Forschungsbericht 2024-15

Knowledge-based Engineering to provide Aircraft Fuselage Design Details for Multidisciplinary and Multifidelity Analysis Model Generation

Jan-Niclas Walther

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Systemarchitekturen in der Luftfahrt
Hamburg

DLR Deutsches Zentrum
für Luft- und Raumfahrt

Forschungsbericht 2024-15

Knowledge-based Engineering to provide Aircraft Fuselage Design Details for Multidisciplinary and Multifidelity Analysis Model Generation

Jan-Niclas Walther

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Systemarchitekturen in der
Luftfahrt
Hamburg

288	Seiten
155	Bilder
17	Tabellen
460	Literaturstellen

DLR Deutsches Zentrum
für Luft- und Raumfahrt

Herausgeber:

Deutsches Zentrum
für Luft- und Raumfahrt e. V.
Wissenschaftliche Information
Linder Höhe
D-51147 Köln

ISSN 1434-8454
ISRN DLR-FB-2024-15
Erscheinungsjahr 2024
DOI: 10.57676/9ghg-3f88

Erklärung des Herausgebers

Lizenz

KBE, Flugzeugentwurf, Rumpf, MDAO, Multi-Modell-Generierung

Jan-Niclas WALTHER
DLR, Institut für Systemarchitekturen in der Luftfahrt, Hamburg

Wissensbasierte Entwurfsmethodik zur Bereitstellung von Flugzeugrumpf-Konstruktionsdetails für multidisziplinäre und fidelitätsvariable Analysemodellgenerierung
TU Berlin

In dieser Arbeit wird eine wissensbasierte Entwurfsmethodik (KBE, engl.: knowledge-based engineering) vorgestellt, welche einerseits zur Unterstützung der konsistenten disziplinären Modellgenerierung für multidisziplinäre Entwurfsanalyse- und -optimierungsanwendungen (MDAO, engl.: multidisciplinary design analysis and optimization) geeignet ist und andererseits die Möglichkeit zur Einbindung neuartiger Produktarchitekturen bietet.

Auf dem Weg zu einer nachhaltigeren und CO2-neutralen Luftfahrt, ist die inkrementelle Verbesserung von hochoptimierten Luftfahrzeugkonfigurationen mit zunehmend hohem Entwicklungsaufwand verbunden. Die Einführung neuer Technologien wie Flüssigwasserstoffantrieben geht zudem mit unausweichlichen fundamentalen Änderungen der Produktarchitektur einher.

Vor diesem Hintergrund ist der Einsatz rechnergestützter Entwurfs- und Analysemethoden unabdingbar, um zum einen schnellere und präzisere Entwurfszyklen zu ermögliche und zum anderen die frühe Bewertung unkonventioneller Flugzeugarchitekturen zu ermöglichen, die nicht durch empirische Entwurfsmethoden abgedeckt werden können. Zudem ist der Entwurf häufig durch multidisziplinäre Integrationseffekte getrieben, wodurch kollaborative MDAO-Kampagnen nötig werden, an denen eine Vielzahl von Experten und Entwurfsprogrammen beteiligt ist. Um die Kommunikation zu vereinfachen, werden häufig zentrale Datenmodelle wie CPACS (Common Parametric Aircraft Configuration Schema) zum Datenaustausch eingesetzt.

Der Flugzeugrumpf ist von besonderem Interesse, da der Entwurf von vielen höchst unterschiedlichen Disziplinen, wie Struktur- und Kabinenentwurf, Industrialisierung oder Human Factors, beeinflusst wird. Um mit Hilfe von MDAO aussagekräftige Ergebnisse erzielen zu können, müssen all diese Aspekte berücksichtigt werden. Die große Breite eingesetzter Entwurfs- und Analysemethoden und die stark unterschiedlichen Anforderungen, welche Produktdetails bekannt sein müssen, stehen einer Integration detaillierter und umfassender Rumpfanalysen derzeit allerdings im Weg.

Um dieser Problematik entgegenzutreten, werden mit der in dieser Arbeit präsentierten KBE-Methodik Multi-Modellgenerierungsfähigkeiten entwickelt, die es ermöglichen konsistente Analysemodelle in unterschiedlichsten Detaillierungsgraden bereitzustellen. Aufbauend auf CPACS als zentralem Datenmodell verbindet der Ansatz Parameter und ausführbare Regeln, die in deklarativer Form zur Verfügung gestellt werden, zu einem Wissensgraphen. Eine graphenbasierte Formulierung bietet zahlreiche Vorteile, z.B. die Möglichkeit Abhängigkeiten zwischen Parametern nachzuverfolgen oder neues Wissen modular einzubinden. Zudem wird der Programmablauf dynamisch zur Laufzeit auf Basis von Nutzerabfragen bestimmt. Regeln werden demnach nicht ausgeführt, bis sie zur Bearbeitung einer Nutzerabfrage benötigt werden, wodurch die unnötige Ausführung rechenintensiver Regeln vermieden wird.

Das System enthält sowohl Geometriemodellierungs- als auch Entwurfsregeln. Es wird eine parametrische Modellierungsengine für CPACS vorgestellt, die detaillierte Regeln für die Geometriegenerierung von Rumpfstruktur und Kabine bereithält. Die Entwurfsregeln ermöglichen die Erstellung komplexer Layouts auf Basis einiger weniger Entwurfsvariablen. Dies erlaubt es, Datensätze, beispielsweise aus dem Flugzeuggesamtentwurf, die nicht alle zur Modellerstellung notwendigen Details beinhalten, mit entsprechenden Daten anzureichern. Im Zusammenspiel erlauben die Regeln somit die flexible Erstellung von Geometrien, die exakt den Genauigkeitsanforderungen unterschiedlicher Analysemodelltypen angepasst sind.

Des Weiteren sind zwei Beispiele für Systemmodifikationen gegeben, welche der Integration neuer Produktarchitekturen dienen sollen: Eine Modifikation ist einer Nurflüglerkonfiguration gewidmet, während die zweite der Integration großer kryogener Tanks im Rumpf dient.

Die Vielseitigkeit des KBE-Ansatzes wird mit Hilfe einiger Anwendungsstudien aufgezeigt.

Zunächst werden zwei konventionelle Konfigurationen betrachtet, eine Single-Aisle-Konfiguration und eine Twin-Aisle-Konfiguration mit zwei Decks. Beide Studien werden mit dem konventionellen Basisregelsatz durchgeführt. Anschließend werden Studien für eine unkonventionelle Nurflüglerkonfiguration, sowie eine Wasserstofftankintegration vorgestellt, in denen die jeweiligen Systemmodifikationen zum Einsatz kommen. Zum Abschluss wird die Fähigkeit des Systems demonstriert, Analysemodelle bereitzustellen, die dem Stand der Technik entsprechen. Dazu werden für die Single-Aisle-Konfiguration ein globales Finite-Elemente-Modell und ein Modell zur interaktiven Kabinenvisualisierung für Human-Factors-Analysen erstellt.

KBE, aircraft design, fuselage, MDAO, multi-model generation

(Published in English)

Jan-Niclas WALTHER
German Aerospace Center (DLR), Institute of System Architectures in Aeronautics, Hamburg

Knowledge-based Engineering to provide Aircraft Fuselage Design Details for Multidisciplinary and Multifidelity Analysis Model Generation

TU Berlin

In this work, a knowledge-based engineering (KBE) methodology is introduced, which is capable of supporting the generation of consistent disciplinary analysis models of the aircraft fuselage for multidisciplinary design analysis and optimization (MDAO) applications on the one hand, and open to novel architectures on the other.

On the path towards more sustainable and carbon neutral aviation, increasing efforts are necessary to achieve incremental product improvements for a highly optimized aircraft configuration. Meanwhile the introduction of new technologies, such as liquid hydrogen (LH2), inevitably entails fundamental changes to the overall aircraft architecture.

Against this backdrop, advanced computational design and analysis methods are indispensable to enable faster, more accurate design cycles on the one hand, and to allow for the early assessment of unconventional aircraft architectures beyond the scope of empirical design methods on the other. In addition, the design is often driven by multidisciplinary integration effects, resulting in a need for collaborative MDAO campaigns involving a multitude of experts and computational design tools. To facilitate communication, central data models such as the Common Parametric Aircraft Configuration Schema (CPACS) are often used for design data exchange.

The aircraft fuselage is of particular interest, since its design is impacted by many diverse disciplines, including structural and cabin design, industrialization, and human factors. In MDAO, all of these aspects need to be accounted for to achieve meaningful results. Yet, the large variety of analysis and design methods and the vastly different requirements in terms of available product details currently pose an obstruction to the integration of detailed fuselage assessment.

To address this issue, the KBE methodology presented in this work provides multi-model generation capabilities to enable consistent analysis model generation at very different levels of fidelity. Built upon CPACS as central data model, the approach combines sets of parameters and executable rules, formulated in a declarative manner to assemble a knowledge graph. The graph-based formulation provides a number of benefits, e.g. the possibility to trace dependencies between parameters or incorporate new knowledge in a modular fashion. Furthermore, the program flow is determined dynamically at run time, based on user requests. Consequently, rules are not executed until they are needed to fulfill a user request, thus avoiding the unnecessary performance of computationally expensive tasks.

Both geometry modeling and design rules are included in the system. A parametric modeling engine for the fuselage based on CPACS is presented, which includes detailed modeling capabilities for the structure and the cabin. The design rules enable the generation of complex layouts based on a small set of design variables. This makes it possible to enrich data sets, e.g. from overall aircraft design, which lack necessary details for the model generation. In combination, the rules enable the flexible generation of geometry models to match the fidelity requirements of different types of analysis models.

Two examples of system modifications to accommodate novel architectures are furthermore given, one for a blended wing-body configuration and another for the integration of large cryogenic tanks in the fuselage.

Several application studies are presented to highlight the versatility of the KBE approach. First, two conventional configurations are considered, one single-aisle and one twin-aisle and twin-deck. Both studies are performed using the basic conventional fuselage design rules. Next, studies for the unconventional blended wing-body and LH2 tank integration problems are shown, applying the respective system modifications. Finally, the capacity to provide state of the art analysis models is demonstrated by deriving models for global finite element analysis and

interactive visualization for human factors evaluation for the single-aisle configuration.

Knowledge-based Engineering to provide Aircraft Fuselage Design Details for Multidisciplinary and Multifidelity Analysis Model Generation

vorgelegt von
Dipl.-Ing.
Jan-Niclas Walther
ORCID: 0000-0001-5738-658X

an der Fakultät V – Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr.-Ing. Julien Weiss
Gutachter: Prof. Dr.-Ing. Andreas Bardenhagen
Gutachter: Dr.-Ing. Björn Nagel

Tag der wissenschaftlichen Aussprache: 8. Dezember 2023

Berlin 2024

For my father Horst
and my daughter Joséphine,
who could never meet.

Abstract

In this work, a knowledge-based engineering (KBE) methodology is introduced, which is capable of supporting the generation of consistent disciplinary analysis models of the aircraft fuselage for multidisciplinary design analysis and optimization (MDAO) applications on the one hand, and open to novel architectures on the other.

On the path towards more sustainable and carbon neutral aviation, increasing efforts are necessary to achieve incremental product improvements for a highly optimized aircraft configuration. Meanwhile the introduction of new technologies, such as liquid hydrogen (LH_2), inevitably entails fundamental changes to the overall aircraft architecture.

Against this backdrop, advanced computational design and analysis methods are indispensable to enable faster, more accurate design cycles on the one hand, and to allow for the early assessment of unconventional aircraft architectures beyond the scope of empirical design methods on the other. In addition, the design is often driven by multidisciplinary integration effects, resulting in a need for collaborative MDAO campaigns involving a multitude of experts and computational design tools. To facilitate communication, central data models such as the Common Parametric Aircraft Configuration Schema (CPACS) are often used for design data exchange.

The aircraft fuselage is of particular interest, since its design is impacted by many diverse disciplines, including structural and cabin design, industrialization, and human factors. In MDAO, all of these aspects need to be accounted for to achieve meaningful results. Yet, the large variety of analysis and design methods and the vastly different requirements in terms of available product details currently pose an obstruction to the integration of detailed fuselage assessment.

To address this issue, the KBE methodology presented in this work provides multi-model generation capabilities to enable consistent analysis model generation at very different levels of fidelity. Built upon CPACS as central data model, the approach combines sets of parameters and executable rules, formulated in a declarative manner to assemble a knowledge graph. The graph-based formulation provides a number of benefits, e.g. the possibility to trace dependencies between parameters or incorporate new knowledge in a modular fashion. Furthermore, the program flow is determined dynamically at run time, based on user requests. Consequently, rules are not executed until they are needed to fulfill a user request, thus avoiding the unnecessary performance of computationally expensive tasks.

Both geometry modeling and design rules are included in the system. A parametric modeling engine for the fuselage based on CPACS is presented, which includes detailed modeling capabilities for the structure and the cabin. The design rules enable the generation of complex layouts based on a small set of design variables. This makes it possible to enrich data sets, e.g. from overall aircraft design, which lack necessary details for the model generation. In combination, the rules enable the flexible generation of geometry models to match the fidelity requirements of different types of analysis models.

Two examples of system modifications to accommodate novel architectures are furthermore given, one for a blended wing-body configuration and another for the integration of large cryogenic tanks in the fuselage.

Several application studies are presented to highlight the versatility of the KBE approach. First, two conventional configurations are considered, one single-aisle and one twin-aisle and twin-deck. Both studies are performed using the basic conventional fuselage design rules. Next, studies for the unconventional blended wing-body and LH_2 tank integration problems are shown, applying the respective system modifications. Finally, the capacity to provide state of the art analysis models is demonstrated by deriving models for global finite element analysis and interactive visualization for human factors evaluation for the single-aisle configuration.

Kurzfassung

In dieser Arbeit wird eine wissensbasierte Entwurfsmethodik (KBE, engl.: knowledge-based engineering) vorgestellt, welche einerseits zur Unterstützung der konsistenten disziplinären Modellgenerierung für multidisziplinäre Entwurfsanalyse- und -optimierungsanwendungen (MDAO, engl.: multidisciplinary design analysis and optimization) geeignet ist und andererseits die Möglichkeit zur Einbindung neuartiger Produktarchitekturen bietet.

Auf dem Weg zu einer nachhaltigeren und CO_2-neutralen Luftfahrt, ist die inkrementelle Verbesserung von hochoptimierten Luftfahrzeugkonfigurationen mit zunehmend hohem Entwicklungsaufwand verbunden. Die Einführung neuer Technologien wie Flüssigwasserstoffantrieben geht zudem mit unausweichlichen fundamentalen Änderungen der Produktarchitektur einher.

Vor diesem Hintergrund ist der Einsatz rechnergestützter Entwurfs- und Analysemethoden unabdingbar, um zum einen schnellere und präzisere Entwurfszyklen zu ermögliche und zum anderen die frühe Bewertung unkonventioneller Flugzeugarchitekturen zu ermöglichen, die nicht durch empirische Entwurfsmethoden abgedeckt werden können. Zudem ist der Entwurf häufig durch multidisziplinäre Integrationseffekte getrieben, wodurch kollaborative MDAO-Kampagnen nötig werden, an denen eine Vielzahl von Experten und Entwurfsprogrammen beteiligt ist. Um die Kommunikation zu vereinfachen, werden häufig zentrale Datenmodelle wie CPACS (Common Parametric Aircraft Configuration Schema) zum Datenaustausch eingesetzt.

Der Flugzeugrumpf ist von besonderem Interesse, da der Entwurf von vielen höchst unterschiedlichen Disziplinen, wie Struktur- und Kabinenentwurf, Industrialisierung oder Human Factors, beeinflusst wird. Um mit Hilfe von MDAO aussagekräftige Ergebnisse erzielen zu können, müssen all diese Aspekte berücksichtigt werden. Die große Breite eingesetzter Entwurfs- und Analysemethoden und die stark unterschiedlichen Anforderungen, welche Produktdetails bekannt sein müssen, stehen einer Integration detaillierter und umfassender Rumpfanalysen derzeit allerdings im Weg.

Um dieser Problematik entgegenzutreten, werden mit der in dieser Arbeit präsentierten KBE-Methodik Multi-Modellgenerierungsfähigkeiten entwickelt, die es ermöglichen konsistente Analysemodelle in unterschiedlichsten Detaillierungsgraden bereitzustellen. Aufbauend auf CPACS als zentralem Datenmodell verbindet der Ansatz Parameter und ausführbare Regeln, die in deklarativer Form zur Verfügung gestellt werden, zu einem Wissensgraphen. Eine graphenbasierte Formulierung bietet zahlreiche Vorteile, z.B. die Möglichkeit Abhängigkeiten zwischen Parametern nachzuverfolgen oder neues Wissen modular einzubinden. Zudem wird der Programmablauf dynamisch zur Laufzeit auf Basis von Nutzerabfragen bestimmt. Regeln werden demnach nicht ausgeführt, bis sie zur Bearbeitung einer Nutzerabfrage benötigt werden, wodurch die unnötige Ausführung rechenintensiver Regeln vermieden wird.

Das System enthält sowohl Geometriemodellierungs- als auch Entwurfsregeln. Es wird eine parametrische Modellierungsengine für CPACS vorgestellt, die detaillierte Regeln für die Geometriegenerierung von Rumpfstruktur und Kabine bereithält. Die Entwurfsregeln ermöglichen die Erstellung komplexer Layouts auf Basis einiger weniger Entwurfsvariablen. Dies erlaubt es, Datensätze, beispielsweise aus dem Flugzeuggesamtentwurf, die nicht alle zur Modellerstellung notwendigen Details beinhalten, mit entsprechenden Daten anzureichern. Im Zusammenspiel erlauben die Regeln somit die flexible Erstellung von Geometrien, die exakt den Genauigkeitsanforderungen unterschiedlicher Analysemodelltypen angepasst sind.

Des Weiteren sind zwei Beispiele für Systemmodifikationen gegeben, welche der Integration neuer Produktarchitekturen dienen sollen: Eine Modifikation ist einer Nurflüglerkonfiguration gewidmet, während die zweite der Integration großer kryogener Tanks im Rumpf dient.

Die Vielseitigkeit des KBE-Ansatzes wird mit Hilfe einiger Anwendungsstudien aufgezeigt. Zunächst werden zwei konventionelle Konfigurationen betrachtet, eine Single-Aisle-Konfiguration und eine Twin-Aisle-Konfiguration mit zwei Decks. Beide Studien werden mit dem konventionellen Basisregelsatz durchgeführt. Anschließend werden Studien für eine unkonventionelle Nurflüglerkonfiguration, sowie eine Wasserstofftankintegration vorgestellt, in denen die jeweiligen Systemmodifikationen zum Einsatz kommen. Zum Abschluss wird die Fähigkeit des Systems demonstriert, Analysemodelle bereitzustellen, die dem Stand der Technik entsprechen. Dazu werden für die Single-Aisle-Konfiguration ein globales Finite-Elemente-Modell und ein Modell zur interaktiven Kabinenvisualisierung für Human-Factors-Analysen erstellt.

Acknowledgments

> No one would say that what they were doing was complicated. It wouldn't even be considered new. Except maybe in the geological sense. They took from their surroundings what was needed, and made of it something more.
>
> *Shane Carruth as Aaron in "Primer"*

This thesis marks the closing chapter of my 8 years and 9 months working as a research scientist at the German Aerospace Center (DLR) between April 2014 and December 2022.

Over the last year and a half – while I was accumulating all the little pieces that make up this book you are holding – the above excerpt from the opening voice-over of my favorite film kept creeping into my mind. Once I got over the thought that someone really ought to explain to Mr. Carruth the difference between geology and topography, I could not stop contemplating how accurately these lines describe the essence of my own work as an engineer in research (even if, unlike the protagonists in the movie, I have not accidentally discovered time travel yet).

On the one hand, the underlying concept of KBE systems, a central element of this thesis, is to literally take the necessary pieces from a body of available data and transform them into something more. On the other hand, as researchers, we, too, depend on the world around us, our peers and colleagues, to provide us with the foundation on top of which we can begin to build our own work.

I was fortunate enough to be able to work with and learn from many fantastic, talented people from a large variety of technical and non-technical backgrounds. This thesis is as much a reflection of their influence on me, as it hopefully is something more for me to give back to the research community. Consequently, there are lots of people to whom I owe a great dept of gratitude for their role in making this thesis what it is. In the following, I would like to take the opportunity to thank at least some of you.

To begin with, I would like to thank Prof. Dr.-Ing. Andreas Bardenhagen, who agreed to supervise this thesis, solely on the basis of a two-page document, with no prior knowledge of me or my work. His profound interest as well as the sympathetic and helpful feedback have been a continuous source of motivation and inspiration for me.

Furthermore, I would like to thank Dr.-Ing. Björn Nagel, who agreed to be the second examiner of this thesis. Björn is also the head of the DLR Institute of System Architectures in Aeronautics (SL), which has been my professional home for over six years. Not only did he give me the opportunity to join his institute in 2016, but also continuously helped me find new ways to grow professionally. As such, when the time came to pin down the topic for this thesis, his input and experience were invaluable.

Many thanks also to Prof. Dr.-Ing. Julien Weiss, who was kind enough to complete the doctoral committee as chair.

There are many more colleagues from my various stations within DLR, whom I need to thank, as well:

From the Cabin and Payload Systems department of SL, this includes Frank Meller, the department head, who invited me into the group and provided me with the necessary freedom to focus on this thesis project. I would also like to thank Prof. Dr. Jörn Biedermann, for his unrelenting support and positivity. I am convinced he will lead the cabin and industrialization research at the DLR on to great things and I am still looking forward to retiring together, even if it might be from different employers. Furthermore, this thesis would not have been what it is without the support from my favorite co-author Dr. Christian Hesse. His enthusiasm for my developments, his vision for new fields of application as well as his uncanny ability to ask all the wrong questions, which made me rethink so many things I had taken for granted, make him the best kind of colleague I could ever have hoped to work with. He also provided invaluable feedback on the manuscript for this thesis. Moreover, I would like to thank Mara Fuchs for all the groundwork she did to build up the department before I joined as well as abiding with my silly questions about Unity and, more importantly, the good times spent together inside and outside the office. In addition, a big thank you to Fabian Reimer, Alex Gindorf and the late Thomas Bock, for the beautiful component models and the inspiration for the passenger-centric cabin layout.

From the MDO group at SL, I would like to thank my former team lead Pier Davide Ciampa, for broadening my stress engineer's horizon and for the high standard to which he holds not only his employees, but also himself. Furthermore, I need to mention my former teammates Mr. Prajwal Shiva Prakasha, Mr. Francesco Torrigiani, and Dr. Luca Boggero, with whom I shared some fun and formative experiences in the early days of SL. Over time, they have grown to be more friends than coworkers to me.

I would also like to thank Daniel Silberhorn, Georgi Atanasov and Sebastian Wöhler from the Aircraft Design group for all the expertise they have put into openAD as well as the countless CPACS files they have shared over the years, some of which are used here. Furthermore, I would like to thank Dr. Marko Alder for his support and the fantastic work he is doing with CPACS.

From my first station at the DLR, the Institute of Structures and Design (BT) in Stuttgart, I would like to thank Dieter Kohlgrüber and Dr. Nathalie Toso, who decided to hire me back in 2014 as well as Stefan Ritt, who acted as the catalyst (in addition to being a coffee/music guru and all around great guy). Echoes of my work with BT reverberate throughout this thesis, and I would like to thank Dieter, who was also my first team lead as well as my colleagues Michael Petsch, Julian Scherer, Dominik Schwinn and Dr. Martin Siemann for the pleasant and productive time together.

None of this would have been possible without the continued love and support from my parents Hannelore and Horst Walther as well as my sister and brother in law Andrea and Markus Wolf, who gave me my very first engineering and programming projects.

Most of all, I want to thank my life partner Florence and our daughter Joséphine. No book could ever hold all the big and little reasons why I am thankful for having them in my life, nor describe the way they make anything and everything worthwhile.

Finally, I would like to thank *you* – the researcher from the future reading this thesis. It is my profound hope that you are going to be able to take from this work something that is needed, and make of it something more.

Hamburg, September 2023 Jan-Niclas Walther

Contents

Acronyms

AD	algorithmic differentiation
AI	artificial intelligence
AiX	Aircraft Exchange
APU	auxiliary power unit
B-rep	boundary representation
BWB	blended wing-body
CAD	computer-aided design
CAGD	computer-aided geometric design
CEASIOM	Computerised Environment for Aircraft Synthesis and Integrated Optimisation Methods
CFD	computational fluid dynamics
CFRP	carbon fiber reinforced polymers
CM	capability module
COTS	commercial off-the-shelf
CPACS	Common Parametric Aircraft Configuration Schema
CS	complex step
CS-25	certification specifications and acceptable means of compliance for large aeroplanes
CSM	constructive solid modeling
CST	class function/shape function transformation
DAG	directed acyclic graph
DEE	Design and Engineering Engine
DFEM	detailed fimite element model
DMU	digital mock-up
DLR	German Aerospace Center
DOC	direct operating cost
DoE	design of experiments
DSM	design structure matrix
EKL	Engineering Knowledge Language
ELWIS	Finite Element Wing Structure
ESP	Engineering Sketch Pad
FEM	finite-element method
FD	finite differences
FFD	free-form deformation
FPG	fundamental problem graph
FSD	fully-stressed design
FST	full-size trolley
FUGA	Fuselage Geometry Assembler
GFEM	global finite-element model
GUI	graphical user interface
HLP	high-level primitive
HTP	horizontal tailplane
IGES	Initial Graphics Exchange Specification
JSON	Java Script Object Notation
KBE	knowledge-based engineering
KBS	knowledge-based system
$\mathbf{LH_2}$	liquid hydrogen
LOPA	layout of passenger accommodation
MALE	medium altitude long endurance
MBSE	model-based systems engineering
MCG	maximal connectivity graph
MDA	multidisciplinary design analysis
MDAO	multidisciplinary design analysis and optimization
MDF	multidisciplinary feasible
MDO	multidisciplinary design optimization
MICADO	Multidisciplinary Integrated Conceptual Aircraft Design and Optimization environment
MINLP	mixed-integer non-linear programming
MILP	mixed-integer linear programming
MMG	multi-model generator
MOKA	Methodology and Tools Oriented to KBE Applications
NASA	National Aeronautics and Space Administration
NLP	nonlinear programming problem
NURBS	non-uniform rational B-splines
OAD	overall aircraft design
OCCT	Open CASCADE Technology
OEM	original equipment manufacturer
OHSC	overhead stowage compartment
OML	outer mold line
OOP	object-oriented programming
openAD	open Aircraft Design
OpenVSP	Open Vehicle Sketchpad
OWL	Web Ontology Language
PADLab	Preliminary Aircraft Design Lab

PANDORA	Parametric Numerical Design and Optimization Routines for Aircraft
PIDO	process integration and design optimization
PLM	product lifecycle management
PrADO	Preliminary Aircraft Design and Optimisation program
PreSTo	Aircraft Preliminary Sizing Tool
PSC	passenger service channel
PSG	problem solution graph
RANS	Reynolds-averaged Navier-Stokes
RAPID	Robust Aircraft Parametric Interactive Design
RCE	Remote Component Environment
RPB	rear pressure bulkhead
SAF	sustainable aviation fuel
SBW	strut-braced wing
SPH	smoothed-particle hydrodynamics
SQP	sequential quadratic programming
STEP	Standard for the Exchange of Product model data
SUAVE	Stanford University Aerospace Vehicle Environment
SWRL	Semantic Web Rule Language
SysML	System Modeling Language
TiGL	TiGL Geometry Library
TLAR	top-level aircraft requirements
TRAFUMO	Transport Aircraft Fuselage Model
UAV	unmanned aerial vehicle
UDF	user-defined feature
UF	utilization factor
UI	user interface
uID	unique identifier
UML	Unified Modeling Language
URI	uniform resource identifier
VBA	Visual Basic for Applications
VR	virtual reality
VTK	Visualization Toolkit
VTP	vertical tailplane
XML	eXtensible Markup Language
XDSM	eXtended Design Structure Matrix

Symbols

Latin letters

σ	stress
σ_{allow}	stress allowable
$a_{ellipse}$	ellipse major semi axis
A_i	CST tuning coefficient
$\mathbf{A}$	rotation matrix
a_{panel}	skin panel width
A	wing aspect ratio
B	Bezier polynomial basis
$b_{ellipse}$	ellipse minor semi axis
$\mathbf{B}$	strain-displacement vector
b_{panel}	skin panel height
$\mathbf{c}_{cap,i}$	cap sphere center point
$\mathbf{C}$	section curve
$\mathbf{C}_{ref}$	reference section curve
c_d	pressure wave transmission speed
$c_{ellipse}$	ellipse linear eccentricity
C	class function (CST)
$\mathbf{C}$	material matrix
$c_{PAX,exit}$	passenger capacity for exit type
$\mathbf{c}$	control point
c	specific fuel consumption
$\mathbf{d}$	displacement vector
D	drag
$\mathbf{d}'$	displacement vector in element coordinates
d_{far}	far plane distance
d_Γ	displacements at the domain interface
d_{mesh}	meshing linear deflection
d_{near}	near plane distance
$d_{seat-wall}$	distance from seats to fuselage wall
d_x	translation in x-direction
d_y	translation in y-direction
Δl_{cabin}	difference between cabin space length and cabin length
$\Delta l_{fuselage}$	change in fuselage overall length
Δl_{tank}	overall tank space length
Δp	pressure difference
$\Delta s_{st,nom}$	nominal stringer pitch at reference section
Δt_{stable}	stable time step
Δx	local longitudinal section displacement
$\Delta x_{cockpit}$	cockpit length
Δx_{exit}	distance between exits
$\Delta x_{exit,CA}$	allowable cross aisle distance to exit
$\Delta x_{fr,nom}$	nominal frame pitch
Δx_{seat}	seat pitch
$\Delta x_{seat,biz}$	business class seat pitch
$\Delta x_{seat,eco}$	economy class seat pitch
$\Delta x_{seat,eco+}$	economy-plus seat pitch
$\Delta x_{tank,i}$	length of tank i
Δz_{APU}	tail vertical eccentricity (at APU center)
Δz_{CB}	crossbeam vertical offset
Δz_{LB}	longitudinal beam vertical offset
Δz_{nose}	nose vertical eccentricity
E	graph edges
E	Young's modulus
E_M	maximal connectivity graph edges
$\mathbf{f}$	load vector
f_Γ	loads at the domain interface
$f_{pitch,glob}$	global seat pitch factor
G	graph
M	maximal connectivity graph
$h_{cab,upper}$	cabin height (upper deck)
h_{cabin}	overall cabin height
$h_{constant}$	constant section height
h_{door}	door height
h_{FE}	floor element height
H	homogenizing transformation
$\mathbf{H}$	transformation matrix in homogeneous coordinates
h_{tail}	tail section height
h_{window}	window height
i	polynomial index
I_{dir}	intensity of incoming light
i_{exit}	exit index
i_{lat}	lateral index
i_{long}	longitudinal index
I	intensity of reflected light
i_{width}	width position index
k_C	average seat pitch (empirical)

$\mathbf{K}$	stiffness matrix
l	number of control points
l_{cabin}	cabin length
$l_{cap,i}$	cap depth
$l_{cockpit}$	cockpit section length
$l_{constant}$	constant section length
L_e	allowable element edge length
l_{FE}	floor element length
$l_{fuselage}$	fuselage overall length
L	lift
l_{pos}	positioning distance
l_{tail}	tail section length
$\mathbf{m}$	numbers of sample points for dimension
$\mathbf{M}_{pp}$	transformation matrix from perspective to parallel projection
n	polynomial degree
N_1	class manipulation variable (CST)
N_2	class manipulation variable (CST)
n_{FST}	number of full-size trolleys
n_{aisle}	number of aisles
n_{assist}	number of assist spaces for exit type
N	B-spline basis
n_{bulk}	number of bulkheads
$n_{container}$	number of cargo containers
n_{dim}	number of spatial dimensions
$n_{exittype}$	number of exits of given type
$n_{exittypes}$	number of possible exit types
$n_{fr,span}$	number of frame bays spanned by floor panels
$n_{iter,opt}$	number of optimization iterations
$n_{iter,root}$	number of root finding iterations
n_{lav}	number of lavatories
n_{PAX}	number of passengers
$n_{PAX,tgt,i}$	approximate cumulative number of passengers at exit i
$n_{reinf,h}$	number of horizontal reinforcements (flat)
$n_{reinf,r}$	number of radial reinforcements (spherical)
$n_{reinf,v}$	number of vertical reinforcements (flat)
n_{rows}	seat block number of rows
n_{SA}	number of seats abreast
n_{seat}	number of seats
$n_{sections,i}$	number of frame bays at the i-th main frame
$n_{stringer}$	number of stringers around circumference
n_{tanks}	number of liquid hydrogen tanks
n_{typeA}	number of exits of type A
n_{typeB}	number of exits of type B
$\mathbf{p}_{con}$	stringer confluence point
$\mathbf{p}_{i,tria}$	triangle corner point
$\mathbf{p}_{prof,CB}$	crossbeam profile points
$\mathbf{p}_{prof,LB}$	longitudinal beam profile points
$\mathbf{p}_{sphere}$	frame curve centroid at bulkhead
$\mathbf{p}_{st}$	stringer curve point
$\mathbf{p}_{st,ref}$	stringer definition reference point
$\mathbf{p}_{target,i}$	deformation target point for the i-th attachment point
$\mathbf{p}_{0,PB}$	calotte sphere center
$\mathbf{p}_{0,bbox}$	bounding box origin
$\mathbf{p}_{0,ray}$	ray origin
$\mathbf{q}_i$	attachment point locations
$\mathbf{Q}$	array of untransformed vectors in homogeneous coordinates
$\mathbf{q}$	untransformed vector in homogeneous coordinates
$r_{cap,i}$	cap sphere radius
r_{corner}	door corner radius
r_{fillet}	cap fillet radius
r_{frame}	bulkhead radius at frame
R	range
r_{sphere}	bulkhead sphere radius
r_{window}	window corner radius
$R_{p,0.2}$	yield strength
s	curve segment length
s_{12}	curve segment length between first and second coordinate
S	shape function (CST)
S_{glob}	global scaling factor
$\mathbf{S}$	scaling matrix
s_{ref}	reference section curve length
S_x	model length scale factor
S_y	model width scale factor
S_z	model height scale factor
t	curve internal parameter
$\mathbf{t}$	curve internal knot sequence
t_{dome}	calotte depth
t_i	request i runtime
t_{panel}	floor panel thickness
$t_{sheet,bulk}$	bulkhead sheet thickness
t_{skin}	skin panel thickness
$\mathbf{t}_{skin}$	vector of circumferential skin thicknesses
$\mathbf{T}$	trajectory curve (in CARTESIAN space)
$\mathbf{T}_{\mathbf{uv}}$	trajectory p-curve (in surface uv-space)

t_{wall}	tank wall thickness including insulation
u	shape running variable (first topological dimension)
$\mathbf{u}_i$	local box coordinates of i-th attachment point
u_{surf}	p-curve coordinate value
$\mathbf{u}$	camera sideways direction
UF_{cargo}	cargo volume utilization factor
UF_{galley}	lavatory utilization factor
UF_i	utilization factor of panel i
UF_{lav}	lavatory utilization factor
v	shape running variable (second topological dimension)
v_{AC}	aricraft velocity
V_{cargo}	cargo volume
$\mathbf{v}_{light}$	incoming light vector
$\mathbf{v}_{look}$	camera view direction
$\mathbf{v}_{ray}$	ray direction
v_{surf}	p-curve coordinate value
V_{tanks}	overall liquid hydrogen tank volume
$\mathbf{v}_{u,bbox}$	box edge vector in u-direction
$\mathbf{v}_{up}$	camera up-vector
$\mathbf{v}_{v,bbox}$	box edge vector in v-direction
$\mathbf{v}$	camera upward direction unit vector
V	graph vertices
$\mathbf{v}_{w,bbox}$	box edge vector in w-direction
V_M	maximal connectivity graph vertices
w_{FST}	full-size trolley width
w_{aisle}	aisle width
w_{assist}	width of assist space for exit type
w_{cabin}	cabin width
$w_{constant}$	constant section width
w_{door}	door width
w_{exits}	overall exit length penalty
$w_{exittype}$	exit type length penalty
w_{FE}	floor element width
W_f	fuel mass
$w_{passageway}$	width of passageway for exit type
w_{seat}	seat width
W_i	structural mass
w_{tail}	tail section width
$\mathbf{w}$	camera backward direction unit vector
w_{window}	window width
w_{xaisle}	cross-aisle width
w	NURBS weight factor

x	generic CARTESIAN dimension/aircraft longitudinal axis
$x_{bulk,i}$	number of bulkheads
$\mathbf{x}_c$	camera location
$x_{f,ij}$	longitudinal position of the j-th frame following the i-th main frame
$\mathbf{x}_{fuse}$	point on fuselage surface
$\mathbf{x}_{mfr}$	main frame position vector
$x_{mfr,i}$	longitudinal position of the i-th main frame
$x_{min,door}$	longitudinal coordinate of door forward bound
$\mathbf{x}$	point on curve/surface
$\mathbf{x}_{ray}$	point on ray
$x_{ref,door}$	longitudinal coordinate of door reference point
$\mathbf{x}_{swept}$	point on swept surface
$x_{tanks,max}$	tank space aft bound
$x_{tanks,min}$	tank space forward bound
$\mathbf{x}_{tria}$	point on triangle
y	generic CARTESIAN dimension/aircraft lateral axis
y_{long}	longitudinal beam lateral position
y_{strut}	crossbeam strut lateral position
z	generic CARTESIAN dimension/aircraft vertical axis
z_{cab}	vertical coordinate above cabin floor
$z_{cab,window}$	window distance to cabin floor
z_{cb}	crossbeam vertical position
z_{floor}	floor vertical position
$z_{floor,main}$	floor vertical position (main deck)
$z_{floor,upper}$	floor vertical position (upper deck)
z_{long}	longitudinal beam z-position
$z_{min,door}$	vertical coordinate of door lower bound
$z_{ref,door}$	vertical coordinate of door reference point

Greek letters

α_{mesh}	meshing angular deflection
γ	rotation angle
Γ	dihedral angle
ε	strains
ζ	relative profile thickness
ζ_T	trailing edge thickness
η	wing midplane spanwise coordinate
θ_h	horizontal field of view angle

θ_v	vertical field of view angle
θ	incidence angle between light and surface
λ_i	Lagrange multiplier
Λ	sweep angle
ξ	wing midplane chordwise coordinate
ρ	density
φ	stringer plane orientation angle
φ_{strut}	crossbeam strut angle
ψ	relative chordwise position
ω	relaxation parameter

1. Introduction

Decarbonization and digital transformation are two of the most preeminent topics in the aviation community today. To meet the goals of the Paris Agreement on climate change [UNF15], which calls for a limitation of the global temperature increase to $1.5°C$ compared to pre-industrial levels, ambitious emission goals have been formulated for the aviation sector e.g. in the road map to net zero carbon dioxide emissions until 2050 by the Air Transport Action Group [ATA21]. Similar goals have been expressed by different organizations on German national and European [CF20; DLR21a] level in reference to the European Green Deal [EC19], which targets being the first continent to achieve net zero emissions by 2050.

These goals for aviation can be achieved only by combining gains through sustainable aviation fuels (SAFs), operations, market-based measures and technology improvements. Whereas improvements in the first three fields can be implemented almost immediately and require virtually no changes to the existing fleet, improvements in technology demand more lead time for product development. Nevertheless, the potential emission savings due to new technologies may amount to up to 34% of the total projected carbon dioxide emissions by 2050 [ATA21]. This forecast is based on the assumption that new, unconventional propulsion technologies such as electric or hydrogen will be brought to the market by 2040. Introducing these novel technologies has the advantage that, differently from SAFs, direct emissions of CO_2 and other greenhouse gases such as NO_x are reduced. On aircraft scale, the technological changes introduce fundamentally new requirements, such as the need for cryogenic fuel storage, resulting in new and unfamiliar architectural solutions.

Towards this goal, digital transformation holds the promise of integrating all available competences in an organization to form a digital continuity, also referred to as the "digital thread", to reduce cost and time to market. Programs to foster digital transformation are found both at major original equipment manufacturers (OEMs) [Air23] and in research organizations, such as the German Aerospace Center (DLR) [DLR21a].

Due to the novelty of the product architecture, virtual testing using numerical methods is indispensable to assure the safety and performance of new developments and thus manage the entrepreneurial risks for the manufacturer. While the aerospace industry has been among the very early adopters of computational methods for product development, driving the development of state-of-the-art technology, such as computer-aided design (CAD) and the finite-element method (FEM), the forthcoming challenges require digital transformation beyond simply using computers for isolated analyses. Instead, the multidisciplinary nature of the aircraft development process, where the influence of integration effects between multiple disciplines can often eliminate the impact of improvements in individual disciplines, must be reflected in interdisciplinary and collaborative digital processes.

The field of multidisciplinary design analysis and optimization (MDAO) aims to couple different types of numerical analyses to better assess these effects. Often, automated collaborative processes are deployed, where information is exchanged among experts using a common central data model, such as the Common Parametric Aircraft Configuration Schema (CPACS). However, whereas the coupling of a few disciplines, predominantly aerodynamic and structural analysis, is well-represented in the literature, the ambition of a digital end-to-end continuity associated with digital transformation is yet to be realized. On the one hand, a link to take into account stakeholder requirements should be established leveraging model-based systems engineering (MBSE) techniques in order to correctly formulate the design problem. On the other hand, more types of disciplinary analysis must be made available to the MDAO process for a more complete evaluation of the design, which is often driven by aspects outside the classical aero-structural domain.

With respect to the latter aspect, the aircraft fuselage is a particularly interesting example. Its design is driven not only by performance considerations, such as a lightweight structure to reduce emissions, but also by passenger needs and safety requirements, which drive the design of the cabin in particular. At the same time, OEMs need to be able to manufacture and sell the aircraft at a profit, which means requirements pertaining to manufacturing and industrialization are increasingly important. Most hydrogen aircraft concepts found in the literature furthermore assume a tank, which is installed in the fuselage.

The diverse requirements of the analysis models w.r.t. the level of fidelity of the underlying geometric representation of the product pose a particular challenge. They can not usually be fulfilled using the established approach of a central CAD-based digital mock-up (DMU), requiring either manual addition or removal of geometric features.

As an alternative to a central DMU, multi-model generators (MMGs) based on a central parametric description, such as CPACS, and a knowledge-based engineering (KBE) framework, where process knowledge is formalized as a set of rules, have been proposed. However, few works have examined the specific requirements of the fuselage in a multidisciplinary setting. Furthermore, the knowledge base in the tools is typically difficult to adapt to support fundamentally different architectural solutions. This type of capability is, however, critical in order to enable effective assessment of the novel types of aircraft needed to meet the decarbonization goals.

To examine these issues, this thesis is structured as follows:

Chapter 2 The state of the art for digital systems for aircraft design is discussed, covering the range from handbook methods to sophisticated MDAO systems. Special attention is given to the field of fuselage and cabin design. A closer look is taken at the steps necessary for integrating high-fidelity fuselage analysis into predominantly aeroelasticity-driven MDAO processes and possible roadblocks are identified.

Chapter 3 A research hypothesis is derived based on the findings of the previous chapter. It is then broken down into four working hypotheses, which provide guidance for the subsequent discussion.

Chapter 4 Enabling technologies for successful deployment of large-scale fuselage and cabin MDAO are discussed, such as standardized geometry- or data-centric common models of aircraft and automated design and analysis capabilities for multiple disciplines and at multiple levels of fidelity. Finally, knowledge-based engineering techniques are introduced as a means of augmenting design details and handling diverging geometry requirements from different disciplines.

