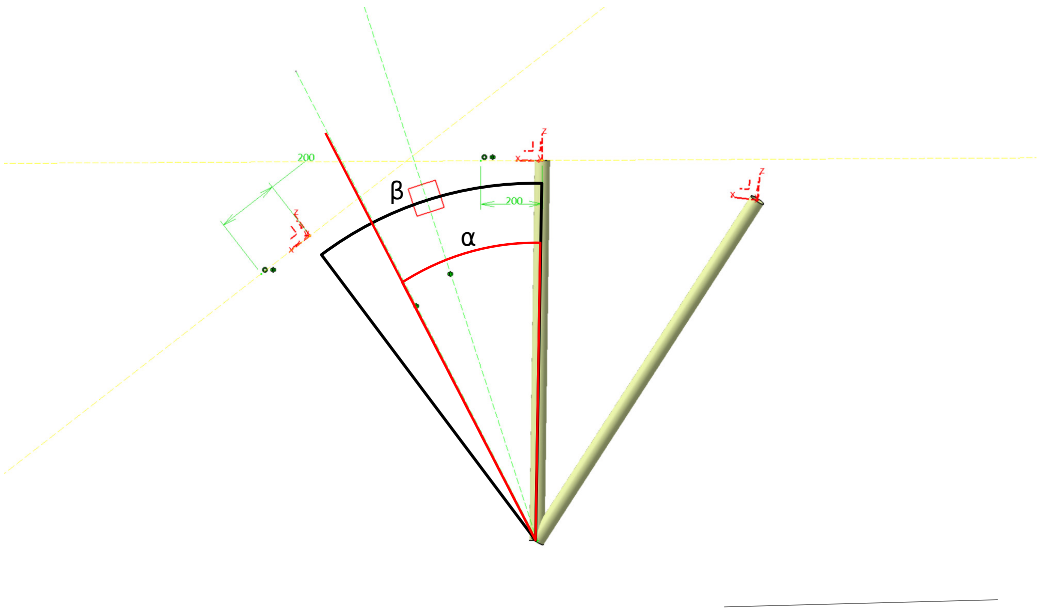


Figure H.4: Parameters for deployment mechanism for drooped hinge.