Chapter 5 A KBE methodology for providing suitable geometry models of the aircraft fuselage and cabin, which are tailored to the specific level of fidelity required for analysis model generation, is proposed based on the data-centric CPACS aircraft data schema. The underlying knowledge-based system (KBS) implementation is explored, which accepts data provided via CPACS and links it through various sets of rules using graph-based reasoning. In this way, a graph-based parameter engine for CPACS is implemented. Furthermore, a modular approach to rule set definition is proposed, which enables the quick implementation of additional rule sets to adapt the KBS to new aircraft system architectures.

Chapter 6 In this chapter, the design rules applied to generate aircraft fuselages and cabins are presented. The structure of the chapter mirrors that of the corresponding rule sets, i.e. the outer shape of the fuselage, the structure and the cabin. A dedicated section to discuss the management of geometric fidelity levels is also provided. Finally, new rule sets - one for the integration of a simplified parametric model of an liquid hydrogen (LH_2) storage tank and one for the determination of the cabin space of a blended wing-body configuration - are introduced as examples for architecture modification.

Chapter 7 The applicability of the approach to actual design studies is investigated in this chapter using a total of six example cases. First, the capabilities for augmenting design details are shown in a cabin design synthesis of a single-aisle narrow-body configuration for an empty outer mold line. To illustrate the support for large parameter spaces, the study is repeated for a twin-aisle wide-body configuration and a multi-deck layout.

Next, the capabilities for adapting the system to novel architectures are shown by performing a design of a fuselage with an integrated LH_2 storage tank and a blended wing-body (BWB) cabin layout, leveraging the corresponding new rule sets.

Finally, the capability to derive consistent analysis models at different levels of fidelity is demonstrated. On the one hand, a geometry model is generated based on the requirements for fuselage structure sizing using a global finite-element model (GFEM), to demonstrate support for lower levels of fidelity. On the other hand, a detailed model for immersive visualization is generated, which covers the high-fidelity end of the spectrum of analysis models.

Chapter 8 The key findings of this thesis are collected and summarized. Based on those insights, suggestions for further research trajectories are provided.

2. Background: Digital systems for aircraft and fuselage design

Over the years, the way in which aircraft are being designed and developed has evolved, embracing technological advances in other fields and adjusting to market needs. While advances e.g. in materials science and engine concepts have enabled improvements at product level, the continuous increase in computing power has furthermore had a significant impact on the development process. Computer-aided design (CAD) software for virtual product modeling and numerical analysis tools are commonplace in industrial environments today.

Comprehensive and continuous digitization holds the promise of a highly integrated and flexible enterprise process, commonly referred to as a digital thread [SP+17; SW18], which can take full advantage of these technologies. However, current development processes, which are discussed in detail in section 2.1, are still rather strictly separated into multiple design phases, which are traversed sequentially, and disciplinary silos, which inhibit exchange of information.

Furthermore, incremental, evolutionary developments are currently the norm. That said, many authors including Mavris, Tenorio, and Armstrong [MTA08] and Ciampa and Nagel [CN21] stress the need to evaluate a large variety of aircraft system architectures early in the design process to properly assess the true potential of emerging technologies.

By stepping from evolutionary to revolutionary developments – a consequence of considering novel aircraft system architectures – many established evolutionary aircraft design procedures are invalidated. In the past, empirical methods built upon knowledge of previous programs have served to understand the complex underlying interdisciplinary relationships in aircraft design. Yet, the further removed a given architecture is from the empirical knowledge base, the more it becomes necessary to turn towards numerical analyses based on physics and detailed knowledge of the product, to develop an understanding of the aircraft configuration. The challenge of connecting the various design disciplines in a meaningful and efficient way has led to the field of multidisciplinary design analysis and optimization (MDAO) [Sob95], which is discussed in section 2.2. Distributed, collaborative processes, where dedicated analysis and design capabilities are contributed by disciplinary experts, have become a viable solution to manage the complexity of modern MDAO processes [Kro04; MD+16; CN20].

2.1. Fuselage and cabin in the aircraft design process

Aircraft are highly complex products. Accordingly, aircraft design is a complex process. Over time, a number of texts have appeared, e.g. by Torenbeek [Tor76], Raymer [Ray89], and Roskam [Ros97], proposing a structured methodological design approach in order to manage this complexity. Despite their age and reliance on empirical data, many of them are still considered standard literature in aircraft design. All methodologies mentioned share the common trait that the aircraft design process is broken down into three phases.

Conceptual design phase During this phase, fundamental decisions about the aircraft architecture are made, based on mission and efficiency requirements. According to La Rocca [LaR11], approximately 1% of the engineering staff are involved, over a duration of weeks to months. This phase is concerned with covering the breadth of the design space to find the most promising architecture for a given transportation task, which is formulated in the top-level aircraft requirements (TLAR). Aside from the cruise mach number and constraints related to airport operations, such as take-off field length or wing span limit, the number of passengers and the design range are elementary components of the TLAR for civil passenger aircraft.

The range R for steady flight at climb cruise conditions, i.e. at constant angle of attack α, velocity v_{AC} and specific fuel consumption c, after BRÉGUET (s. e.g. [Tor76; Cav06]) given by

$$R = \frac{v_{AC}}{c} \frac{L}{D} \ln \left(\frac{W_i}{W_i - W_f} \right) \tag{2.1.1}$$

provides a good example for a very simple mission performance evaluation. It illustrates how the range can be influenced by aerodynamic efficiency $\frac{L}{D}$, the structural mass W_i and fuel mass W_f, aside from the aforementioned velocity. Even though the design missions of real-life aircraft programs are usually more complex, it remains a fundamental task during the conceptual design to determine estimates of these product properties. The estimates should be sufficiently accurate to perform meaningful trade studies for different architectures and to properly assess the benefit of new technologies.

Preliminary design phase In this phase, the configuration is frozen, and more detailed analysis is performed. To this end, key product details, such as the lofting must be specified. The goal is to provide an accurate cost estimate before proceeding to the next phase. Around 9% of the engineering staff are involved in this phase, which lasts in the order of months to years [LaR11].

Detail design phase All the details and components necessary to assemble the physical aircraft must be designed, including brackets, riveting, wiring etc. Furthermore, the manufacturing process and necessary tooling are traditionally mainly developed in this phase. It also includes large structural tests required e.g. for certification. La Rocca [LaR11] states that 90% of the engineering staff will be working on this phase for several years.

The design phases are bookended by a market survey or customer request on the one hand, which results in the postulation of the TLAR for a new design, and the manufacturing and operation phase of the finished product on the other. More recently, the decommissioning of the product has become an additional important aspect to consider, which enables the assessment of the complete product life-cycle [JS14; EWP18].

In figure 2.1.1, the development of available product knowledge and committed cost over the design phases is shown. It highlights the importance of the early design phases. Even though the duration and cost of these phases is relatively small compared to detail design, the decisions taken in the early phases based on very little knowledge largely determine the design and hence both the cost and ultimately the success of the entire aircraft program. As stated e.g. by Mavris and DeLaurentis [MD00] and Ciampa and Nagel [CN17], the earlier detailed product knowledge is made available in the design process, the more design freedom is given to the engineers, since adverse effects and potential risks can be identified early. In return, the true committed cost is very likely lower, since fundamental errors requiring expensive redesigns are less probable.

According to e.g. La Rocca [LaR11] and Air Transport Action Group [ATA21], conventional configurations have currently reached a phase of saturation, where evolutionary improvements, while requiring substantial development efforts, have increasingly minor effects. Instead revolutionary changes are required, to address growing environmental concerns in society and achieve the ambitious international decarbonization goals outlined in chapter 1. Emission penalization schemes as well as the volatile energy market, furthermore provide economic incentives. Strengthening the role of preliminary design by increasing the breadth and depth of the product knowledge made available during this phase becomes imperative, not only to provide a technically sound product, but also to manage the entrepreneurial risk for original equipment manufacturers (OEMs) due to uncertainties associated with revolutionary changes.

In the following, first a look is taken at existing aircraft design frameworks in section 2.1.1, which are then inspected in more detail w.r.t. their integration of details of the cabin and fuselage in section 2.1.2.

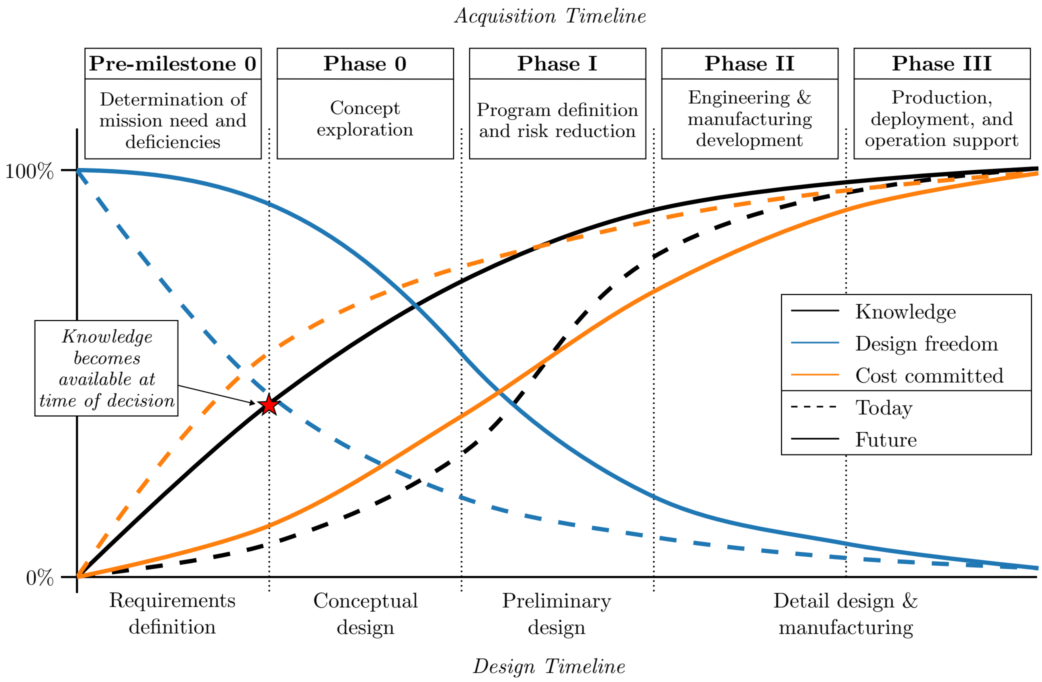


Figure 2.1.1.: Current and future relationship between available knowledge, design freedom and committed cost over the design process phases (after [MD00])

Unconventional aircraft system architectures and the corresponding new challenges for conventional aircraft design frameworks are explored in section 2.1.3. High-level approaches for digital processes capable of handling these challenges are discussed in section 2.1.4.

2.1.1. Review of computational aircraft design frameworks

The formalized methodology provided in literature on the one hand, and the need for exploration of large design spaces and thus for fast computation on the other, have resulted in the development of a large number of computational aircraft design frameworks. Table 2.1.1 provides a non-exhaustive list of frameworks developed over several decades, which are explained in the following.

Early codes, such as FLOPS (Flight Optimization System) and PASS (Program for Aircraft Synthesis) were developed in the United States by the National Aeronautics and Space Administration (NASA). PrADO (Preliminary Aircraft Design and Optimization program) is among the earliest tools published in Europe and pays heed to the multidisciplinary nature of the design process by splitting it into separate design functions. The functions are then passed through sequentially. Consistency is asserted by comparing the design results to the initial assumptions in an iterative loop. Later implementations, such as PADLab (Preliminary Aircraft Design Lab), openAD (open Aircraft Design), MICADO (Multidisciplinary Integrated Conceptual Aircraft Design and Optimization environment) or PreSTo (Aircraft Preliminary Sizing Tool) follow a similar philosophy.

The design functions are based on regressions on empirical data of existing aircraft. Whereas this method is mature for conventional configurations and leads to very good results, its use is limited for configurations, which are too far removed from the knowledge base. This led to efforts to add physics-based analysis capabilities to the existing tools e.g. by Österheld [Öst03], who added numerical structural analysis to PrADO. This approach is taken further in the CEASIOM (Computerized Environment for Aircraft Synthesis and Integrated Optimization Methods) design system, where several dedicated disciplinary analysis tools are orchestrated via a central graphical user interface (GUI) for overall aircraft design (OAD). A comparable solution is SUAVE (Stanford University Aerospace Vehicle Environment), which is, however controlled via a Python script instead of a GUI. Initiator is the conceptual design part of the Design and Engineering Engine (DEE) developed by TU Delft,

Table 2.1.1.: List of aircraft design frameworks

Name	Institution	Publication
AAA	DARCORP	[Ros97; Ros18]
CAPS/pyCAPS	MIT	[AD+16; DR19]
CEASIOM	KTH Stockholm	[BRI08; RZ+12]
FAST-OAD	ONERA	[Sch18; DD+21]
FLOPS	NASA	[McC84]
Initiator/DEE	TU Delft	[LLB12; EVL14]
MICADO/UNICADO	RWTH Aachen	[RA+12; Ris16; ASS21; SA+21]
openAD (formerly VAMPzero)	DLR	[Böh15; WA+20; MZM21]
OpenVSP	NASA	[GDG96; Hah10; MG22]
Pacelab ACE	PACE	[SE07]
PADLab	TU Berlin	[Gob15; TUB17]
PASS	NASA	[Joh77]
PrADO	TU Braunschweig	[Poh86; Hei94]
RAPID	Linköping University	[SM+12; MS+15; Mun17]
RDSwin	NASA	[Ray16]
SAS/PreSTo/OPerA	HAW Hamburg	[SS10; ASS11; Hei12; Nit13]
SUAVE	Stanford University	[LW+15; BW+16; MC+17]
TASOPT	MIT	[Dre10]
VisualCAPDA	TU Berlin	[SH+96]

which also provides a multi-model generator (MMG) to automatically derive consistent disciplinary analysis models. The models are analyzed to evaluate the flight performance. The overall process can be controlled by an optimization.

Tools like CAPS (Computational Aircraft Prototype Syntheses) and OpenVSP (Open Vehicle Sketchpad) deserve a special mention, due to their strong focus on the geometry of the outer mold line (OML). This makes them significantly more versatile than the aforementioned tools, as evidenced by several studies on general aviation configurations [AD+16; MG22]. Notably, SUAVE also provides an interface to OpenVSP. That said, the amount of detail, when it comes to the structure or payload is limited in these tools, even though CAPS offers some support for structural model generation [JG+19].

Expanding upon the approach used in PrADO and CEASIOM, some recent conceptual design tools like openAD have been designed specifically to provide a preliminary design synthesis, which serves as an input to subsequent analyses at higher levels of fidelity performed using separate tools resulting in modular, possibly distributed design systems. To communicate the design data, it is stored in a common data exchange model (e.g. CPACS, s. section 4.1.2). In this way, the preliminary design process development can be decoupled from the physics-based analysis tools, to a certain extent. The merits and limitations of this approach are discussed further in section 2.2.4.

Recent versions of Initiator or MICADO also offer support for common data exchange models by providing interfaces to export the designs, which are generated from the TLAR, to CPACS data sets. This enables the tools to fulfill a role comparable to openAD. For CEASIOM, on the other hand, CPACS has even been adopted for internal data storage and transfer between individual design components. UNICADO (UNIversity Conceptual Aircraft Design and Optimization) is another notable project to adopt CPACS for data exchange. The aim of UNICADO is to cluster the conceptual aircraft design capabilities of several German universities based on lessons learned during the development of tools like PrADO, PADLab or MICADO [SS21].

2.1.2. Integration of fuselage and cabin in preliminary design environments

As stated previously, the number of passengers is among the key information given by the TLAR. In a conventional passenger aircraft configuration, the primary purpose of the fuselage is to provide the space for the payload, i.e. the passengers, while producing as little drag and using as little structural mass as possible. This makes it a crucial component of the aircraft even in conceptual design. Therefore, the aircraft design frameworks presented in table 2.1.1 are examined closer w.r.t. the level of detail of the fuselage and cabin modeling in this section.

Fidelity level and parametric description of the fuselage geometry The drag contribution of the fuselage is dictated by the OML. As such, a parametric description of the OML is an integral part of any aircraft design tool. Rough drag assessment is possible using very basic parametric descriptions. For instance, Roskam [Ros02] proposes a very reduced description, where the fuselage is considered as a cylinder with a diameter d_f and a length l_f. The parameters can be used to compute a fuselage fineness or slenderness ratio $\lambda_f = \frac{l_f}{d_f}$, which can be related to an estimate of the fuselage drag coefficient c_{D_B}.

For more detailed analysis, Roskam proposes splitting the fuselage into a spherical cockpit section (also referred to as forward section), a cylindrical mid-section and a conic aft section. This subdivision is the basis to many preliminary fuselage design tools for conventional aircraft [Met08; God08; Şen10; NS10] and is also introduced in texts on structural design e.g. by Niu [Niu88]. Differences arise in the parametrization of the various sections, most importantly the cross-section of the middle section. Aside from a circular cross-section, elliptic or double bubble shapes are often implemented. A rather complete collection of the cross-section types that occur in aviation is provided by OpenVSP [MG22]. In addition to the aforementioned section types point, super-ellipse, rounded rectangle, CST (Class-function/shape-function transformation, s. section 4.1.1.1) and general fuselage (free form) cross-sections are available.

The importance of the cross-section to the overall design is documented e.g. by Metzger [Met08] and Fuchte [Fuc14], who show the effects of the choice of cross-section in conjunction with the cabin seating layout on the direct operating cost of the aircraft. Other authors, such as Şen [Şen10] investigate the effect of the cross-section on the structural loads and therefore mass.

PreSTo [ASS11] and PADLab [TUB17] can provide detailed fuselage shapes, which also include the cockpit and tail sections. However, these sections are not fully parametric, instead relying on predefined geometry, which is scaled to fit the design.

A different approach is chosen in CPACS, which is the basis for tools like openAD and CEASIOM. Here, the fuselage is described using a sequence of sections distributed along the length of the fuselage, which gives additional freedom in the description of the non-cylindrical sections. Sections can be defined either by a point list, or using a rectangle, superellipse or CST parametrization to separately describe the upper and lower half. Circles and ellipses are also supported, being special cases of superellipses. In the simplest case, the fuselage surface is built by linear interpolation between the sections. This has the disadvantage that the resulting fuselage surfaces are not smooth, but contain discontinuities at the profile curves. Therefore, optional guide curves can be specified to control the transition between adjacent segments at a section.

A comparable approach is implemented in ParaFuse by Jonge [Jon17], which provides a detailed parametrization of the cockpit and tail sections using a combination of cross-sectional profile curves and guide curves at the top, the bottom and side of the fuselage as shown in figure 2.1.2. The implementation is based on the commercial KBE software ParaPy [DB23].

A very similar method based on the commercial CAD program CATIA (s. section 4.1.1.4) is implemented in RAPID (Robust Aircraft Parametric Interactive Design) as described by Staack et al. [SM+12]. The OML is described using three guide curves, which span the entire fuselage length. For the interpolation between the cubic Bézier curves (s. section 4.1.1) are applied, which can be modified freely to model a large variety of fuselage shapes.

In CAPS [AD+16], a more general solution based on CAD feature trees (s. section 4.1.1.3) is chosen. It is built using the Engineering Sketch Pad (ESP) [HD13], a custom CAD environment based on a

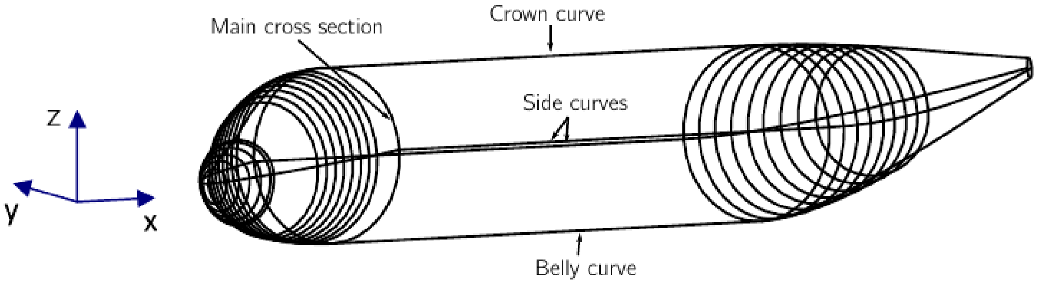


Figure 2.1.2.: Fuselage definition in ParaFuse (from [Jon17])

modified version of the Open CASCADE Technology (OCCT) kernel (s. section 4.1.1.4). Nonetheless, similar fuselage modeling techniques are ultimately applied, as well. The resulting model is sufficiently detailed to derive surface meshes for computational fluid dynamics (CFD) analyses, which can provide a superior drag assessment both in accuracy and flexibility than the aforementioned approach by Roskam.

Similar functionality is available in OpenVSP. Unlike the previous examples, OpenVSP is not reliant on a dedicated CAD kernel. Instead, a custom geometry kernel is provided. Despite this, a clear trend towards integrating CAD capabilities to enable modeling of smooth fuselage surfaces is discernible. A custom geometry kernel may be a viable solution for lean, highly specialized tools. However, the availability of the capabilities a fully realized CAD kernel is an enabling factor when trying to add further details to the design. Nonetheless, it remains important to identify key parameters, as proposed by Roskam to understand general relationships.

Integration of cabin design details Information about the cabin is of essence when designing the OML of the fuselage. Most importantly, the dimension of the cross-section is governed by the number of seats abreast n_{SA} and the fuselage length depends on the cabin length l_{cabin}. Commonly, the TLAR prescribe a certain number of passengers n_{PAX}, which need to be accommodated by the fuselage. An empirical relationship between the number of seats abreast and the number of passengers is given by Raymer [Ray89]:

$$n_{SA} = 0.45 \cdot \sqrt{n_{PAX}}. \tag{2.1.2}$$

The cabin length can then be estimated by

$$l_{cabin} = \frac{n_{PAX}}{n_{SA}} \cdot k_C, \tag{2.1.3}$$

where k_C is an empirically determined average seat pitch, which takes into account the real seat pitch for all classes, cross aisles due to exits and monuments, e.g. galleys and lavatories. Many of these parameters are in turn functions of the desired level of comfort in the cabin. Similarly, the width of the cabin can be computed using

$$w_{cabin} = w_{aisle} \cdot n_{aisle} + w_{seat} \cdot n_{seat} + 2 \cdot d_{seat-wall}. \tag{2.1.4}$$

Once again, the computation of the width requires additional details, such as the seat width w_{seat}, the number n_{aisle} and width w_{aisle} of the aisles, and the distance $d_{seat-wall}$ to the fuselage wall. Moreover, many of the parameters involved are subject to constraints by the certification specification [EAS21], which prescribes e.g. minimum aisle widths (CS 25.815) and the maximum number of seats between a seat and the nearest aisle (CS 25.817).

The underlying approach outlined by the above equations is referred to as *inside-out* design. This means the cabin dimensions are computed first and the fuselage dimensions are derived based on the results. The approach can be applied for clean-sheet designs. *Outside-in* design is the counterpart to *inside-out* design. It implies that the fuselage dimensions are already known and a fitting cabin must be designed subsequently. Aircraft cabins can be exchanged multiple times across the life-cycle of an aircraft, which provides the opportunity to install improved, customized cabins. The design of such cabins is a typical example for an *outside-in* design task. However, the distinction between the two types of design is not clear cut. For instance, the design of cabins for families of aircraft is constrained by the shared cross-section, but free w.r.t. the cabin length, resulting in a *hybrid inside-out/outside-in* design task.

Equations 2.1.3 and 2.1.4 furthermore illustrate that detailed information about the cabin is necessary to provide a reliable assessment of the cabin dimensions. Therefore, more detailed cabin design capabilities have been developed for some of the design tools listed in table 2.1.1.

For example, ParaFuse [Jon17] provides the capability to initialize a cabin layout based on the fuselage definition as shown in figure 2.1.3. The layout includes a seating arrangement as well as simplified galleys, lavatories and overhead stowage compartments. Seat rails are considered in the seat layout. The tool is a development of the cabin design capabilities provided in the DEE by TU Delft first described by Brouwers [Bro11] under the name DARfuse. The design can be performed using either the outside-in or inside-out approach. A similar distinction is made in CabLab by Gobbin [Gob15]. However, unlike ParaFuse, CabLab performs the design exclusively in two dimensions, based on a cabin basis surface. A 3D CAD model can be derived from the results only in postprocessing.

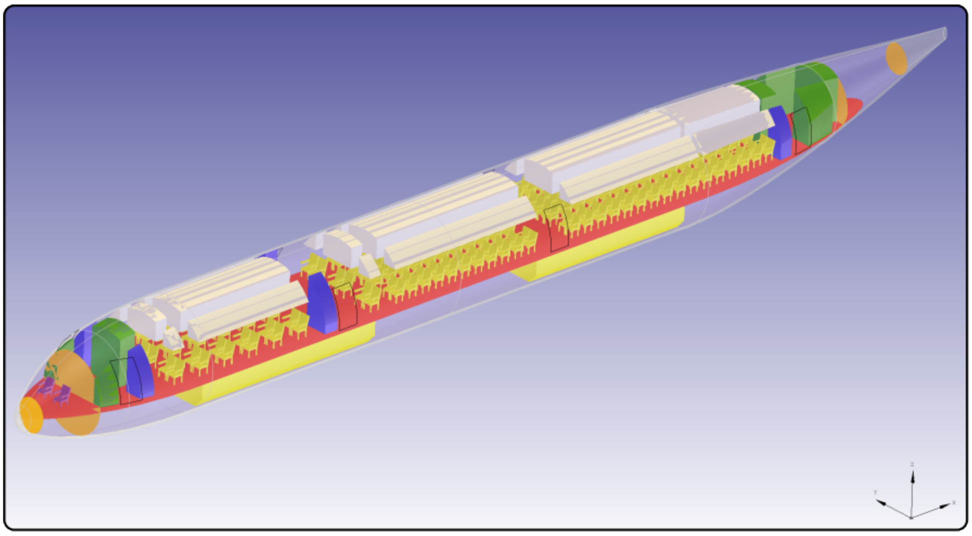


Figure 2.1.3.: ParaFuse cabin model (from [Jon17])

Munjulury et al. [MS+15] present an extension of the RAPID design tool, which enables derivation of details, including the aircraft interior, based on the previously mentioned design of the OML. Aside from the cabin layout, which includes the seating layout, doors, windows, galleys, lavatories and containers, the cockpit area is also represented. The configuration of the cabin layout is performed using a Microsoft Excel spreadsheet, where conformity with the certification specifications and acceptable means of compliance for large aeroplanes (CS-25) [EAS21] and comfort standards is evaluated [Mun14]. However, no information is given concerning the use of automation of the cabin layout generation. The stated purpose of the cabin design is the determination of a required cabin length, which can be compared to the available length provided by the OML. Furthermore, the cabin mass can be estimated.

Beyond the previously mentioned design tools, another three-dimensional cabin design tool is proposed by Motzer [Mot16]. It is shown, how very detailed geometry models of the cabin can be produced in CATIA using graph-based design languages (s. section 4.1.2), deployed via complementary external software. The models are used to perform wire and pipe routing for the electrical and and ventilation system. However, Motzer considers the cabin design activities to be strictly downstream from the preliminary and structure design processes. As such, inputs are received, but no feedback is provided, which affects e.g. the exit layout design as only one type of exit can be installed, due to limitations from the prescribed frame pitch.

Automated design capabilities for only the fuselage have been demonstrated by Fuchte, Gollnick, and Nagel [FGN13]. Using a list of constants and user-defined settings, a fuselage shape is provided along with the corresponding layouts of the structure and cabin, based on a requested number of passengers. The cabin layout includes the floor plan or layout of passenger accommodation (LOPA) and a simplified representation of the secondary structure. Similarly to preliminary design tools like PrADO, the design process consists of a series of individual steps, which are executed according to a predefined sequence. A final analysis step provides cabin data, such as the number of passengers, masses and comfort standards. Based on these results, an iteration is performed to fulfill the number of passengers requirement. The design tool is implemented in MATLAB, but provides an interface to CATIA implemented via Visual Basic for Applications (VBA) scripts for visualization or analysis mesh generation. The authors furthermore highlight the interoperability with other advanced analysis tools via a common central data model (CPACS, s. section 4.1.1).

In industry, the myCabin extension to the commercial Pacelab ACE preliminary design framework is a common choice for cabin configuration. Schneegans and Ehlermann [SE07] provide some background on the methodology. Applications have been published e.g. by Szasz [Sza09] and Abritta, Thorbeck, and Mattos [ATM12] illustrating some very powerful knowledge-based engineering features (s. section 4.3). However, it has not been shown, how the tool can be applied for automatic design studies over large parameter spaces, as is required in preliminary design campaigns [FGN13].

Topics related to the cabin, which require a higher level of detail, such as manufacturing or human factors, are hardly considered by any of the above authors. Nevertheless, these issues have a major impact on the economic success of the final product. The importance of simultaneous design of the product and planning of the manufacturing or assembly, which is sometimes referred to as co-engineering or co-design, is illustrated by an increasing interest in the aerospace industry [PR+17; BC+18]. Consequently, it is desirable, to include such considerations in the conceptual and preliminary design stages, in order to provide a more complete understanding of the product and quickly identify potential showstoppers, before selecting a design for a new aircraft program.

2.1.3. Challenges of preliminary design of novel aircraft system architectures

As the level of sophistication of conventional tube-and-wing configurations approaches saturation, engineers are increasingly pushed towards considering novel concepts or aircraft system architectures to further reduce the ecological footprint of aviation [LaR11]. Configurations are usually proposed due to significant advantages in one particular discipline. However, it can often be shown that the detailed integration of these technologies will have adverse effects in other disciplines, which may counterbalance or even outweigh the initial benefits [Rec22].

For instance, researchers have proposed a high aspect ratio wing to improve the aerodynamic performance. However, the aspect ratio of the wings is also limited by structural constraints e.g. the root bending moment [JS+14; KM14]. This has led to the investigation of strut-braced wing (SBW) configurations [Gra98; PM+17; HG+20], where the bending moment is reduced by introducing an additional support strut. However, the introduction of the strut for improved structural support, results in a compromised aerodynamic performance again.

Another example is the blended wing-body (BWB) [Lie04; Han09; DV14; BV18], which promises gains in aerodynamic performance and structural mass by placing payload in lift-producing areas and reducing the wetted surface. Yet, the concept has never been realized in a large scale passenger aircraft program due to unresolved issues e.g. w.r.t. stability and control, pressurization, evacuation or ground handling.

In light of the advancing climate change, driven also by greenhouse gas emissions [LF+21], electric propulsion and clean energy carriers are also receiving more attention [DLR21a; CF20; ATA21]. Liquid hydrogen (LH_2) has been identified as a potential replacement for fossil fuels as energy carrier in aeronautics [See10; SA+19; DP+22]. Possible scenarios include direct combustion as well as fuel cells for electrical energy generation for all-electric or hybrid electric propulsion architectures [Por18]. It has also been pointed out e.g. by Hoelzen et al. [HS+22] that, along with the introduction of

LH_2, deep structural changes in the air transportation and energy supply system will be necessary. On the vehicle scale, the storage requirements of LH_2 compared to kerosene necessitate substantial architectural changes. These include installation of dedicated tanks, which can fulfill the updated storage temperature and pressure requirements [WK+18; GS19; BC+21] and need to be taken into account in the design process.

Conceptual and preliminary design systems based on handbook methods will not be able to give reliable estimates to perform the trades outlined for the above aircraft system architectures unless significant changes to the systems are implemented. This is exemplified by the work of Hansen [Han09] and Schürmann [Sch16], who describe the substantial efforts necessary to adapt PrADO for BWB and supersonic transport architectures respectively, based on the work of Österheld [Öst03]. Similarly, Jungo [Jun14] illustrates the effort required to implement support for SBW configurations in the CEASIOM system. Moreover, Ko et al. [KM+02] note significant integration effects due to the detailed design of the strut for a SBW configurations, such as shock formation at the wing-strut intersection due to a channel effect. Correct assessment of the penalty due to this effect requires detailed consideration of the fairing of the intersection region.

This need for additional information also applies to the cabin design. For instance, details of the cabin and exit layout of a BWB configuration, are critical to accurately determine evacuation times, which are commonly cited as a big risk in conjunction with such a configuration. Several published examples of cabin designs for BWB configurations are available, e.g. by Lee [Lee03], Eelman et al. [ES+04], Nickol [Nic12], and Baan [Baa15]. However, only Baan attempts to automate the BWB cabin generation for integrated design processes in the Cabin Configurator tool. Two examples of BWB layouts generated by the tool are given in figure 2.1.4. Fuchte et al. [FP+14] provide an assessment of the optimal overall shape of the revenue space without providing a detailed cabin layout.

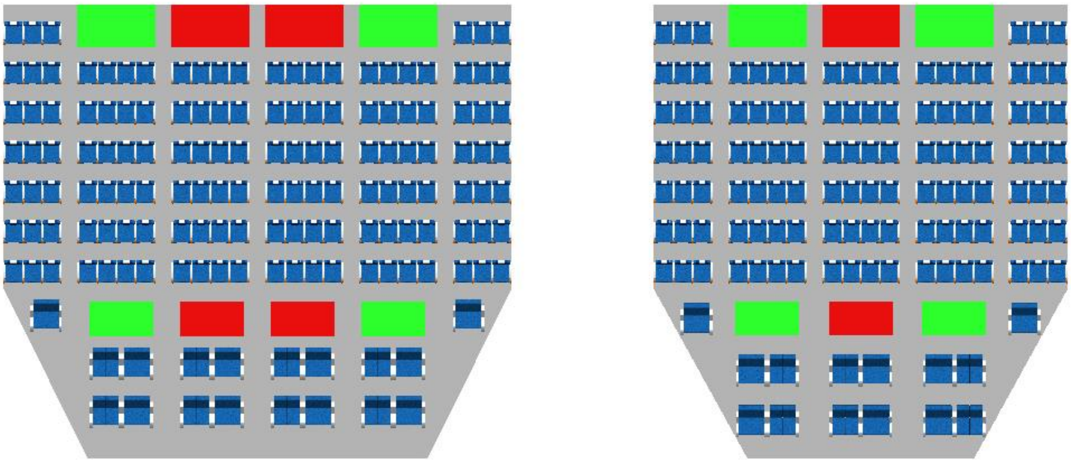


Figure 2.1.4.: Cabin Configurator BWB layouts including galleys (red) and lavatories (green; from [Baa15])

Another particular challenge of novel architectures is that their parametric description may be fundamentally different to that of conventional architectures described in section 2.1.2. For instance, the modeling strategy for BWB configurations often analogous to wings [CZN10], which makes the fuselage a section of the wing as opposed to a discrete body. In this context, the previously established segmentation of the fuselage into three sections also no longer applies. Yet another complicating factor for larger BWB passenger aircraft is the limited availability of mature structural concepts [VT10; QA21a].

For LH_2-fueled designs, the placement of the tanks is an active research topic. Various architectural solutions have been proposed [DP+22], with tanks mounted on the wings as pods, or integrated into the fuselage as illustrated by Silberhorn et al. [SA+19] and Troeltsch et al. [TE+20]. In the latter case, tanks may be placed either in front of or behind the cabin, which requires an increase in fuselage length.

Another option is to place the tanks above the cabin, which entails a larger cross section. Either way, a drag penalty is expected, due to the increase in surface area and an adverse effect on the fuselage slenderness ratio. Furthermore, a stretched fuselage will have implications w.r.t. weight and balance, rotation clearance during takeoff, control surface effectiveness etc. In this case, accurate knowledge of the cabin and tank dimensions will not only make the fuselage length estimation more reliable, but also provide parameters for trade studies to assess e.g. the potential for gains in performance due to a decrease in comfort level. Moreover, whereas the effects mentioned can be identified on a preliminary design level, it must be expected that many other complications due to the tank integration will only arise at a more detailed integration level e.g. of the subsystems.

It follows that, to ensure their relevance given the future challenges in aviation, conceptual and preliminary aircraft design environments need to be flexible to support fundamentally new aircraft system architectures on the one hand and at the same time be capable of providing a high level of detail as required for a meaningful assessment on the other. Since state of the art handbook methods are unable to handle this discrepancy, numerical analysis must be introduced on a larger scale on the one hand, but at the same time be made more accessible for design integration on the other.

2.1.4. Digital process for evaluating and comparing arbitrary aircraft system architectures

In the previous sections, the need to augment the digital aircraft design process in the early design stages has been discussed. On the one hand, the goal is to assess new configurations at a higher level of detail, to better include economically critical aspects like manufacturing. On the other hand, the approach must allow for the evaluation of novel aircraft system architectures, which lie outside of the empirical knowledge base reflected by traditional handbook methods, but hold the promise of a significantly improved product. In this context, numerical analysis can help anticipate unexpected effects on a detailed integration level.

At the same time, Mavris, Tenorio, and Armstrong [MTA08] raise the issue of comparing fundamentally different concepts fairly, i.e. finding ways to determine the “best” solution for vastly different concepts. To this end they propose the definition of a product via functional requirements, which are modeled from stakeholder inputs and broken down into TLAR. Brooker [Bro06] also raises the issue of conflicting design goals and how the importance of a given goal might change over time due to reevaluation of externalities such as climate impact. This means, the functional requirements might evolve over time, as well. Crawley, Cameron, and Selva [CCS15] describe the system architecture as the link between a functional architecture, as expressed by the requirements and the formal structure, i.e. the possible engineering solutions to fulfill said requirements. These considerations have all contributed to the rise of the vision of aircraft system architecture optimization.

In pursuit of this vision, Ciampa and Nagel [CN21] propose the Complex System Development Framework shown in figure 2.1.5, to enable the propagation of stakeholder requirements down to TLAR. It is applied to model the functional architecture of the aircraft, using techniques from model-based systems engineering (MBSE) in order to provide different aircraft system architectures. Compared to traditional document-based systems engineering, MBSE facilitates data communication, improves traceability and handling of information and reduces ambiguity by storing information in digital models [DF+21]. Changes can not only be communicated quickly to connected stakeholders, but automated processes can be deployed to react to these changes. According to Ciampa and Nagel, MBSE can be applied for stakeholder definition, requirements management and architecture generation, but also for life-cycle modeling and resource planning.

Aside from MBSE, MDAO is the second important component of the framework. As illustrated by figure 2.1.5, MDAO is applied for the downstream product design, based on the requirements and architecture provided by the upstream MBSE process. As discussed in more detail in section 2.2, a key contribution of MDAO is the combination and coordination of numerical analysis tools for the different aircraft design disciplines, which are deployed to provide a design solution for a given architecture.

A complete realization of such a process for architecture optimization, places massive demands w.r.t.

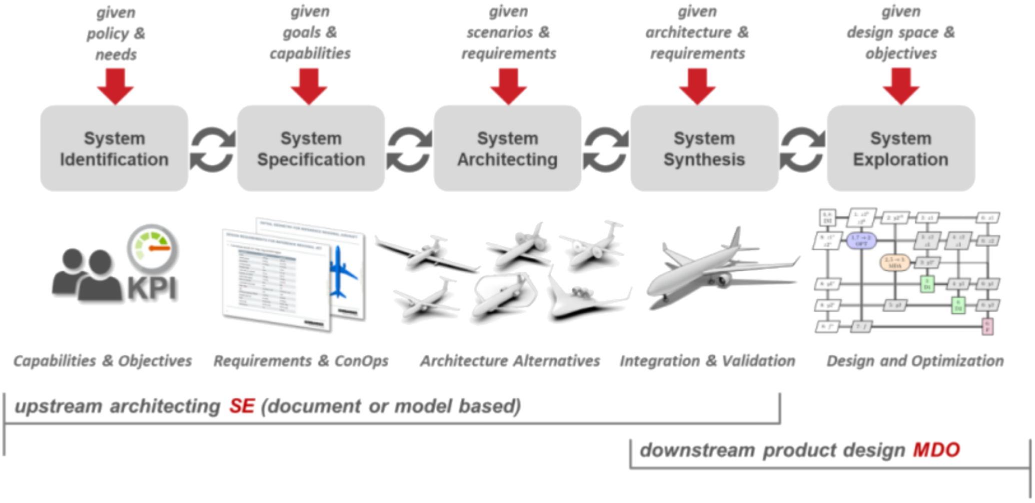

given policy & needs
given goals & capabilities
given scenarios & requirements
given architecture & requirements
given design space & objectives
System Identification
System Specification
System Architecting
System Synthesis
System Exploration
KPI
Capabilities & Objectives
Requirements & ConOps
Architecture Alternatives
Integration & Validation
Design and Optimization
upstream architecting SE (document or model based)
downstream product design MDO

Figure 2.1.5.: Complex System Development Framework combining MBSE and MDO (from [CN21])

the flexibility on the MDAO process and its constituting analysis tools. They must not only support a wide range of design variables, but fundamentally different product architectures. The knowledge-based geometry generation approach proposed in chapter 3 is aimed to facilitate the analysis model generation for the aircraft fuselage and cabin by providing usable geometry models irrespective of the top-level architecture.

2.2. Multidisciplinary optimization and integrated collaborative design workflows

It was established in section 2.1 that a sequential design process is applied in the majority of preliminary design tools, where different disciplinary design modules are traversed in a predefined order. An iteration is performed to ensure consistency of the assumptions. As more and higher fidelity analyses are introduced into the design process, this approach becomes prohibitively slow. This is true for complex system development frameworks designed to support different aircraft system architectures, which are dependent on high-fidelity numerical analysis. Therefore, the techniques from multidisciplinary design analysis and optimization outlined in the following are applied.

First, the problem of multidisciplinary analysis is discussed in section 2.2.1 and then developed into multidisciplinary optimization in section 2.2.2. After this, current research trajectories are highlighted. On the one hand, the problem of finding an efficient geometry description to enable gradient-based optimization is discussed in section 2.2.3. On the other hand, approaches to handle increasingly complex multidisciplinary systems including more and more disciplines are considered in section 2.2.4.

2.2.1. Multidisciplinary analysis in aircraft design

Aeroelasticity, especially of the wing, is the most prevalent example for multidisciplinary problems in aircraft design. As illustrated by the aeroelastic triangle [Col46], effects due to the interactions between fluid dynamics, structural elasticity and inertial forces, which can have a significant impact on aircraft safety, can only be evaluated by combining analyses from the different disciplines. Consequently fluid-structure coupling is the earliest example of multidisciplinary design analysis (MDA).

Early fluid-structure MDA was performed using dedicated monolithic codes, however as analysis methods for the individual disciplines became more sophisticated, partitioned approaches, which couple disciplinary analysis codes in an iterative process have become the norm [WG+19; WD+21; Tra16; KHH13; SD+12]. Figure 2.2.1 illustrates a static aeroelasticity analysis process using eXtended Design Structure Matrix (XDSM) notation [LM12]. The disciplinary analyses are placed on the main diagonal, while coupling variables can be found in the off-diagonal entries. Entries in the same column as the analysis block denote inputs, whereas entries in the same row are outputs. Since the analyses are performed according to the sequence of the diagonal, entries in the upper triangular describe feed-forward connections, whereas entries on the lower diagonal introduce feedback connections. Taking these rules into account, figure 2.2.1 expresses that the aerodynamic loads at the interface f_Γ, which are provided by the aerodynamic analysis, depend on the displacements at the interface d_Γ. The displacements are a result of the structural analysis, which in turn requires the loads as an input. These interface variables can also be referred to as state variables of the system. Since all off-diagonal connections in the XDSM are filled, the system is considered fully coupled.

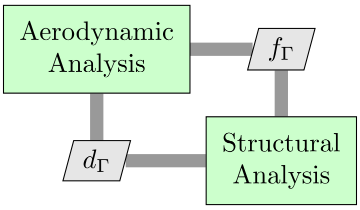


Figure 2.2.1.: XDSM of the fluid-structure interaction problem

In order to solve the MDA problem, the Block Gauss-Seidel procedure can be applied [LM12], as illustrated in figure 2.2.2. A new coordinator block named MDA is introduced to the process, which performs a fixed-point iteration on the coupled problem. To resolve the feedback loop, the MDA component provides an initial guess for the interface displacements d^t_Γ and compares it to the output after running all the tools once. If the residual, which is often chosen to be the ℓ^2-norm $\left\| d_\Gamma - d^t_\Gamma \right\|_2$, is below a given tolerance, the iteration terminates and the current values of d_Γ and f_Γ are accepted as results. Otherwise, a new guess for the displacements d^t_Γ is computed. While an abundance of methods exists to estimate the new guess (s. [MY10] for a comprehensive evaluation), good convergence can often be achieved by simply setting $d^{t,i+1}_\Gamma = d^i_\Gamma$.

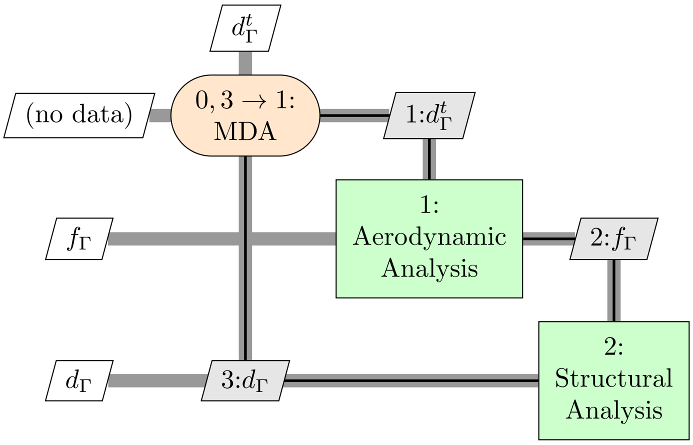


Figure 2.2.2.: XDSM of the Gauss-Seidel MDA for the fluid-structure interaction problem

Other MDA procedures besides the Gauss-Seidel iteration exist, such as the Jacobi iteration [LM12], where all state variables from the last iteration are provided to all analysis blocks by the MDA coordinator block, instead of passing data sequentially between analysis blocks. On the one hand, this procedure has the advantage that all analyses can be executed in parallel, which can yield a significant run time benefit. On the other hand, it is more difficult to assert feasibility of the result, i.e. consistency of the different outputs.

2.2.2. From analysis to optimization

The purpose of analysis in aircraft design is to inform design decisions. So far, the MDA presented in the previous section only provides consistent load, displacement distributions. To leverage these capabilities to automatically make design decisions, the process in figure 2.2.2 can be expanded to include an optimization. Sobieszczanski-Sobieski [Sob95] refers to this methodology as multidisciplinary design optimization (MDO). Over time, the more general acronym MDAO, a conglomerate of MDA and MDO, has also gained popularity [SM+15].

A typical simple MDO task could be to determine the wing aspect ratio A, which results in the longest range R according to equation 2.1.1. A more slender wing improves the aerodynamic performance $\frac{L}{D}$. However, it is also necessary for the wing structure to withstand the loads, i.e. all the stresses σ in the wing must be lower than the smallest allowable stress σ_{allow} for any given structural component. The allowable stress is typically be computed for each failure mode, e.g. static strength, buckling or fatigue and depends on both material properties, such as the yield strength $R_{p,0.2}$ and the geometric properties of the structure, e.g. the panel aspect ratio a_{panel}/b_{panel} for single panel buckling [Bru73]. Failure to meet the criteria can be resolved e.g. by increasing the skin thickness t_{skin}, at the cost of increased structural mass W_i and thus reduced range. In addition to the aforementioned stress-based criteria, strain-based criteria exist for carbon fiber reinforced polymers (CFRP) [Nah86; SKH04]. They are, however, omitted in the following due to their limited range of applicability according to Soden, Kaddour, and Hinton [SKH04].

Based on the above, an optimization problem can be formulated:

$$\begin{aligned} \underset{A,t}{\text{minimize}} \quad & -R(A,t) \\ \text{subject to} \quad & A_{min} \leq A \leq A_{max}, \\ & t_{skin,min} \leq t_{skin} \leq t_{skin,max}, \\ & \sigma(t) - \sigma_{allow}\left(R_{p,0.2}, a_{panel}, b_{panel}, t_{skin}, ...\right) \leq 0, \end{aligned} \tag{2.2.1}$$

The aspect ratio A and the skin thickness t are the design variables, the negative range is the objective function, and the stress limit and the limits for A and t are constraints. The evaluation of the range and the stress constraint requires additional state variables w.r.t. to the original MDA. The stress distribution σ and the structural mass W_i can be provided by the structural analysis in addition to the displacements, while the aerodynamic performance $\frac{L}{D}$ can be computed as part of the aerodynamic analysis. All other inputs, e.g. the yield strength $R_{p,0.2}$ or the specific fuel consumption c, are assumed to be constant.

Figure 2.2.3 shows the XDSM of one possible MDO process implementation for the problem in equation 2.2.1 using the multidisciplinary feasible (MDF) architecture [CJ+94]. The nested MDA loop is treated by the optimizer like a single disciplinary analysis. MDF is a popular architecture choice, because the consistency of the state variables is automatically ensured by the MDA [ML13]. However, a large variety of different MDAO architectures have been proposed to address specific applications, of which Martins and Lambe [ML13] provide a comprehensive overview.

The goal of optimization is to minimize a given objective function $f(\mathbf{x})$ w.r.t. the design variable vector $\mathbf{x}$ under the inequality constraints $c_i(\mathbf{x}) \geq \mathbf{0}$. Optimality is defined by the KARUSH-KUHN-TUKER (KKT) conditions [Kar13; KT51], which are sometimes also referred to as the design equation [IM+20]. For unconstrained problems, the gradient of the objective function disappears at the minimum, while the Hessian is positive definite, i.e.

$$\nabla f(\mathbf{x}^*) = \mathbf{0} \, \text{and} \, \nabla^2 f(\mathbf{x}^*) \, \text{is positive definite.} \tag{2.2.2}$$

If constraints are present, the first-order criterion must be expanded to take into account the possibility of minima on the constraint boundary. To this end, the gradient of the LAGRANGIAN function for the optimization problem is considered, instead of the objective function. If only inequality constraints are present, this results in

$$\nabla \mathcal{L}(\mathbf{x}^*, \lambda^*) = \nabla f(\mathbf{x}^*) + \sum_j \lambda_j^* \nabla c_j(\mathbf{x}^*) = \mathbf{0}. \tag{2.2.3}$$

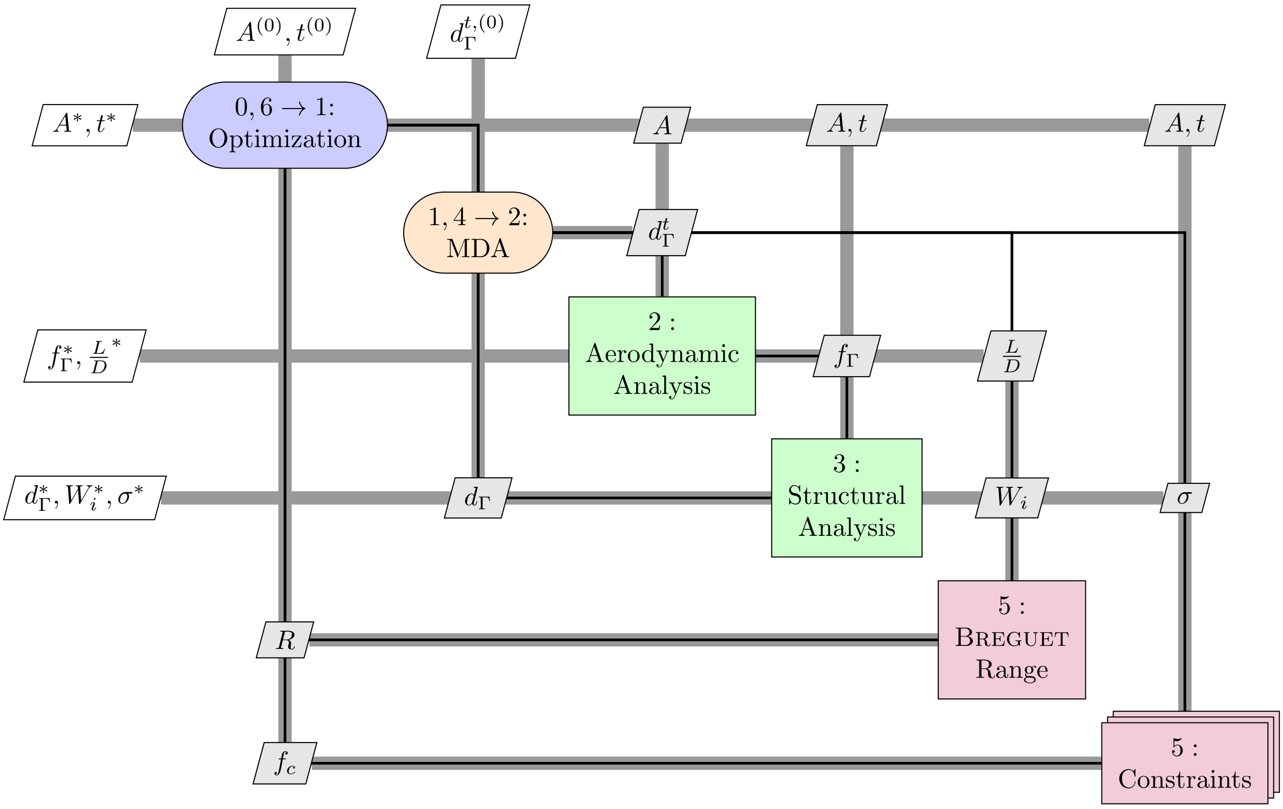


Figure 2.2.3.: XDSM of the optimization process for the range problem using the MDF architecture

The LAGRANGE multipliers λ_i are only active, if the corresponding constraint is active, i.e. $\lambda_i^* c_i(\mathbf{x}^*) = \mathbf{0}$ for all constraints. Furthermore, the problem must be feasible, which means all constraint conditions must be fulfilled and $\lambda_i \geq 0$ for all LAGRANGE multipliers. It follows that a potential minimum is reached if the gradient of the objective function can be equalized by a weighted sum of the gradients of the active constraint functions.

Depending on the type of objective function, constraints and design variables, different classes of optimization problems exist, each with different corresponding solution algorithms [Kel99; NW06; PT+07]. MDAO problems are typically nonlinear programming problems (NLPs). Furthermore, the objective function may not be explicitly known, but the result of e.g. a call to a closed-source commercial analysis tool. Several types of optimization algorithms exist for such a case, which are briefly discussed in the following.

Gradient-based optimizers Gradient-based optimizers use quasi-Newton gradient-descent methods to iteratively solve the NLP. Commonly, constrained and unconstrained optimizers are distinguished, as are mono- and multidimensional optimizers. MDAO problems, like the example given in equation 2.2.1, tend to be constrained multidimensional problems. These types of problems can be solved using sequential quadratic programming (SQP). Implementations e.g. NLPQL [Sch86] or SLSQP [Kra88], are available in open-source scientific computing libraries.

A key enabler for large-scale gradient-based optimization is the availability of sensitivities of the objective function in the form of total derivatives w.r.t. the design variables [MH13]. The total derivative for the objective function f is computed using the chain rule [MN21] and given by

$$\frac{df}{dx} = \frac{\partial f}{\partial x} + \sum_{i=0}^{n} \frac{\partial f}{\partial y_i} \frac{dy_i}{dx}. \tag{2.2.4}$$

The sensitivity terms $\frac{dy_i}{dx}$ contain contributions from each component of the multidisciplinary system.

As discussed e.g. by Moore [Moo12], the computation can become rather complex depending on the system architecture. However, it can be concluded that every analysis tool involved in the process must provide not only the analysis result, but also the derivatives of the analysis result w.r.t. the overall design variables.

Several approaches have been proposed to compute the derivatives of analysis tools. If the underlying analytic functions are known manually implementing the derived functions is an option. This is, however, rarely the case for real-life engineering problems, where complex numerical codes are often used. Instead it may be necessary to apply techniques such as finite differences (FD), complex step (CS)[MSA03] or algorithmic differentiation (AD)[GW08]. In FD, tools are considered as black boxes, thus making it the easiest method to implement. Some optimization algorithms even provide a built-in internal FD scheme [Pow94; Pow09]. However, one additional function call is required per design variable, which is unacceptable if many design variables or computationally expensive high-fidelity analyses are involved. CS and AD are more efficient, but need to be supported by the analysis tools themselves.

A common drawback of gradient-based optimizers is that they are only capable of finding local optima for non-convex functions.

Bayesian optimization The problem of potentially unavailable analysis sensitivities is addressed in Bayesian optimization. Instead of providing the sensitives directly, a surrogate model of the multidisciplinary system is constructed from a low number of sample points, selected using a design of experiments (DoE) scheme. Many authors choose Kriging [SW+89], also referred to as Gaussian process regression, as surrogate model [SAZ03; LK+09; BL+17], but other approaches have been demonstrated as well [Kri05; HM18; BH+19]. The surrogate model provides a smooth interpolation between the known samples, the gradients of which are known. This allows for gradient-based optimizers to be deployed on the surrogate model. In an iterative loop, the optimum value of the surrogate model can then be compared to the true result of the multidisciplinary system, and the process is either repeated for a new model, which includes the new sample point, or stopped if the results are sufficiently close.

Aside from a predicted function value, the surrogate model also provides an estimate of the variance, i.e. the reliability of the estimate of the function value. This enables the computation of different infill criteria, which can be optimized instead of the objective function itself. The weighting of the function value against the variance can be tuned to pursue an optimization strategy, which is focused on either exploration, i.e. the broad evaluation of the design space to avoid getting stuck in local minima, or exploitation, i.e. quick convergence towards a local minimum [SPG02].

Stochastic optimizers In this class of optimizers, stochastic methods, where the design variables are modified randomly or semi-randomly, are applied to increase the chances of finding the global minimum. Another advantage is that unlike e.g. gradient-based optimizers, stochastic optimizers can handle discrete design variables, such as integers. Thus, they provide an interesting option for the evaluation of entirely new classes of optimization problems, such as mixed-integer linear (MILP) and nonlinear (MINLP) programming problems.

On the downside, stochastic optimizers require a very large number of function evaluations compared to gradient-based optimizers, which makes them ill-suited for MDAO problems with long running disciplinary analyses, e.g. CFD. [MN21]

Over the years, numerous publications on MDAO applications have appeared, the detailed discussion of which exceeds the scope of this thesis. Sobieszczanski-Sobieski and Haftka [SH97] list early examples, where MDAO has been applied for simultaneous aerodynamic and structural optimization as well as for simultaneous optimization of structures and control. A more recent review of MDAO research by Papageorgiou et al. [PT+18] still finds an overwhelming bias towards aerodynamics and structures in literature. Cabin design as well as connected analyses e.g. of human factors or manufacturing aspects are not present. The study furthermore shows that the vast majority of applications so far has been in academia. Nonetheless, publications e.g. by Giesing and Barthelemy [GB98], Gazaix et al. [GG+11], Piperni, DeBlois, and Henderson [PDH13], and Cavalcanti et al. [CL+18] are indicative of a sustained interest, also in industry.

In many studies on MDAO applications, the authors apply either an ad-hoc implementation, or custom-built frameworks [AL+04; LH+04; AK05] to assemble the MDAO process. While this is a relatively simple task for basic MDAO problems, it becomes more difficult as the complexity of the problem increases. As a result, the openMDAO library [GMN10; HG12; GH+14; GH+19] provides a unified open-source programming framework for facilitating the development of MDAO applications. It not only facilitates the formulation and exploration of MDAO systems, but also provides access to nonlinear solvers for MDA and optimizers. Furthermore, the computation of the total derivative is handled, based on the sensitivities for the individual disciplines, which can also be provided.

Weck et al. [WA+07] describe the range of possible trajectories for new research in MDAO in terms of two extremes: horizontal and vertical growth. Essentially, horizontal growth is concerned with the increasing the fidelity and efficiency of current MDAO processes. In contrast, vertical growth is concerned with increasing the number of disciplines involved to provide a more comprehensive understanding of the product to ultimately enable the evaluation of more challenging and complex optimization problems. According to the authors, MDAO research must address both these issues for an ideal growth direction, which they refer to as "dream growth". Therefore, the challenges and and implications of both extremes are discussed in the following.

2.2.3. Horizontal growth: Geometry description for efficient high-fidelity gradient-based optimization

Research activities towards horizontal growth of MDAO look to enable numerically efficient gradient-based optimization including high-fidelity analysis and increasingly detailed design variables. However, the set of disciplines involved remains largely untouched.

As discussed in section 2.2.2, tools must provide sensitivities w.r.t. the design variables in addition to the analysis result in order to enable efficient gradient-based optimization. While few commercial analysis codes provide this feature, eligible research codes have been published e.g. for aerodynamic [WB+10; EP+16; TW+19] or structural analysis [SKH96; KM14]. Combined aero-structural frameworks supporting gradient-based optimization have also been proposed [SD+12; Rei15; Sán17; KS+19].

To enable fully gradient-based MDAO, it is furthermore necessary, to provide derivatives of the geometry model generation w.r.t. the top-level geometric design variables, such as the wing aspect ratio. To this end, Samareh [Sam99] describes three geometry parametrization schemes for MDAO processes, the discrete approach, the free-form-deformation approach, and the CAD approach, which are discussed in the following.

Discrete parametrization In the discrete approach, no central geometry model is provisioned, from which the analysis meshes are derived. Instead, the points of the analysis grids are manipulated directly. Whereas this approach is simple to implement and offers fine grained, localized control of the geometry, it is impractical for real-life MDAO applications for a number of reasons. These include difficulties to maintain a smooth geometry and ensure manufacturability of the results. Furthermore, given the number of degrees of freedom in modern CFD surface meshes, there are too many design variables for the optimization to remain efficient. Since the disciplinary meshes are all different from one another it is also difficult to link disciplinary design variables in an MDAO context.

Free-form deformation The free-form deformation (FFD) scheme is adopted by many authors in aerodynamic shape optimization or aero-structural optimization [Sam04; Ron06; Gu17] to overcome some of the limitations of the discrete approach. In FFD, a control volume, described using a Bézier

[SP86], B-spline [GP89] or NURBS [LW94] formulation (s. section 4.1.1), is placed around the component of interest and deformed using the control points of the volume as design variables. The analysis mesh points are mapped to the local coordinate space of the bounding volume, which yields the corresponding local coordinates $[u, v, w]$. Evaluating the deformed bounding volume at these local coordinates yields the deformed mesh points. This results in a smoother deformed surface and can significantly reduce the number of design variables. However, the volume control points still do not correlate well with intuitive engineering parameters. In fact, the problem is still artificially complicated for many parameters, e.g. for the span. Instead of changing a single number, the span in FFD must be modified by adjusting the lateral positions of at least half of the FFD control points. Moreover, this makes it difficult to map optimization results back to product model parameters later.

CAD-in-the-loop modeling As a third option, a CAD-based approach is considered. Kenway, Kennedy, and Martins [KKM10], Hwang and Martins [HM12], and Hwang, Kenway, and Martins [HKM14] propose a purpose-built differentiated parametric geometry modeler, where a watertight OML geometry is stitched together from a collection of B-spline surfaces. They also provide the possibility to model simple internal structures, such as wing spars and ribs. However, as the requirements w.r.t. the level of details of the geometry model increase, this approach is not sufficiently flexible and the inclusion of a geometry model based on a full CAD kernel becomes an imperative.

Bobrowski et al. [BF+17] propose a reduced-order model (CAD-ROM) approach, where a surrogate model of the surface mesh generation results is generated based on parametric CAD geometry built in a commercial environment. Similar meta-modeling approaches have been discussed previously for analysis tools in general [VS+14]. However, a sufficiently accurate surrogate model will still require a prohibitively high number of sample points, each of which implies a call to the commercial tool [MSR17]. The pyGeo library [HY+23] provides both FFD and parametric CAD capabilities by providing interfaces to OpenVSP and ESP. However, analytical derivatives are only available for FFD, whereas the parametric CAD interfaces rely on FD.

This leaves the application of a fully differentiated CAD kernel. Solutions have been proposed by Dannenhoffer and Haimes [DH15] who implement FD for the ESP and by Banović [Ban19] who shows a working example using AD. In both cases the OCCT CAD kernel [OCC23a] provides the basis for the development. Even though, neither of the codes have been shown to work for large-scale aircraft MDAO problems, promising results have been published e.g. by Xu, Jahn, and Müller [XJM13], Dannenhoffer and Haimes [DH17], and Banović [Ban19]. A pathway towards making this technology available e.g. for CPACS by leveraging parametric trees to chain algorithms has been proposed by Kleinert et al. [KR+23].

The above shows that gradient-based MDAO is a very active field of research, where development is bound to continue in the future. For the purpose of this thesis, it can be deduced that a geometry model generation tool for an MDAO setting should be based on a CAD-kernel. Another requirement is that a new implementation should be open to gradient evaluation using CS or AD. Consequently, open source solutions should be preferred to proprietary commercial codes.

2.2.4. Vertical growth: Increased variety of disciplines and assessments through collaboration and facilitated process setup

The notion that the success of new configurations will not exclusively be defined by the classical disciplines of aircraft aeroelasticity has spawned a complementary movement to the research activities outlined previously. In many cases, it can be advantageous from an economic perspective to sacrifice a certain amount of mass or a few drag counts for the sake of e.g. improved production rate or passenger comfort. In order to take these aspects into account in an MDAO process, the objective function must be adapted to include these parameters and weigh them against the classical performance parameters. Furthermore, analysis capability must be available to evaluate a given design w.r.t. these aspects.

The expansion of the scope of the MDAO entails a number of new challenges. Chiefly, the growing variety and respective complexity of the disciplines involved make it impossible for a single person to overlook the entire process and adequately assess all of its results. This has motivated research on collaborative design workflows, where each disciplinary tool or design competence is the responsibility of a disciplinary expert [Kro04; CN20]. In this context Kroo [Kro97] identifies three distinct generations of collaborative MDAO systems, which have later been corroborated by Zill [Zil13] and Ciampa and Nagel [CN20]:

In the *first generation*, which Kroo describes as integrated analysis with optimization, the optimizer and the analysis capabilities are tightly integrated in a monolithic system. This requires low-level access from the optimizer to the respective analysis codes, which on the one hand, makes it difficult to incorporate established disciplinary analysis tools and on the other hand creates a significant barrier for the addition of new analyses.

Therefore, in the *second generation* of MDAO, the coupling between the analysis tools is loosened to enable distributed analysis. While the optimizer retains control of all the design variables, the actual analysis is performed by dedicated disciplinary processes. It is possible for these processes to be executed either on the same machine or on separate machines connected via network. The results are fed back to the optimizer, which provides a new set of parameters to be evaluated in the following iteration. State-of-the-art high-fidelity gradient-based MDAO frameworks such as the FlowSimulator [Rei15] or the fluid-structure interaction component of SU2 [Sán17] can be considered part of the second generation. However, distributed analysis has the drawback that the disciplinary analysis tool providers can not directly influence the design variables, which are under control of the top level optimizer.

This paradigm is therefore abandoned in the *third generation* of MDAO, which introduces distributed design processes instead. In such a process, the participants do not only provide analysis capabilities, but actively participate in the design by contributing design decisions. For instance, a structural design tool would no longer just return the stress distribution or reserve factors, but contribute a feasible or even optimal thickness distribution. This procedure has several advantages. To begin with, design proposals are contributed by disciplinary experts, which are better trained to judge the design results than the process integrator. Consequently, the task of the optimizer is no longer to process the analysis output and propose design changes, but to coordinate and synchronize all contributions.

The shifting of the responsibility to make design decisions towards the design competence level also facilitates the implementation of multi-fidelity processes, as described e.g. by Lazzara, Haimes, and Willcox [LHW09], Böhnke [Böh15], and Dannenhoffer and Haimes [DH16]. Multi-fidelity implies that various levels of abstraction of the model can be analyzed even within a single discipline. In this way, low-fidelity analyses with low run time requirements can be deployed for large design space explorations. As promising designs are identified, the level of fidelity is increased and higher fidelity methods are deployed, which take into account additional design details at the cost of longer run times. Analyses of the same model at different fidelity levels can be considered two separate design competences. To facilitate the coordination of multi-fidelity contributions, Moerland, Becker, and Nagel [MBN15] propose a division of the design competences into four discrete fidelity levels, ranging from statistical and empirical tools (*level 0*) to highly accurate numerical simulation capabilities to capture detailed local effects (*level 3*).

In theory, the third generation MDAO approach facilitates the addition of new disciplinary design competences to the process, since each new competence will act as a black-box w.r.t. the overall process. However, third generation MDAO processes are subject to an entirely new set of challenges:

Design competence provision In practice, the preparation of a disciplinary design tool for distributed design processes requires non-negligible initial efforts by the design competence provider. Specific technical requirements exist in terms of both tool automation and data interfaces [WCN22]. On the one hand, the design tool should be capable of running fully automatically without user intervention (hands-off automation). This includes not only the analysis, but also the subsequent decision making, which can be automated e.g. using a nested optimizer. On the other hand, the tool should be capable of providing and receiving all relevant information on the current state of the product from the process.

This is usually implemented using wrapper interfaces to a central product data model as described in section 4.1.2.

Furthermore, the tool should, to some extent, be robust towards missing or unexpected input data. Some design details necessary for the analysis may not always have been made available by the preceding tools in some stages of the design process. As a result, the missing details, for instance the structural layout of a fuselage, need to be initialized before the actual execution of the design competence. So far, no unified solution for coordinating initial assumptions has been proposed. Instead, design competence providers choose initialization parameters independently, without centralized checks for overlaps. This workaround is acceptable if boundaries between disciplines are clear cut, as is the case for structures and aerodynamics. However, it makes the design process vulnerable to inconsistencies if responsibilities of disciplines can not clearly be separated, as is the case e.g. for the fuselage structure and the cabin layout.

Non-technical barriers Aside from the technical difficulties, Belie [Bel02] also identifies some non-technical barriers to MDAO, which fall into three general categories: Confidence, culture and complexity. These barriers also translate into challenges for third generation MDAO processes.

Due to the black-box nature of disciplinary design competences and the ensuing lack of detailed knowledge of the implementation, the results provided can be difficult to comprehend for most process participants. This makes it harder to instill confidence in the design competence results. However, according to Belie, this problem exists even in current development processes, since experts from one department will have to accept results from another, without looking deep into the details. He therefore proposes to build on proven, verified work as much as possible and provide high visibility of both the process flow and the results.

For barriers due to organizational culture Belie lists inertia, e.g. due to the required initial development efforts, territorial concerns between departments and fear of loss of control. The necessary steps for disciplinary analysis experts to provide a third generation MDAO design competence listed by Walther, Ciampa, and Nagel [WCN22], i.e. hands-off automation, willingness to share results, and adherence to a central data schema, require competence providers to overcome these cultural barriers. Both publications suggest that highlighting the benefits of MDAO and providing education will help convince people to overcome their inertia. Furthermore, Belie recommends putting in place cross-cutting organizational units with sufficient authority to foster interdepartmental exchange. Baalbergen, Lammen, and Kos [BLK12] address the problem of inter-organizational barriers, by giving contributors in distributed MDAO processes full control over when and how often their tools get accessed to prevent unwanted data extraction.

Finally, Belie mentions barriers due to complexity, which results from the interplay of data, human, discipline and organizational layers. Even though complexity is presented as a non-technical barrier, many of the solutions proposed to manage it, which are presented in the subsequent paragraph, are technical in nature.

Collaborative workflow integration The collaborative distributed design systems typical for third generation of MDAO can be implemented using e.g. a process integration and design optimization (PIDO) framework. The open-source Remote Component Environment (RCE) [SF+12] and the commercial ModelCenter [ANS23b], modeFrontier [EST23], Optimus [Noe23] or Isight [Das23] are established tools in this field, which allow for individual design competences to be connected remotely via network. A common trait of these systems is a visual programming interface, which allows the user to connect blocks representing executable tools using arrows which represent data flow in a GUI. The approach has been applied successfully in a number of projects both within a single organization [KA+15; GK+18; Hei18] and in a cross-organizational setting [CP+19]. The design process is controlled by an integrator, who is responsible for assembling the workflow and running the optimization. Many of the integration tasks surrounding collaborative MDAO have been already identified by Sobieszczanski-Sobieski [Sob95] in the early 1990s. Among the most time consuming tasks is the set up of the workflow, i.e. building the connections between available design competences.

As the number of disciplines or design competences increases further, the task of manually setting up the design workflows becomes prohibitively extensive. In addition, slow reconfigurability is an obstacle to rapid tool integration and MDAO system architecture exploration. The automation of the problem setup has therefore been identified as another important research topic. Pate, Gray, and German [PGG13] are among the first to propose the application of graph data structures to perform dependency analysis on design competences and build MDAO workflows automatically. The idea has since been picked up e.g. by Gent [Gen19a] and Page Risueño et al. [PB+20], who not only implement and refine the graph based reasoning on the connections, but also provide interfaces to PIDO frameworks or environments like openMDAO to create executable workflows automatically. Ciampa et al. [CP+19] showcase the potential for reducing the MDAO process setup time using this approach. A comparable framework has been presented by Gallard et al. [GV+18]. While a classical MDF architecture is usually applied [TB+18], other dedicated architectures for collaborative optimization have been proposed [KM00; IM+20]. For a more customized approach, Aigner et al. [AG+18], Lafage, Defoort, and Lefebvre [LDL19], and Page Risueño et al. [PB+20] furthermore propose user interfaces for interactive MDAO system exploration and manipulation.

To preserve the computational efficiency of the optimization and avoid exaggerated run times, the increased number of disciplines must typically be balanced with sacrifices in analysis fidelity. Furthermore, gradient-based optimization as outlined above is usually not possible due to limitations of the tools involved. Instead, Bayesian optimization of surrogate model built from an initial sampling plan of the design space is commonly used.

2.2.5. Summary

MDAO is introduced as a promising solution to handle the complexity of interdisciplinary dependencies in aircraft design. Due to its coupling of numerical, often physics-based, analysis tools, MDAO is deemed more fit to handle novel configurations than the traditional empirics-based aircraft design methods discussed in section 2.1.1. Whereas in the past, much of the focus of MDAO has been placed on the interaction of aerodynamics and structural design, more recent efforts strive to include other design aspects, such as human factors or industrialization. In these fields, the aircraft cabin design has a significant impact, which is why detailed knowledge of the cabin needs to be made available within the respective MDAO processes.

Distributed collaborative processes break down the complex overall design task into disciplinary sub-design-tasks, which can be handled by disciplinary experts. Product knowledge is typically exchanged using a central data model, which serves as a single source of truth. However, the sub-tasks usually involve an element of augmentation, where data, which is not yet available in the overall process but required for the analysis, is added by the respective experts. Currently, no mechanism is available for distributed processes to ensure the consistency of the assumptions, which undermines the authority of the central data model. For instance, knowledge of the positions of the door cutouts is important for both the structural and cabin design engineer. If both assume different door positions without communicating their assumptions, it may severely compromise the results of the overall design process.

3. Research hypothesis

In the following, a research hypothesis is formulated, based on the findings of chapter 2. In section 2.1, the state of the art in fuselage and cabin design for early design stages has been explored, and the applicability of the methods to new and unconventional configurations, necessary to meet the emission goals stated in chapter 1. It is shown that the commonly used empirical methods are insufficient and should be substituted with physics-based methods.

To handle the increased complexity and coordinate the different disciplinary contributions, collaborative third-generation MDAO techniques are introduced in section 2.2, where different experts participate in a single distributed integrated design process. In this context, different disciplinary analyses typically have different requirements w.r.t. design details, which need to be available. The details are commonly augmented by the disciplinary experts as required, without assuring consistency with other disciplinary assumptions, which can lead to inconsistent designs. On the other hand, the fragmentation of the design knowledge impedes adaptations for novel configurations, as changes to a multitude of tools are required.

To overcome these shortcomings, the application of knowledge-based engineering techniques (s. section 4.3) to provide consistent and adaptive design detail generation capabilities is proposed in this thesis, resulting in the following research hypothesis:

Research hypothesis Knowledge-based engineering techniques can be applied in the context of multidisciplinary design analysis and optimization processes, both to generate new design details as they are required within the process and to assemble geometry models for disciplinary analysis model generation at the exact required level of fidelity. This methodology furthermore allows adaptation to fundamentally new product system architectures. Linking the generation of models and design details enables the automatic selection and generation of only those design details, which are required inputs for the model generation task at hand.

To assess the validity of the research hypothesis, the proposed methodology is applied to the use case of the aircraft fuselage and cabin. Due to the close relationship e.g. between the structural design and cabin design, this is an interesting use case in terms of interdisciplinary dependencies. To support the discussion of the hypothesis in the subsequent chapters, it is furthermore broken down into the following four working hypotheses:

Working hypothesis 1 Aircraft fuselages and cabins can be described using parameters, which can be organized in a knowledge graph by providing rules, which define relationships between the parameters. This architecture enables the modular inclusion of new knowledge and the setup of dynamically configured design and modeling processes without predefined program flow.

Working hypothesis 2 Based on the knowledge graph, consistent geometry for disciplinary analysis can be derived. The level of fidelity is tailored to the disciplinary application through targeted querying of the knowledge graph. Based on the disciplinary models, analyses can be performed, enabling consistent multidisciplinary and multifidelity evaluation of the present design as needed within third-generation MDAO processes.

Working hypothesis 3 Using graph analysis, missing data for modeling a product instance can be identified and the knowledge graph can be extended with design functionality. This includes knowledge from diverse disciplines such as preliminary design and structural design, thus enabling integrated interdisciplinary fuselage and cabin design.

Working hypothesis 4 The process is not only parametric and suited for exploration of large design spaces, but also inclusive to novel product architectures through extension of the knowledge graph, thus supporting fundamental decisions on system architectures.

4. Computer aided engineering methods to enable fuselage MDAO

The collaborative approach in third generation MDAO entails the need for exchange interfaces for product data, which allow experts to share newly generated product knowledge with collaborators in the design process. In section 4.1 the underlying concepts and theory behind the prevalent CAD-modeling approach are discussed. To overcome the shortcomings of this approach w.r.t. collaborative MDAO applications, more general data-centric modeling approaches as described e.g. by Nagel et al. [NB+12] are introduced as well.

To evaluate the applicability of the existing means of data exchange, different existing numerical analysis and design capabilities are presented in section 4.2. In line with the **research hypothesis** of this thesis stated in chapter 3, the focus is specifically placed on capabilities for fuselage and cabin evaluation. The competences are evaluated in terms of which data is required and in what form. On the one hand, this serves to illustrate the types of input data and level of detail required to enable a given disciplinary analysis. On the other hand, gaps and overlaps between data requirements for different analyses are identified.

Finally, the knowledge-based engineering (KBE) methodology is presented in section 4.3 as a potential solution for providing product information tailored to the requirements of disciplinary analysis. As shown e.g. by La Rocca [LaR11], the methodology enables the implementation of a MMG based on a central product data model as well as automatic generation of product knowledge by formulating executable relationships between product features.

4.1. The product in a digital engineering process

Commonly accepted, unambiguous interfaces between different design competences are a key enabler for successful collaborative MDAO processes. As shown e.g. by Böhnke, Nagel, and Gollnick [BNG11], a central common model can help reduce the number of interfaces, which must be implemented to connect all disciplines involved in a given design process. Since all product data is stored in the central common model, it represents the virtual product in digital engineering processes.

With geometry being a central aspect of engineering products, a central geometry model, sometimes referred to as the central digital mock-up (DMU), is often selected to represent the virtual product (s. section 4.2). Consequently, the description of geometry in computer-aided geometric design (CAGD) systems is examined in section 4.1.1. That said, the amount of information available may be insufficient to build an accurate geometry model of the configuration in the early aircraft design phases, as exemplified by Raymer's cylinder model for the fuselage from section 2.1.2. Conversely, in later stages of MDAO, detailed information beyond sheer geometry may be required, to perform a given analysis, such as material properties or mission details. To this end, data-centric product models have been proposed, which are discussed in section 4.1.2.

The gap between data-centric parametric modeling and geometry-centric modeling using CAGD is reflected in the working hypotheses. Whereas a parametric, i.e. data-centric description is mentioned in **working hypothesis 1**, the need for a geometry-centric model is stated in **working hypothesis 2**. To bridge this gap between the two formulations, the concept of a parametric modeling engine is introduced in section 4.1.3, which can provide geometry-centric views based on the data-centric product models.

4.1.1. Geometric product models and CAGD systems

The need to transition from a data-centric parametric model to a geometric product description expressed in **working hypothesis 2** is further supported by section 2.2.3, where it is found that a CAD-in-the-loop geometry model should be available for high-performance MDAO. An exasperating factor in aircraft design is that the geometric shape has a direct effect on the performance on the product. As a result, shapes e.g. the outer mold line often consist of complex free-form surfaces, which must be understood in order to derive accurate geometries from the parametric description. For the efficient implementation if subsequent details a good understanding of CAGD systems is furthermore required. Therefore, the theory behind the most common types of free-form surfaces is discussed in the following as well as some more general background on CAGD systems.

The idea of using computers to describe geometry dates back to the early days of computing [CM60; Sut63] and CAGD systems are commonplace in industry today. There are numerous texts that outline the theoretical foundations of CAGD, e.g. by Farin, Hoschek, and Kim [FHK02], Faux and Pratt [FP79], and Sederberg [Sed12]. Nonetheless, some essential background is provided in the following.

Fundamentally, geometry is typically described by univariate curves $\mathbf{x}(u)$, which are discussed in section 4.1.1.1, and bivariate surfaces $\mathbf{x}(u, v)$ described in section 4.1.1.2. Trivariate volume descriptions $\mathbf{x}(u, v, w)$ are also possible, but more rarely used. Instead, different advanced techniques have been developed to describe volumes, which are introduced in section 4.1.1.3.

4.1.1.1. Mathematical description of curves in CAGD

Considering the fuselage modeling techniques discussed in section 2.1.2, the defining role of curves quickly becomes apparent. Every modeling approach assumes a cross-sectional profile, which is defined by a curve. The assumption may be either implicit, e.g. the circular cross-section for the cylinder model, or explicit e.g. the OpenVSP profile definitions. In addition, the need for guide curves to realize more detailed fuselage representations is stated.

In CAGD systems, the simplest curve types are straight lines, which may be trimmed or untrimmed. Other common types of analytical curves are conic sections e.g. circles, ellipses, parabolas or hyperbolas. All of these types of curves can be described analytically. However, they are usually not suited to represent the fuselage profile and guide curves. Therefore, polynomial curve formulations, most commonly Bézier curves, B-splines or non-uniform rational B-splines (NURBS) are available in most systems to allow for a more flexible description of curves. NURBS curves and surfaces in particular, have become the norm for representing the OML of aircraft, as they are the most general of the three curve formulations. Comprehensive texts on NURBS theory are available, e.g. by Boor [Boo01], Farin [Far01], and Piegl and Tiller [PT96]. The following explanations of the theoretic foundations are primarily based on Farin [Far01].

In the following, some fundamental aspects of the three types of curves are discussed, which are relevant when trying to describe aircraft component geometry.

Bézier curves Bézier curves are constructed from a control point polygon using a Bernstein polynomial basis

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}, \tag{4.1.1}$$

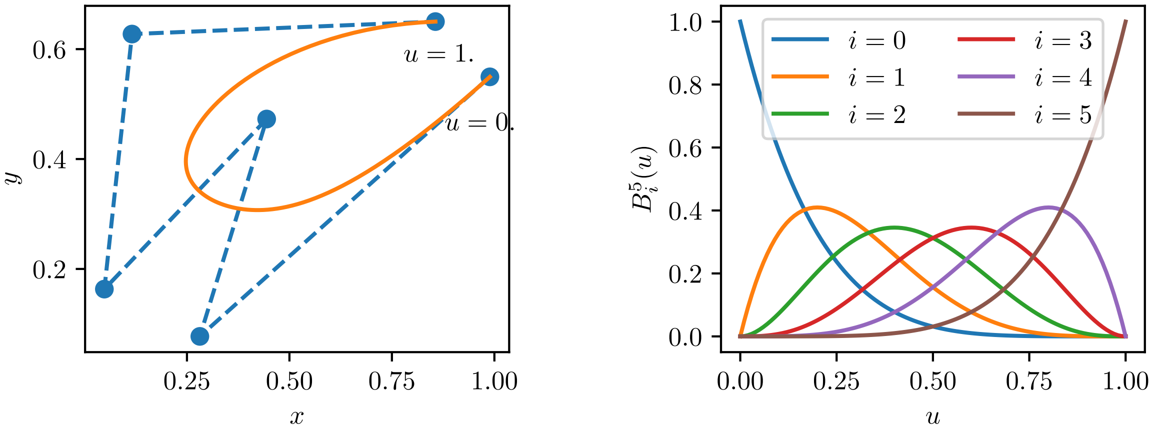
as illustrated by figure 4.1.1. The variable u is the running variable along the length of the curve, which is defined in the interval $[0, 1]$. The polynomial degree of the curve is prescribed by n. An important property of Bernstein polynomials and all subsequent basis functions, is that they form a *partition of unity*, i.e. the sum of all basis curves always amounts to one, as expressed by

$$\sum_{i=0}^{n} B_i^n(u) \equiv 1. \tag{4.1.2}$$

The curve $\mathbf{x}(u)$ is built by summing the BERNSTEIN polynomials for all $i \leq n$ multiplied by a corresponding control point $\mathbf{c}_i$:

$$\mathbf{x}(u) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(u)\,. \tag{4.1.3}$$

Equation 4.1.3 also illustrates the major weakness of BÉZIER curves. Since the number of basis functions to evaluate is directly coupled to the number of control points, the function becomes expensive to evaluate as curves become more complex and require more control points. Furthermore, the polynomial degree increases with the number of control points, which makes them more difficult to handle. On the other hand, only the control polygon is required to build a BÉZIER curve, since the degree is determined by the number of control points.



(a) BÉZIER curve

(b) BERNSTEIN basis functions

Figure 4.1.1.: BÉZIER curve and corresponding BERNSTEIN basis functions for example control points

Many applications, such as visualization, require the evaluation of the curve at a discrete number of sample positions $\mathbf{u} = [u_1, ..., u_m]$. In this case, the evaluation of the BÉZIER curve as given in equation 4.1.3 can be reduced to the scalar product

$$\mathbf{x}(\mathbf{u}) = \mathbf{B}(\mathbf{u}) \cdot \mathbf{c}, \tag{4.1.4}$$

where $\mathbf{c}$ is a matrix containing the control point vector coordinate values and has the shape $n_{dim} \times n$ and the $n \times m$-matrix $\mathbf{B}(\mathbf{u})$ contains each polynomial basis evaluated at the positions $\mathbf{u}$. Consequently, the result $\mathbf{x}(\mathbf{u})$ is a vector of curve point vectors and has the shape $n_{dim} \times m$. It follows from equation 4.1.1 that $\mathbf{B}(\mathbf{u})$ is a full matrix for BÉZIER curves, making evaluations expensive for both large numbers of control points (i.e. high-order curves) and large numbers of sampling positions.

A useful property of BÉZIER curves is their *affine invariance property*. Affine maps are coordinate transformations, which, among other things, preserve collinearity, parallelism, convexity, and ratios of lengths and include commonly known translation, rotation, and scaling operations as well as combinations thereof. The affine invariance property says that, if an affine map is to be applied to a BÉZIER curve, it is sufficient to apply it to the control points.