POWER DRIVE UNIT

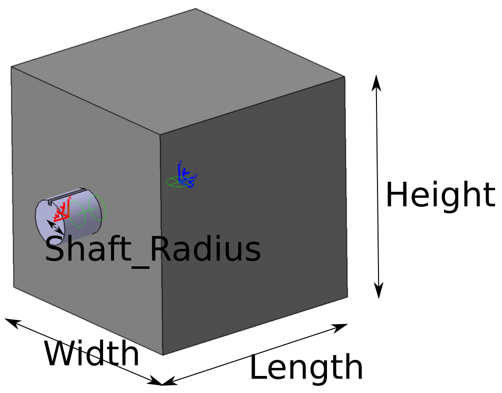


Figure H.5: power drive unit parameters

BRANCHED GEARBOX

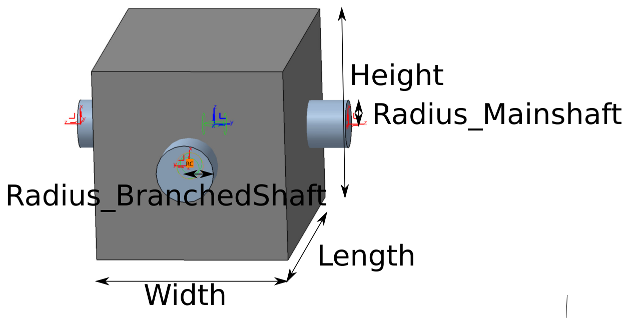


Figure H.6: Branched gearbox parameters

BEVEL GEARBOX

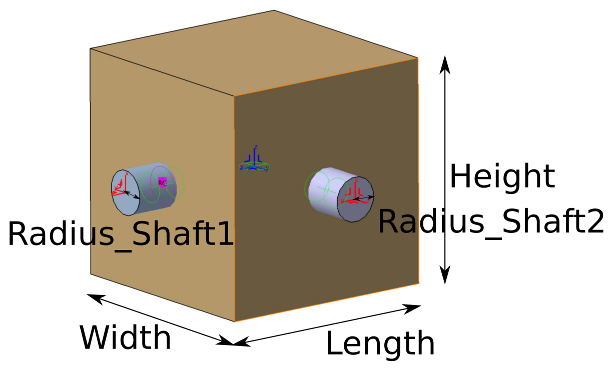


Figure H.7: Bevel gearbox parameters

CONTROLLER

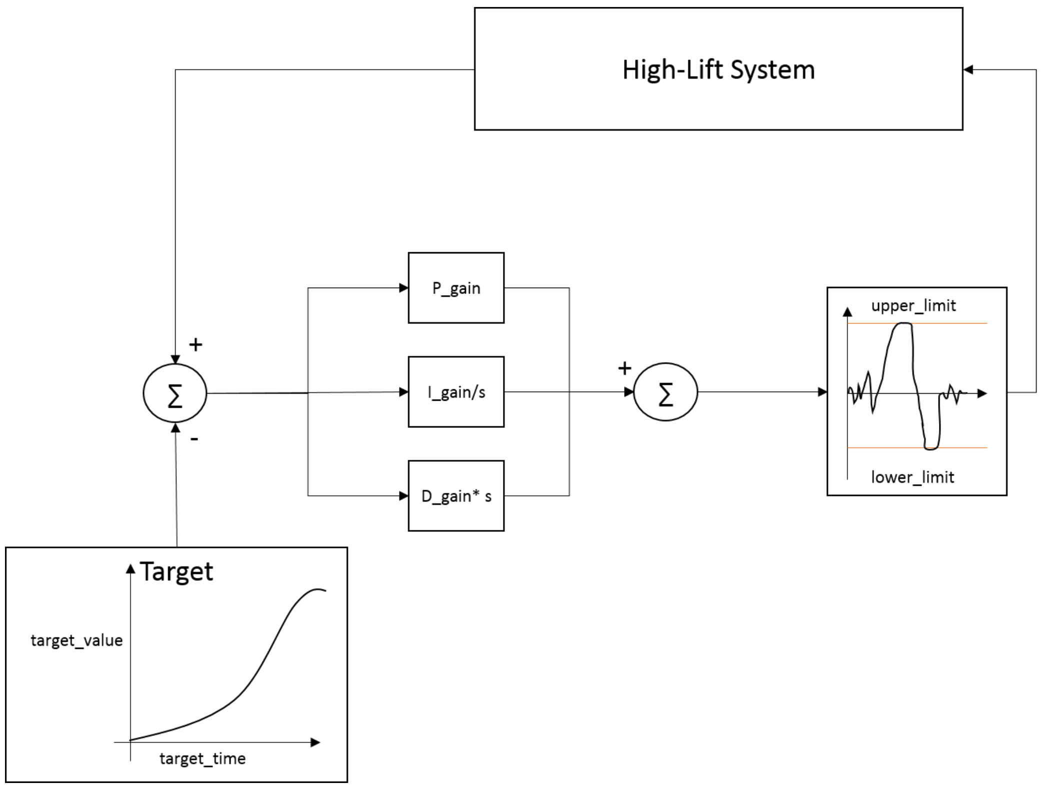


Figure H.8: Controller parameters

BIBLIOGRAPHY

[1] G. Warwick and G. Norris, *Designs for Success,* Aviation week & Space Technology , 72 (2010).

[2] T. Baumann, *Simulation-driven design,* Aerospace Engineering , 6 (2011).

[3] S. Becz, A. Pinto, L. E. Zeidner, R. Khire, A. Banaszuk, and H. M. Reeve, *Design system for managing complexity in aerospace systems,* in *10th AIAA ATIO/ISSMO Conference* (AIAA, Fort Worth, TX, 2010) pp. 1–7.

[4] M. H. Sadraey, *Aircraft Design: A Systems Engineering Approach* (John Wiley & Sons, 2012).

[5] S. Szykman, S. J. Fenves, W. Keirouz, and S. B. Shooter, *A foundation for interoperability in next-generation product development systems,* Computer-Aided Design **33**, 545 (2001).

[6] W. J. Verhagen, P. Bermell-Garcia, R. E. C. Van Dijk, and R. Curran, *A critical review of Knowledge-Based Engineering: An identification of research challenges,* Advanced Engineering Informatics **26**, 5 (2012).