Another interesting property, which can be exploited e.g. in intersection algorithms is the *convex hull property*. It says that a BÉZIER curve will always be contained within the convex hull of its control polyline. Therefore, if an object is outside of the convex hull, it is guaranteed not to touch the curve (in 2D). A related property in this context is the *variation diminishing property*, which states that for a curve in space, a plane will never intersect the curve more often than the control polyline.

Finally, the differentiability depends on the polynomial degree of the curve, and hence the number of control points n. A BÉZIER curve is always C^{n-1} continuous.

Non-rational B-splines The limitations of Bézier curves can be avoided in non-rational B-splines (often simply referred to as B-splines), by applying a piecewise polynomial basis function, where the polynomial degree is decoupled from the number of control points. This is accomplished using a recursive basis function:

$$N_l^n(u) = \frac{u - t_{l-1}}{t_{l+n+1} - t_{l-1}} N_l^{n-1}(u) + \frac{t_{l+n} - u}{t_{l+n} - t_l} N_{l+1}^{n-1}(u), \tag{4.1.5}$$

with

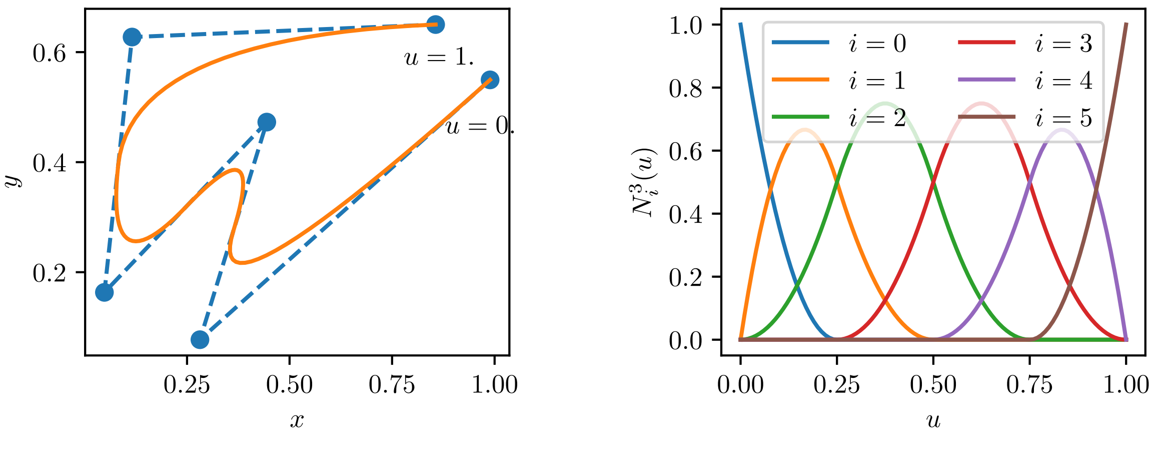
$$N_i^0(u) = \begin{cases} 1 & \text{if}\, t_i \le u < t_{i+1} \\ 0 & \text{else} \end{cases}. \tag{4.1.6}$$

Visibly, the piecewise polynomial approach requires an additional knot sequence $\mathbf{t}$ with the entries t_i to be provided, which defines the intervals in which the polynomials are defined. Furthermore, n is once again the polynomial degree of the curve, while l is the number of control points.

Using this basis, the curves are computed in a similar way as the Bézier curves in equation 4.1.3 using

$$\mathbf{x}(u) = \sum_{i=0}^{l+n+1} \mathbf{c}_i N_i^n(u), \tag{4.1.7}$$

where $\mathbf{c}_i$ is the i-th point in the control polygon. Consequently, unlike Bézier curves, the control polygon $\mathbf{c}$ is not sufficient to describe a B-spline. Instead, the polynomial degree n must be stated explicitly as well as the internal knot sequence $\mathbf{t}$. A cubic B-spline curve using the same control points as the Bézier curve in figure 4.1.1 is provided in figure 4.1.2, along with the basis functions. Just like the Bernstein basis of the Bézier curve, the B-spline basis functions form a partition of unity. However, each basis function covers only a subset of the full range of t. As a result modifications of individual control points of a B-spline curve only have a localized effect. Curve segments that lie sufficiently far away from the control point, will remain unchanged. This is referred to as a *localized modification scheme.*



(a) B-spline curve

(b) B-spline basis functions

Figure 4.1.2.: B-spline curve and corresponding basis functions for example control points

Like for Bézier curves, the evaluation of the B-spline curve at the m discrete sample positions $\mathbf{u}$ can be interpreted as a scalar product given by

$$\mathbf{x}(\mathbf{u}) = \mathbf{N}(\mathbf{u}) \cdot \mathbf{c}. \tag{4.1.8}$$

Following equation 4.1.5 the matrix $\mathbf{N}(\mathbf{u})$ has the shape $l + n + 1 \times m$. Unlike $\mathbf{B}(\mathbf{u})$, which is a full matrix, $\mathbf{N}(\mathbf{u})$ is a band matrix due to the localized modification scheme.

A Bézier curve can be understood to be a special case of a B-spline curve, where $n = l$, with $2(n+1)$ knots and half of the knots clamped at each end respectively. Consequently, B-spline curves share many of the properties of Bézier curves, such as affine invariance and convex hull. However, it is also possible in B-splines for a value to appear multiple times in the knot sequence $\mathbf{t}$. This is expressed by the knot multiplicity k and can be used to introduce breaks in continuity into the curve. The curve will be only C^{n-k} continuous at the knot of multiplicity k. This makes it possible to model smooth curves with discontinuities or corners. The first and last knot of a clamped non-periodic B-spline curve as shown in figure 4.1.2 always have the multiplicity $n+1$.

NURBS curves B-splines are a powerful tool to efficiently describe a large variety of shapes. However, conic sections cannot be described exactly. To this end, NURBS are commonly available in CAGD environments.

As explained e.g. by Farin [Far01], a conic section in $\mathbb{E}^2$ can be understood as the projection of a quadratic parabola in $\mathbb{E}^3$ to a plane using

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}. \tag{4.1.9}$$

Applying the same rule, $\begin{bmatrix} wx & wy & w \end{bmatrix}^T$ will project to $\begin{bmatrix} x & y \end{bmatrix}^T$, where w is an additional weight factor. This illustrates how B-spline or Bézier curves can be modified to describe conics by introducing an additional weight per control point. The vector $\begin{bmatrix} wx & wy & w \end{bmatrix}^T$ is referred to as the non-rational projection of the NURBS curve. As such, this type of curve can be handled like a common non-rational B-spline. To evaluate the rational curve, the projection to rational space must be performed, which yields

$$\mathbf{x}(u) = \frac{\sum_{i=0}^{l+n+1} w_i \mathbf{c}_i N_i^n(u)}{\sum_{i=0}^{l+n+1} w_i N_i^n(u)}. \tag{4.1.10}$$

Figure 4.1.3 provides an example of a circle described using a NURBS curve.

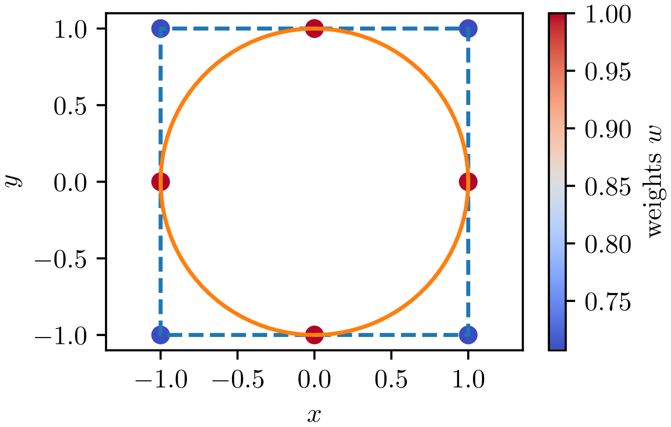


Figure 4.1.3.: Circle represented by a NURBS curve

Spline curve interpolation techniques Defining the curves via a control polyline has several convenient mathematical implications, which have already been addressed. However, it is not helpful for practical engineering applications, as engineers are usually more concerned with points, which lie on the actual curve. Spline interpolation techniques are helpful in determining a spline curve, which passes through a given set of points. The approach is demonstrated here for non-rational B-spline curves, but can analogously be applied to Bézier and NURBS curves.

Essentially, the problem of B-spline interpolation is the problem of finding the control polygon based on a given sequence of points by solving equation 4.1.8. The challenge is to choose the sample parameter vector $\mathbf{u}$, the knot vector $\mathbf{t}$ and the polynomial degree n in such a way that $\mathbf{N}(\mathbf{u})$ becomes a square matrix, which can be inverted. Typically, the polynomial degree is prescribed by the user before the interpolation, leaving the determination of the vectors $\mathbf{u}$ and $\mathbf{t}$.

The length of the sample point vector $\mathbf{u}$ is determined by the number of points to be interpolated m. The distribution of the values of $\mathbf{u}$ within the value range of the curve can, however, be chosen freely. Common approaches include the *uniform*, *chord length* and *centripetal* method. Lee [Lee89] proposes a unified formulation for all three approaches:

$$u_i - u_{i-1} = \frac{|\mathbf{x}_i - \mathbf{x}_{i-1}|^e}{\sum_{j=1}^{m} |\mathbf{x}_j - \mathbf{x}_{j-1}|^e}, 1 \leq i \leq m. \tag{4.1.11}$$

For the chord length method, the exponent is set to $e = 1$. This means that the parameter difference for a segment between two points is equivalent to the distance between them, i.e. the chord length of the segment, divided by the summed lengths between all point pairs. For the uniform method, the exponent is set to $e = 0$, which means each chord length evaluates to 1. Therefore, the parameter difference is the same for each segment. For the centripetal method, the exponent is chosen to be $e = 0.5$.

From the resulting vector $\mathbf{u}$ the knot vector $\mathbf{t}$ can be computed using a moving average approach:

$$t_{j+n} = \frac{1}{l} \sum_{i=j}^{j+n-1} u_i, j = 1, 2, ..., m - n. \tag{4.1.12}$$

End knots must be appended for open B-splines.

Figure 4.1.4 shows the interpolation results for a simple example. Comparing the curve segment lengths to their corresponding chord lengths, the chord length method is biased towards longer segments, whereas the uniform method is biased towards shorter segments. The centripetal method seems to provide a good compromise between the two in many cases.

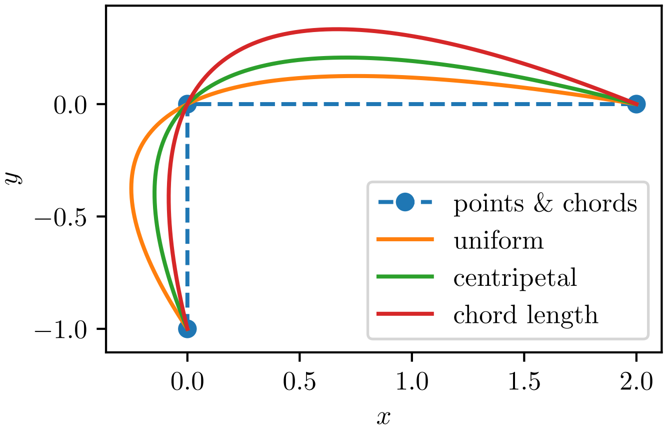


Figure 4.1.4.: Interpolation results using different parametrization methods

A drawback of the interpolation approach is that the number of control points of the resulting curve is coupled to the number of input points. Therefore, a high number of input points will result in a curve that is expensive to evaluate. Instead of an interpolation, a least-squares approximation can be performed, e.g. by solving

$$\mathbf{N}(\mathbf{u})^T \mathbf{x}(\mathbf{u}) = \mathbf{N}(\mathbf{u})^T \mathbf{N}(\mathbf{u}) \cdot \mathbf{c} \tag{4.1.13}$$

for $\mathbf{c}$. Here, $\mathbf{N}(\mathbf{u})$ must no longer be square and the desired number of control points n_c can be specified by the user, independently from the number of input points. Compared to the interpolation approach, this also changes the way the knot vector is determined. Instead of the moving average approach given in equation 4.1.12, an evenly spaced distribution can be assumed.

Class function / Shape function Transformation (CST) curves While accurate knowledge of the product shape is essential especially in later development stages, e.g. for manufacturing, in the earlier stages it can be easier to describe aircraft geometry using just a few top-level parameters, such as wing span or fuselage diameter. These parameters are also substantially more meaningful to the design engineer than e.g. NURBS control points and weights and better suited as design variables for MDO.

The CST introduced by Kulfan [Kul08], represents an attempt to provide a formulation, which combines an accurate geometric curve description with a meaningful parametrization for aerodynamic shape optimization. It is used to describe both wing and fuselage profiles ζ using

$$\zeta\left(\psi\right) = C_{N_2}^{N_1}\left(\psi\right) S\left(\psi\right) + \psi\zeta_T, \tag{4.1.14}$$

where $C_{N_2}^{N_1}\left(\psi\right)$ is the class function, $S\left(\psi\right)$ is the shape function and ζ_T is the trailing edge thickness ratio. The class function is defined as

$$C_{N_2}^{N_1}\left(\psi\right) = \psi^{N_1}\left(1-\psi\right)^{N_2}. \tag{4.1.15}$$

The exponents $N1$ and $N2$ can be used to manipulate the class of the profile. For instance, $N_1 = 0.5$ and $N_2 = 1.0$ are used for round-nose airfoils and $N_1 = 0.5$ and $N_2 = 0.5$ for elliptic profiles. The shape function has the form

$$S\left(\psi\right) = \sum_{i=0}^{n} A_i S_i\left(\psi\right), \tag{4.1.16}$$

which is exactly the same as for Bézier curves or B-splines. Kulfan proposes the Bernstein polynomial basis from equation 4.1.1 to represent the shape function, which then effectively amounts to a one-dimensional Bézier curve. However, the overall shape of the curve is also modified by the class function. Therefore the control points coordinate values A_i are considered coefficients, which can be used to modify the profile. For instance, using a second degree Bernstein polynomial would provide three shape functions with three corresponding coefficients. The first shape function is dominant in the forward part of the profile, the second function controls the middle, and the third one the trailing edge. It follows that the respective coefficients control the corresponding sections of the profile.

The class function, on the other hand, enforces the boundary conditions e.g. a round leading edge and a sharp trailing edge for $N_1 = 0.5$ and $N_2 = 1.0$. In this case, the first coefficient affects the radius of the leading edge, the second coefficient affects the thickness and the third one affects the trailing edge tangent, providing more targeted design variables for optimization. Therefore, the CST formulation provides an interesting example of how the complex curve formulations outlined above can be related to more relevant engineering parameters. That said, the actual coefficient values A_i are usually computed numerically and are difficult to interpret by themselves.

In addition to aerodynamic profiles, Kulfan proposes the application of CST curves to describe fuselage cross sections, as shown in figure 4.1.5. Moreover, Jonge [Jon17] provides support for CST cross-sections in ParaFuse. However, problems arise when trying to represent CST curves as NURBS curves, e.g. to visualize the curves in CAD environments, which do not usually support CST curves. Marshall [Mar13b] shows that an exact representation of CST curves using NURBS is possible for the most common, but not all combinations of exponents. A prerequisite for Marshall's method is a pointed trailing edge, i.e. $N_2 = 1.0$. This is not the case for the fuselage profiles proposed by Kulfan. Consequently, Jonge applies a B-spline fitting technique to implement CST profiles in ParaFuse, albeit without providing further details.

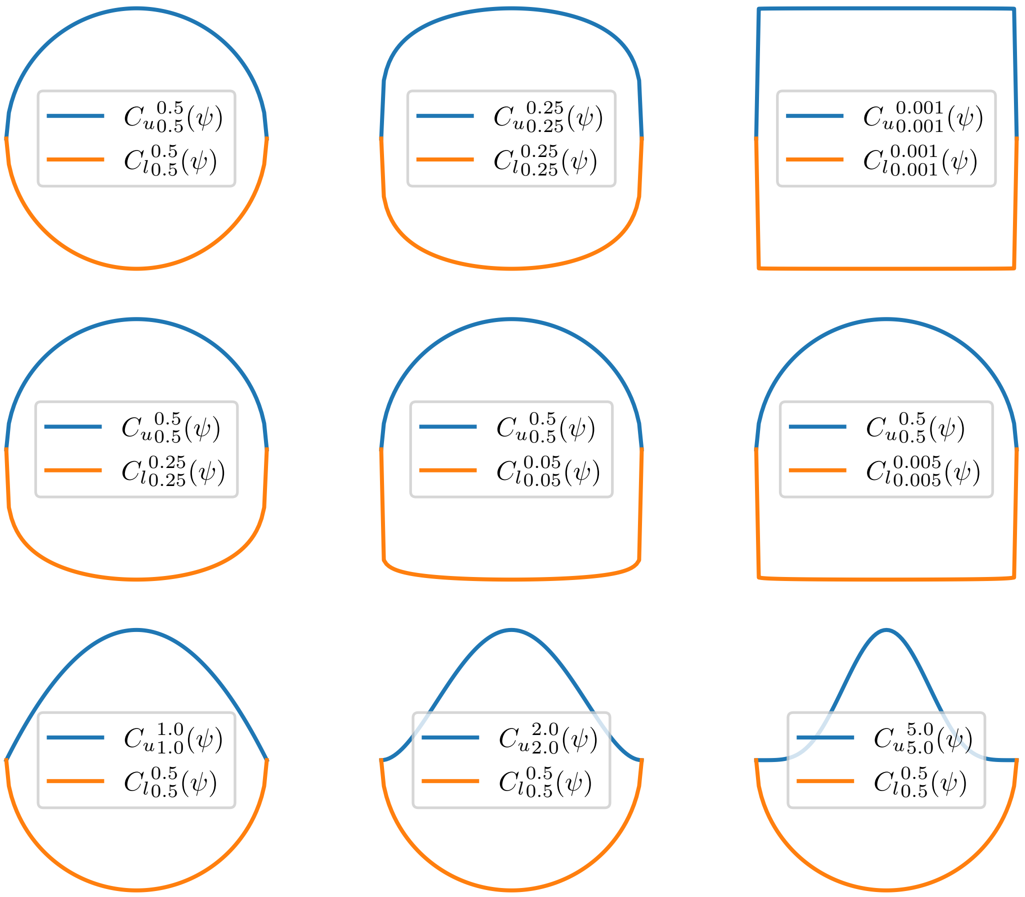


Figure 4.1.5.: CST representations for different fuselage profiles (after [Kul08])

4.1.1.2. Spline-based description of surfaces and volumes

So far, the discussion was focused on univariate curves $\mathbf{x}(u)$. However, the curve descriptions can easily be transferred to bivariate surfaces $\mathbf{x}(u, v)$. Analogously to analytical curves, analytical surfaces, such as planes, cylindrical or spherical surfaces are also available in most CAGD environments.

With regards to spline curves, an additional coordinate direction v must be introduced to describe surfaces. The topologically one-dimensional control point polygon is extended to a topologically two-dimensional control point grid, as shown in figure 4.1.6. Furthermore, the basis functions need to be evaluated twice, once for each topological direction. For the case of a B-spline surface, this results in

$$\mathbf{x}\left(u, v\right) = \sum_{i=0}^{l_1+n_1+1} \sum_{j=0}^{l_2+n_2+1} \mathbf{c}_{ij} N_i^{n_1}\left(u\right) N_j^{n_2}\left(v\right). \tag{4.1.17}$$

The basis functions, i.e. the polynomial degree n_i and the knot vector $\mathbf{t}_i$ can be selected independently for each topological direction. In this way, it is possible to describe surfaces, which are defined e.g. by a cubic polynomial in one direction and a linear polynomial in the other.

Several techniques for building surfaces from points or curves have been proposed [PT96]. Some relevant examples are listed in figure 4.1.7 and briefly discussed in the following.

Bivariate point grid interpolation (figure 4.1.7a) Bivariate grid point interpolation is an extension of the curve interpolation approach in two dimensions. Instead of a polyline of points, a point grid is required as input. Simply put, the curve interpolation is performed twice, once for each direction of the point grid.

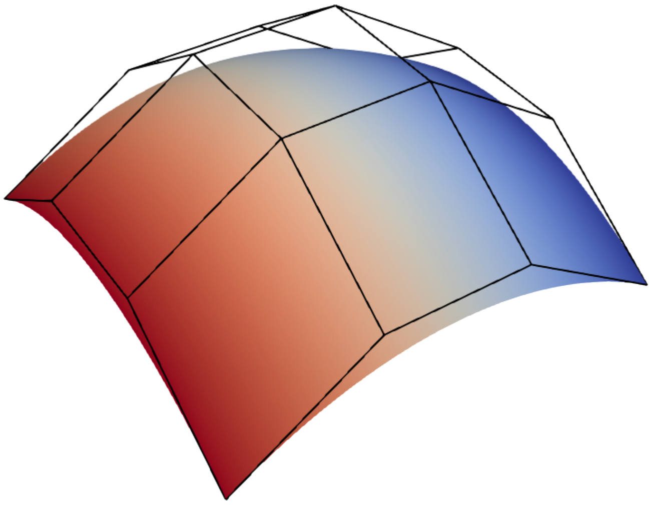


Figure 4.1.6.: B-spline surface with corresponding control point grid

Point cloud approximation (figure 4.1.7b) Instead of a structured point grid, a surface can also be constructed from an unstructured point cloud in 3D. In this case, a principal component analysis must first be performed, to define the topological directions of the surface and estimate the uv-coordinates of the points. A specific number of points must be provided to perform an interpolating fit of the point cloud, which depends on the assumptions made for the surface. Therefore, it is usually more useful in many cases to perform an approximation instead. The resulting surfaces have a tendency to run off towards infinity outside the bounds of the point cloud.

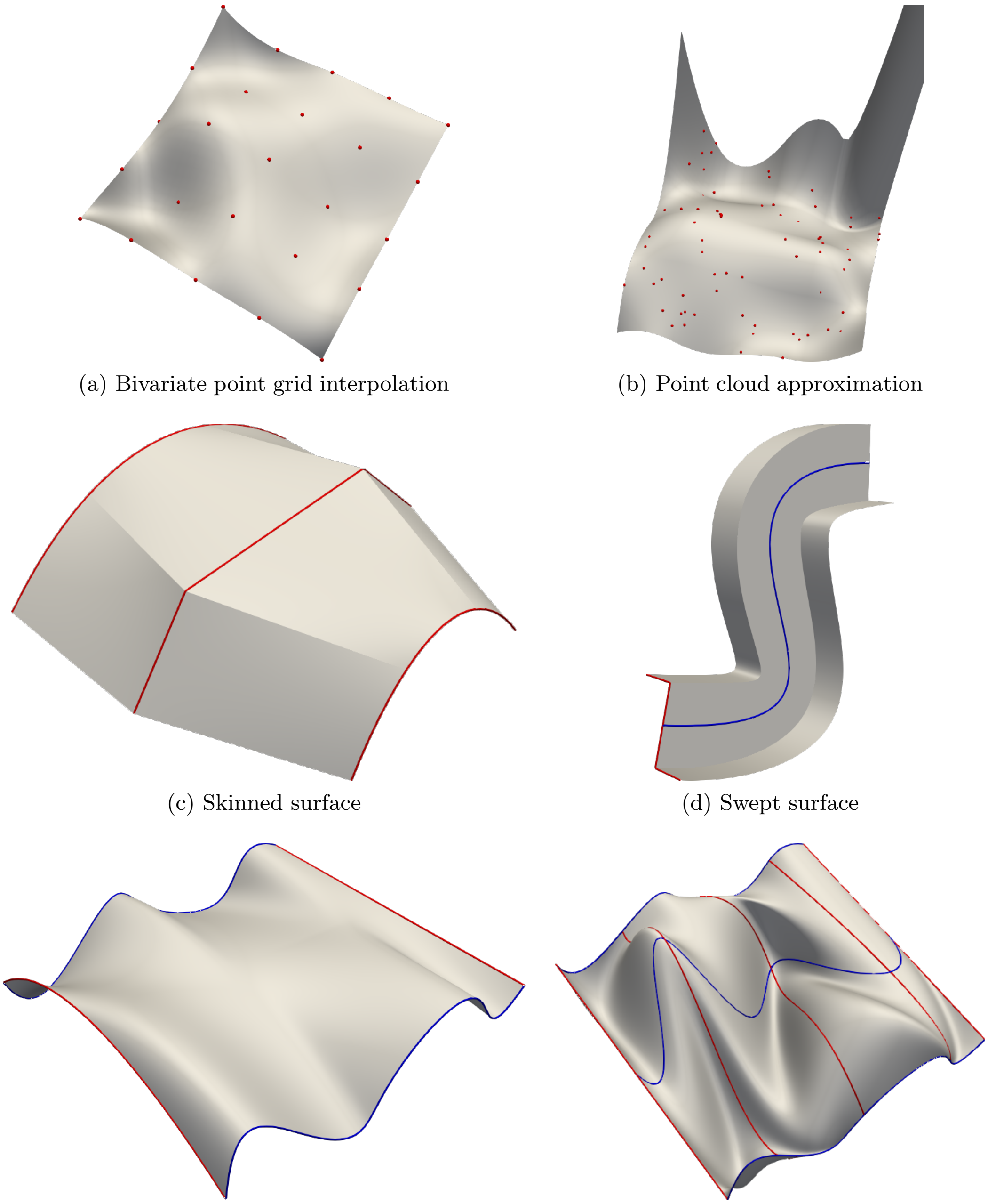
Skinned surface (figure 4.1.7c) A surface is generated by an interpolation between a series of curves. A challenge here is that the curves must be made compatible, i.e. they all need to share the same polynomial order n and knot vector $\mathbf{t}$. This requires application of advanced spline manipulation techniques like degree elevation and knot insertion.

Swept surface (figure 4.1.7d) A surface is generated by extruding a profile curve along a trajectory curve. This is essentially a special case of the skinned surface algorithm, which is applied to transformed copies of the profile curve. The translations are computed based on the trajectory curve. It is also possible to manipulate the orientations of the translated profiles.

Coons patch (figure 4.1.7e) A Coons patch is defined by four surrounding curves. The surface is assembled by summing the linear skinned surfaces between the two pairs of opposite curves and subtracting the bilinear patch defined by the corner points. Advanced implementations also support the prescription of gradient distributions along the edges of the patch.

Gordon surface (figure 4.1.7f) A Gordon surface is an extension of the Coons patch, which operates on a topologically structured curve network. Instead of computing a surface only between two opposite curves, a skinned surface is computed for each of the two surface directions based on a sequence of curves. The patch, which is subtracted must contain all curve intersection points in the curve network. In this context, Siggel et al. [SK+19] provide some further background on the practical challenges to overcome when building compatible curve networks. A big advantage of Gordon surfaces over Coons patches is that continuity over shared edges between adjacent cells is ensured without additional effort.

Analogously to equation 4.1.17, trivariate volumes can be created by further expanding the control point grid from 2D to 3D and adding further topological directions [FH85]. An example for a trivariate



(a) Bivariate point grid interpolation

(b) Point cloud approximation

(c) Skinned surface

(d) Swept surface

(e) Coons patch

(f) Gordon surface

Figure 4.1.7.: Spline surface construction techniques

B-spline volume is shown in figure 4.1.8. Whereas such volumes are rarely used in CAD environment, they are commonly applied in FFD schemes, as described in section 2.2.3.

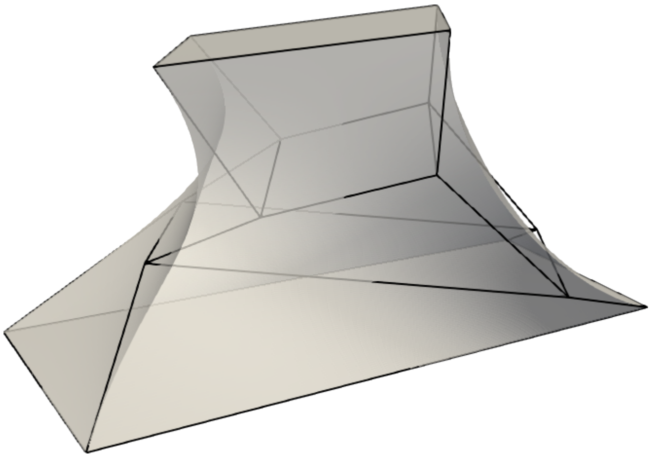


Figure 4.1.8.: B-spline volume and corresponding control point grid

4.1.1.3. Advanced CAGD modeling concepts based on tree structures

Building upon the previously established concepts of geometric curves and surfaces as well as analytically defined primitives, such as ellipsoids, boxes or cones, more advanced geometry description techniques can be applied as described e.g. by Hoffmann [Hof89]. The common underlying idea is to express shapes in terms of combinations of simpler shapes. The relationships can be stored in a tree structure, where the final shape is the root and its source components are the leaves.

Boundary representation (B-rep) As mentioned previously, trivariate volume descriptions are rarely found in CAGD systems. Instead, a boundary representation (B-rep) approach is often adopted, where volumes, also referred to as solids, are defined by their bounding surfaces. To differentiate between mathematical surface descriptions, as discussed in the previous section, and bounding surfaces in B-rep, the latter are often referred to as faces. Analogously to solids, faces can be represented by their bounding edges. A complete overview of the B-rep components implemented in the OCCT kernel is given in table 4.1.1. As illustrated by figure 4.1.9, the B-rep formulation results in a hierarchical structure, where a solid is described in terms of a collection of faces, each of which is in turn defined by a collection of edges.

Table 4.1.1.: Terminology of B-rep compared to geometry as implemented in OCCT [OCC23a]

Dim.	Geometry	B-rep	Comment
0D	Point	Vertex	
1D	Curve	Edge	An edge may be bounded by vertices
		Wire	A wire is composed of multiple edges
2D	Surface	Face	A face may be bounded by a closed wire
		Shell	A shell is composed of multiple faces
3D	Volume	Solid	A solid may be bounded by a closed shell
		Compound solid	A compound solid is composed of multiple solids

Constructive solid modeling (CSM) Volumetric solids provide the foundation for the constructive solid modeling (CSM) approach. In CSM, geometry is described in terms of Boolean operations on geometric sets. The Boolean operations include union ($\cup$), intersection ($\cap$) and difference ($-$).

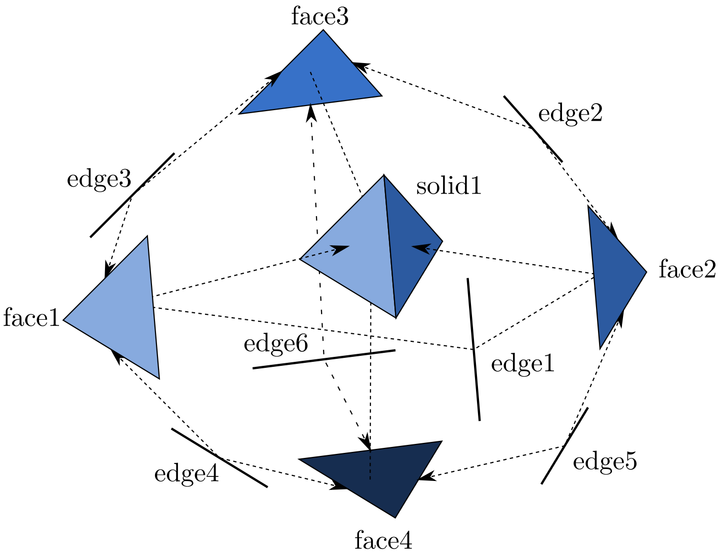


Figure 4.1.9.: B-rep representation of a tetrahedron (after [Hof89])

Using these operations, solids can be combined recursively, resulting in a tree structure as illustrated by figure 4.1.10. To a certain extent, the approach can furthermore be extended to simpler shape types e.g. faces and edges. CSM capabilities can be found in any modern CAGD framework. To facilitate the use of CSM, most vendors furthermore provide predefined parametric primitives, such as boxes, spheres, cylinders or cones. Schmidt [Sch12] proposes a formalization based on the Unified Modeling Language (UML) for a number of primitives and subsequently demonstrates CSM capabilities, with the goal to connect those to superordinate design processes. Based on ESP, Dannenhoffer [Dan13] introduces the OpenCSM framework, which has been designed to provide built-in support for geometric sensitivities for gradient-based MDO. It also includes a dedicated language to formulate CSM trees.

The mathematical implementation of these operations depends on the underlying geometry. For Boolean operations involving polynomial curves or surfaces, it is necessary to apply iterative numerical methods [PM10], which makes them computationally expensive and potentially unstable. In addition to the numerical aspects, both Hoffmann [Hof89] and Dannenhoffer and Haimes [DH15] point out the difficulty to properly identify and react to changes in topology when applying Boolean operations to parametric models.

While B-rep and CSM are related in that they are both based on tree formulations, the two types of trees must be kept separate. For instance, a difference operation, as shown for the sphere and the cube in figure 4.1.10, accepts two B-rep trees as input, from which the intersection curves are computed. A new B-rep tree is provided as output, which contains elements of both input trees, but is effectively a new copy.

Parametric feature trees Parametric feature trees, which are very common especially in interactive CAD environments, provide a generalization of the tree approach in B-rep and CSM linking them to more general engineering parameters. They allow the user to combine custom defined parameters using a set of operations. These include generative operations to build B-rep shapes or primitives from scratch, such as rotations, extrusions and offsets, but also affine transformations. Boolean operations can also be included, enabling CSM capability.

To illustrate, the simple example of a thick-walled cylindrical barrel as shown in figure 4.1.11 is considered. It is defined by its length l_{barrel}, inner diameter d_{inner} and wall thickness t_{barrel}. Two possible feature trees to assemble the geometry are given in figure 4.1.12. On the one hand, a rectangle with the length l_{barrel} and the width t_{barrel} can be created. It is then moved the distance d_{barrel} away from the rotation axis around which it is subsequently extruded into a rotational body. The second approach leverages CSM by creating two cylinders of the length l_{barrel}. The inner cylinder has the

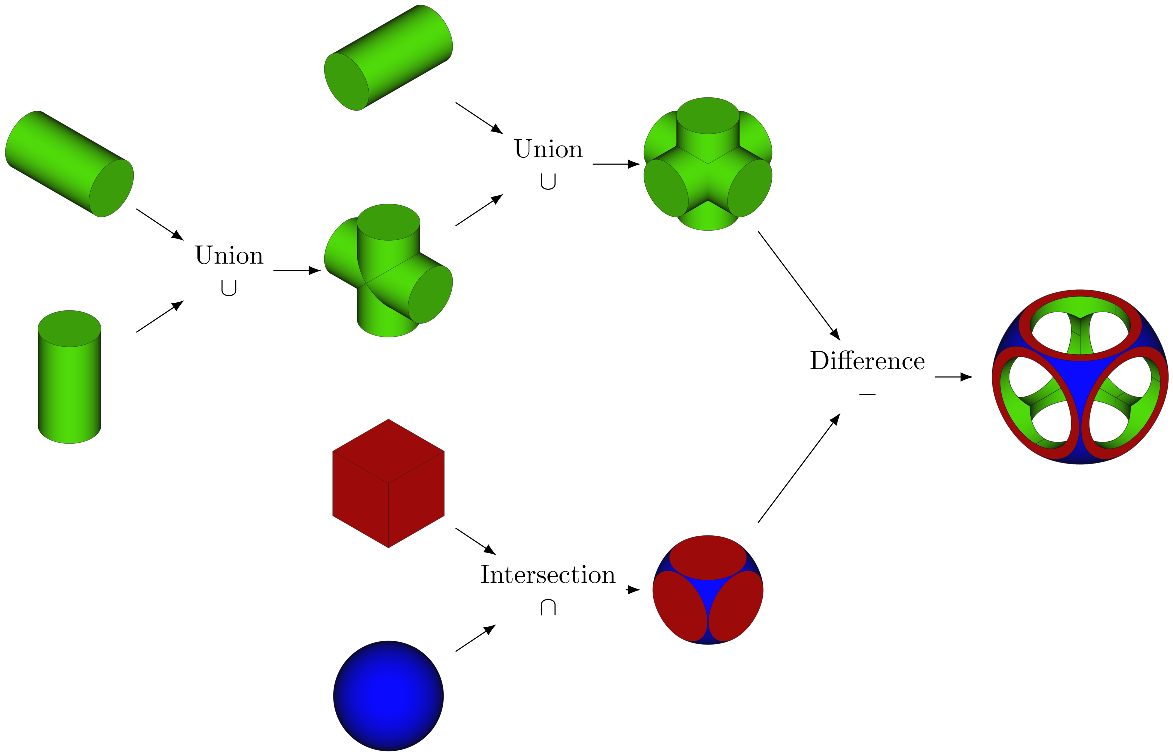


Figure 4.1.10.: CSM tree for a shape built using Boolean operations (after Wassermann et al. [WB+16])

diameter d_{inner}, whereas the outer one has the diameter $d_{outer} = d_{inner} + 2t_{barrel}$. The barrel geometry is then simply created by subtracting the inner cylinder from the outer. The example exemplifies that there is rarely one singular feature tree to describe a given geometry.

Another more detailed example is given by Dannenhoffer and Haimes [DH15], who use a feature tree to describe a bolt.

Dannenhoffer [Dan13] furthermore lists drawbacks of the feature tree implementations in prevalent CAD applications. This includes a very limited and proprietary language for the formulation of build recipes. As a result, the number of parameters is usually kept low and the intended use is limited to combining predefined primitives, resulting in a high modeling bias. This means that changes in the fundamental product architecture, e.g. adopting a non-circular cross-section will require significant changes in the model. Due to these drawbacks, in addition to missing support for derivative computation and license cost, Dannenhoffer opts to introduce a new description language for feature trees.

4.1.1.4. Review of CAD environments and data standards

The market for CAD solutions is vast, with an ever growing number of vendors and products. To compile a comprehensive review of all available solutions would exceed the scope of this thesis. However, it is worth to highlight a few preeminent CAD solutions in various fields connected to this thesis. In addition, a brief overview of CAD file formats is given.

Notably, the scope of many solutions exceeds the purely geometric product modeling, which is why the more specific acronym CAGD is nowadays largely abandoned in favor of the more general CAD.

CATIA / 3DEXPERIENCE In aerospace, CATIA by Dassault Systèmes [3DS23b] remains the standard solution for OEMs. It provides an interactive user interface and a vast library of tools specific

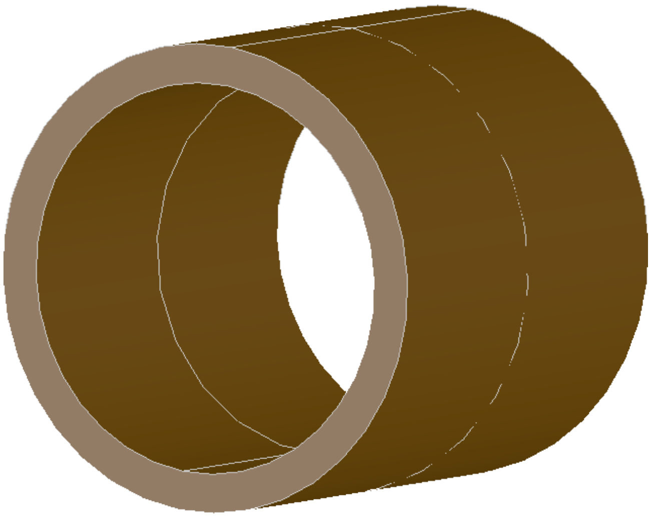
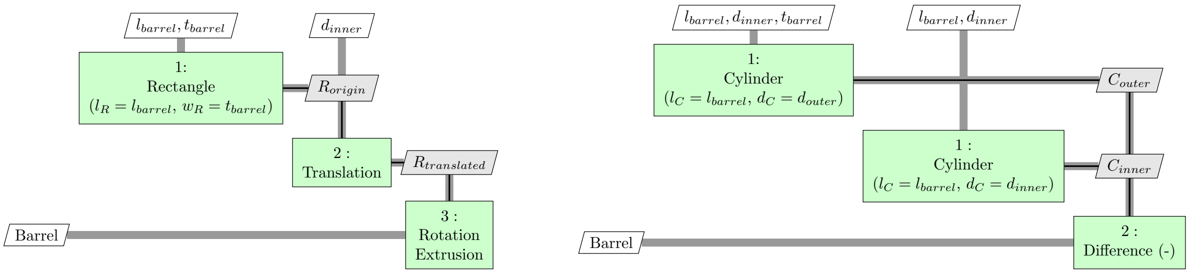


Figure 4.1.11.: Thick-walled barrel example geometry



(a) Rectangle rotation

(b) Cylinder difference

Figure 4.1.12.: Possible feature trees for a thick-walled barrel in XDSM notation

to aerospace applications. Automation of CATIA is possible using either the COM-interface, a VBA macro interface, or the built-in Engineering Knowledge Language (EKL). EKL can also be used to formulate moderately advanced relationships within the model, including loops and if/else constructs. Segments of the feature tree can be re-used using the Power Copy or user-defined feature (UDF) capabilities, which can be compared to high-level primitives in KBE (s. section 4.3.2.4). Knowledge Patterns can be applied to create and manage multiple instances of a Power Copy or UDF controlled by EKL code. That said, many capabilities for CATIA can only be accessed by purchasing additional tool kits. Furthermore, code must typically be written in the user interface (UI) and is not externally accessible, which results in a cumbersome development process and inhibits application of commonly accepted best practices in modern software engineering.

Recent versions of CATIA are distributed as part of the 3DEXPERIENCE platform [3DS23c]. Whereas the CAD functionality is largely the same as in older versions, data is no longer stored locally in the file system, but in a central product lifecycle management (PLM) server. The proprietary centralized data management solution facilitates the interaction with other tools inside the platform, but further impedes the accessibility via third-party tools.

Rhinoceros 3D A popular solution in the field of 3D computer graphics is Rhinoceros 3D [McN23]. Compared to CATIA, Rhinoceros places a stronger emphasis on interactive NURBS-based surface modeling. A Python-based visual programming environment called Grasshopper is available, promising rapid setup of parametric modeling pipelines by domain experts with minimal coding skills required.

The visual programming approach is very similar to the PIDO frameworks used for MDAO workflow integration (cf. section 2.2.4). As such, it is particularly well-suited for small, easily comprehensible example problems, but does not adapt well to the high level of complexity commonly found in real-world applications, as indicated by Kirchhof et al. [KJ+23].

Open CASCADE Technology Finally, the Open CASCADE Technology (OCCT) library [OCC23a] provides an open source implementation of a CAD kernel. Aside from being free of charge, OCCT has the advantage that it provides access to functionality at a very low level, which comes at the expen of a steeper learning curve compared to the commercial tools. Projects like FreeCAD, where a user interface is implemented on top of the OCCT kernel, aim to improve accessibility and provide an open source alternative to commercial CAD solutions. OCCT is written in the C++ programming language, but bindings for other languages such as Python are available [Pav20].

File formats All of the above CAD tools have their own dedicated file formats, which are designed to support a large subset of the feature set of the respective application. For instance, CATIA V5 provides the CATPart and CATProduct format, which can be used to store not only the geometry, but also the feature trees. Analogously, Rhinoceros 3D provides the *Rhino 3D model* (3dm) format and FreeCAD the *FreeCAD Document* (FCStd) format. OCCT provides a lower level geometry-only binary format called BRep.

The mentioned formats tend to work well with the tool for which they have been designed, but interoperability is usually a problem. Interfaces for import and export are either not available at all or support only very small subsets of a given format's feature set. The need for a common exchange format for geometry has led to the introduction of the *Initial Graphics Exchange Specification* (IGES) format, which provides a standardized definition of product geometry. It was later superseded by the *Standard for the Exchange of Product model data* (STEP), which is the current ISO standard for product model data exchange [ISO21]. Compared to IGES, STEP supports a larger set of product data, including some metadata. However, a common drawback of STEP and IGES is the lack of support e.g. for feature trees and parametric relationships. Furthermore, both are text formats by default, rather than binary formats, which results in slow import and export operations.

4.1.2. Data-centric product model

In the previous section, the unambiguous mathematical description of the geometry or shape of the product is presented as one possible way of sharing engineering data in a collaborative process. However, in aircraft design, significant amounts of non-geometric data as diverse as mass properties, mission profiles, material properties and engine performance maps are necessary to perform a proper analysis.

Modern CAD systems provide support for metadata connected to the geometry, such as component group affiliation or assigned materials. Furthermore, visualization settings can be stored using common exchange formats like STEP. Nevertheless, the sheer amount and heterogeneity of data required effectively prohibits its inclusion into the CAD model. Moreover, the geometry description using e.g. NURBS curve formulations, though mathematically accurate, rarely provides parameters, which correspond to the intuition of a design engineer or are expressive for an optimization. Consequently, CAD-based geometry must be considered a necessary, but insufficient part of the digital product description for the aircraft design process. This is reflected in **working hypothesis 1working hypothesis 1**, where a description based on parameters is explicitly called for.

Consequently, the evolution of data-centric product models for aircraft design is discussed in the following, which aim to provide a more meaningful product description compared to a CAD model.

Design data input In its most basic form, a data-centric product model is a way of providing all necessary input data to a design tool. This is usually a list containing a limited set of design parameters. The tool then takes care of the distribution of the parameters and provides the desired outputs, which could range from an estimate of the mass, block fuel, or direct operating cost (DOC) to a detailed CAD model. To the tool, the design parameter list provides a sufficiently complete description of the product, since any other necessary information is computed using the implemented routines.

Some design tools, especially with an educational background e.g. OpenVSP or PADLab, also provide graphical user interfaces, which allow interactive modification of parameters. The set of parameters made available by the GUI effectively represents a parameter list as well.

Design data serialization As mentioned in section 2.1.1, most aircraft design frameworks consist of an iterative loop, where new information about the product becomes available successively. This means that the values of certain design parameters become available only later in the process. Therefore, the design parameters are managed in memory in a central database in tools like PrADO or PADLab.

The need for a structured and predictable way to store data permanently becomes apparent when implementing a feature for saving and loading design projects. As a result, developers have been forced to find ways to store product data using dedicated data structures or ontologies. The process of translating data structures or objects in memory to a form, which can be stored and reconstructed later is commonly referred to as serialization. PADLab uses a struct-type data structure from MATLAB as a product data container. The corresponding serialization tools are applied for saving and loading. In PrADO a data management system layer is provided, which is accessed by the individual design modules to manipulate the product data via an interface in FORTRAN. Both tools store the data in binary form, which has advantages in terms of performance, but results in files, which are not readable for humans. It also makes it difficult to access the data via third-party software, since the structure of the data is controlled only by the respective design tool.

To overcome these limitations Risse et al. [RA+12] propose the Aircraft Exchange (AiX) format, an aircraft definition schema based on the eXtensible Markup Language (XML)[W3C08]. XML has several technical advantages that make it well-suited for data exchange. First, XML files are formatted text files, which can be read and understood by humans. It also allows for information to be organized in a hierarchy, which facilitates the navigation of the data. Finally, it provides schema definition and validation capabilities, allowing for prescription and enforcement of the structure of the data set. This makes it possible to implement software in adherence to the schema, which will function with any valid data set.

AiX has been developed as a container format for the MICADO conceptual design tool. As such, the purpose of the format is to contain the inputs i.e. requirements and specifications on the one hand, and the product definition, including geometry, masses, accommodation etc. on the other. However, Schültke et al. [SA+21] indicate that MICADO was conceived as a standalone-framework, implying that AiX is not intended for use outside of MICADO. Instead they point to CPACS as an exchange format for outward compatibility, which is discussed in the following.

Design data exchange The Common Parametric Aircraft Configuration Schema (CPACS) [LH11] is also based on XML and thus shares the technical basis with AiX. However, the ambition for CPACS, first stated by Nagel et al. [NB+12], is fundamentally different. Instead of merely providing a product data schema to support a single closed framework, CPACS is designed to be a "common language" for aircraft design. As such, it must be capable of providing data to a large variety of analysis and design tools. Therefore, disciplinary analysis tool providers have been actively involved in the development process. The initial geometry definition was proposed by Liersch and Hepperle [LH11], who not only provide ways to describe e.g. the OML, but also necessary design requirements e.g. mission definitions. The definition of wing structures is outlined by Dorbath, Nagel, and Gollnick [DNG11]. Scherer and Kohlgrüber [SK16] later contributed a description schema for fuselage structures, whereas Fuchte, Gollnick, and Nagel [FGN13] contributed a first cabin definition, which has recently been revised by Walther et al. [WH+22a]. In this way, product descriptions for all common disciplines in aircraft design have been included in CPACS.

It is worth noting that CPACS – or any product data model – is only a container for the exchange of product information. The actual knowledge on how to create this information must be provided via disciplinary analysis and design capabilities. In this context, the hierarchical structure of CPACS is exploited by storing more detailed data on deeper levels of the hierarchy [Zil13]. On the one hand, this enables an evaluation of the design on various levels of fidelity, which is discussed in depth by Böhnke [Böh15], allowing for successively higher fidelity analyses to add more and more design details. On the other hand, other authors have applied CPACS for analyses at air transportation system level [GS+11; KL+11].

Alder et al. [AM+20] provide a good overview of projects, where CPACS has been used successfully in the past as well as perspectives for further development. Remarkably, many of the challenges mentioned are not related to the technical definitions in CPACS, but to the supporting processes and ecosystem. Despite extensive documentation being available [DLR23], the aspect of training has been cited as a particularly important prerequisite to lower the entry barrier to a format as complex as CPACS.

Towards design knowledge exchange Increasing interest can be noted in even more advanced data modeling languages, such as the Unified Modeling Language (UML) [OMG17] and its derivatives like the System Modeling Language (SysML) [OMG19] as well as the Web Ontology Language (OWL) [W3C12]. They allow for storage and exchange of more complex relationships between pieces of information. UML in particular is well-known from object-oriented programming (OOP), where objects are described using classes with attributes (data) and methods (operations on the data). Relationships between classes can be established through interactions. In this way, the behavior of the product can be exchanged in addition to the structure. This combination is commonly referred to as design knowledge (s. section 4.3.1). SysML is a derivative of UML, which aims to make the language more suitable for complex systems engineering applications. OWL provides a modified understanding of classes and additional description capabilities, which enable semantic reasoning on the available data. This includes e.g. type inference. All languages can be serialized using XML syntax, however, the additional overhead due to the modeling language means a decrease in readability compared to plain XML.

Published applications include cabin systems modeling [Mot16; FH+21], manufacturing [PN19; MS+22] and geometry description [Sch12; ZZM20]. OWL in particular provides significant additional value w.r.t. XML. On the one hand, entries from one ontology are made available globally using an uniform resource identifier (URI) and can be referenced from any other ontology. The result is referred to as the semantic web [W3C23] and enables highly modular ontologies. The additional freedom comes at the cost of a loss of control over individual ontologies and thus an increased risk of inconsistent or duplicate definitions.

On the other hand, the type inferencing capabilities (s. also section 4.3) are a very powerful feature of OWL. Using proven reasoners, properties of types can be determined at run-time based on predefined rules using associated rule description languages like the Semantic Web Rule Language (SWRL), which permits very efficient modeling, compared to the aforementioned languages. However, this also means that the modeling procedure is significantly more complex, thus exacerbating the issue of the high entry barrier observed for CPACS.

Evaluation of suitability for collaborative fuselage design The most important formats described in the preceding paragraphs have been collected in table 4.1.2, in order to provide a comparison and make an assessment w.r.t. their applicability for collaborative fuselage design use cases. To this end, several criteria are identified to take into account requirements from both a software engineering and an aircraft design perspective. For the former, aspects such as portability between different systems as well as the availability of an open source library and a well-established long-term support are of particular interest. For the latter, the possibility to explore the raw data in a human readable format, a mature and stable data structure definition dedicated to the aircraft application case and a manageable complexity of the modeling are key issues.

Table 4.1.2.: Assessment of applicability of data formats for collaborative fuselage design

Storage format	PrADO parameter list	PADLab struct	CPACS	SysML/ UML	OWL
Human readable	+	−	+	o	o
Exchangeable	o	−	+	+	+
Open source library	−	−	+	o	o
Long term support	−	−	+	o	o
Maturity for aircraft design	+	+	+	−	−
Modeling complexity	+	o	o	−	−
Rule support	−	−	−	o	o

A clear distinction must be made between the first three columns, which contain dedicated data formats for aircraft description, and the latter two, which are better compared to a technology like XML. This means that they provide the means to describe a data model, but it is up to the user to implement it for the actual use case e.g. aircraft. As long as no mature model for aircraft design, similar to CPACS is available publicly, this rules out these technologies for the purpose of design data exchange in the scope of this thesis. Furthermore, the capabilities of the included rule languages like SWRL cannot match the versatility of an established general purpose programming language like Python for describing complex relationships found in engineering. This is likely to diminish the benefit of the languages builtin rule support, as the associated knowledge will likely need to be split between the model and an associated software library anyway.

Out of the remaining entries, CPACS emerges as the most suitable format, outperforming the PrADO and PADLab formats. Considering that CPACS has been designed explicitly for the purpose of data exchange, this is to be expected. Notably, some authors investigating modeling formats based on SysML or OWL [FH+21; MS+22] explicitly state that their developments are designed to be complementary to CPACS i.e. build upon basic product information provided via that format. This not only highlights the ubiquity of CPACS as an aircraft design data format, but also shows a development path, where the aircraft design know-how contained in CPACS can be leveraged along with the advantages of more modern knowledge modeling approaches.

4.1.3. Geometric interpretation via a parametric modeling engine

If geometry-centric models are necessary, but insufficient, to completely describe an aircraft in the design process, this is also true for data-centric models. Whereas the CAD descriptions provide unambiguous mathematical representations, most data-centric models are inherently vulnerable to misinterpretation. This is not a major problem, as long as the model is designed for use with a single program, since the correct rules for interpreting the parameters are already included in the program. For input data and storage models, this is usually the case.

The set of rules to translate the parameters to CAD models is referred to as a parametric modeling engine. Implementations of parametric modeling engines of varying degrees of sophistication are found in most OAD environments. For example, interfaces to CATIA are provided by tools such as PreSTo [ASS11] and PADLab [TUB17], whereas an interface to OCCT is available for MIKADO [Ris16]. PrADO and OpenVSP, on the other hand, have built-in visualization solutions. In all of these cases, changes to the schema of the internal data-centric model are usually driven by changes in the tool, which means the tool is adapted to changes in the format automatically.

For data-centric exchange models, by contrast, the situation is more complicated. A format such as CPACS is designed to support a large number and variety of tools, all of which have a different emphasis. The development of the format is therefore driven by the requirements and contributions of stakeholders, instead of the development of any single tool. This results in a breadth of data represented, which exceeds the scope of any single tool. Furthermore, the multi-fidelity paradigm of a format like CPACS means that options exist to provide data to a very high level of detail. However, these are usually only required and thus supported by the respective disciplinary specialist.

The lack of a single central tool for data-centric exchange models means that consistent derivation of geometry becomes an issue, due to the aforementioned inherent ambiguity of data-centric models. One example is the definition of fuselage profile curves in CPACS. These are usually defined by a list of points, ordered in circumferential direction. To generate e.g. a NURBS curve an interpolation algorithm, as presented in 4.1.1.1, must be applied. However, this still leaves an infinite number of possible curves, since the parameter distribution and polynomial order of the curve can still be chosen freely by the user. Even though engineering intuition will clearly show that some results are closer to the original intent of the designer than others, as far as the schema is concerned all possible solutions are correct.

In the perimeter of CPACS, the TiGL Geometry Library (TiGL) [SK+19; SS+22] has been introduced to address the issue of geometric interpretation. Implemented in C++ and built using OCCT, TiGL accepts a CPACS file as input and provides CAD geometry representations of selected components. Among other things, the OML, engines, the wing structure and wing movables are supported. TiGL not only provides a GUI for interactive visualization and, to some extent, manipulation [Dro18] of CPACS files, but also a programming interface. The idea is, for disciplinary tool developers to build their analysis and design capability on top of the geometry capabilities of TiGL. Several examples for structural analysis tools built using TiGL are provided in section 4.2.2.

A major weakness of TiGL is that it does not offer support for all geometric details, which can be defined in CPACS, due to the vast scope of the format. For tool developers, this usually means that at some point during the development, they will reach a point where they have exhausted the capabilities of TiGL and must implement their own geometry rules to generate further geometric details. In the best case the additional details are implemented in a way that is compatible to TiGL, as exemplified by the development of the wing structure geometry for the tool Descartes [MPD13], which allows them to be merged back into the TiGL library. However, the entry barrier for this approach is fairly high, since it demands a high proficiency in software development using C++ and detailed understanding of the OCCT kernel. As a result, some authors refrain from using TiGL altogether and opt to develop a dedicated solution for their specific use case instead. Examples include the tool jPAD [NM+16; DeM18] and a model generator based on CATIA described by Wunderlich et al. [WD+21], which are intended to provide smooth outer surfaces and detailed representations of the wing movables for aerodynamic analysis using CFD. Furthermore, in the structural model generators cpacs-MONA [KS+19] and PANDORA [PKH18; WPK17], low-level B-spline libraries are applied for surface modeling to accelerate e.g. the computation of intersection points between the fuselage surface and structural component definition vectors (s. also section 4.2.2).

It is worth noting that in all parametric modeling engines mentioned above CPACS is considered to be the single source of truth, while the geometric model is only a view of the parametric data in CPACS. This means that changes in the geometry are only valid if they are based on a change in the underlying CPACS data.

Nevertheless, the examples give an impression of the breadth of disciplines, to which the parametric modeling engine has to cater. Furthermore, the difficulty for a library with a conventional object oriented architecture such as TiGL to comply with the heterogeneous user requirements is revealed. Similarly, the breadth of fidelity requirements must be handled on the individual tool level.

The graph-based methodology described in **working hypotheses 1 und 2** is intended to overcome some of these limitations by facilitating the model generation for different disciplines and fostering re-use of existing geometry.

4.1.4. Summary

In this section, two types of product representations for digital engineering processes are discussed and compared to the requirements from **working hypotheses 1 und 2**. On the one hand, classical

CAD geometry models are considered, which can provide an unambiguous description of geometry, but have limitations w.r.t. non-geometric data. On the other hand, data-centric models provide nearly unlimited freedom in terms of what kinds of data can be stored. However, if geometric information is stored, it must usually be interpreted, which complicates working with the format and may lead to ambiguities.

For complex products, such as aircraft, which require substantial amounts of both geometric and non-geometric data during design, a hybrid approach using a parametric modeling engine is considered. In this approach, the geometric parameters are stored in a data-centric format. The modeling engine, which is deployed along with the format, provides rules on how the parameters are to be interpreted and can be used to derive a CAD representation of the geometric subset of the product data.

The CPACS format is a well-established data-centric format for aircraft design, which provides the means to describe the aircraft from the point of view of many disciplines, including cabin and structural design. A complementary parametric modeling engine named TiGL is available, which, however, does not support all geometric features of CPACS. Therefore, additional modeling relationships must be implemented to describe fuselage structures and cabins at a sufficient level of detail to enable the investigation of **working hypothesis 2**.

The considerations in this section show that both a data-centric parametric description and a CAD-based geometric description are by themselves necessary but not sufficient to completely describe the product. This corroborates **working hypothesis 1**, in which it is proposed to augment the parametric data with rules to form a knowledge graph.

4.2. Computational analysis and design of the fuselage and cabin

In **working hypothesis 2**, the need to generate analysis models based on different disciplinary analysis models is stated. In this section, previous work on computational analysis and design methods for disciplines concerned with the fuselage and cabin, which could potentially be incorporated as design competences in collaborative third generation MDAO processes, is therefore examined. The goal is to identify necessary product details and geometric features, which must be provided by the virtual product model in order to enable seamless and consistent integration of the respective disciplines.

A particular challenge is that the fuselage structure and the cabin design are closely interlinked and cannot be treated independently from one another. Therefore, special attention is paid to overlapping requirements between fuselage and cabin analysis methods.

Before discussing the existing work on fuselage analysis, a short theoretical introduction to relevant analysis methods in the different fields is given in section 4.2.1. Examples of structural analysis of the fuselage are then reviewed in 4.2.2. In section 4.2.3, different examples of human factors analysis are considered. A short overview of industrialization evaluation of the cabin is given in 4.2.4. Finally, a summary is provided in section 4.2.5.

4.2.1. Review of relevant computational analysis methods

In this subsection, the fundamentals of the underlying methods relevant to the analysis of the fuselage and cabin w.r.t. **working hypothesis 2** are discussed briefly. First, the guiding principles of structural analysis using the finite element method are outlined in section 4.2.1.1. This includes structural sizing methods, which are required for mass estimation. In section 4.2.1.2, the basic concepts of 3D visualization are explained, which have an influence e.g. on the model generation for human-factors analysis models.

4.2.1.1. Finite Element Method for structural analysis and sizing

The core of structural mass estimation based on numerical analysis is a sizing process, by means of which necessary dimensions of the structural components to withstand all critical loads across the flight envelope [Lom96] are computed.

The fully-stressed design (FSD) method is a well-established approach to determine these dimensions and has been described e.g. by Wallerstein and Haggenmacher [WH76] and Nagel, Kintscher, and Streit [NKS08]. It is built upon the finite-element method (FEM), which is widely used and well documented in literature [Sch84; ZT05; Bat08]. In FEM the structural deformations $\mathbf{d}$ at the nodes of an analysis mesh, due to the load vector $\mathbf{f}$ are computed for the static case by solving

$$\mathbf{f} = \mathbf{K} \cdot \mathbf{d}. \tag{4.2.1}$$

$\mathbf{K}$ is the stiffness matrix, which is assembled based on formulations of structural elements connecting the nodes, providing simplified (usually linear) formulations of the structural behavior using

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV. \tag{4.2.2}$$

The strain-displacement matrix $\mathbf{B}$ represents the kinematic behavior of the elements used to decompose the structural domain and therefore depends on the type of element used. While a large variety of different element types exists, topologically one-dimensional beam elements (s. section A.1 for the derivation of the stiffness matrix) and topologically two dimensional shell elements are most commonly used for the thin-walled structures usually found in aerospace. The matrix $\mathbf{C}$ represents the material behavior.

For linear elements the volume integral in equation 4.2.2 can be computed cheaply using numerical integration schemes such as Gauss-Legendre quadrature. Since $\mathbf{K}$ is usually symmetric and positive definite the solution for $\mathbf{d}$ in equation 4.2.1 can be performed efficiently for multiple load cases, i.e. different load vectors $\mathbf{f}$ by Cholesky factorization [Sch84].

Based on the deformations, which must be transformed to element coordinates, a strain distribution ε in the elements can be computed using the element kinematics matrix $\mathbf{B}$:

$$\varepsilon = \mathbf{B} \cdot \mathbf{d}'. \tag{4.2.3}$$

The strains can be translated into a stress distribution σ via the material matrix $\mathbf{C}$:

$$\sigma = \mathbf{C} \cdot \varepsilon. \tag{4.2.4}$$

For sizing, the stress results are compared to allowable stresses, which are computed using applicable failure criteria, to ensure the structural integrity. For isotropic materials under tensile loads, the von Mises stress can be compared to the yield strength of the material. More advanced criteria are provided e.g. by Bruhn [Bru73] for buckling of skin panels under compression. The geometry, e.g. skin thickness, is adjusted according to the ratio of computed stress and allowable stress:

$$t_i^{new} = \left(\frac{\sigma_i}{\sigma_{allow}} \right)^{\omega} t_i^{old}. \tag{4.2.5}$$

To avoid unrealistic results, minimum and maximum values for the thickness are usually provided, as well. In an iterative process, this procedure is repeated, until the geometric design variables t_i (and as a result the structural mass) converge. The relaxation parameter ω can be adapted to accelerate the convergence or improve the stability.

The stress-based sizing is commonly used for isotropic materials. In CFRP the stress may vary depending on the orientation of a given layer. This has led to the introduction of different, usually more complex failure criteria [Puc96; IJs11] for CFRP. For very complex structural designs, failure criteria based on global-local analysis approaches have also been proposed [PM20].

FSD is a popular choice for structural sizing in preliminary aircraft design because of its comparatively simple implementation and good, if not optimal, results [Dor14]. The limitations of FSD have been discussed by Patnaik, Guptill, and Berke [PGB95]. Instead, a structural sizing optimization can be performed, as has been described e.g. by Ainsworth et al. [AC+10], Schuhmacher et al. [SD+12], and Klimmek [Kli16].

Review of finite element analysis and structural sizing tools At this point a short review of eligible finite element analysis (FEA) tools is given. Among the most popular commercial FEA programs [Nag21] are MSC Nastran [Hex23], ANSYS Mechanical [ANS23a] and Abaqus [3DS23a] . Even though Nastran is among the earliest implementations of the finite element method for engineering and still dominates the market in the aerospace industry, both ANSYS Mechanical and Abaqus are sophisticated and mature tools in their own right. ANSYS provides advanced scripting capabilities via the ANSYS parametric design language (APDL), whereas the Abaqus CAE user interface can be controlled using Python. All three codes provide mature capabilities for linear-elastic analysis, which are adequate for preliminary structure sizing [WPK17]. In addition, all of the programs provide different sets of additional analysis types.

Several open-source alternatives are available as well, which can, however, not compete with maturity and the wide range of capabilities offered by commercial tools, yet [Ayy16]. Nonetheless, tools like Code_Aster [EDF23] or B2000++ [SMR18] can be applied if a large number of comparatively basic analyses is required, e.g. to manage license cost of commercial tools.

Aside from finite element analysis tools, specialized solutions for structural design are also available. At the German Aerospace Center (DLR), the sizing tool S_BOT+ makes the FSD methodology available in ANSYS [Dor14; Nag21]. On the other hand, MSC Nastran provides optimization capabilities via the solution 200 mode, which are leveraged e.g. by Klimmek [Kli16]. Similar capabilities are found in the tool LAGRANGE developed by Airbus Defence & Space [SKH96]. Finally, HyperX (formerly HyperSizer) [AC+10] is a commercial solution providing advanced features w.r.t. composite design and failure criteria.

4.2.1.2. Fundamentals of 3D visualization

In section 2.1.2, human factors analysis is mentioned as a relevant type of analysis for the cabin. Often, human in the loop simulations using virtual reality are performed to quickly evaluate accessibility or customization options (s. section 4.2.3). The need for high performance visualization to achieve good levels of immersion results in special requirements for the disciplinary analysis model generation for **working hypothesis 2**. To derive these requirements some background on the basic theory of 3D visualization is provided in the following.

Although development in the field is ongoing, the foundations of 3D visualization have been well-established for many decades, and comprehensive literature is available [Hil90; SAM09; HD+13]. An exemplary graphics pipeline is given in figure 4.2.1. It shows how the visualization is broken down into a sequence of separate tasks, which need to be performed each time a frame is rendered. From the details of the implementation for some of these steps, requirements for the provision of geometric models can be deduced.

The first steps, i.e. *Simulate* and *Pose* are usually implemented within the interactive engine, in order to provide the static state of the geometry at a given point in time. *Culling* is then used to remove invisible faces from the view, e.g. because they are facing away from the camera (front face culling), to improve performance. The reduced 3D geometry then enters the actual rendering pipeline. The main steps are outlined in the following.

Camera Transformations & Projection The first step during rendering is a coordinate transformation from the global geometry coordinate system to the camera coordinate system. The camera coordinate system may be defined by a camera location $\mathbf{x}_c$, a view direction $\mathbf{v}_{look}$ and an upwards pointing vector $\mathbf{v}_{up}$.

In a first step, the origin of the global coordinate system is translated to the camera location $\mathbf{x}_c$ by subtracting it from all point coordinates. From the camera vectors, a rotation matrix $\mathbf{M}_c =$

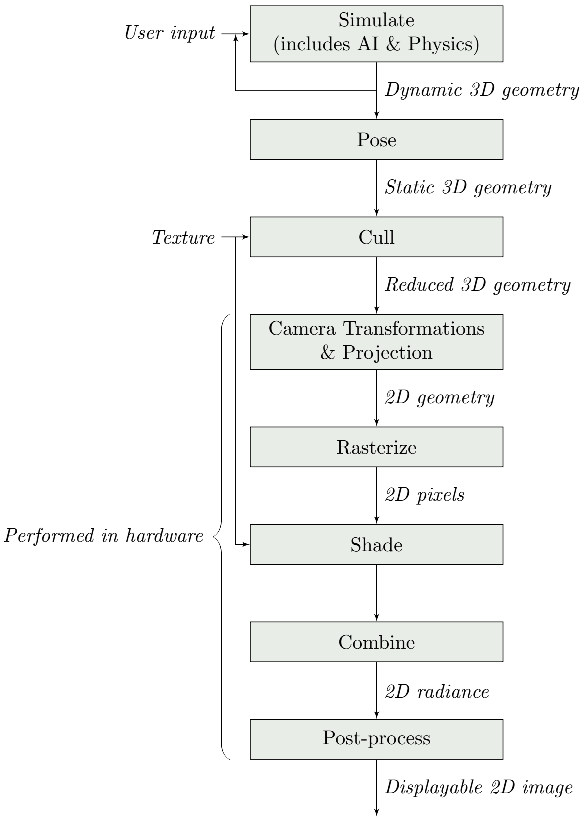


Figure 4.2.1.: Basic rasterizing renderer pipeline steps (after [HD+13])

$\begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{w} \end{bmatrix}$ can be assembled using

$$\begin{aligned} \mathbf{w} &= \frac{-\mathbf{v}_{look}}{\|\mathbf{v}_{look}\|}, && (4.2.6) \\ \bar{\mathbf{v}} &= \mathbf{v}_{up} - (\mathbf{v}_{up} \cdot \mathbf{w})\mathbf{w}, && (4.2.7) \\ \mathbf{v} &= \frac{\bar{\mathbf{v}}}{\|\bar{\mathbf{v}}\|}, && (4.2.8) \\ \mathbf{u} &= \mathbf{v} \times \mathbf{w}. && (4.2.9) \end{aligned}$$

By convention, the new z-axis denoted by the vector $\mathbf{w}$ points in the negative looking direction. The resulting system is illustrated in figure 4.2.2.

Furthermore, the camera frame is scaled to the interval $[-1, 1]$ for the x and y directions. To this end, field of view angles in both directions θ_h and θ_v are given. In addition, a far plane, with the distance d_{far} to the camera is given, which is set to be $z = -1$. This results in the scaling vector $\begin{bmatrix} \frac{1}{f \tan \frac{\theta_h}{2}} & \frac{1}{f \tan \frac{\theta_v}{2}} & \frac{1}{z} \end{bmatrix}$.

Following the transformation of the camera, it is possible to compute the projection. The goal is to transform the frustum defined by the above camera specification and an additional near clipping plane distance d_{near} to a standard parallel view volume, in order to make distant objects appear smaller. This requires a projective transformation on $\mathbb{R}^3$, which corresponds to a linear transformation on $\mathbb{R}^4$ and a subsequent homogenizing transformation H using a similar mathematical approach to non-rational projection in NURBS curves:

$$H(x, y, z, w) = \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right). \qquad (4.2.10)$$

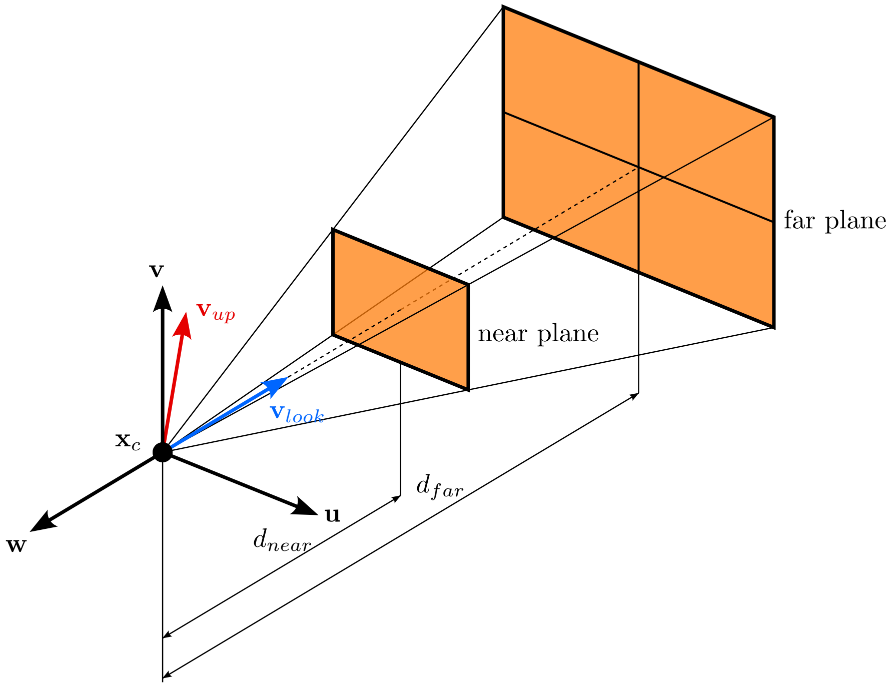


Figure 4.2.2.: Camera projection system including $\mathbf{v}_{look}$ and $\mathbf{v}_{up}$ (after [HD+13])

The transformation matrix $\mathbf{M}_{pp}$ from perspective to parallel projection on $\mathbb{R}^4$ is given by

$$\mathbf{M}_{pp} = \begin{bmatrix} d_{far} - d_{near} & 0 & 0 & 0 \\ 0 & d_{far} - d_{near} & 0 & 0 \\ 0 & 0 & f & d_{near} \\ 0 & 0 & -(d_{far} - d_{near}) & 0 \end{bmatrix}. \tag{4.2.11}$$

The result from the camera transformation and projection steps, is a 2D projection of the geometry in the parallel view volume, along with depth information, which is the input for the next step in the pipeline.

Rasterization Rasterization is the step of translating geometric data to pixels, i.e. image points, which can be displayed on a screen. To this end, the intersections between the geometry and a ray originating in the xy plane and pointing into negative z direction must be computed for each pixel in the interval $[-1, 1]$ in x and y direction.

A classical intersection algorithm has been described by Möller and Trumbore [MT97]. It works on geometry, which consists only of triangular faces. The advantage of triangles over any other type of polygon is that they are guaranteed to be planar and convex. A point on any triangular facet can be described using barycentric coordinates

$$\mathbf{x}_{tria}(u, v) = (1 - u - v)\mathbf{p}_{0,tria} + u\mathbf{p}_{1,tria} + v\mathbf{p}_{2,tria}, \tag{4.2.12}$$

where $\mathbf{p}_{i-1,tria}$ is the i-th corner point in the triangle. The ray is defined by its origin $\mathbf{p}_{0,ray}$ and unit direction $\mathbf{v}_{ray}$ through $\mathbf{x}_{ray}(w) = \mathbf{p}_{0,ray} + w\mathbf{v}_{ray}$, where w is the distance from the origin. Setting $\mathbf{x}_{ray}(w) = \mathbf{x}_{tria}(u, v)$ yields

$$\begin{bmatrix} -\mathbf{v}_{ray}, & \mathbf{p}_{1,tria} - \mathbf{p}_{0,tria}, & \mathbf{p}_{2,tria} - \mathbf{p}_{0,tria} \end{bmatrix} \begin{bmatrix} w \\ u \\ v \end{bmatrix} = \mathbf{p}_{0,ray} - \mathbf{p}_{0,tria}, \tag{4.2.13}$$

which can be solved to find the barycentric coordinates u and v and the distance from the origin w e.g. using Cramer's rule. An intersection is found if both u and v are in the interval $[0, 1]$ and $u + v \leq 1$.

In the case of multiple intersections, which triangle is to be drawn is determined by the lowest distance to the origin w, as shown in figure 4.2.3.

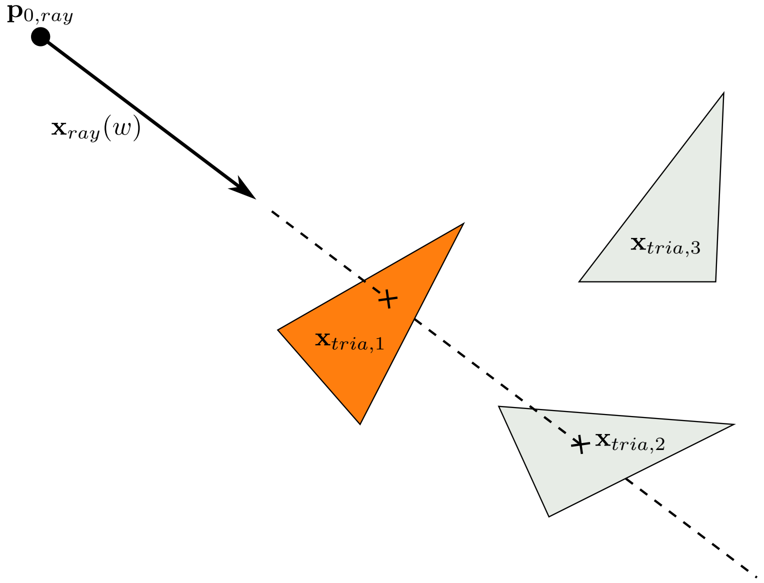


Figure 4.2.3.: Ray tracing applied to find the closest triangle for each pixel (after [SAM09])

Depending on the sequence of the loops, the image generation technique is referred to either as ray casting or rasterization. In ray casting, the outer loop is performed over the pixel centers. For each pixel, a ray is constructed and intersected with all triangles. The closest intersection is returned. In rasterization, on the other hand, the outer loop is over the triangles. This means all pixel positions are computed for one triangle before moving to the next. This has advantages in terms of memory use and allows for simpler deployment of more efficient sampling techniques e.g. bounding-box optimization.

It follows from the above that, in order to perform rasterization efficiently, the fuselage and cabin geometries for visualization must be provided in the form of a triangulated mesh. Common triangulation algorithms such as Delaunay triangulation [Wat81; Reb93] are available in open source libraries such as Gmsh [GR09] or OCCT.

Shading Once the closest intersections are known, the color of the corresponding pixels must be determined. On the one hand, this entails the evaluation of the color or texture data of the eligible faces, based on the uv coordinates computed during rasterization. On the other hand, a realistic rendering of the brightness of the faces due to a given light source significantly improves the quality of the image.

The basic assumption for computing intensity I of the light reflected by a face is given by Lambert's cosine rule

$$I = I_{dir} \cos\theta, \tag{4.2.14}$$

where I_{dir} is the intensity of the incoming light and θ is the angle between the light vector $\mathbf{v}_{light}$ and the surface normal. A face, which is perpendicular to the incoming light vector, will reflect all the light and thus appear bright, whereas a face parallel to the light vector will appear dark.

Applied to a triangulated mesh, Lambert's rule will yield a constant intensity for each triangle. This is referred to as flat shading and provides acceptable results for objects with flat faces e.g. boxes or pyramids. However, for curved surfaces, a smoother distribution is desirable to provide a better impression of the underlying geometry. Increasing the mesh resolution alone does not prove sufficient, as the eye is very sensitive to discontinuities across mesh edges. Furthermore, significantly increasing the number of faces will have adverse effects on the performance of the rasterization.

Gouraud [Gou71] proposes an approach, where the intensity is instead computed for each vertex of the triangle and then interpolated across the face. The approach has later been refined by Phong

[Pho75], who proposes to interpolate the normals themselves to support more complex lighting models. In both cases, knowledge of the normal vectors at the vertices is required, which can be provided by an external algorithm, or approximated by averaging the normals of the adjacent faces. As illustrated by figure 4.2.4, the method yields smooth renderings even for coarse meshes. Therefore, vertex normals should be provided when creating visualizations of smooth surfaces, such as the outer fuselage surface, in order to keep the polygon count low and improve rendering performance of the model. Whereas it can be difficult to determine accurate normals based on a triangle mesh without any additional information, the normal vectors at CAD surface points can be computed easily from the tangent vectors in the two topological directions.

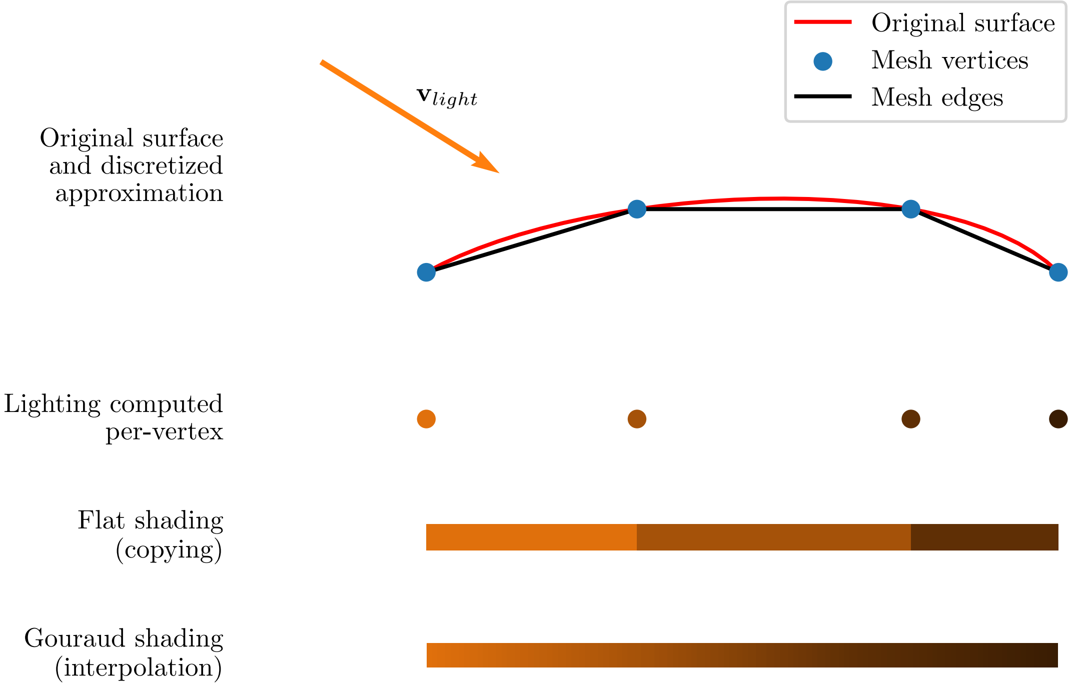

Original surface
Mesh vertices
Mesh edges
$\mathbf{v}_{light}$
Original surface and discretized approximation
Lighting computed per-vertex
Flat shading (copying)
Gouraud shading (interpolation)

Figure 4.2.4.: Illustration of flat vs. Gouraud shading (after [HD+13])

Classification of 3D visualization tools With 3D visualization firmly established in modern computing, a number of solutions, both proprietary and open source, are available to build 3D visualization applications. In the following, a few solutions will be examined more closely along with possible use cases.

Visualization library A visualization library is the lowest level of 3D visualization application. It is used to create visualizations from the source code level. The Visualization Toolkit (VTK) [SML06], an open source library for scientific 3D visualization originally written in C++ is an example for a visualization library. It provides bindings to many other programming languages including Python. A visualization library provides a lot of freedom to developers, but also requires high expertise.

Scientific visualization environment Built upon VTK, ParaView [Aya15] makes the capabilities of the library available in a GUI. This facilitates interactive analysis of scientific data, e.g. FEM stress results, or CFD flow fields. In addition to the interactive UI, a Python scripting interface is provided, to support task automation.

Tecplot [Tec23] is another widely used desktop application for scientific data visualization. Unlike ParaView, Tecplot is proprietary software.

3D graphics environment Unlike scientific visualization environments, the primary use case for a general purpose 3D graphics environment like Blender [Ble23] is interactive asset development for 3D applications. To this end, limited geometry modeling functionality is available, including primitives and spline curves, which serves to create inputs for mesh generation. However, the capabilities are not on the same level as those of a fully realized CAD kernel. In return, mesh modification, texture mapping and many more interactive features are provided in addition to the geometry modeling. Many capabilities of Blender are also accessible via scripting, using the built-in Python interpreter. Furthermore, an advanced ray tracing engine for high-quality rendering is provided.

Video game engine In contrast to the above applications, a video game engine such as Unity [Uni23] provides a complete environment for game development. As such the features are not limited to visualization, but also provide advanced possibilities for real-time user interaction, which can be customized via a scripting interface. In the case of Unity, this interface supports the C# programming language. Notably, a game engine is usually not intended to create or modify assets, but to compose and present many assets in an interactive scene. A similar engine to Unity is the Unreal Engine [Epi23]. An open source alternative is Godot [LMc23].

Mesh file formats In order to pass geometry models to scientific visualization tools or game engines, suitable exchange formats are necessary. Among the most widely-used formats is the STL (originally stereolithography) format [Gri04], which provides a basic notation for triangular meshes. STL files can be provided in plain text or binary form. It is possible to store face normals in STL, to save time for the computation, but vertex normals are not supported. Hence, STL is not an ideal format to share models of curved surfaces.

The Wavefront OBJ format [Wav92], on the other hand, provides not only support for vertex normals, but also *uv*-coordinates for texture mapping. The drawback of OBJ is that it is a pure text format, which makes import and export operations expensive. Furthermore, the format has been out of active development for a long time. Still, OBJ is widely supported by modeling and visualization environments.

A more advanced alternative to OBJ, which is, however, proprietary, is the FBX (Autodesk Filmbox) format [Aut23]. FBX adds support e.g. for motion data to the feature set of OBJ. It is available both in text and binary form. However, due to the proprietary license, FBX is not supported by all open source environments.

The glTF (GL Transmission Format) format [Khr17] is a comparatively recent development, which aims to reproduce the feature set of FBX in an open standard. Like FBX it supports motion data in addition to the geometric information and allows for model storage in plain text or binary format. While support for glTF in open source projects is growing, support for many commercial tools including Rhinoceros 3D and the Unity Engine is only available via external plugins at the time of publication.

4.2.2. Analysis and design of fuselage structures

Asserting the structural integrity of the airframe has been an essential engineering task since the earliest days of aviation. At the same time, minimizing structural mass is key when trying to maximize performance, as evidenced by the Breguet equation (2.1.1), which has led to a comparatively aggressive approach to lightweight structural design. Safe lightweight structures require a very good understanding of the structural behavior under all possible load conditions. The introduction of large-scale CFRP-components in recent aircraft programs has amplified this need. Consequently, detailed structural analysis using advanced numerical methods is a well-established in the aircraft industry, especially during detail design. Thus, it is an important discipline to consider, when evaluating the analysis model generation capabilities described in **working hypothesis 2.**

When it comes to the earlier design phases, numerical structural analysis is widely used for mass estimation. However, the amount of scientific work w.r.t. the fuselage structure is eclipsed by the research on wing design [Dor14; HKM14; Kli16; QG16; MG+18], where aeroelastic effects have a much more immediate impact, and multidisciplinary design techniques like aeroelastic tailoring promise significant potential for mass reduction. Nevertheless, the fuselage has its own set of relevant structural challenges, ranging from pressurization, to crashworthiness, which merit closer consideration.

4.2.2.1. Global FEM approach for structure sizing

FEM-based sizing of aircraft structures requires a dedicated structural model, which is commonly referred to as the global finite-element model (GFEM). In contrast to detailed fimite element models (DFEMs) of individual components, such as brackets, which are often modeled using mainly solid elements, the GFEM approach aims to represent the thin-walled structure of aircraft using simplified structural elements like beams or rods e.g. for stiffeners and shells e.g. for skin panels.