[7] G. La Rocca, *Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design,* Advanced Engineering Informatics **26**, 159 (2012).

[8] F. Donida, G. Ferretti, S. M. Savaresi, and M. Tanelli, *Object-oriented modelling and simulation of a motorcycle,* Mathematical and Computer Modelling of Dynamical Systems **14**, 79 (2008).

[9] W. Borutzky, *Bond graph modelling and simulation of multidisciplinary systems - An introduction,* Simulation Modelling Practice and Theory **17**, 3 (2009).

[10] Y. Lemmens, W. Dehandschutter, I. Becuwe, and T. Olbrechts, *Architecture-driven Design Study Of An Electrically-Powered UAV,* in *EMEASEC 2014* (Cape Town, South Africa, 2014) pp. 1–12.

[11] Anonymous, *INCOSE Systems Engineering Vision 2020,* Tech. Rep. September (INCOSE, 2007).

[12] Anonymous, *A World in Motion: Systems Engineering Vision 2025,* Tech. Rep. (INCOSE, 2014).

[13] P. K. C. Rudolph, *NASA Contractor Report,* Tech. Rep. 4746 (NASA, Seattle, Washington, 1996).

[14] R. Marty, *Object oriented programming,* Computer Physics Communications **38**, 181 (1985).

[15] J. Ludewig, *Models in software engineering – an introduction,* Softw Syst Model Digital Object Identifier **2**, 5 (2003).

[16] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language* (Morgan Kaufmann, 2014).

[17] INCOSE, *INCOSE Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities* (John Wiley & Sons, Inc., 2015).

[18] C. Chapman and M. Pinfold, *Design engineering—a need to rethink the solution using knowledge based engineering,* Knowledge-Based Systems **12**, 257 (1999).

[19] D. Rondeau and K. Soumilas, *The primary structure of commercial transport aircraft wings: Rapid generation of finite element models using knowledge-based methods abstract,* in *Proceedings of the 1999 Aerospace Users' Conference* (MacNeal-Schwendler Corporation, 1999).

[20] A. Corallo, R. Laubacher, A. Margherita, and G. Turrisi, *Enhancing product development through knowledge-based engineering (KBE): A case study in the aerospace industry,* Journal of Manufacturing Technology Management **20**, 1070 (2009).

[21] Z. Zhu, M. van Tooren, and S. der Elst, *On the development of a heuristic routing application for the automatic wire harness design in the aircraft,* in *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference* (2011).

[22] G. La Rocca and M. J. L. van Tooren, *Enabling distributed multi-disciplinary design of complex products: a knowledge based engineering approach,* Journal of Design Research **5**, 333 (2007).

[23] M. Negnevitsky, *Artificial intelligence: a guide to intelligent systems* (Pearson Education, 2005).

[24] A. Morris, P. Arendsen, G. La Rocca, M. Laban, R. Vos, and H. Hönlinger, *Mob-a european project on multidisciplinary design optimisation,* in *Proceedings of the 24th Congress of International Council of the Aeronautical Science* (2004) pp. 1–13.

[25] A. Sangiovanni-Vincentelli, *Is a unified methodology for system-level design possible?* IEEE Design and Test of Computers **25**, 346 (2008).

[26] G. La Rocca, T. H. M. Langen, and Y. H. A. Brouwers, *The Design and Engineering Engine. Towards a Modular System for Collaborative Aircraft Design,* 28th International Congress of the Aeronautical Sciences , 1 (2012).

[27] G. Muller, *CAFCR: A multi-view method for embedded systems architecting*, Ph.D. thesis, Delft University of Technology (2004).

[28] B. Graaf, *Model-driven evolution of software architectures*, Ph.D. thesis, Delft University of Technology (2007).

[29] L. Horvath, J. Fodor, and I. J. Rudas, *Manufacturing aspect of the IBCA structure for active knowledge content representation in product model,* IFAC-PapersOnLine **48**, 1616 (2015).

[30] M. Efatmaneshnik and M. J. Ryan, *On Optimal Modularity for System Construction,* Complexity **21**, 176 (2016).