Among the first efforts to include physics-based sizing methodology in an OAD context is the work of Österheld [Öst03], who integrates FEM-based structural sizing into PrADO. She applies a modified version of FSD on a half-model of the aircraft primary structure, shown in figure 4.2.5. A classical structural layout for passenger aircraft fuselages with a thin-walled structure reinforced by longitudinal stringers and circumferential frames as described e.g. by Niu [Niu88] is adopted. The parameter values for the model generation are provided by the PrADO database. Based on the work of Österheld, adaptations for different aircraft system architectures have been developed later by Werner-Westphal, Heinze, and Horst [WHH08] and Hansen [Han09]. Further sizing capabilities for CFRP structures were later added by Rieke [Rie13].

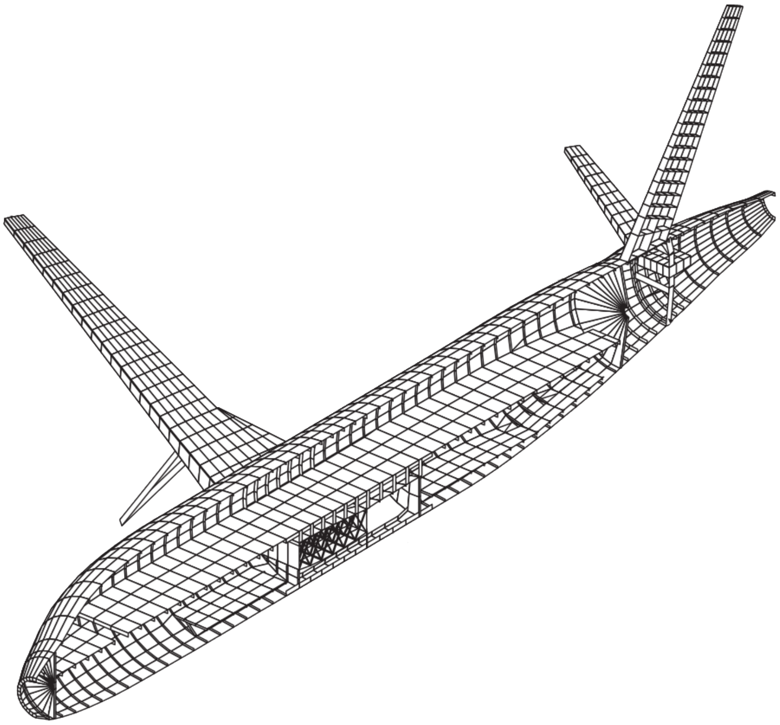


Figure 4.2.5.: FEM model from PrADO (from [Öst03])

In table 4.2.1 the components considered in the model by Österheld are listed and mapped to their respective FEM representation. For comparison, the table then provides the FEM modeling capabilities of subsequent tools, which are described in the following.

Parametric-associative CAD A different approach built around a central CAD model has been demonstrated by Ledermann, Ermanni, and Kelm [LEK06]. Using a parametric-associative approach built upon CATIA Knowledge Patterns, the authors propose the introduction of UDFs, which represent parametric models of individual components. Indeed, many elements of KBE (s. section 4.3) are already present in this approach. The UDFs can be combined to yield a complete CAD model of

Table 4.2.1.: Structural components modeled in PrADO compared to later structural analysis tools

Component group	**Components implemented**	**PrADO [Öst03]**	**CATIA [LEK06]**	**TRAFUMO [SK16]**	**PyPAD [Tra16]**	**Descartes [MPD13]**	**GeoMACH [HKM14]**
Paneling	Skin	shell	shell	shell	shell	shell	shell
	Stringers	orthothropic skin properties	beam	beam	beam	beam/properties	shell
Frames	Standard frames	beam	beam	beam	beam	beam	shell
	Main frames	beam	beam	beam	beam	beam	shell
	Pressure bulkheads	shell	shell	shell + beam	shell	shell	-
Floor structure	Panels	shell	shell	-	shell	n/a	shell
	Crossbeams	beam	beam	beam	beam	n/a	-
	Struts	beam	beam	beam	beam	n/a	-
	Seat rails / longitudinal beams	-	-	beam	beam	n/a	-
Center wing box	Spars (with spar caps)	shell (beam)	shell (n/a)	shell (beam)	shell (beam)	shell (beam)	shell (-)
	Skin	shell	shell	shell	shell	shell	shell
	Stringers	orthothropic skin properties	beam	beam	shell	beam	shell
	Connecting ribs	shell	shell	shell	shell	shell	shell
	Diagonal struts	beam	beam	shell (ribs)	shell (ribs)	shell (ribs)	-
Landing gear bay	Skin	shell	n/a	shell	shell	-	shell
	Stringers	orthothropic skin properties	n/a	n/a	n/a	-	-
	Pressure bulkheads	shell	shell	shell + beam	shell + beam	-	-
	Keel beam	shell	n/a	shell	shell	-	-
Tailplane attachments		yes	cutout	beam	wingbox	wingbox	wingbox
Doors and windows		point mass	cutout	property	-	-	-
Payload		point mass	n/a	point mass	-	point mass	-
Fuel in center wing box		point mass	n/a	s. [Dor14]	yes	point mass	-
CFRP support		s. [Rie13]	no	s. [Dor14]	experimental	yes	yes

the airframe controlled by about 200 parameters. From this model, an FEM mesh is derived, as illustrated by figure 4.2.6. Compared to Österheld, the model by Ledermann, Ermanni, and Kelm features discrete stringer modeling and door cutouts, as shown by table 4.2.1. In addition a full model is used, as opposed to the half-model described by Österheld, which makes it possible to take into account unsymmetrical load cases. The model is applied to perform a topology optimization on the wing box using rib positions and thicknesses of skin, spars and ribs as design variables and structural mass as objective function. A genetic optimization algorithm is applied. The wing is only analyzed for a 2.5g maneuver load case, expressed as shear, moment and torque curves along a loads reference axis. No specific information is given on how non-geometric information, such as material properties, are associated with the model, nor whether thicknesses can be stored in the central model or are only found in the FEM representation. The latter option not only poses a risk w.r.t. design data consistency, but also complicates data exchange.

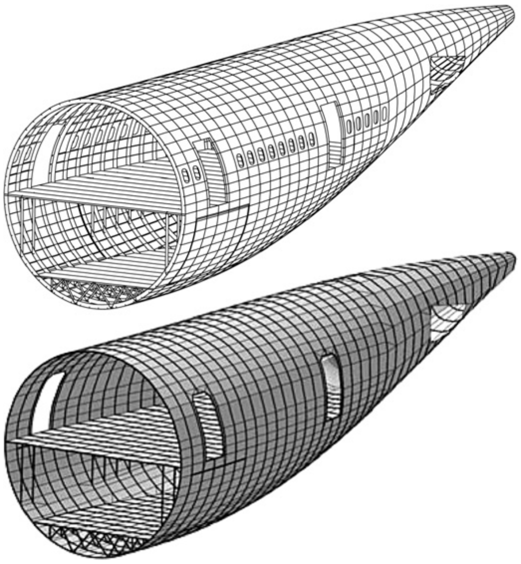


Figure 4.2.6.: Fuselage geometry model (top) and analysis mesh as shown by Ledermann, Ermanni, and Kelm [LEK06]

The modeling approach shown by Ledermann, Ermanni, and Kelm was subsequently developed further by Weiss [Wei09], who turns the parameters and unstructured Knowledge Patterns into a more formal parametric feature tree formulation. He also shows that attributes can be assigned to entries in a CATIA feature tree. However, for complex parts this quickly results in a cluttered tree that becomes difficult to manage and navigate. The examples provided by Weiss are mostly taken from the automotive industry and on a detail design level, but serve to illustrate the potential for applying optimization on the feature tree. Notably, spline points are used as design variables. In contrast to Weiss, Hürlimann [Hür10] further pursues the wing design aspects of the work of Ledermann, Ermanni, and Kelm, implementing a loads process driven by CFD, improved model details e.g. landing gears, and more advanced failure criteria.

Wenzel, Sinapius, and Gabbert [WSG11] also adopt the parametric-associative approach. Unlike Hürlimann, they also demonstrate it for constant cross-section fuselage barrels and the wing-fuselage intersection area in addition to the wing. A distinguishing detail to the previous works is the introduction of uniform design regions, i.e. regions of the FEM model, where the properties such as skin thickness are the same for all elements. This not only reduces the number of design variables and thus the complexity of the optimization problem, but is also more in line with current manufacturing practices. Another example for structural optimization leveraging the parametric-associative capabilities of CATIA is given by Amadori, Jouannet, and Krus [AJK08].

In a later study, Dannenhoffer and Haimes [DH16] show that the feature-tree-based CAD modeling approach implemented in the ESP can also be applied to derive structural analysis models in a similar fashion.

CPACS-based In the context of integrated collaborative design activities at the DLR, a number of analysis and design tools have been implemented based on the CPACS aircraft description schema (s. section 4.1.2). One of the earliest examples is provided by Ciampa, Zill, and Nagel [CZN10] who derive a vortex lattice mesh and a simple structural analysis model of a BWB configuration from a CPACS data set. The geometry in CPACS is generated from a CST parametrization. Based on these design variables and a structural sizing, an aero-structural optimization is performed to minimize the structural mass.

Dorbath [Dor14] introduces ELWIS (Finite Element Wing Structure), a model generator for highly detailed conventional wing models, including movables and landing gears. He also integrates a loads process, which includes aerodynamic load computations using vortex-lattice methods, but also fuel loads. All necessary analysis models are provided by what is referred too as a central multi-model generator.

Complementary developments for modeling the fuselage structure have been carried out by Schwinn et al. [SS+13] in the tool TRAFUMO (Transport Aircraft Fuselage Model), which has later been renamed to PANDORA (Parametric Numerical Design and Optimization Routines for Aircraft) [PKH18]. Scherer and Kohlgrüber [SK16] provide a complete list of the structural components, which are part of the fuselage model, along with their corresponding CPACS definitions. The components mostly coincide with the list by Ledermann, Ermanni, and Kelm. However, the parametric description by Scherer and Kohlgrüber goes to significantly higher levels of detail pertaining to e.g. the cross-sectional profile of extruded stiffeners like stringers and frames, assigned materials, or skin panel groups, which are similar to the uniform design regions proposed by Wenzel, Sinapius, and Gabbert. A resulting fuselage model from TRAFUMO for a single-aisle configuration is given in figure 4.2.7.

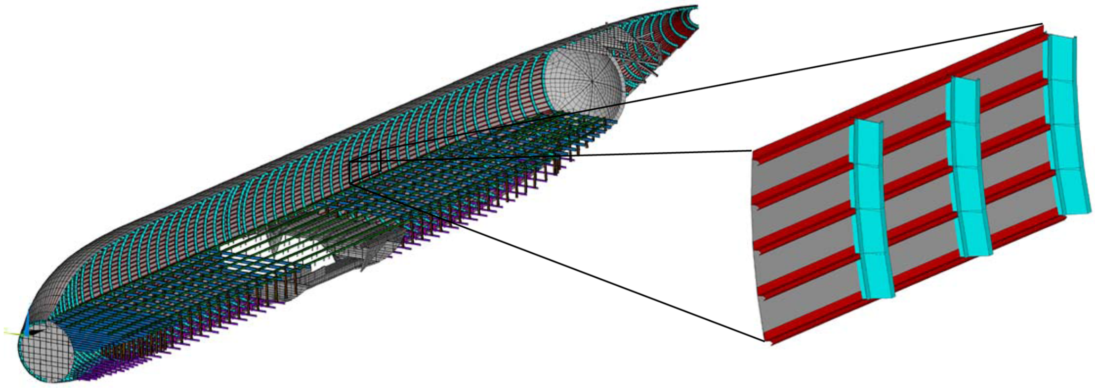


Figure 4.2.7.: Fuselage structure model with detailed stiffener cross-section definition (from [SK16])

Scherer et al. [SK+13] furthermore demonstrate the compatibility of the wing and fuselage models, by assembling them into a detailed model of a full configuration, generated solely from information provided via CPACS. An example distribution of the von Mises stresses for a 2.5g load case is given in figure 4.2.8. Petsch [Pet15] later proposes a more general way of modeling the wing-fuselage intersection region. The sizing for all of the models is performed in ANSYS Mechanical using the S-BOT+ sizing tool [Nag21], which implements FSD. In addition, Walther, Petsch, and Kohlgrüber [WPK17] present a solver-agnostic implementation of the fuselage GFEM generation, enabling the solution of the models using different solvers.

DELiS (Design Environment for Lightweight Structures), another GFEM generator based on CPACS, has been introduced by Freund et al. [FH+14] and Führer et al. [FW+16]. The models provided by DELiS provide fewer details than the model by Scherer et al. [SK+13], e.g. no explicitly modeled stringers and less detailed wing-fuselage-intersection regions. On the other hand, the whole aircraft, i.e. both fuselage and wing, is covered by a single tool, which contrasts the partitioned approach

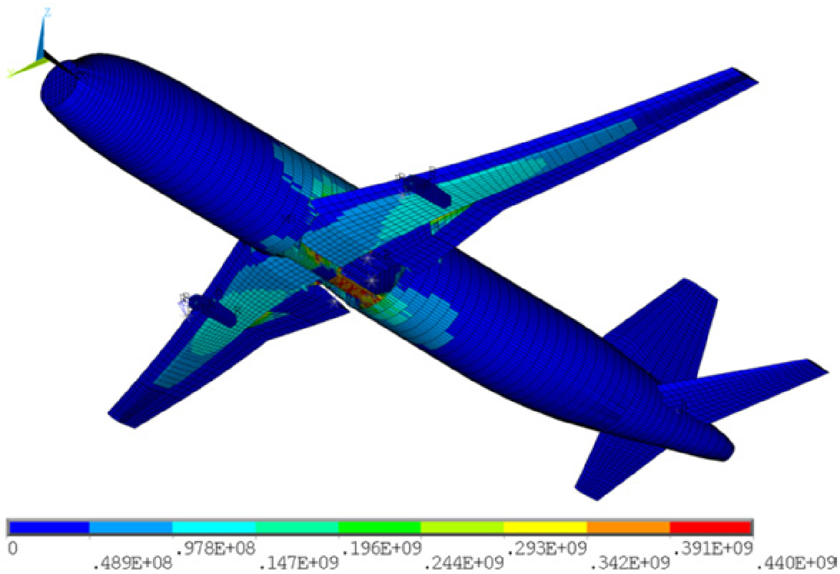


Figure 4.2.8.: Von Mises stresses (in Pa) for a 2.5g maneuver load case on a full configuration model (from [SK+13])

described by Scherer et al. Furthermore, Bach et al. [BF+16] demonstrate a sizing optimization for CFRP structures based on a DELiS wing model using the HyperSizer framework, whereas Scherer et al. assume an isotropic material for the fuselage.

All of the above CPACS-based tools have been built on top of TiGL [SK+19] (s. section 4.1.3), which is used to determine mesh point locations by computing intersections between the outer surface and vectors computed from the definition planes of the structural components in CPACS. However, Walther, Petsch, and Kohlgrüber [WPK17] show that a fuselage model generation implemented independently from TiGL can have significant performance benefits, due to better suited geometric component descriptions and low-level geometry access, which enables e.g. efficient computation of surface normals at the intersection points. Similarly, in cpacs-MONA (CPACS-ModGen-Nastran) by Klimmek et al. [KS+19] a custom geometry modeling approach using B-spline surfaces is applied [Kli16] to build full configuration FEM models for MSC Nastran. A subsequent optimization enables the sizing of the structure not only due to static aeroelasticity, but also due to flutter and divergence for isotropic and anisotropic materials.

The PyPAD (Python package for Preliminary Aircraft Design) tool suite presented by Travaglini [Tra16] is a noteworthy example for a full configuration FEM model generator based on CPACS developed outside the DLR. Travaglini applies Abaqus CAE to build the model, using the TiGL library as interface to CPACS. While its primary focus is on the wing modeling, the fuselage is represented not only via frames, stringers and skin panels, but also a detailed center fuselage area for accurate load transfer. PyPAD was initially developed as a successor to NeoCASS (Next generation Conceptual Aero-Structural Sizing Suite) [CRT11], which is the structural sizing component in CEASIOM. Therefore, similarly to the work of Dorbath, it also provides surface meshes for aerodynamic analysis using an unsteady panel method in addition to the structural model in the sense of a multi-model generator. In his thesis, Travaglini considers the possibility to use the OCCT CAD library for geometry modeling instead of the commercial Abaqus CAE. He ultimately dismisses the former approach in favor of the latter, citing the better documentation and Python scripting capabilities of Abaqus CAE as reasons. An important drawback of Travaglini's model is that the skin mesh is generated independently from the stiffeners, which are subsequently connected using constraint elements. This not only introduces numerical complexities, but also limits the design freedom for the skin sizing, since it is not possible to properly assign sizing regions as shown by Wenzel, Sinapius, and Gabbert.

Another example for a non-DLR structural model generator is Descartes by Maierl, Petersson, and

Daoud [MPD13]. Unlike Travaglini, the authors select the OCCT kernel and expand on the geometry modeling capabilities provided by TiGL. While the structural modeling capabilities were initially focused on the wing, with only rudimentary support for fuselage structures, more detailed fuselage modeling, including beam-based frames and stringers and shell-based skin panels and bulkheads, has been added later, enabling advanced analyses such as subsystem space allocation, in addition to FEM-based structural sizing [TD+21]. Like PyPAD, Descartes can also provide aerodynamic analysis models for loads calculation [Dei16]. Many published applications of Descartes use a medium altitude long endurance (MALE) configuration as an example [WG+19; TD+21], which has many common aspects w.r.t. a conventional tube and wing configuration. The structural models created by Descartes are intended for use with the proprietary structural optimization tool LAGRANGE, which is a limitation to a more general applicability of Descartes. However, many of the structural geometry modeling algorithms developed by Maierl, Petersson, and Daoud have subsequently been made available in TiGL [SK+19], which is possible due to the common open source CAD kernel.

Support for novel concepts In **working hypothesis 4**, the issue of compatibility with novel concepts is raised. Most of the model generators considered thus far have mainly been focused on conventional tube-and-wing configurations. Exceptions include Ciampa, Zill, and Nagel [CZN10], who model a strongly simplified BWB configuration using wing modeling techniques, Travaglini [Tra16], who demonstrates the capacity of his tool to analyze tailless aircraft, and Maierl, Petersson, and Daoud [MPD13], who investigate a MALE configuration of an unmanned aerial vehicle (UAV), which is, however, closely related to a conventional tube-and-wing configuration.

More detailed analyses of novel aircraft system architectures in the past have required significant adaptations of existing tools, or even specialized analysis tool developments entirely from scratch. For example, Hansen [Han09] shows the implementation of BWB support in PrADO, requiring significant changes to initial assumptions and deep understanding of the overall code. Similarly, Gern [Ger15] introduces the HCDstruct tool dedicated to generating finite element models of BWB configurations based on OML data provided by OpenVSP. The tool was later adapted to add support for conventional and double-bubble layouts [QG17]. The level of detail of the structure design represented by the model is low compared to the aforementioned tools. Furthermore, the authors do not specify, how the necessary details w.r.t. the structural layout are added to the OpenVSP data. The OML is once again used to generate an aerodynamic analysis mesh to provide loads.

Another example for a BWB model generator is presented by Qian and Alonso [QA21a]. The level of detail is significantly increased compared to Gern, as explicitly modeled fuselage stiffeners and cabin boundaries are considered. Notably, the frames are modeled as shells, unlike the previous tools. The model generation is implemented in Python [QA21b].

Picchi Scardaoni, Binante, and Cipolla [PBC17] describe a dedicated model generator for the fuselage of a boxwing configuration based on Abaqus CAE. It supports single- and double-deck layouts and provides frames, stringers, and floor structures all modeled using shell elements. The data is provided in a custom XML file.

Hwang, Kenway, and Martins [HKM14] show applications of the GeoMACH (geometry-centric MDO of aircraft configurations with high fidelity) geometry modeling tool [HM12], which include conventional, BWB, double-bubble and SBW configurations. The geometry is described in a Python script by combining different parametric geometric primitives and interpolants. The modeling approach is somewhat similar to CPACS, as the geometry of the wing and body is described via sections. The structural mesh consists exclusively of shells and includes frames, stringers, skin and floor panels as well as a detailed wing structure. Only models with a significantly reduced number of modeled stringers compared to real-life designs are shown and some details like bulkheads and floor support structures are omitted altogether. In return, the structure can be expressed completely in terms of parametric positions on the OML surface. This has the advantage that the shape of the structural members can easily be differentiated. Furthermore, the structure can follow along with large deformations of the OML. The drawback with the approach is that it quickly leads to warped structures, which may be difficult to realize, e.g. due to manufacturability constraints. Furthermore, the script-based modeling approach, makes it hard to exchange models without sharing the library.

A similar approach to structural modeling to Hwang, Kenway, and Martins, albeit based on a CAD kernel, has been proposed by Haimes and Drela [HD12].

Summary Current fuselage GFEM for structural sizing usually consist of shell and beam elements, where shell elements represent the skin panels and beam elements represent the frames and stringers as well as the floor beams. Bulkheads are commonly modeled, in order to create a watertight model of the pressurized region. A detailed representation of the wing-fuselage intersection area is useful for realistic load transfer.

Furthermore, many model generators are part of multi-model generation tool suites, which also include an aerodynamic mesh generation component for loads analysis. Hence, knowledge of the wetted surface is usually required. The different geometry inputs are managed via a central geometry model, which is either CAD- or parameter-based.

FSD or optimization are used to determine structural parameters, such as skin thickness or beam profile parameters, to assure structural integrity under all load conditions. The results can be used to compute the mass of the primary structure, which provides a starting point for the estimation of the overall structural mass.

4.2.2.2. Beyond GFEM

The model resolution of the GFEM is usually sufficient to perform a sizing of the structure under aerodynamic loading. However, as standards in safety and comfort are increasing, it is necessary to investigate other relevant scenarios, which require more detailed knowledge of the structure and usually involve nonlinear structural analysis.

One such scenario is crashworthiness analysis, where the fuselage structure plays a key role in dissipating the energy of the impact and thus increasing survivability of the event. As a result of advances in analysis capabilities and computing power, these analyses are now also required to certify new passenger aircraft. Schatrow and Waimer [SW16] provide an example for a suitable FEM analysis mesh to assess the kinematic behavior of a fuselage barrel under crash loading, which takes into account nonlinear material behavior. Notably, cabin details, such as seat models, are also taken into account in order to accurately determine the accelerations to which passengers are subjected, which in turn allows for assessment of the sustained injuries [Eib59; Wai13]. As inertial effects must be taken into account, the distribution of the mass must be known as well. Furthermore, the mesh has a significantly lower element size compared to GFEM meshes to ascertain a sufficiently fine resolution of failure regions on the one hand and to allow for a high time resolution for transient analysis on the other [Wai13]. The stable time step Δt_{stable} is linked to the allowable element edge length L_e via the transmission speed of longitudinal pressure waves in the material, i.e. the speed of sound c_d:

$$\Delta t_{stable} = \frac{L_e}{c_d}. \tag{4.2.15}$$

For one-dimensional solids $c_d = \sqrt{\frac{E}{\rho}}$.

Schwinn [Sch15] describes an effort to generate full fuselage models for crash analysis based on CPACS introducing a locally refined mesh in the region of impact, which also contains simplified representations of the seats as well as the door opening and detailed door surround structures. Siemann et al. [SS+17] subsequently adopt this modeling approach for ditching simulation, where the water is represented using the smoothed-particle hydrodynamics (SPH) method.

On the side of comfort improvement, interior noise reduction is an important topic. The analysis of structural nose transmission is briefly discussed in section 4.2.3.2, where the structural model of the fuselage requires a comparable level of detail to the crash case.

These applications illustrate that more detailed representations of the structure than GFEM will be required eventually, as the level of detail of the analyses in the development process increases. This emphasizes the need for consistent multi-fidelity capabilities in structural analysis model generation.

4.2.3. Cabin analysis tools for Human Factors evaluation

Analysis of the aircraft cabin is often concerned with the comfort of the passenger. As such, the topic should be taken into account in the selection of disciplines to illustrate the multidisciplinary analysis model generation capabilities described by **working hypothesis 2.** In classical aircraft design, typical measures for comfort are the seat pitch and width, the lavatory utilization, i.e. number of passengers per lavatory, or trolley utilization, i.e. the number of passengers per full-size trolley (FST) [Tor76; Lee03; Gob15]. However, in keeping with the vertical growth trajectory for MDAO, the goal is to incorporate more detailed human-centered design aspects in early design phases, as proposed by Hall et al. [HM+13], Bagassi et al. [BL+18], and Reimer et al. [RR+20]. This means that design decisions should be made based on an assessment of their effects on passenger comfort, or more generally put, the user experience [RN+12].

In the following, several types of analyses are reviewed, which may be connected to the analysis process to enhance the assessment of comfort. First, simulation of human interaction with the cabin for comfort assessment using human in the loop and human model approaches are discussed in section 4.2.3.1. Then, aspects of noise and cabin airflow simulation using physics-based simulation are examined in section 4.2.3.2. Finally, simulation approaches for boarding and evacuation are considered in section 4.2.3.3.

4.2.3.1. Comfort assessment using human in the loop and human model simulation

Bagassi et al. [BL+18] propose comfort evaluation on a detailed level using both geometric evaluations based on a human model and human in the loop simulations, which also allow for the evaluation of subjective measures such as aesthetics. For this approach, detailed cabin geometry models must be available.

Even though virtual reality (VR) methods have been applied in industry for a long time [Zep07; BV16], advances in 3D visualization and VR technology [CFS17] have renewed interest of researchers in rapid evaluation of the aircraft cabin from a passenger's perspective using VR. De Crescenzio et al. [DB+19] show an example for human-centric design evaluation for a business jet cabin using human in the loop simulation. To this end, test subjects were shown two variants of a business jet cabin layout in a VR environment. They were then asked to provide a rating of the variants on a Likert scale in different predefined categories. The two variants of the cabin differ only in terms of material choices for the surfaces, which are represented by two different sets of textures on the same underlying 3D model, which must be provided a priori. The two options are shown in figure 4.2.9.

Using a similar experimental procedure, the interactive evaluation of components in a regional aircraft cabin based on human in the loop simulation has been demonstrated by De Crescenzio, Bagassi, and Starita [DBS21]. In the simulation, built using the Unity video game engine, subjects did not only move through the cabin, but were asked to interact with components e.g. open an overhead stowage compartment, access items in a galley, or step inside a lavatory. However, only a single variant of the cabin was assessed.

Since the human in the loop analysis requires the interaction of a human user for each analysis, it is not directly applicable for deployment within an MDAO process. Nonetheless, it is useful to make these models available early in the design process and provide the option to perform the analysis during post-processing of design a run.

A different approach has been proposed by Fuchs, Fuchte, and Biedermann [FFB19], who show a multidisciplinary design evaluation of the passenger service channel (PSC) including a human factors assessment. Given the relative position of the PSC and a seat row, accessibility e.g. of oxygen masks is evaluated, based on distances between key points, which are then compared to requirements e.g. from

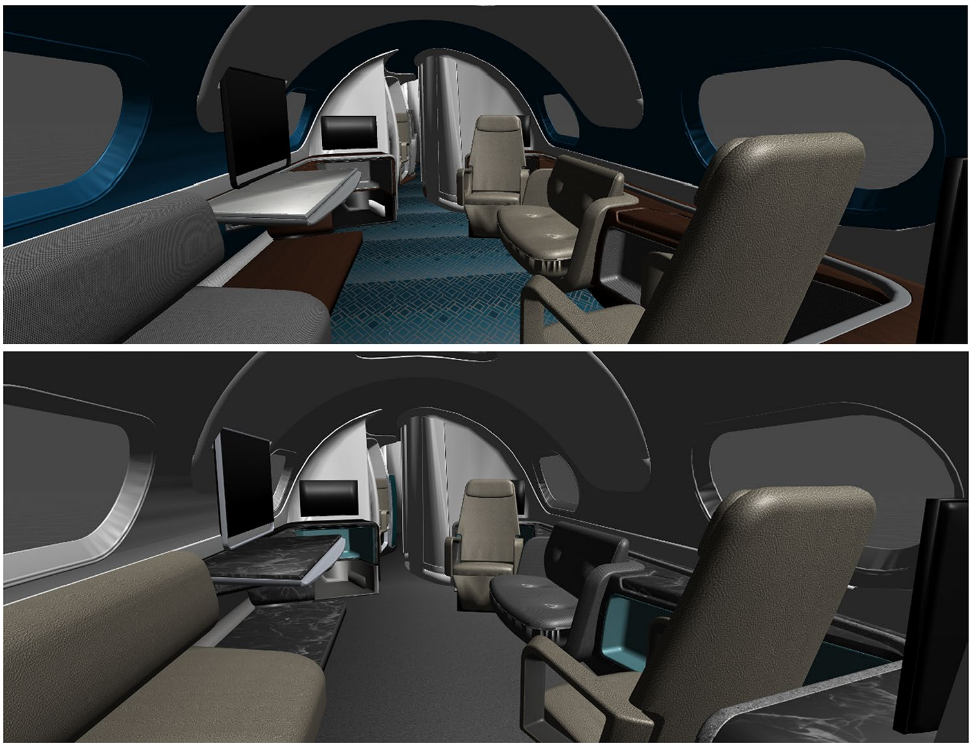


Figure 4.2.9.: Different cabin options proposed by De Crescenzio et al. [DB+19]

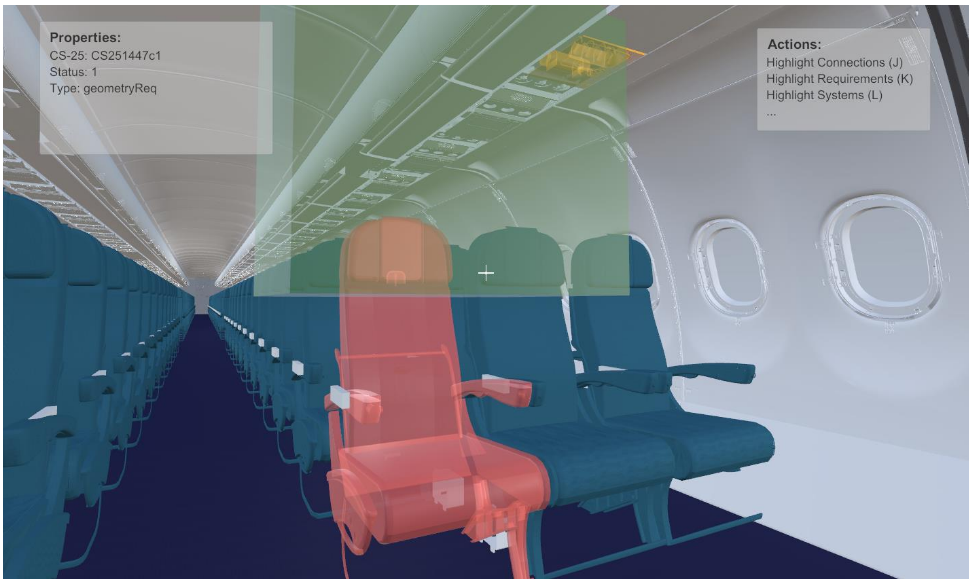
the certification specification [EAS21]. This approach can essentially be reduced to an underlying human model as proposed by Bagassi et al., which is expressed implicitly through the requirements. Unlike human in the loop simulation, the human model approach requires no user interaction, which makes it eligible for application in MDAO [FH+19].

The capabilities for the PSC can also be applied to full cabin configurations and different components including the air conditioning system [FB+21]. Furthermore, a VR visualization environment is implemented using the Unity engine to make the complexity of the subsystem architecture more accessible to the engineer. Figure 4.2.10 shows the accessibility assessment of the PSC for a selected seat in VR. The underlying product data is stored in a modular fashion [FH+21]: Basic aircraft data e.g. the fuselage cross-section and the frame positions are extracted from CPACS and used to place models of the frames. Cabin and subsystems information is stored in a separate XML file. A system layout algorithm implemented in MATLAB provides the design rules, which are based on geometric input parameters and certification requirements. The final design is passed to Blender via a custom XML file. Using Blender Python scripting, the VR model is assembled from external geometry models. Like De Crescenzio, Bagassi, and Starita, Fuchs et al. apply the Unity engine for interactive visualization.

Engelmann, Drust, and Hornung [EDH20] show a similar model generation approach using Blender. Since subsystems are not considered, they build the model solely based on the CPACS cabin description initially introduced by Fuchte, Gollnick, and Nagel [FGN13]. Similarly to Fuchs et al., component 3D models are retrieved from a separate 3D model database. Engelmann, Drust, and Hornung identify image rendering and geometry assessment, e.g. the detection of component collisions as principal applications for the 3D cabin model.

4.2.3.2. High fidelity physics-based human-factors analysis

High fidelity physics-based analyses to provide an evaluation of the cabin w.r.t. human factors are more rare than the geometry-based assessments discussed in the previous section. Nevertheless, some studies related to cabin noise have been published. Langer and Blech [LB19] propose an analysis



Properties:
CS-25: CS251447c1
Status: 1
Type: geometryReq

Figure 4.2.10.: VR-driven accessibility evaluation as shown by Fuchs et al. [FB+21]

pipeline, which couples flow simulation using the Reynolds-averaged Navier-Stokes (RANS) equations, modeling of fluctuating sound sources and structural modeling of the aircraft fuselage. High-fidelity FEM analysis is required to properly determine the acoustic transmission properties of the fuselage [TS19; HB19]. Papantoni [Pap17] furthermore takes into account the human perception of noise and investigates active structural control measures to reduce the perceived noise based on an FEM model. Efforts towards auralization of cabin noise pursue a similar goal of evaluating the human perception [SM+22]. However, none of these works consider detailed aspects of the cabin, such as the secondary structure. Only Hesse, Allebrodt, and Rege [HAR20] take detailed representations of components such as sidewall panels and isolation materials into account when evaluating acoustic properties of the fuselage.

Cabin airflow simulation is another type of analysis, which can be found in literature. Simulation of the interior airflow using CFD requires very detailed and watertight representations of the cabin geometry [FRW11]. Schmeling et al. [SS+20] show an application where the thermal distribution of the cabin is analyzed to improve comfort and energy efficiency. The approach has also been applied for the assessment of aerosol dispersion to evaluate health risks [SS+21]. A simpler approach to evaluate thermal distribution using a two-dimensional cabin representation has been proposed by Gil et al. [GS+22].

4.2.3.3. Boarding and Evacuation

Passenger flow simulations for boarding, deboarding and evacuation scenarios are also often found in relation to cabin concept evaluation. Arguably, they can be understood as a subset of human model simulations discussed in section 4.2.3.1. However, compared to passenger comfort, the implications for operations and safety are more easily measured. On the one hand, improved boarding times lead to shorter turnaround times and therefore promise a better utilization of the aircraft. On the other hand, proof that the aircraft can be evacuated within a given time frame, must be provided by the OEM for certification as specified in CS 25.803 [EAS21].

It is therefore not surprising that many authors of cabin design tools, such as Fuchte, Dzikus, and Gollnick [FDG11] and Gobbin [Gob15], also look towards boarding and deboarding analysis to evaluate their designs [FDG11; Fuc14; GKB21]. The PAXelerate tool by Schmidt et al. [SE+16] is another example for this trend.

Fuchte, Dzikus, and Gollnick [FDG11] use a boarding simulation, shown in figure 4.2.11, to evaluate and compare different cabin and fuselage layouts for a given number of passengers. They show that the inclusion of an additional quarter door in a configuration for 200 passengers can yield an advantage in boarding time of up to two minutes. The introduction of a second aisle, on the other hand, can result in an advantage of over 12 minutes, at the expense of a larger cross-section. The capabilities of the tool are later expanded to support a complete turnaround simulation including deboarding, cleaning, refueling and boarding. Furthermore, a preliminary design tool is connected to assess the DOC for configuration with the modified fuselage. The results show that the time saved during turnaround almost never justifies the additional fuel consumption due to the increased cross-section [Fuc14] from an economical perspective. A similar study of the effect of the cross-section has been performed by Smeets [Sme17] using ParaFuse.

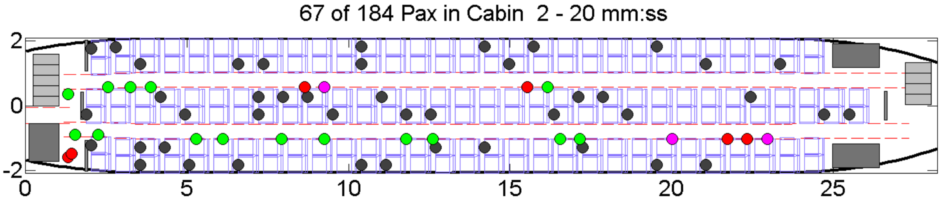


Figure 4.2.11.: Boarding simulation (from [FDG11])

Engelmann and Hornung [EH19] apply the PAXelerate tool, to assess various boarding strategies, such as random boarding, rear-to-front or window-to-aisle, for a given layout. They show that random boarding is among the fastest possible strategies, due to the high number of passenger interruptions for other strategies. Only the optimized strategy proposed by Steffen [Ste08] performs better, but is also significantly more difficult to implement, due to its low robustness towards deviations. No changes in the layout are proposed as a result of the analysis.

Contrary to the previous examples, Gobbin, Khosravi, and Bardenhagen [GKB21] investigate emergency evacuation situations. According to CS 25.803 a manufacturer must demonstrate the capability to evacuate the an aircraft of 40 passengers or more within a time frame of 90s, which is accomplished in large-scale tests. The primary goal of the study is to tune the simulation to get as close as possible to actual times measured by manufacturers. Again, no changes to the layout are considered. In terms of geometry, the locations, dimensions and orientations of the emergency slides must be known in addition to the aisles, seats and exit positions, as shown in figure 4.2.12.

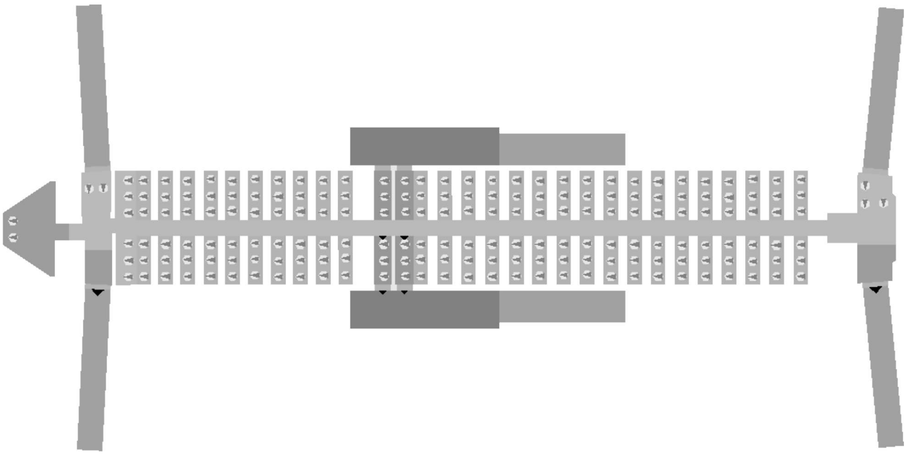


Figure 4.2.12.: Cabin model including slides for evacuation simulation (from [GKB21])

A common trait of all these examples is that the simulation is essentially performed using a 2D representation of the cabin floor layout. Seats and monuments amount to blocked areas. However, Engelmann, Kleinheinz, and Hornung [EKH20] propose to take into account seat geometry, to evaluate the available aisle width at various heights for a more detailed assessment. Gopani [Gop21] even demonstrates a fully realized three-dimensional approach by implementing the agent-based approach for the Unity video game engine. As illustrated by figure 4.2.13 the simulation includes not only the cabin, but also the airport waiting area and the bridge. However, unlike the other authors, Gopani only provides a model of a small portion of the aircraft cabin.

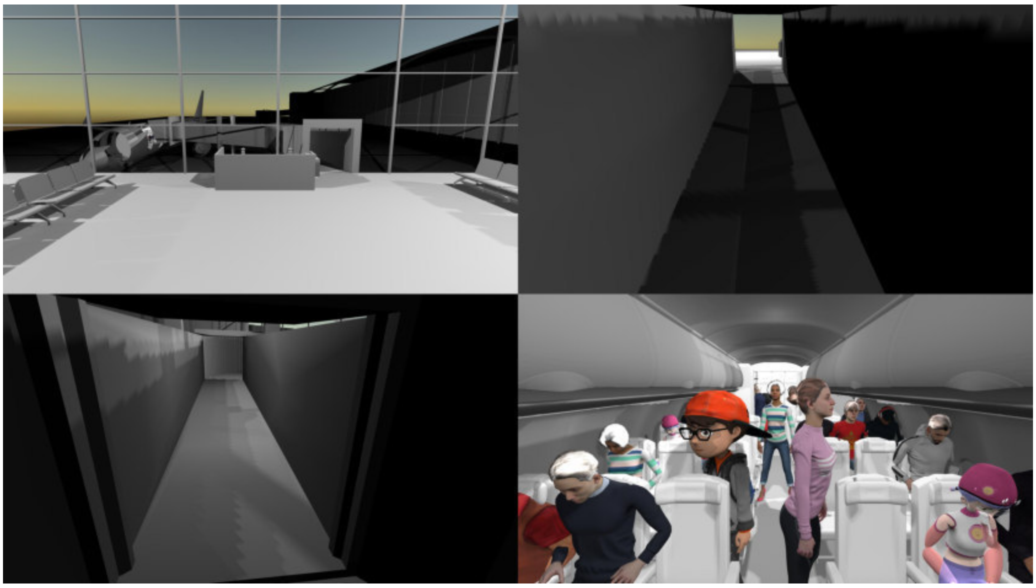


Figure 4.2.13.: Boarding simulation in Unity (from [Gop21])

More so than the representation of the geometry, the correct prediction of the passenger movement is a challenge in these analyses. It is often performed using commercial [GKB21; RR+21] or custom-built [Ric07; FDG11; App14; SE+16] agent-based environments for motion flow simulation. Fuchte, Dzikus, and Gollnick [FDG11] note the significant impact of user-provided estimates on how much time is required for certain actions. Gobbin, Khosravi, and Bardenhagen [GKB21] try to improve the accuracy of such assumptions w.r.t. the passenger behavior by investigating different age distributions where each age class has different associated properties.

4.2.4. Cabin analysis scenarios for industrialization

Aside from human factors evaluation, applications for cabin manufacturing and assembly planning will contribute substantially not only towards a more complete understanding of new aircraft design candidates but also towards a more fully realized implementation of co-design in industrial processes. To this end, Page Risueño and Nagel [PN19] propose a knowledge-based production modeling system to describe the assembly of an aircraft cabin, based on semantic web technologies (s. section 4.1.2).

It becomes clear that reliable predictions on manufacturing time and cost are possible only if detailed detailed information on available means of production and cost estimation relationships are available. This kind of information is usually highly confidential, which makes it difficult to validate methodologies for production modeling in a research setting. Therefore, Markusheska et al. [MS+22] apply the aforementioned methodology to a simplified assembly process using robots, which can be realized in a research lab. Srinivasan et al. [SM+21] describe the corresponding implementation at shop floor level. Live process data is made available digitally. In this way assessment of the process is possible to some degree.