[31] N. Chungoora, R. I. Young, G. Gunendran, C. Palmer, Z. Usman, N. A. Anjum, A. F. Cutting-Decelle, J. A. Harding, and K. Case, *A model-driven ontology approach for manufacturing system interoperability and knowledge sharing,* Computers in Industry **64**, 392 (2013).

[32] J. Hirtz, R. Stone, D. McAdams, S. Szykman, and K. Wood, *A functional basis for engineering design: Reconciling and evolving previous efforts,* Research in Engineering Design **13**, 65 (2002).

[33] T. Kurtoglu and M. I. Campbell, *Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping,* Journal of Engineering Design **20**, 83 (2009).

[34] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama, *Supporting conceptual design based on the function-behavior-state modeler,* Ai Edam **10**, 275 (1996).

[35] M. S. Erden, H. Komoto, T. J. Van Beek, V. D'Amelio, E. Echavarria, and T. Tomiyama, *A Review of Function Modeling: Approaches and Applications,* Ai Edam **22**, 147 (2008).

[36] A. Alvarez Cabrera, *Architecture-Centric Design: Modeling and Applications to Control Architecture Generation,* Phd thesis, Delft University of Technology (2011).

[37] Y. Umeda and T. Tomiyama, *FBS Modeling: Modeling Scheme of Function for Conceptual Design,* Proceedings of the 9th International Workshop on Qualitative Reasoning , 271 (1995).

[38] Y. Umeda, S. Fukushige, K. Tonoike, and S. Kondoh, *Product modularity for life cycle design,* CIRP Annals - Manufacturing Technology **57**, 13 (2008).

[39] S. Tamaskar, K. Neema, and D. A. DeLaurentis, *Framework for measuring complexity of aerospace systems,* Research in Engineering Design **25**, 125 (2014).

[40] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks,* Physical Review E - Statistical, Nonlinear, and Soft Matter Physics **69**, 1 (2004).

[41] T. J. Van Beek, M. S. Erden, and T. Tomiyama, *Modular design of mechatronic systems with function modeling,* Mechatronics **20**, 850 (2010).

[42] B. Shekar, R. Venkataram, and B. M. Satish, *Managing Complexity in Aircraft Design Using Design Structure Matrix,* Concurrent Engineering **19**, 283 (2011).

[43] E. J. Haug, *Computer aided kinematics and dynamics of mechanical systems*, Vol. 1 (Allyn and Bacon Boston, 1989).

[44] A. A. Shabana, *Dynamics of multibody systems* (Cambridge university press, 2013).

[45] W. Schiehlen, *Multibody System Dynamics: Roots and Perspectives,* Multibody System Dynamics **1**, 149 (1997).

[46] Anonymous, *LMS Virtual.Lab Motion On-line help*, Siemens PLM (2017).

[47] E. Obert, *Aerodynamic design of transport aircraft* (IOS press, 2009).

[48] A. M. O. Smith, *High-Lift Aerodynamics,* Journal of Aircraft **12**, 501 (1975).

[49] J. Cole, *Airfoil flap conical extension mechanism,* (1979), uS Patent 4,172,575.

[50] R. Balaji, F. Bramkamp, M. Hesse, and J. Ballmann, *Effect of flap and slat riggings on 2-D high-lift aerodynamics,* Journal of Aircraft **43**, 1259 (2006).

[51] S. P. Schoensleben, *Integrated Trailing Edge Flap Track Mechanism for Commercial Aircraft,* Masters thesis, ETH Zurich (2005).

[52] EASA, *Acceptable Means of Compliance for Large Aeroplanes CS-25*, Tech. Rep. (European Aviation Safety Agency, 2013).

[53] D. Zaccai, *Design Framework for Trailing Edge High-Lift Systems*, Masters thesis, Delft University of Technology (2014).

[54] K. Beyer and L. Krueger, *Design Validation Through Kinematic Simulation: Airplane Flap Design,* in *PLM Conference and technifair* (Las Vegas, Nevada, 2010).

[55] D. Zaccai, F. Bertels, and R. Vos, *Design methodology for trailing-edge high-lift mechanisms,* CEAS Aeronautical Journal **7**, 521 (2016).

[56] C. P. van Dam, *The aerodynamic design of multi-element high-lift systems for transport airplanes,* Progress in Aerospace Sciences **38**, 101 (2002).