As highlighted by Markusheska et al., product details are necessary as well as to process details, which can partly be provided by CPACS. This includes e.g. positions and dimensions of the cabin components. Furthermore, knowledge of constraints w.r.t. possible installation sequences, i.e. if one part must be installed first before another can be installed, is also required. Currently, this data must be provided in addition to the CPACS product description. For robot path planning more details on the cabin components and their structural attachment points must also be provided, to determine e.g. gripping points. These details are usually not available in preliminary design processes.

As a result, the industrialization analysis of aircraft cabins will also benefit if more details are made available in the early design stages. However, as a possible use case in the scope of this thesis, it is excluded, since a meaningful assessment of complete cabins cannot yet be realized due to the aforementioned limitations.

4.2.5. Summary

In this section a survey of computational analysis methods is provided, which may be applied to inform design decisions on the fuselage and cabin in an MDAO process. The different analysis types, which have been identified, are taken into consideration as candidates for examples to validate **working hypothesis 2** in the following.

On the one hand, example implementations for structural sizing using FEM analysis are given, which are typically performed using a GFEM mesh representing the thin-walled fuselage structure. The skin of the fuselage is modeled using two-dimensional shell elements, whereas the stiffeners and floor structure are represented by one-dimensional beams. Details of the beam cross-section are provided not in the form of explicit geometry but using beam properties.

That said, other types of FEM structural analyses to evaluate e.g. crash scenarios or vibro-acoustics require substantially more detailed models. Here, stiffeners are usually modeled explicitly using shell elements and the overall element edge length is reduced significantly. Simplified representations of cabin components may be considered in order to take into account inertia loads due to their mass in dynamic simulations.

On the other hand, examples for detailed cabin analysis to evaluate human factors are given. Interest in human in the loop VR, where detailed cabin component models must be available as CAD models or triangle meshes, is growing as availability of hardware and ease of use are improving. Textures to represent materials and animations of the kinematics e.g. of a overhead stowage compartment opening, greatly enhance the effectiveness, but require the use of video game engines to perform the simulation. On the other hand, simulation approaches based on a simplified a human model can also provide meaningful results e.g. w.r.t. certification or boarding duration requirements, if sufficiently detailed geometry is available.

The challenge, when bringing these analyses together in a single MDAO process, is the heterogeneity of the requirements for the geometry, which must be provided as the basis of the respective disciplinary analysis models. Whereas the GFEM generation can be implemented via the intersection of the outer fuselage surface with the frame and stringer definition planes, the models for crash simulations may require extrusion of the cross-sectional profiles to provide detailed frame and stringer geometries. Cabin components, which may simply be represented as a mass point in the structural analysis, must be available in utmost detail for human in the loop VR analysis. To make matters worse, it must be assured by the geometry process that the different representations of the product are all consistent with one another to ensure correct results in the MDAO process.

The integration of the analysis methods into the MDAO process consequently requires the management of both multiple levels of fidelity and multiple disciplines. From the perspective of a provider of geometry models the distinction between multi-fidelity and multidisciplinarity can conveniently be neglected, since different disciplines simply lead to different fidelity requirements for different geometric components. Therefore, the problem of providing geometry models for multi-fidelity and multidisciplinary analyses in MDAO processes is reduced to the problem of providing multi-fidelity geometry models, which consist of different combinations of component models at different levels of detail.

In the following section, knowledge-based engineering is introduced as a possible way of providing multi-fidelity models, while ensuring consistency of the underlying design assumptions.

4.3. Knowledge-Based Engineering

In the previous sections, two main difficulties for large-scale MDAO processes involving simultaneous design of the fuselage structure and the cabin have been identified:

1. As already stated in **working hypothesis 2**, consistent geometry to derive models for various disciplines must be available. Some authors e.g. La Rocca [LaR11] and Dorbath [Dor14] propose a multi-model generator to generate the required geometry based on a small set of parameters. As illustrated by the previous section, another dimension of this issue is the need for multi-fidelity, i.e. providing geometric representations of the same component at different levels of detail, as described e.g. by Böhnke [Böh15].

2. As pointed out in **working hypothesis 3**, there is a need for generation of new details. The generation must support any state of a given product model, taking into account all relevant and available product data. Centralized availability of this capability in an MDAO process, can serve to better orchestrate augmentation of inputs and requirements and reduce inconsistencies.

Knowledge-based engineering (KBE) is a methodological approach, which has the potential to address both these tasks simultaneously. According to La Rocca [LaR15]

> "KBE is engineering using product and process knowledge that has been captured and stored in dedicated software applications, to enable its direct exploitation and reuse in the design of new products and variants."

A number of followup questions arise from this definition, e.g. what exactly terms like product and process knowledge entail, how their meaning is different from product and process data or information and how knowledge can be formalized as executable software and deployed to design new products. These matters are discussed in sections 4.3.1 and 4.3.2. Afterwards, in section 4.3.3, a review of applications of KBE both for multi-model generation and design synthesis are provided. Finally, the relationship between KBE and MDAO is briefly discussed.

4.3.1. Terminology

For most people, the understanding of terms such as knowledge, information and data is informed rather by intuition than by concrete definitions. Therefore, the terms are treated as being somewhat interchangeable. However, in KBE there is a clear distinction between the terms. For the purpose of this thesis, the DIKW hierarchy [BCM04; Row07], which is shown in figure 4.3.1, is adopted. It is named after the initials of the different hierarchy layers: Data, Information, Knowledge and Wisdom.

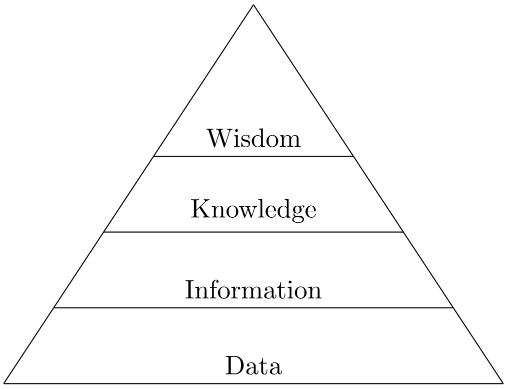


Figure 4.3.1.: The DIKW hierarchy (after Rowley [Row07])

Following the hierarchy, data can be understood to be isolated numbers or symbols without context. To gain information from data, it must be processed and put into context. Knowledge can be understood as connected data and information, which can be applied to make predictions. Finally, wisdom involves an understanding of the underlying principles within the knowledge.

According to Rowley, concepts become harder to implement using computers, as one moves up in the hierarchy. Therefore, only data, information and knowledge are taken into account in the following sections.

4.3.2. Methodology

As explained by Chapman and Pinfold [CP99], KBE was conceived by merging CAD technology with OOP and artificial intelligence (AI) techniques. Consequently, the initial understanding of a product in KBE is closely related to traditional CAD models, as described in section 4.1.1. The goal of KBE is to accelerate the engineering design process and eliminate repetitive work by collecting engineering expertise and design practices necessary for the creation of a given part and make them available as executable software. Chapman and Pinfold do not specify any particular form for this information. Instead a central corporate “knowledge base”, containing e.g. databases and external company programs is envisioned.

4.3.2.1. KBE life cycle and knowledge capture

An early attempt to formalize not only the knowledge representation but the larger framework necessary to successfully deploy KBE for actual product development is the Methodology and Tools Oriented to KBE Applications (MOKA) [Sto01]. The idea behind MOKA is to formalize the life cycle of KBE applications, so that they can be deployed in a corporate setting. The MOKA life cycle consists of six steps.

***Identify*:** Determine economical and technical feasibility for a KBE application. Provide technical concept specification.

***Justify*:** Assess necessary resources and potential risks. If all is well, the project receives go-ahead.

***Capture*:** Collect and structure raw data from experts. The outcome is an informal model of potentially used knowledge.

***Formalize*:** The captured knowledge is assembled into a formal model, which consists of a product model and a design process model, by a knowledge engineer.

***Package*:** The knowledge from the formal model is translated to source code, from which a KBE application is assembled.

***Activate*:** The KBE application is deployed and users are trained.

Whereas many of the surrounding steps are aimed towards deployment of KBE on an enterprise level, the detailed description of the *Capture* step in MOKA provides very concrete background on the KBE system itself. Here, it is specified, which kind of information should be collected for the informal model, by introducing the ICARE forms. The acronym is derived from the initials of the five different categories for capturing engineering knowledge provided by the forms:

***Illustration*:** Supplementary information, e.g. example cases, plots, etc. to support understanding of the other forms.

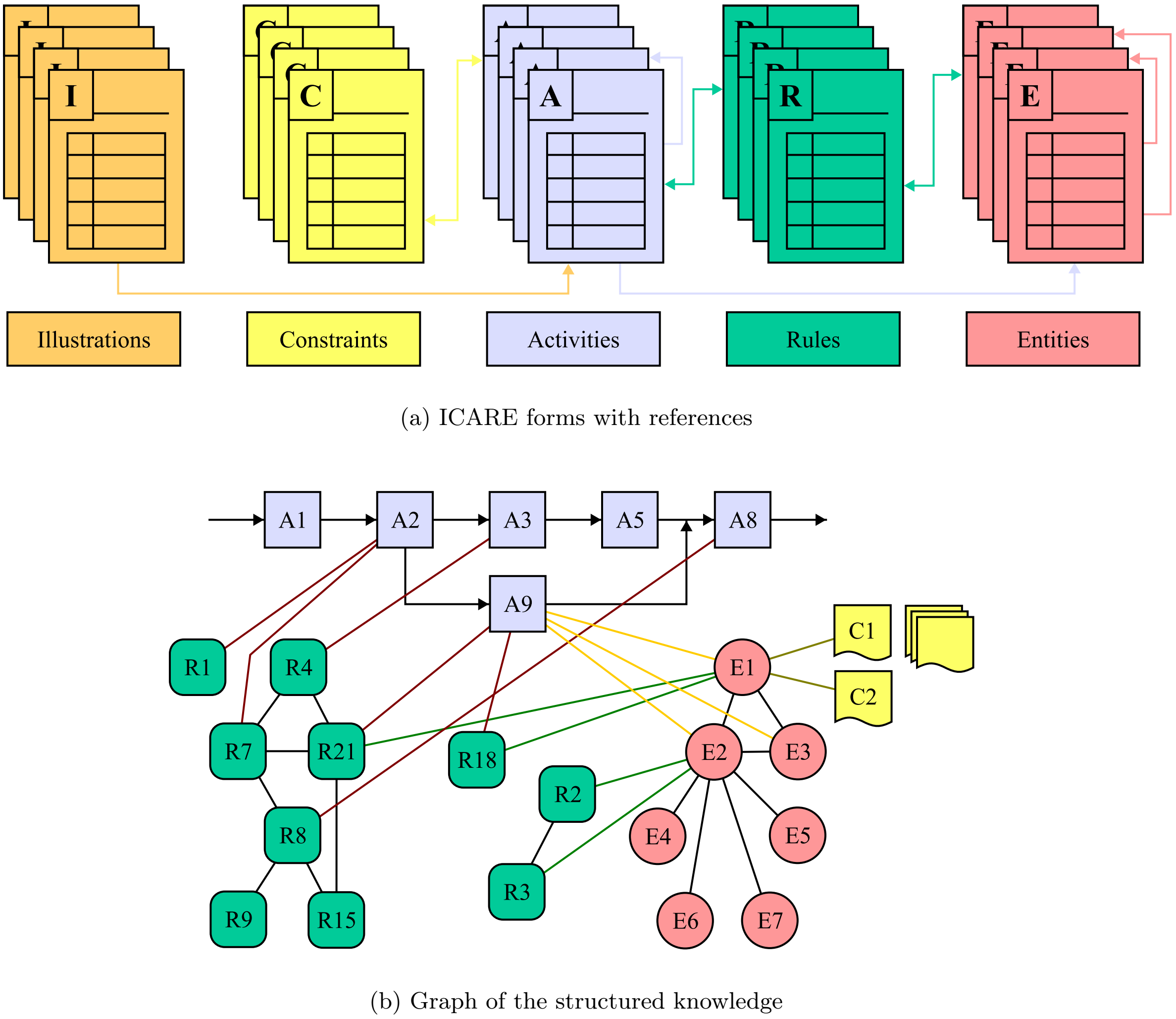
***Constraint*:** Specification of limitations or boundary conditions for entities.

***Activity*:** Description of design process steps.

Rule: Description of rules which affect the design process.

Entity: Description of aspects of the final product including form, function and behavior.

The ICARE forms furthermore provide the means to link different forms. For instance, an activity will provide a reference to related entities and vice versa, as illustrated by figure 4.3.2. It is shown, how the dependencies recorded in the forms naturally form a graph representation of the captured knowledge.



(a) ICARE forms with references

(b) Graph of the structured knowledge

Figure 4.3.2.: Relationships between ICARE forms and resulting graph structure of the knowledge (after [Sto01])

In the formal model, the ICARE forms are grouped into the product model, which contains the *Entity* and *Constraint* forms, and the design process model, which contains the *Activity* and *Rule* forms. The *Illustration* form is kept for documentation only, but is not translated into a formal model. Referring back to the KBE definition by Chapman and Pinfold, the CAD model can be understood as the product model within the formal model in MOKA, whereas the engineering expertise and best practices are represented by the design process model.

4.3.2.2. Commercial off-the shelf KBE solutions

The application of commercial off-the-shelf (COTS) solutions holds the promise of reducing long-term cost by outsourcing the development effort. The MMG presented by La Rocca [LaR11] is built using the commercial KBE program ICAD. Whereas ICAD has since been discontinued [LaR15], comparable solutions are available today, such as Genworks GDL [CL07; Gen19b], ParaPy [DB23] and CATIA KnowledgeWare [3DS23b], the latter of which is marketed as the successor of ICAD.

However, the COTS approach also entails some drawbacks. These include a reduced flexibility to integrate new libraries, including e.g. a differentiated version of the OCCT CAD kernel as discussed in section 2.2.3 to support high-fidelity gradient-based MDAO. It can also be argued that the majority of the development effort for a KBE system will be spent on the design knowledge rather than the framework. Since COTS solutions provide only the framework, the actual benefit in development effort might therefore be relatively small. Furthermore, storing critical knowledge inside a commercial system makes corporations susceptible to vendor lock-in, which may severely increase long term cost.

4.3.2.3. Elements of a KBE system

In contrast to the focus on organizational structure, which characterizes MOKA, La Rocca [LaR11] more closely examines the implementation aspects on the software side for his MMG. He establishes an understanding of KBE systems as a further development of knowledge-based systems (KBSs), i.e. computer applications that use knowledge to solve problems in a specific domain [Neg11], thus placing a stronger emphasis on the aspects of AI. Five key components for a KBE system, which are introduced in the following are identified. A different terminology from La Rocca is adopted in this thesis for the data and knowledge base components to avoid confusion with established concepts in computing like relational databases. The components are instead referred to as data and knowledge *repository* in reference to the well-established repository pattern (s. e.g. [PG20]). The original terms used by La Rocca are given in brackets.

Knowledge repository (knowledge base) All engineering rules are stored in the knowledge repository. The distinction between activities and rules from MOKA is dropped altogether. Instead, La Rocca breaks down the rules into five subcategories:

- *Logic rules (or conditional expressions):* If-then-else type relationships.
- *Mathematical rules:* Simple arithmetic relationships e.g. addition, subtraction, multiplication etc.
- *Geometry handling rules:* Generation and manipulation of geometric entities.
- *Configuration selection rules (or topology rules):* Control the number of instances of an object in the object tree.
- *Communication rules:* Link to external data, e.g. in databases or lookup tables.

ICAD provides the ICAD Design Language (IDL) to describe engineering rules, which is based on the Common Lisp programming language. Lisp has a few unique features that make it a popular choice for rule definition in commercial KBE tools. One of the key features is its support for a declarative programming style, where code need not be written in the order of its execution. Instead the knowledge repository can be understood as an unstructured collection of relationships between parameters.

That said, downside of IDL and even Lisp is their niche status, compared to more popular languages such as Python. As a result, capable knowledge engineers to develop rules will be difficult to find in the labor market and new candidates likely require extensive training. This puts the companies at risk of expert lock-in, where competences and tools may get lost or become unusable upon the departure of very few key experts.

Workspace or data repository (data base) Unlike the knowledge repository, which is meant to be stored long-term, the workspace is a short-term data storage for the product design problem at hand. As new results are made available by rules from the knowledge repository, they are added to the workspace. Negnevitsky [Neg11] refers to a data repository, instead of a workspace, in the context of general KBSs. This term provides a better analogy to the term knowledge repository, given the terminology established in section 4.3.1.

Due to the implementation based on a CAD environment, the data repository in the case of La Rocca provides many entries, which are essentially parameters of a CAD model. However, he also applies it to store and trace non-geometric data such as masses. It has already been discussed in section 4.1 that a data-centric product model such as CPACS could in fact be better suited in such a case. This is corroborated by La Rocca, Langen, and Brouwers [LLB12], who introduce an interface to CPACS in a later version of the MMG.

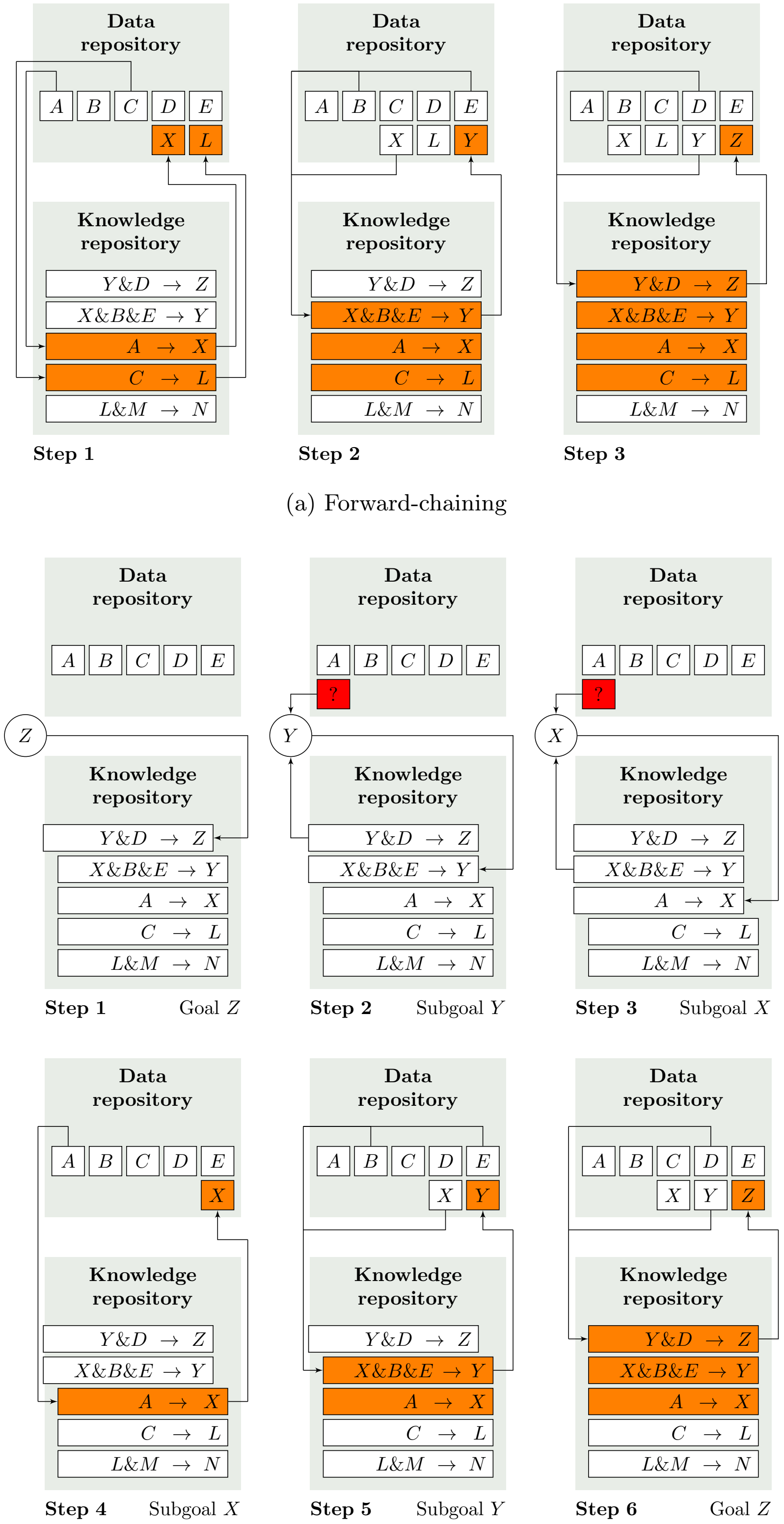
Inference engine The inference engine is the core of the KBS. It provides reasoning mechanisms, which attempt to deploy applicable rules from the knowledge repository in a structured way to solve a given design task. The task is formulated based on a user query requesting a certain piece of information and the data that is available in the data repository. Given these boundaries, a combination of rules is found, which can provide the requested data as an output when executed in sequence by applying reasoning algorithms.

Negnevitsky [Neg11] introduces forward- and backward-chaining as common reasoning algorithms. In forward-chaining, eligible rules, which only require inputs that are already provided in the data repository, are identified and executed. The procedure is repeated iteratively. Since new data is added to the data repository with each iteration new rules become eligible for execution. The problem is solved if a rule can be triggered, which can provide the information requested by the user. An illustrative example for the approach is given in figure 4.3.3a. Here, the goal is to determine the unknown variable Z by evaluating the available rules in the knowledge repository. A drawback of the forward-chaining approach is that it is not known, if a given function will help progress towards the requested data. Therefore, forward-chaining may produce a lot of unnecessary calls to rules, exemplified by the rule to compute the variable L, which is called in figure 4.3.3a, but does not contribute to the goal of determining the unknown variable Z.

To address this issue, backward-chaining, which is also referred to as goal-driven or lazy inference, starts the process by identifying a rule, which can provide the requested piece of information. If not all required inputs to that rule are available in the data repository, it is stored and further rules, which can provide the additional missing inputs are identified. This procedure is repeated iteratively. Only once all inputs to a rule are available in the data repository, it can be executed to provide additional data repository entries. With new data available, it is reevaluated whether any of the functions in storage can be executed. Figure 4.3.3b provides an illustration of the approach. Visibly one less rule was triggered than in the forward-chaining example in figure 4.3.3a, as the unnecessary computation of the variable L could be avoided.

If a solution is found by the inference engine, it is guaranteed to be consistent with the rules in the knowledge repository and the inputs provided in the data repository. However, depending on the available rules and data, the inference engine may not be capable of solving a given problem.

Explanation subsystem If the inference engine can provide a result, the result can be explained, i.e. the exact sequence in which rules have been executed can be retraced, along with the related inputs and outputs. Making this explanation available to the user not only helps to understand the reasoning behind successful solutions, but also to detect, which data or rules might be missing in failed solutions. Consequently, an explanation subsystem is considered an essential component of any KBE system according to La Rocca.



(a) Forward-chaining

(b) Backward-chaining

Figure 4.3.3.: Illustrative examples of inference mechanisms (after [Neg11])

User interface Finally, La Rocca points out the need for a UI to interact with the KBE system. To specify requirements for the UI, he identifies three user roles: The *domain expert* or knowledge engineer to build and maintain the knowledge repository, the *software engineer* to develop the inference system and the *end user*, who will apply the system to generate product designs. The roles are distinct and should not interfere with one another. As such, different views of the system must be provided for each task.

4.3.2.4. Structuring knowledge

La Rocca [LaR11] acknowledges that the low-level concepts contained in both the data and rule repository of KBE applications often do not correspond to the intuition of the end users, e.g. preliminary design engineers. He therefore proposes ways to wrap the knowledge contained in the rules in higher level objects, which he calls high-level primitives (HLPs) and capability modules (CMs), to make the application more accessible.

HLPs, named as a counterpart to low-level primitives such as cylinders or boxes found in CAD, are designed to represent different components of the aircraft. La Rocca proposes HLPs e.g. for wings, fuselages and engines. Following an OOP paradigm, all the rules necessary to build the wing surface are collected as methods in the HLP class definition. As a result only the most basic top level parameters are exposed, which are familiar to design engineers. A beneficial side effect is that the parameters usually also correspond to meaningful design variables for optimization. Additional capabilities can be packaged with HLPs, including the generation of an internal structure [Laa08] to encapsulate the complete design of a given component.

Moreover, different HLPs can be deployed to support different aircraft architectures. For instance, the MMG provides support for conventional tube-and-wing, BWB, SBW and boxwing configurations. Knowledge can be shared among different classes to some extent by introducing abstract base classes, from which the HLPs can inherit. However, this approach quickly results in a complex object-oriented structure.

A disadvantage of HLPs is an increased bias of the model and thus a reduced design freedom. For instance, 4-digit NACA profiles are assumed for wings in the MMG, excluding modeling of CST profiles. Therefore, the existing HLP would either have to be adapted, increasing the complexity of the HLP, or a new HLP for CST profiles would have to be created, which could lead to a large number of very similar HLPs, which would again impair accessibility.

Aside from HLPs, CMs have been introduced to capture more complex procedural knowledge required to derive analysis models for multidisciplinary analysis, e.g. the segmentation of the wing surface, or the computation of the tank volume. While the rules are once again collected in classes, they cannot be instantiated on their own, but only in terms of a HLP. Therefore, CMs can be understood as operators on HLPs and thus ultimately constitute a more complex version of a rule.

4.3.2.5. Graph-based design languages

Expanding on the idea of a graph-based system representation found in MOKA, Schmidt and Rudolph [SR16] propose the application of a graph-based design language based on UML (s. section 4.1.2). The corresponding KBE system architecture is given in figure 4.3.4.

Vocabulary, i.e. parametric descriptions of components provided in UML, which mirror the concept of HLPs, and rules, which can be provided as executable bits of Java code [Mot16], form a knowledge repository, which the authors refer to as a production system. The inference mechanism, referred to as design compiler, receives the system as input and builds a design graph, which serves as the central data format. The design graph provides the input for a host of dedicated builder modules, which can generate consistent CAD geometry, FEM and thermal analysis models as well as subsystem models and routing.

The approach enables not only vendor-agnostic support of different CAD environments, but also provides a basis for the implementation of a multi-model generator. Furthermore, results from analyses

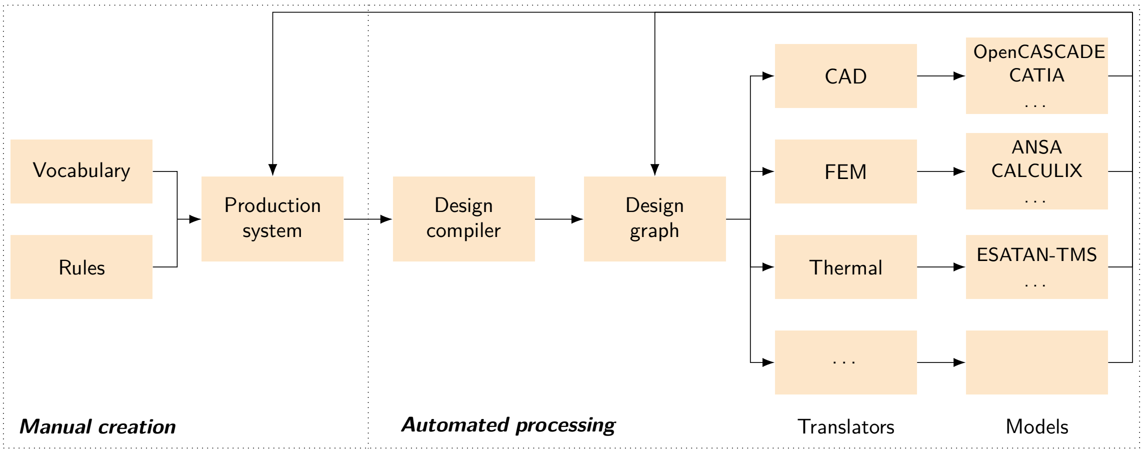


Figure 4.3.4.: KBE system based on graph-based languages as proposed by Schmidt and Rudolph [SR16]

can be fed back to the production system in a similar way to an MDAO system. In this context, the role of the design graph is similar to that of CPACS in the distributed design workflows introduced in section 2.2.4. That said, no information is available if the multi-fidelity requirements for model generation are propagated back to the design compiler, i.e. if the design compiler can be configured to compute only those aspects of the design, which are necessary for the subsequent model generation. This type of feature can significantly enhance the efficiency of the model generation.

4.3.3. Application examples of KBE

Successful applications of KBE have been shown e.g. in the automotive [CP99; CP01], aerospace [LKT02; Dor14] and civil engineering [Sin14; HB21] fields. In the following, a few examples of applications are discussed, which relate to the two target use cases, stated in the beginning of this chapter. First, the design synthesis aspect is considered in section 4.3.3.1, then multi-model generation is discussed in section 4.3.3.2. Finally, in section 4.3.3.3, the relationship between KBE and MDAO process architecting tools is discussed.

4.3.3.1. Product and component design synthesis

Early motivation for the development of KBE was the automation of repetitive design tasks. Chapman and Pinfold [CP99] show a simple example, where the frame structure of a car is created automatically based on the outline.

Rentema, Jansen, and Torenbeek [RJT98] are among the first to propose application of KBE techniques in conceptual aircraft design. A first implementation using ICAD is presented by La Rocca, Krakers, and Tooren [LKT02]. Laan [Laa08] later shows an application of ICAD to generate wing movables and structural details. In DARfuse, Brouwers [Bro11] presents a fuselage HLP, i.e. a parametric model capable of creating fuselage and cabin details, implemented using Genworks GDL. Based on the work of Laan, Kulkarni et al. [KL+17] show a KBE system for a detailed rudder implemented in ParaPy. ParaFuse [Jon17] also leverages ParaPy to provide an updated fuselage HLP, based on the work by Brouwers in DARfuse. However, the main focus is on the parametric description and regulatory constraints, rather than the KBE methodology. In terms of the fuselage structure, only the seat rails are taken into account.

Another example is shown by Munjulury [Mun17], who provides a very detailed overall aircraft model including interior, structures, engines and lading gears. The model is built using CATIA, leveraging the available KBE features i.e. UDF, Power Copy and Knowledge Patterns. The approach is similar to the parametric-associative modeling described by Ledermann, Ermanni, and Kelm [LEK06] or Wenzel, Sinapius, and Gabbert [WSG11] mentioned in section 4.2.2.1, but the model is meant to cover a larger scope, also taking into account e.g. aspects of the cabin. Munjulury acknowledges the additional time and expertise necessary to build and update the system, including the need for proficiency in multiple programming languages.

Motzer [Mot16] demonstrates capabilities to provide fuselage and designs at a very high level of detail down to subsystem level by applying graph-based design languages. He also takes into account structural information, though strictly as input, thus neglecting the mutual effects of fuselage structure and cabin design.

4.3.3.2. Multi-Model-Generation

The application of KBE has always been closely related to the automatic generation and evaluation of analysis models. Based on their frame design, Chapman and Pinfold [CP01] show that FEM models can be derived, where the walls are modeled as shells. To this end, the full model representation, which has thick walls is simplified to a model with a surface representation of the wall.

In aircraft design, the need to support multiple disciplines, all of which have different model requirements, has spawned multiple attempts to create a MMG using KBE techniques, as described by La Rocca [LaR11]. As illustrated by figure 4.3.5, La Rocca demonstrates the application of the MMG to derive aerodynamic and structural analysis models. For the aerodynamic method, both panel methods and CFD are supported. Notably, the generation of details for multi-fidelity analysis is mentioned as well.

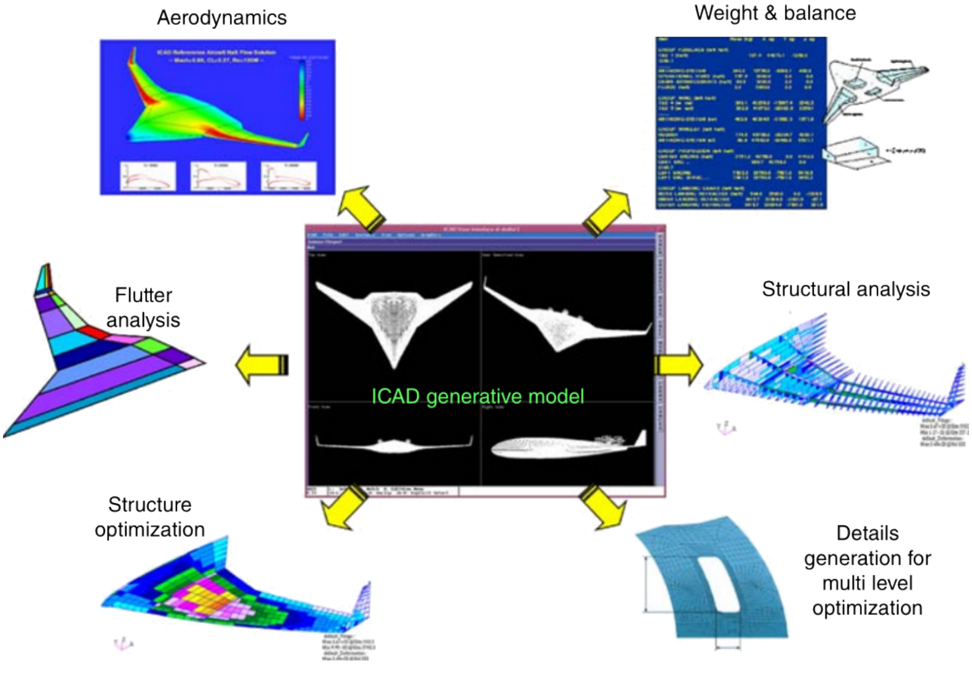


Figure 4.3.5.: MMG applications (from [LaR11])

For the wing, Dorbath [Dor14] proposes a similar approach, where a MMG is used to derive a large number of tool-specific parameters which serve as inputs to different analysis tools, based on a small and manageable number of user-defined input variables. The specific analysis tools mentioned by Dorbath are AVL (Athena Vortex Lattice, Drela and Youngren [DY24]) for aerodynamics and ANSYS for structures. The MMG provides a knowledge base implemented in MATLAB to perform the translation. However, it does not strictly qualify as a KBE application, as Dorbath opts for a procedural implementation, instead of the declarative approach advocated by La Rocca. The element of the inference engine, which is an integral component of a KBS, is therefore eliminated.

Bhagat and Alyanak [BA14], Alyanak et al. [AD+16], and Dannenhoffer and Haimes [DH16] demonstrate multi-model generation using the ESP. They apply an approach similar to HLPs to generate and combine the individual aircraft components, which is implemented using the scripting language of ESP. Like Dorbath, they adopt a procedural implementation. Once again, structural and aerodynamic models are derived.

For the fuselage, Tooren and Krakers [TK07] demonstrate the generation of structural, acoustic and thermal models of a fuselage barrel for coupled analysis using Abaqus CAE. Like La Rocca, the authors use ICAD to generate the geometry. That said, a very basic cylindrical barrel geometry is assumed, which does not properly reflect the geometric complexities of complete configurations.

4.3.3.3. Relationship to MDAO workflow generation

The techniques mentioned in 2.2.4 to assemble MDAO workflows automatically, bear close resemblance to the approach used in KBE. Gent, La Rocca, and Hoogreef [GLH18] propose the CMDOWS container format to describe MDAO systems. Several stages of the synthesis of the workflow are addressed. The first stage is the tool repository, where parameters, separated into design variables, objective variables, constraint variables and state variables, and executable blocks, which can be either a mathematical function or a design competence that performs an unknown black-box operation, must be provided. Each executable block provides sets of input and output parameters, which are required and provided by the function respectively. If the executable block describes a design competence based on CPACS, the corresponding parameters can provide references to data points in a CPACS data set using XML XPath notation. In relation to the aforementioned elements of KBE, the design competences can be considered the knowledge base, whereas, the parameters or by extension, the CPACS data set, can be considered the data base.

From the tool repository, the MDAO problem and an MDAO solution strategy can then be derived using graph-based techniques, which have been implemented in the tool KADMOS [Gen19a], to assemble the executable MDAO workflow. This corresponds to the inference engine in KBE. The final MDAO system graph can then be exported to a PIDO framework, e.g. RCE (s. section 2.2.4), which provides the explanation subsystem and the user interface.

The approach of comparable tools by Gallard et al. [GV+18] and Page Risueño et al. [PB+20] can be mapped to the elements of KBE in a similar way.

4.3.4. Summary

Knowledge-based engineering techniques, which combine a set of rules provided in a declarative manner and an inference engine to dynamically fill a product data base, have been applied successfully to both create new aircraft and fuselage designs from scratch and provide disciplinary models for aero-structural analysis. For the aerodynamic analysis models, multi-fidelity aspects have also been considered e.g. by La Rocca [LaR11], who provides models both for panel method and CFD analysis. As such, a MMG approach based on KBE methodology is promising w.r.t. the **research hypothesis**, particularly **working hypotheses 1 und 2**, to enable dynamically reconfigurable product design and taylored-fidelity model generation processes based on a central product model. Product design here could also entail the generation of new details as described by **working hypothesis 3.**

Fundamentally, KBE is furthermore well-suited to react to changes in the overall system architecture as required by **working hypothesis 4**, which is illustrated by the high-level primitives concept proposed by La Rocca. However, due to the high number of different options, the system can become highly complex very quickly, when using the proposed object-oriented approach.

None of the authors above demonstrate a fully integrated fuselage design approach, which takes into account the interconnections of OML design, structural design and cabin design. Also, due to the focus of the multi-model generation on aero-structural analysis, no example for a fuselage-specific MMG is found, which can support the types of analysis presented in section 4.2. There is also no indication, whether the requirements from the MMG are propagated back to the design process, to avoid the generation of details, which are not necessary for a given analysis model.

4.4. Contributions to the State of the Art

Based on the available research discussed in this chapter, several contributions of the work presented in the subsequent chapters of this thesis to the state of the art can be identified.

Applying a KBE approach as described in section 4.3, the implementation of a comprehensive set of design and modeling rules for the fuselage is described in chapter 6, linking generation of the OML, structural design and cabin layout in such a way that enables consideration of complex interdisciplinary connections. As shown, the KBE methodology implicitly results in a graph-based formulation, as described by **working hypothesis 1**, where product parameters are connected via rules expressing mutual dependencies. A novelty of the KBE implementation in chapter 5 in this thesis is that it is designed around the parametric foundation provided by a central data model, i.e. CPACS. The design capabilities may operate directly on the central model, but are also coupled to KBE formulation of a parametric modeling engine within a single knowledge-based system. This allows for simple and efficient integration of geometry-dependent design rules. As discussed in section 4.1, the parametric modeling engine furthermore serves to bridge the gap between data-centric and geometry-centric product representations.

The literature shows that KBE is well-suited for multi-model generation capabilities in an MDAO context as discussed in section 2.2 and stated by **working hypothesis 2**. However, most applications lie in the domain of OAD and aero-structure coupling. All of the applications shown for the fuselage, such as the GFEM generators, are implemented in a procedural manner, rather than using KBE. It is shown in section 7.3 of this thesis, how the KBE approach in chapter 5 can be applied as a multi-model generator to provide tailored geometry models for disciplinary analyses related to the fuselage, which have been discussed in section 4.2. This includes the creation of detailed analysis models of the cabin, e.g. for human factors evaluation. So far, such models have not played a role in early design processes. Instead, all studies shown rely on manually or semi-automatically created models, created in a subsequent step to the overall design.

Aside from meeting multidisciplinary and multi-fidelity geometry requirements, consistency and efficient automation are key aspects. The KBE approach presented in this thesis enables tracing of the upstream dependencies for given geometry components, which means that, as stated in **working hypothesis 3,** only design details necessary to the model generation need to be evaluated. This is a direct consequence of the integration of the parametric modeling engine as part of the KBE system. A single shared knowledge repository based on a central data model furthermore enables the dynamic linking of rules from different disciplines without relying on a prescribed sequence of disciplinary design contributions. Once generated, data, including geometric objects, is cached and thus available for later re-use, also by other disciplines, which ensures consistency and improves efficiency.

Finally, it is shown in sections 6.3 and 7.2, how the knowledge-based approach provides the means to quickly implement new rules to support different aircraft system architectures, while re-using as much of the existing knowledge as possible. This helps improve the accuracy, efficiency and flexibility of aircraft design processes for novel configurations, which have been discussed in section 2.1. It is stated in **working hypothesis 4** that it should be possible to include novel configurations by manipulating the knowledge graph. However, most examples in the literature are either specialized KBE systems for specific architectures or based on highly generic descriptions, which lack the necessary specificity to describe products in the level of detail necessary for the types of analysis from section 4.2. In contrast to this, suitable means of structuring and modularizing the knowledge are introduced in section 5.2 this thesis as well as mechanisms to minimize the necessary changes to the system and maximizing the exploitation of applicable existing knowledge.

5. Extensible implementation of the KBE methodology

In this thesis, the implementation of the KBE methodology for the fuselage and cabin design tasks in the KBE application Fuselage Geometry Assembler (FUGA) using the Python programming language [Pyt23] is discussed. Several factors have contributed to the decision to implement the methodology from scratch rather than using one of the existing COTS solutions discussed in section 4.3.2.2.

On the one hand, a major strength of Python is its capability to interface with libraries from different languages. In this way, CAD or mesh visualization capabilities which are readily available via the OCCT and VTK libraries can be leveraged easily. On the other hand, Python also provides versatile libraries for data handling, which facilitates the interaction with data e.g. from CPACS. Its wide popularity[1] furthermore reduces the risk of expert lock-in.

The choice of Python as programming language eliminates all existing solutions except for ParaPy. However, available publications show a strong focus of ParaPy on encapsulated parametric component models using objects according to the HLP approach [Sch17; Jon17; Plu21], whereas the goal of this thesis is to investigate more flexible extensible graph-based KBE systems as stated in **working hypothesis 1**. In addition, Schaft [Sch17] lists a number of missing features in the geometry kernel, required to automate the design and model generation of aircraft structures. Thus, the pythonocc library [Pav20] is adopted instead in the scope of this thesis, which provides bindings to the OCCT kernel in Python.

Different capabilities of FUGA have been implemented in separate subpackages as shown in figure 5.0.1. This chapter is predominantly concerned with the implementation of the underlying KBE methodology, which is found in the `fuga.core` subpackage. First, in section 5.1, the implementations of the different KBE system elements listed by La Rocca [LaR11] are introduced. Then, in section 5.2, the approach to structuring knowledge in the remaining subpackages is discussed, which is based on a plugin architecture leveraging user-defined graph composition. Furthermore, ensuing possibilities to assemble use-case specific design systems are considered.

```
fuga
 ├─ core
 ├─ cpacs
 ├─ geometry
 └─ design
```

Figure 5.0.1.: Subpackage structure of FUGA

5.1. Knowledge-based engineering system implementation in FUGA

In this section, the implementation of the KBE system in FUGA is introduced.figure 5.1.1 establishes the different process stages of the system, which will be revisited throughout this section. In general, a setup and initialization phase and an application phase can be distinguished, based on the level of user

[1] Python has been the most popular programming language since 2018 according to Carbonnelle [Car23]

interaction. Whereas not much user input is required during the system setup and initial data update steps, the subsequent steps rely on user requests for the postulation of a specific design problem, which can then be solved. As such, it is up to the user to guide the design process.