[57] A. Flaig and R. Hilbig, *High lift design for large civil aircraft,* in *AGARD-CP-515* (AGARD, Brussels, Belgium, 1993).

[58] R. M. Martins-Pires, V. Lajux, and J. P. Fielding, *Methodology for the Design and Evaluation of Wing Leading Edge and Trailing Edge Devices,* in *25th international congress of the aeronautical sciences* (2006) pp. 1–10.

[59] D. Reckzeh, *Aerodynamic design of the high-lift-wing for a Megaliner aircraft,* Aerospace Science and Technology **7**, 107 (2003).

[60] P. K. C. Rudolph, *NASA Contractor Report*, Tech. Rep. 1998-196709 (NASA, 1998).

[61] R. Anderson, C. Flora, R. Nelson, E. Raymond, and J. Vincent, *Development of weight and cost estimates for lifting surfaces with active controls*, Tech. Rep. (NASA, Seattle, Washington, 1976).

[62] E. Torenbeek, *Prediction of wing group weight for preliminary design,* Aircraft Engineering and Aerospace Technology **43**, 16 (1971).

[63] M. Pfennig and F. Thielecke, *A knowledge-based approach for design and modelling of high lift actuation systems,* Journal of Aerospace Engineering **225**, 302 (2011).

[64] Anonymous, *Clean Sky at a Glance,* (2014).

[65] D. van den Bossche, *the a380 Flight Control Electrohydrostatic Actuators ,*, in *25th international congress of the aeronautical sciences* (2006) pp. 1–8.

[66] R. I. Jones, *The more electric aircraft: assessing the benefits,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **216**, 259 (2002).

[67] I. Chakraborty, D. N. Mavris, M. Emeneth, and A. Schneegans, *A methodology for vehicle and mission level comparison of More Electric Aircraft subsystem solutions: Application to the flight control actuation system,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **229**, 1088 (2015).

[68] P. Y. Papalambros, *Editorial,* Journal of Mechanical Design **132** (2010).

[69] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms second edition* (The MIT Press, 2001).

[70] A. B. Kahn, *Topological sorting of large networks,* Communications of the ACM **5**, 558 (1962).

[71] C. Dickerson and D. N. Mavris, *Architecture and principles of systems engineering* (CRC Press, 2009).

[72] Anonymous, *F.05.01.01 normal force on flaps and controls,* (ESDU, 1973).

[73] T. H. G. Megson, *Aircraft structures for engineering students* (Elsevier, 2012).

[74] G. La Rocca, *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*, Ph.D. thesis, Delft University of Technology (2011).

[75] E. Obert, *Forty Years of High-Lift R&D: An aircraft manufacturer's experience,* in *AGARD-CP-515* (NATO, 1993).

[76] D. Reckzeh, *Aerodynamic Design of Airbus High-Lift Wings,* (2005).

[77] G. Ruijgrok, *Elements of Aircraft Performance* (Delft University Press, 2004).

[78] R. S. Pepper and C. P. Van Dam, *Design Methodology for Multi-Element Civil Transport High-Lift Aircraft Systems*, Tech. Rep. NCC2-5042 (NASA, 1996).

[79] B. Dillner and F. W. May, *Aerodynamic issues in the design of high-lift systems for transport aircraft,* in *Proceedings of the Conferece on Improvmenent of Aerodynamic Performance through Boudary Layer Control and High Lift Systems* (AGARD, Brussels, Belgium, 1984).

[80] J. Roskam and C. T. Lan, *Airplane aerodynamics and performance* (DARcorporation, 1997).

[81] J. Szodruch and H. Schnieder, *High-lift aerodynamics for transport aircraft by interactive experimental and theoretical tool development,* in *27th Aerospace Sciences Meeting* (1989).

[82] T. Nelson, *787 Systems and Performance,* (2005).

[83] Y. Lemmens, M. Barile, and N. Koklas, *Design of composite high lift and load control and alleviation devices for a natural laminar flow wing,* in *NAFEMS NORDIC Seminar* (Stockholm, Sweden, 2015) pp. 1–2.

[84] Anonymous, *Jane's all the world's aircraft,* `www.janes.ihs.com` (2016), accessed on 31/01/2016.

[85] J. Diebel, *Representing attitude: Euler angles, unit quaternions, and rotation vectors,* (2006).