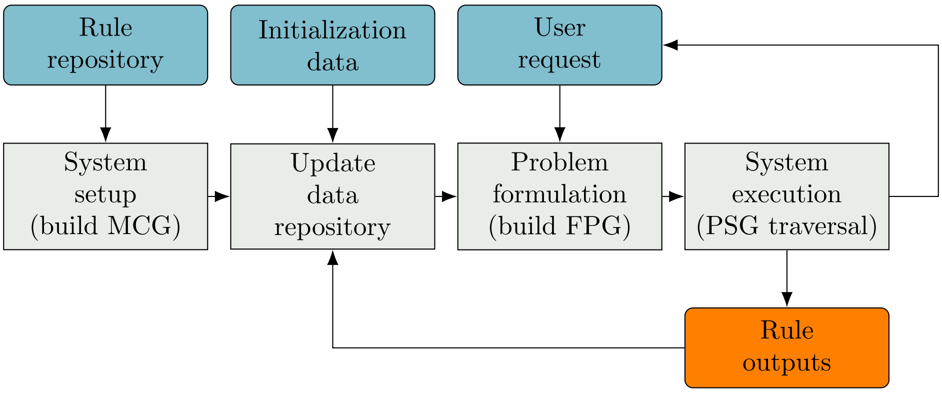


Figure 5.1.1.: KBE system process stages

In section 4.3.2.3, the constituting elements of a KBE system after La Rocca [LaR11] have been listed. The actual implementations of the concepts in FUGA are presented in the following subsections. As mentioned before, the terminology of data and knowledge *repository* is adopted in this thesis, instead of data and knowledge *base* as used by La Rocca.

To begin with, the implementation of the *data repository* is discussed in section 5.1.1. The data-centric exchange format CPACS plays a fundamental role in this context, however, the geometry-centric product description, too, must be accounted for.

Next, a simple protocol for the formulation of functional rules using Python objects is proposed in section 5.1.2. The goal is to at once provide a maximum degree of freedom to the knowledge engineer, i.e. the person concerned with the actual implementation of the rules and make integration into the overall system as simple as possible. By collecting rules written in adherence to the protocol, the *knowledge* or *rule repository* can be assembled.

Several key stages of the third key element, the *inference engine*, are shown in figure 5.1.1. During system setup, all the connections provided in the rule repository are identified. Based on the available data in the data repository and a user request for additional data, a problem is formulated, which is then solved by executing the system, i.e. performing a traversal of all eligible rules. This step can be repeated given a new user request. The implementation of the inference engine in this thesis leverages a graph representation for the different steps. It is discussed in section 5.1.3. The approach is closely related to the automatic collaborative MDAO workflow integration techniques mentioned in section 2.2.4.

Finally, in section 5.1.4, it is discussed how the need for an *explanation subsystem* and a *user interface*, which may be used to communicate the user requests to the system, can be satisfied by using interfaces to established file formats and third-party software, avoiding a time-consuming implementation from scratch.

5.1.1. Data repository for data-centric and geometry-centric product description

In essence, the data repository provides a mapping between the unique name or address of a variable in the KBE system and its value. In Python, the standard data structure for mappings is a dictionary, which efficiently links a hashable key, e.g. a string, and a value, which can be an arbitrary Python object. A Python dictionary thus provides a good foundation for the implementation of the repository. The data repository supports several basic operations, which are inherited from dictionaries:

- `add`: Add a new key-value-pair.

- `update`: Change the value for a key, which already exists in the repository. Fall back to `add` if key does not exist.
- `delete`: Remove a key-value pair from the repository.
- `get`: Retrieve the value for a given key. If the key is not found in the repository, either a null value is returned or an error is raised.

Using these operations, the data repository can gradually be filled with data, as it becomes available. Typically, once the value of a variable has been computed, it is added to the repository, which acts as a cache. This means, the variable does not need to be recomputed if it has to be used again, but can simply be looked up in the repository. This ensures that expensive operations are not executed unnecessarily often. However, a change in value of one variable, i.e an update, can also render other results invalid, if they depend on the changed variable. These results must be removed from the repository using the delete operation.

A data repository based on a Python dictionary provides significant freedom w.r.t. what types of data can be stored. However, without a guiding structure the data quickly becomes difficult to navigate and product data exchange will be tedious. Therefore, a common product description must be established as a means to organize the data. In section 4.1.2, the common parametric aircraft configuration schema CPACS was identified as a promising choice for data-centric or parametric description of aircraft, due to its vast scope, high maturity, and relative ease of use compared to other formats. It has therefore been selected as the basis for the KBE system to store the ground truth of the product. This means that CPACS acts as the source for all modeling activities and a storage and exchange container for all design results.

To establish a link between the data in the CPACS XML structure and the dictionary-based data repository, all entries of the CPACS tree must be assigned a unique string as an identifier. Here, the XPath notation of XML, which allows for navigation of the XML tree hierarchy using a notation similar to file paths in computer operating systems, can be applied. An example is given by the following listing, which provides the path to the first fuselage instance in a given CPACS file:

```
/cpacs/vehicles/aircraft/model/fuselages/fuselage[0]
```

That said, some adjustments are made to the approach when parsing CPACS, in order to take advantage of some advanced features of the Python programming language. The main reason is that CPACS adds the concept of unique identifiers (uIDs) to XML. The uIDs in CPACS have a similar role to foreign keys in relational databases, allowing for references to an XML node from anywhere else in the tree. This allows for the introduction of connections, which break up the strict hierarchical structure of XML. Additionally, there are many instances of collection nodes to be found in CPACS, i.e. nodes, which contain an arbitrary number of children of the same type. For collection nodes, representing the data contained as a table can be more intuitive and practical, than a tree structure.

The pandas package [McK10; Num23] is well-established to work with tabular data in Python, which is stored in so-called data frames. It supports merge operations between data frames, which allow for lookup of uID references to be implemented in a very efficient way. This means that in many cases the entries of collection nodes are not individually stored in the data repository, but in a single data frame representing the entire collection (s. also [WPK17]). An example is provided in table 5.1.1.

Table 5.1.1.: Pandas data frame of the wing node for an example configuration

	name	description	parentUID	symmetry	componentUID
uID					
htp	htp	This is an HTP	fuselage	x-z-plane	AircraftModel
vtp	vtp	This is a VTP	fuselage	NaN	AircraftModel
wing	wing	This is a wing	fuselage	x-z-plane	AircraftModel

The data frame is identified via a generalized notation the XPath of the collection node:

```
/cpacs/vehicles/{aircraft}/model/wings/wing
```

The generalization is necessary in order to take into account that CPACS provides separate but for the most part identical nodes for describing rotorcraft and fixed-wing aircraft. To adapt to this, the value of `{aircraft}` can be customized by the user depending on whether a fixed-wing or rotorcraft use case is evaluated. No index is given to highlight that all XML nodes of this type are contained in the table.

As discussed in section 4.1.3, a geometry-centric model will be required for the generation of the analysis model and certain design tasks, in addition to the data-centric product model provided via CPACS. Consequently, geometric data must also be stored in the data repository, along with the CPACS data. To this end, the Python bindings for the OCCT CAD-kernel provided by the pythonocc package allow for OCCT geometry to be represented by Python objects, which can in turn be stored as values in the repository. Since the geometry models are not included in the XPath address space of CPACS another identifier must be chosen by the knowledge engineer. Using a path based notation similar to XPath providing the tool and subpackage name has proved to be a sensible approach to keep the data repository organized. Borrowing from semantic web terminology, the path used as data repository key will be referred to as *uniform resource identifier (URI)* in the following.

5.1.2. Rule repository enabled by function-based rule protocol

The rule repository is a collection of all rules available in the system. To facilitate processing by the inference engine, all rules should be formulated according to a common standard. The design of this rule formulation standard is motivated by the need to make it sufficiently simple as to allow inexperienced programmers to implement rules quickly. At the same time, it should also grant sufficient freedom to more experienced programmers. To this end, a rule protocol[2] has been established, to provide a basic template to implement and integrate new rules quickly.

The protocol needs to fulfill several fundamental requirements. First of all, the rule needs to be provided as an executable piece of software. In Python, the simplest way to accomplish this is a function. A function provides a single return value, which needs to be assigned to a repository entry. The URI of this entry must be specified by the protocol. In a knowledge-based system, the input values must furthermore be retrieved from the data repository. Hence, the URIs of the input values must also be specified.

The basic design of the protocol is illustrated by figure 5.1.2. A valid implementation of the rule protocol is a class in Python, which implements the attributes `provides` and `requires` a method `compute`. The attribute `provides` of type “string” contains the URI under which the output of the rule ought to be stored in the data repository. The attribute `requires` contains a set of all URIs, which are necessary inputs to the computation that need to be looked up in the data repository. Finally the method `compute` contains the necessary program code to compute the value specified by the `provides` attribute based on the inputs found in the `requires` attribute.

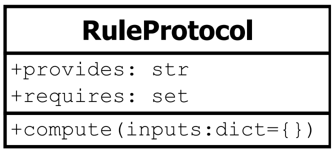


Figure 5.1.2.: UML representation of the rule protocol

By default, a single dictionary is expected as input to the `compute` method, which provides a mapping between the input URIs and the corresponding data repository entries. This has the advantage that the `compute` method does not reference any attributes of the class directly, making it a static method. A static method can be treated exactly like a function, which means that, in its simplest form, the rule definition essentially contains only a function and references to the inputs and output.

[2] For more background on protocols in Python s. PEP 544 [LLL17].

However, the protocol also gives more experienced developers the opportunity to exploit the OOP features of Python to design more advanced and complex classes or leverage inheritance as required.

Formulating the rules in this way has several advantages over a conventional procedural implementation. First of all, the computation itself is stateless, i.e. the result only depends on the inputs and not the state of the remaining system. Statelessness has positive effects on reproducibility of results and thus testability of the program. It also facilitates the implementation of parallel execution. Secondly, the relationship is provided in a declarative way, meaning it is not evaluated right away, but only upon request. This enables e.g. lazy evaluation and corresponds to the findings of La Rocca, who also advocates a declarative rule formulation.

The presented approach to rule formulation based on an established general-purpose programming language has several advantages to other approaches, discussed in chapter 2. Compared to design knowledge exchange formats such as OWL described in section 4.1.2, rules can be formulated with the full language capabilities, including advanced and mature libraries e.g. for numerical computation, data analysis or CAD. Due to the dynamic typing capabilities of Python, no restriction is placed on which kinds of objects are to be used, which also means the architecture is open to future developments in the language without additional integration effort. An advantage compared to a dedicated design language such as IDL, mentioned in section 4.3.2.3, is the higher public awareness of a popular and well-established language such as Python, which not only results in better availability of documentation, but also makes it more likely for newcomers to the KBE tool to already be familiar with the language.

To assemble the rule repository, all relevant rules are collected and assembled into a single design system. The specific rule implementations for fuselage and cabin design and modeling in FUGA are provided by the `fuga.geometry` and `fuga.design` subpackages listed in figure 5.0.1, which are introduced in chapter 6.

5.1.3. Graph-based inference engine

To implement the inference engine, an overall design system graph can be assembled based on the rule repository using the dependencies specified in the individual rules. This representation, enables reasoning on the system as shown e.g. by Pate, Gray, and German [PGG13] and Gent [Gen19a] for MDAO systems. A similar approach is presented here, which is implemented using the NetworkX graph library for Python [HSS08].

In the following, some background on graph theory syntax is provided first in section 5.1.3.1. Then, the different types of graphs that are required for the implementation of the KBE inference engine are introduced as well as their relationships. Finally, in section 5.1.3.3, solution strategies for different types of problems are discussed.

5.1.3.1. Graph fundamentals

Following the notation used by Diestel [Die17], a graph G consists of a set of vertices (or nodes) V and edges (or connections) E

$$G = (V, E), \tag{5.1.1}$$

where $E \subseteq [V]^2$, i.e. all elements of E are two-element subsets of V. The order of the pairs in E has no meaning for *undirected graphs*. It is, however, relevant for *directed graphs* (also referred to as *digraphs*), where the order of each pair indicates the direction of the connection. For instance, the connection from the vertex v to the vertex w would be the edge $e = (v, w)$, where v is the initial and w is the terminal vertex. In figure 5.1.3, an example for a digraph with

$$\begin{aligned} V &= \{a, b, c, d, e, f, g, h\} \\ E &= \{(a, c), (b, c), (c, d), (d, a), (d, e), (d, f), (f, d), (f, g)\} \end{aligned} \tag{5.1.2}$$

is shown.

The *cardinality* $|A|$ of a set A denotes the number of entries. The edges directed out of the vertex v are denoted by the set $E^+(v)$ and its cardinality is referred to as the *outdegree* $\delta^+(v) = |E^+(v)|$.

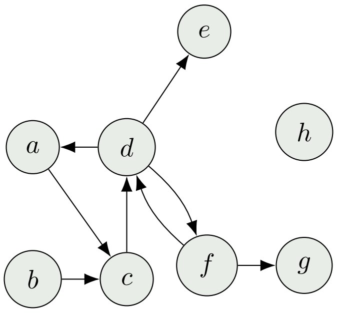


Figure 5.1.3.: Example digraph

Analogously, the edges pointing towards the vertex are denoted by $E^{-}(v)$ with the *indegree* $\delta^{-}(v) = |E^{-}(v)|$. For the example of the vertex d in figure 5.1.3 the outdegree $\delta^{+}(d) = 3$ and the indegree $\delta^{-}(d) = 2$.

A *path* $Q = (V, E)$ is a subgraph of G (i.e. $Q \subseteq G$), where

$$V = \{v_0, v_1, ..., v_k\}$$

and

$$E = \{(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)\}.$$

A *cycle* is a path where $v_0 = v_k$. The graph in figure 5.1.3 contains two cycles with $V_{C1} = \{a, c, d\}$ and $V_{C2} = \{d, f\}$.

A digraph containing cycles is referred to as a *directed cyclic graph*, whereas a digraph without cycles is called *directed acyclic graph (DAG)*. Retaining the vertices of the cyclic digraph from figure 5.1.3, a DAG can be created by updating the edges, using e.g.

$$E_{DAG} = \{(a,c), (b,c), (c,d), (d,e), (d,f), (e,h), (f,g)\}.$$

Due to the updated connections, the two graphs are not the same despite being composed of the same set of vertices. The resulting graph is shown in figure 5.1.4.

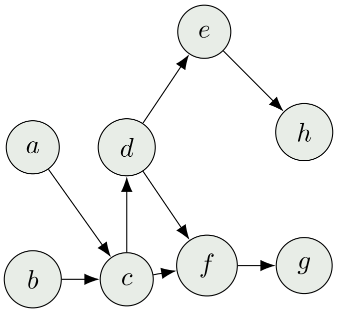


Figure 5.1.4.: Example DAG

In a DAG, the *ancestors* of a vertex v are all vertices with a path to v, whereas the *descendants* of v are given by the set of vertices, which can be visited starting from v. The ancestors d in figure 5.1.4 are given by $\{a, b, c\}$, whereas the vertices $\{e, f, g, h\}$ are descendants.

If a vertex v has no incoming edges, i.e. $\delta^{-}(v) = 0$, it is referred to as a *source*. Conversely, if the vertex has no outgoing edges, i.e. $\delta^{+}(v) = 0$, it is referred to as a *sink*. In figure 5.1.4, the nodes a and b are sources, whereas the nodes g and h are sinks.

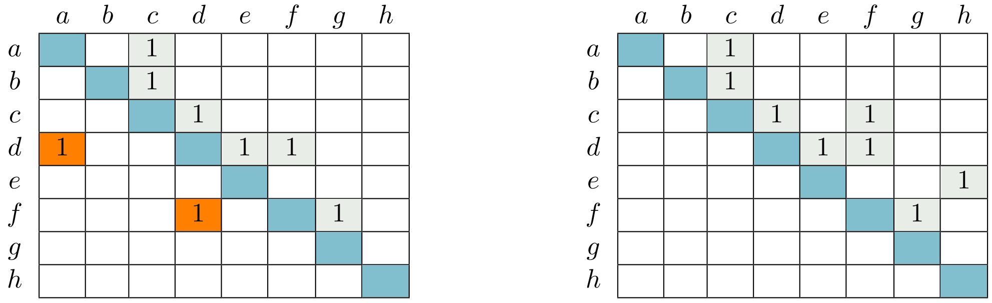
The *topological sorting* algorithm by Kahn [Kah62] can be applied to a DAG G, to determine an order of the vertices in G, in which the initial vertex of any given edge is guaranteed to appear before the terminal vertex. Multiple solutions of the topological sorting are possible. For the DAG in figure 5.1.4, the order (a, b, c, d, e, f, g, h) is one feasible solution.

Two graphs G_1 and G_2 can be combined into $G_c = (V_c, E_c)$ by forming the union of the respective sets of vertices $V = V_1 \cup V_2$ and edges $E_c = E_1 \cup E_2$. Van Gent and Pate also introduce the following notation for unions involving more than two sets:

$$\bigcup_{i \in I} A_i = \bigcup_{A \in \mathcal{A}} A = \{x | x \in A \,\text{for some}\, A \in \mathcal{A}\}\,, \tag{5.1.3}$$

where the set of sets $\mathcal{A} = \{A_i | i \in I\}$ is an indexed family of sets with the index i and the indexing set I [SAE14].

Any graph can be represented by an *adjacency matrix*, where the rows and columns represent the vertices in the graph and nonzero entries represent an edge from the row node to the column node. Adjacency matrices are the foundation e.g. of N² and XDSM diagrams. Undirected graphs are characterized by a symmetric adjacency matrix, whereas for digraphs, the upper triangular denotes feed-forward connections and the lower triangular denotes feedback connections w.r.t. the order of the nodes. The adjacency matrices for the two example graphs are depicted in figure 5.1.5. Visibly, entries are found in the upper and lower triangular for the cyclic graph. For the DAG, however, only entries in the upper triangular are present. This is due to the choice of a topological order for the row and column order, which results in all connections being feed-forward.



	a	b	c	d	e	f	g	h
a			1					
b			1					
c				1				
d	1				1	1		
e								
f				1			1	
g								
h								

(a) Cyclic graph (figure 5.1.3)

	a	b	c	d	e	f	g	h
a			1					
b			1					
c				1		1		
d					1	1		
e								1
f							1	
g								
h								

(b) DAG (figure 5.1.4)

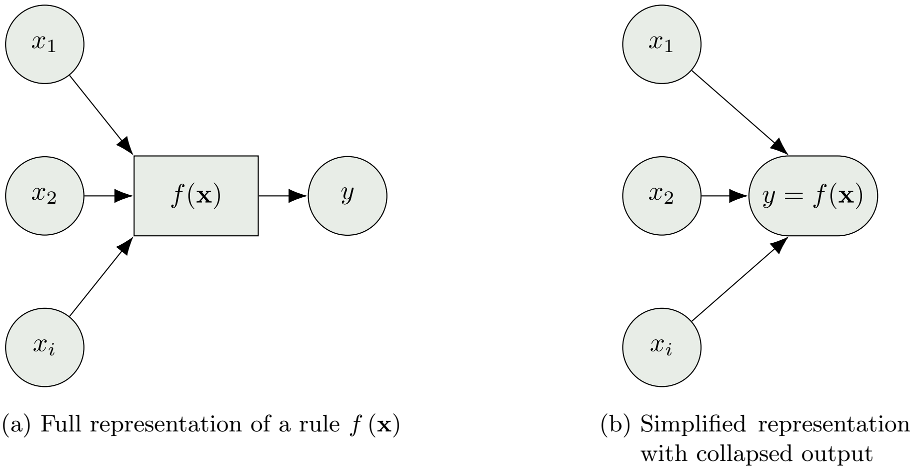
Figure 5.1.5.: Adjacency matrices for example graphs

5.1.3.2. KBE system graph types

Several types of graphs are necessary to implement the KBE inference engine, which are introduced in the following. The terminology is adopted from Pate, Gray, and German [PGG13]. Gent [Gen19a] employs a slightly different nomenclature, which is also cited for comparison. To facilitate the comparison between the methodologies for KBE and MDAO, the term function is used to refer to both KBE rules and MDAO tools.

Maximal connectivity graph (MCG) Based on the information contained in the `provides` and `requires` attributes of the rule protocol, a representation of a given function $y = f_i\,(\mathbf{x})$ using a DAG $G_i = (V_i, E_i)$ can be created. An example is shown in figure 5.1.6a. Notably, the graph contains two different types of nodes. The round nodes represent variable entries in the data repository, whereas the rectangular node represents an executable function. Since each rule only provides a single output, as per the protocol, a simplified representation can also be introduced, where the rule and its output node are merged into a single node, as shown in figure 5.1.6b.

The *maximal connectivity graph (MCG)* $M = (V_M, E_M)$ is assembled by combining these individual



(a) Full representation of a rule $f(\mathbf{x})$

(b) Simplified representation with collapsed output

Figure 5.1.6.: Graph representation of a KBE rule

graphs for all available functions:

$$V_M = \bigcup_{i \in I} V_i, \tag{5.1.4}$$

$$E_M = \bigcup_{i \in I} E_i. \tag{5.1.5}$$

It thus represents the entirety of the connections described by the function repository. In reference to this, Van Gent chooses to adopt the name *repository connectivity graph (RCG)*, instead of MCG.

Relating to the KBE process stages shown in figure 5.1.1, the MCG is instantiated only once during the KBE system setup step, since the set of available rules is not expected to change. It forms the basis from which all subsequent graphs will be extracted.

Fundamental problem graph (FPG) Unlike the MCG, which represents the entirety of the available knowledge, the *fundamental problem graph (FPG)* is assembled at the problem formulation stage to describe the knowledge subset necessary to solve a specific design problem. A design problem is defined based on the information available in the data repository, i.e. the known variables, on the one hand, and a request for a set of variables via their URIs on the other, as illustrated by figure 5.1.1.

The FPG contains all the paths in M, which lead from a known variable or a source to a requested variable without passing another known variable. This can be accomplished by discarding all ancestors of the input nodes and all descendants of the requested nodes in the MCG.

To ensure that the problem can be solved, the source nodes of the MCG should never be unknown. Therefore, the data repository should be populated with seed data e.g. by parsing data from a CPACS file before submitting the first user request. This is referred to as system initialization and reflected by the initial data repository update step in figure 5.1.1.

Problem solution graph (PSG) In order to determine a solution to the FPG, a solution sequence must be defined, which is referred to as the *problem solution graph (PSG)*. As mentioned in section 4.3.2.3, the challenge is that each rule may only be visited, once all required inputs are available in the data repository. The PSG provides an order for the function calls to be executed, which fulfills this requirement. If the FPG is a DAG this is the case for the topological sort, as described in section 5.1.3.3.

In place of a PSG, van Gent introduces two separate graphs, the MDAO data graph and the MDAO process graph. This is related to the XDSM notation, which distinguishes data flow from process flow. The data graph is concerned with which variables are exchanged between which functions. It thus contains both variable and function nodes. In many ways, it corresponds to the FPG. Differences are minor and mostly concerned wit MDAO execution technicalities.

In the context of FUGA, managing the data flow on such a low level is not critical, because the data repository is accessible to all functions. The process graph, on the other hand, provides the function execution sequence and is thus the counterpart to the PSG in FUGA. Like the PSG, it only contains function nodes.

5.1.3.3. Deterministic and numerical system solution strategies

At this point, the issue of the system solution is examined, i.e. how to determine a PSG, which can provide a consistent solution to the design problem stated by the FPG. If no cycles are present in the FPG, i.e. it is a DAG, topological sorting [Kah62] can be applied. Beginning at the source nodes, the algorithm provides a node sequence, where a node is visited only, once all of its immediate ancestors have been visited. An implementation of the topological sorting algorithm is provided in the NetworkX library. Passing through the PSG, the required inputs are retrieved from the data repository and passed to the `compute` method for each rule. The return value is then added to the data repository under the key given by the `provides` attribute of the rule.

This deterministic approach is very straightforward and sufficient for many problems found in FUGA. For instance, the feature-tree-like KBE-formulation of the parametric modeling engine consists entirely of DAGs with CPACS parameters as sources and their geometric representations as sinks. Nevertheless, there are cases, in particular when deploying KBE for design applications, where the deterministic approach cannot be applied. Instead, numerical methods must be used.

One such case is the presence of cycles in the FPG. Cycles represent a dependency loop between parameters, i.e. a rule to compute a given parameter requires information, which only becomes available later in the process. The length increase of the fuselage for the tank integration problem described in section 6.3.1 is an example of this. On the one hand, the length of the fuselage must be known to determine the cabin layout and tank design. On the other hand, the length of the fuselage must later be corrected to accommodate the requested tank volume.

This type of problem is closely related to the fluid-structure-coupling problem discussed in section 2.2.1 and can therefore be resolved in a similar fashion using e.g. Gauss-Seidel iteration. By introducing a coordinator block, the cycle can be broken at one of the nodes. In this way one loop of the cycle is turned into a black box function

$$x_{new} = f\left(x_{old}\right), \tag{5.1.6}$$

where x_{old} is an initial guess for the splitting node value and x_{new} is the value of the same node at the end of the cycle. Same as in MDAO, the problem of resolving the cycle can be reduced to a root finding problem in the coordinator block:

$$g\left(x\right) = \left|f\left(x\right) - x\right| = 0. \tag{5.1.7}$$

Different nonlinear root finding algorithms are applied in FUGA, depending on the type of the coupling variable. Gradient-free techniques must always be employed, since the rule protocol does not foresee derivatives. Therefore, the secant method is used if the variable is scalar, whereas Broyden's method is applied for vectors [Kel03; Rot03]. Implementations of both algorithms for Python are available in the SciPy library [VG+20]. Other, non-numerical types of data repository entries, such as geometry objects are not eligible for use as variables in the fixed-point iteration.

A related problem is the solution of inverse problems, i.e. moving in opposition to the direction of the digraph. These types of problems essentially describe an attempt to derive less detailed information from more detailed information, which is both somewhat outside the scope of FUGA and difficult to accomplish properly in many cases, e.g. when trying to reverse-engineer TLAR from detailed CAD-models. Still, fundamentally, the problem can be reformulated and solved like the cycle using fixed-point iteration. To this end, an FPG can be built, where the requested variable is the source and the given variable the sink, once again providing a black-box function

$$y = f(x) . \tag{5.1.8}$$

Assuming an target value $\hat{y}$ for the requested variable, this can again be formulated as a root finding problem for

$$g(x) = |f(x) - \hat{y}| = 0, \tag{5.1.9}$$

which can be solved using the same methods as for the cycle.

Expanding on the aforementioned strategies, it is possible to formulate optimization problems on the graph. To this end, any single path or set of paths in the graph from a set of nodes with the values $\mathbf{x}$ to another node can be understood as a black box function $f(\mathbf{x})$, as long as the values of $\mathbf{x}$ and $f(\mathbf{x})$ are numeric. Consequently, a nonlinear optimization problem can be formulated to minimize $f(\mathbf{x})$ w.r.t. $\mathbf{x}$. Moreover, constraints can be introduced for the members of $\mathbf{x}$.

Consequently, a variety of complex problems can be addressed by applying numerical methods to the knowledge graph. However, there are still some limitations, pertaining to both the deployment of numerical methods in FUGA and their efficiency compared to deterministic method. For instance, the algorithm in FUGA is not yet capable of introducing converger or optimizer blocks automatically based on user queries. Instead, the user must tell the system explicitly on which variable to perform e.g. the fixed point iteration for a cycle at this point.

That said, a second aspect of the iterative numerical methods is their inherent performance penalty compared to the deterministic solution. Knowledge engineers are therefore encouraged to formulate design systems in a deterministic fashion wherever possible.

5.1.3.4. Cache invalidation

A convenient side effect of the graph-based formulation is that cache invalidation capabilities, as demonstrated in principle by Kuhn et al. [KD+11], can be integrated with almost no additional programming effort. Whereas cache invalidation is not necessary if the data repository is updated during a traversal of the PSG, since the order of rule execution of the rules ensures consistent results, it is required if a repository value is updated manually. This may occur, e.g. if the user would like to modify a top level design parameter. A comparable situation is also at hand if a design variable is updated during an iterative solution run, as described in section 5.1.3.3. In these cases, all descendants of the modified variable (or variables) in the MCG must be identified and deleted from the data repository.

5.1.4. Explanation subsystem and user interface via third-party software

La Rocca [LaR11] stresses the importance of providing an explanation subsystem to communicate the activities of the reasoner and suitable user interfaces to interact with and inspect both the design system and the outputs from various user perspectives. For FUGA, the conscious decision was made not to develop a user interface from scratch, but to leverage existing software, which can provide more features and a higher level of maturity. In the following, an overview of the different use scenarios and the corresponding software solutions is provided.

Explanation subsystem FUGA implements several features to provide the functionality of an explanation subsystem. Most importantly, the different graph representations. i.e. the MCG, FPG and PSG can be exported to a graph exchange format such as graphML [Gra19] using the NetworkX library. Aside from the graph connectivity, the nodes and edges can be enriched with metadata, e.g. availability of a corresponding value in the repository. The graph can be inspected using specialized software such as Cytoscape [SM+03] or Gephi [BHJ09].

N^2-charts provide another way to visualize graphs based on the adjacency matrix. The openMDAO package provides interactive N^2-charts, which can be opened in the browser. FUGA provides the possibility to map the MCG to openMDAO for visualization. The rules are grouped by rule sets (s. section 5.2.1), which can be expanded and collapsed.

The data repository log is another contribution to the explanation subsystem in FUGA. It records any manipulation of the data repository, i.e. `add`, `update` and `delete` operations in a list, including the timestamp. The log can be exported to a Microsoft Excel or CSV file at any time in the process, allowing the user to retrace the activity of the inference engine.

Finally, the tool itself provides output messages informing the user about the progress at different customizable levels of granularity.

User interface According to La Rocca [LaR11], a KBE system must provide three different user interfaces to cater to three different use scenarios. On the one hand, there is the end user, who is applying the system to generate designs and models, on the other hand, there are the knowledge engineer and the software engineer, who develop the rules and inference system respectively.

Since FUGA is written almost entirely in Python, any development can be performed using an established Python IDE (Integrated Development Environment). As illustrated by figure 5.0.1, the separation between code related to the inference engine, which is relevant to the software engineer, and the rule repository, which is the concern of the knowledge engineer, is instead achieved by deploying it in separate subpackages. Specifically, the `core` package provides the inference engine, whereas all other subpackages contain domain knowledge rules. In addition, the rules for different types of applications are also distributed separately. As a result, a CAD geometry knowledge engineer and a simulation engineer would likely develop their rule sets in separate subpackages.

For the end user interface, Jupyter Notebooks have proved to be a good solution. They allow for interactive execution of Python code blocks from a web-browser [GP21; Jup23]. Setting up and distributing a well-documented standard design process in a Jupyter Notebook as shown in figure 5.1.7, has proved sufficient as an interface to perform a large majority of FUGA design tasks. To inspect the results, models can be plotted in the Notebook or written to both CAD (s. section 4.1.1.4) and mesh formats (s. section 4.2.1.2), which can be visualized using specialized tools.

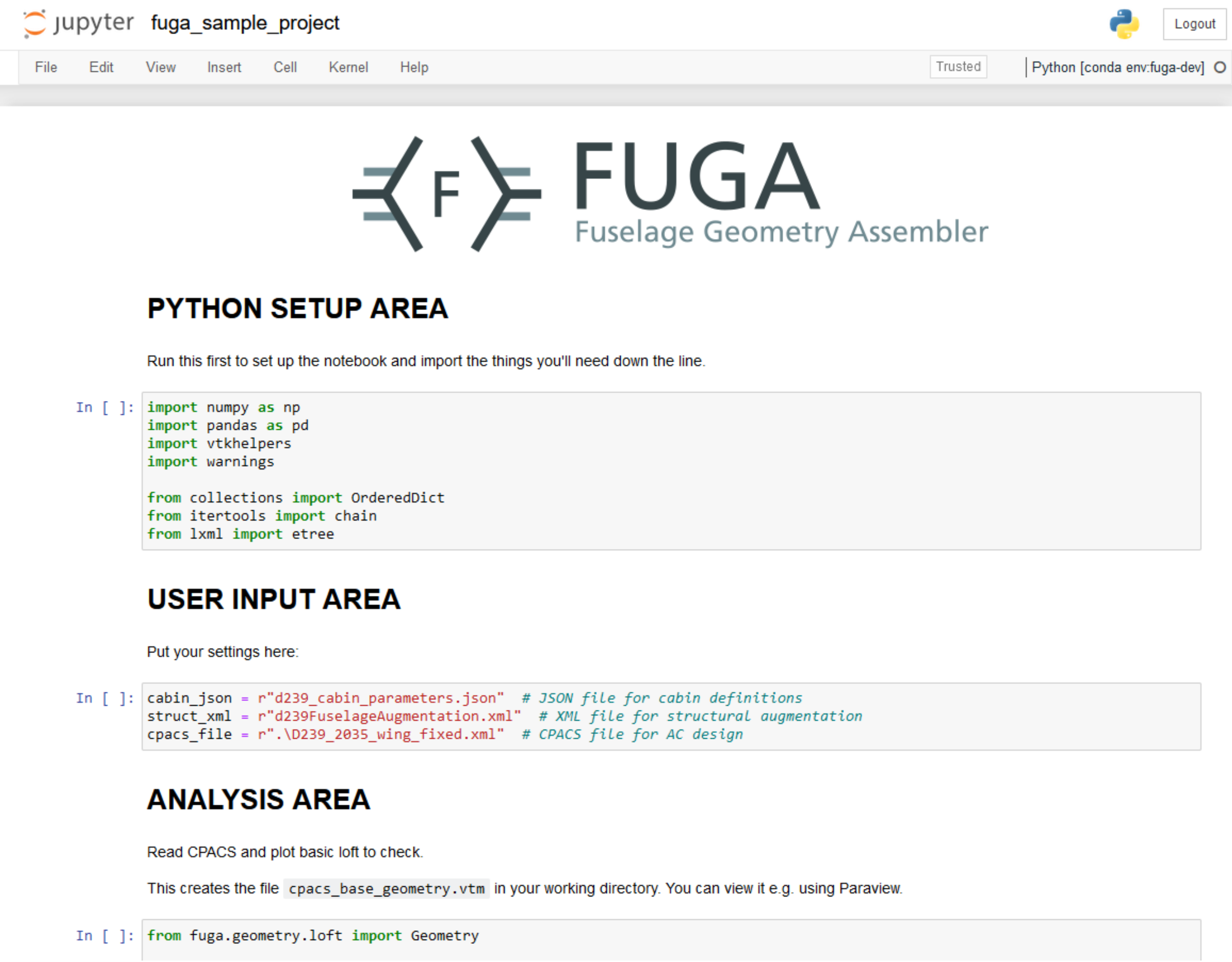
5.2. Plugin-based graph extension for knowledge management and architecture modification

In section 4.3.2.4 the need for structuring knowledge in complex KBE systems to make it accessible to the end user was discussed. Whereas La Rocca [LaR11] introduces HLP and CMs inspired by OOP, which cater to the requirements of preliminary aircraft design experts and have been adopted e.g. in ParaPy, in FUGA the more basic concept of rule sets is introduced instead. More details are provided in section 5.2.1. The rule sets can be leveraged in a plugin-based architecture to tailor the design system to the requirements of the task at hand.

On the other hand, section 2.1.3 illustrates the need for the system to be adaptable to new architectures. This can be accomplished via new rule sets, which model system knowledge that deviates from conventional designs, as shown in section 5.2.2.

5.2.1. Structuring knowledge using subpackages and rule sets

The fundamental approach to structuring the knowledge contained in the knowledge repository is to share the rules in several separate and smaller sub-system graphs instead of one large MCG. To this end, the rules must be grouped using the means described in this section. For each group of rules, a smaller dedicated MCG can be assembled as outlined above. The overall MCG $M = (V_M, E_M)$ can


jupyter fuga_sample_project
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Trusted
Python [conda env:fuga-dev]
FUGA
Fuselage Geometry Assembler

PYTHON SETUP AREA

Run this first to set up the notebook and import the things you'll need down the line.

```
In [ ]: import numpy as np
        import pandas as pd
        import vtkhelpers
        import warnings

        from collections import OrderedDict
        from itertools import chain
        from lxml import etree
```

USER INPUT AREA

Put your settings here:

```
In [ ]: cabin_json = r"d239_cabin_parameters.json"  # JSON file for cabin definitions
        struct_xml = r"d239FuselageAugmentation.xml"  # XML file for structural augmentation
        cpacs_file = r".\D239_2035_wing_fixed.xml"  # CPACS file for AC design
```

ANALYSIS AREA

Read CPACS and plot basic loft to check.

This creates the file `cpacs_base_geometry.vtm` in your working directory. You can view it e.g. using Paraview.

```
In [ ]: from fuga.geometry.loft import Geometry
```

Figure 5.1.7.: FUGA user interface in Jupyter Notebook

be assembled by composing the sub-system MCGs $M_i = (V_{M,i}, E_{M,i})$ using

$$V_M = \bigcup_{i \in I} V_{M,i}, \tag{5.2.1}$$

$$E_M = \bigcup_{i \in I} E_{M,i}. \tag{5.2.2}$$

This approach has the advantage that the different sub-system MCGs, i.e. the active rules in the system can be selected and combined depending on the design problem at hand. In FUGA, a plugin-architecture is implemented, which makes it possible for the user to add new rule groups to the main design system at run time.

Two ways have been established in FUGA to structure and deploy the available knowledge, which pursue two slightly different goals. One the one hand, *subpackages* are available to provide rules, which belong to different tool functionalities. The top level subpackage structure tree of FUGA given in figure 5.0.1 illustrates this grouping:

- The `fuga.core` subpackage provides the basic implementations of the data repository, the rule protocol, and the inference engine. All other subpackages contain a collection of rules dedicated to a specific purpose.
- The `fuga.cpacs` subpackage implements the interface to CPACS data sets. It can be used to parse a given XML file in order to initialize the data repository with the available information.

To this end, data mapping rules are provided to link entries in the CPACS tree to entries in the data repository. On the other hand, the subpackage also contains methods for writing eligible repository data back to a CPACS file. This may include newly generated information from the KBE system.

- The `fuga.geometry` subpackage contains the rules of the parametric modeling engine. The simplest use case of this subpackage is to assemble and plot a geometry model of all product information from a given CPACS file. For this, the `geometry` subpackage depends on the `cpacs` subpackage for data retrieval. In section 6.1, the rules of the subpackage are discussed in detail.

- If geometry models at a specific level of fidelity are required, the rules provided by the above subpackages may be insufficient, as some necessary details may not be provided by the CPACS file. In this case, the `fuga.design` subpackage can be triggered, to generate additional product information. The subpackage contains design rules, which take CPACS data, such as the OML, and additional user parameters as input to compute additional CPACS data, such as a structural layout or a cabin configuration. Since the rules may depend on geometric component representations in some cases, the `design` subpackage depends on both the `cpacs` and the `geometry` subpackage. The design rules are discussed further in section 6.2.

FUGA subpackages are distributed separately and can therefore be installed independently from one another. In this way, it can be controlled, which parts of the business logic are shared and which parts should remain proprietary. For instance, by distributing the `core`, `cpacs` and `geometry` subpackages, a functionality similar to TiGL can be made available to partners. The `design` module is, however, not shared in this scenario, to protect the knowledge contained in this subpackage.

Whereas subpackages serve to group the basic functionality of FUGA, *rule sets* can be used to structure knowledge from different domains within a subpackage. In table 5.2.1, the rule sets for the `cpacs`, `geometry` and `design` subpackages are listed, along with a brief description of the scope. As outlined above, the `fuga.cpacs` subpackage provides a collection of rule sets for file I/O as well as for mapping the data from the XML tree to data repository entries. The ruleset naming roughly mirrors the corresponding CPACS tree nodes. In addition, some data preprocessing rules for the loft and structure data are provided, which are identified by the `views`-prefix and universally applicable for all subsequent subpackages. They include e.g. the routines to ensure that profiles are ordered in a mathematically positive sense, or the assembly of symmetry maps for the wings and fuselages.

In the `geometry` and `design` subpackages, the knowledge is grouped according to the corresponding design discipline, e.g. `loft`, `wing`, (fuselage) `structure` and `deck`. Each rule set contains the necessary knowledge either to build geometric representations of the corresponding components based on CPACS data (`geometry` subpackage), or to generate said CPACS data from design inputs (`design` subpackage). Notably, no wing rule set is given in the design subpackage, as wing design is currently outside the scope of FUGA. Nevertheless, the wing geometry must be taken into account, as it provides important boundary conditions to the fuselage design. Meanwhile, the `loft` and `lh2_tanks` rule sets are optional, as explained below, and thus put in parentheses.

In figure 5.2.1, the rule sets necessary for a conventional outside-in cabin design process are shown in an N^2 chart along with their mutual dependencies. In this type of design process, the cabin layout is generated based on a fixed OML, as described in section 2.1.2. The rule sets are identified by their package name as well as their scope description from table 5.2.1. The chart was built using the openMDAO framework, which was first introduced in section 2.2.2 and to which an interface is provided in FUGA. Visibly, the outside-in design involves rule sets from the `cpacs`, `geometry` and `design` subpackages. Since no design capabilities for the OML is available in the system, it must be provided via CPACS and the corresponding loft geometry must be assembled using the respective rule sets in the `cpacs` and `geometry` subpackages. This information is then used to generate the structural and cabin design data as required, using the `design` subpackage rule sets. The results are passed back to the corresponding model generation rule sets.

package	package scope	rule set	rule set scope
`fuga.cpacs`	CPACS interface	`base`	file I/O & data mapping
		`fuselage`	
		`wing`	
		...	
		`views.geometry`	OML data preprocessing
		`views.structure`	structure data preprocessing
`fuga.geometry`	model generation	`loft`	OML
		`wing`	wing (movables & structure)
		`structure`	(fuselage) structure
		`decks`	cabin/cargo
`fuga.design`	design	(`loft`)	fuselage extension
		`structure`	(fuselage) structure
		`decks`	cabin/cargo
		(`lh2_tanks`)	LH_2 tank design & model generation

Table 5.2.1.: Rule sets provided by the FUGA subpackages

Whereas the N^2 diagram is useful to visualize the dependencies between different rule sets, it is somewhat misleading when it comes to the actual execution flow. As stated previously, feed-forward connections are typically found in the upper triangular , while entries in the lower triangular represent feedback connections. Some rule sets, such as the structural design and cabin design, are connected both via a feed-forward and a feedback connection, which suggests the need for an iteration loop to resolve the relationships. However, this is not actually the case, since the feed-forward and feedback connections link different nodes inside the individual rule sets. Consequently, different nodes from the two rule sets are visited intermittently while traversing the graph, without the need for any actual iteration. This becomes clear when considering the more detailed representation of the MCG for this system, which contains the connections between the individual nodes.

It is given in figure 5.2.2, along with the rule set affiliation of the individual nodes, which is given by the node color. The nodes are arranged according to their topological layers from top to bottom. Consequently, evaluation of the graph is performed by a downward traversal. As a rough orientation aid, ellipses corresponding to the colors representing the respective rule sets are placed in the figure to highlight concentrations of rules from a single rule set. Whereas the sequence of the ellipses suggests a general program flow from design to modeling rules, it also becomes clear that the task of clustering the rules is rendered more difficult by a substantial number of “outlier” rules. For example, rules from structural design are found in the area dominated by cabin design, whereas some cabin design rules can be found in the structural modeling section. These are the interspersed connections between the rule sets, which lead to the perceived loop in the N^2 diagram. These observations once again highlight the close interconnection between the disciplines.

5.2.2. Manipulation of system behavior for novel architectures

The modular approach to knowledge formulation using rule sets and subpackages not only allows knowledge engineers to structure the existing knowledge, but also to extend the existing knowledge base with new rule sets to account for fundamental architectural changes. An example is shown in figure 5.2.3, which extends the initial outside-in design problem to include the integration of an LH_2 tank in the rear fuselage, as described in section 2.1.3. The design problem is discussed in more detail in 7.2.1.

The fundamental approach is to increase the fuselage length as much as necessary for the LH_2 tanks with the required fuel volume to fit in the space between the rear pressure bulkhead (RPB) and the horizontal tailplane (HTP) wingbox. Consequently, a fuselage design rule set for extending

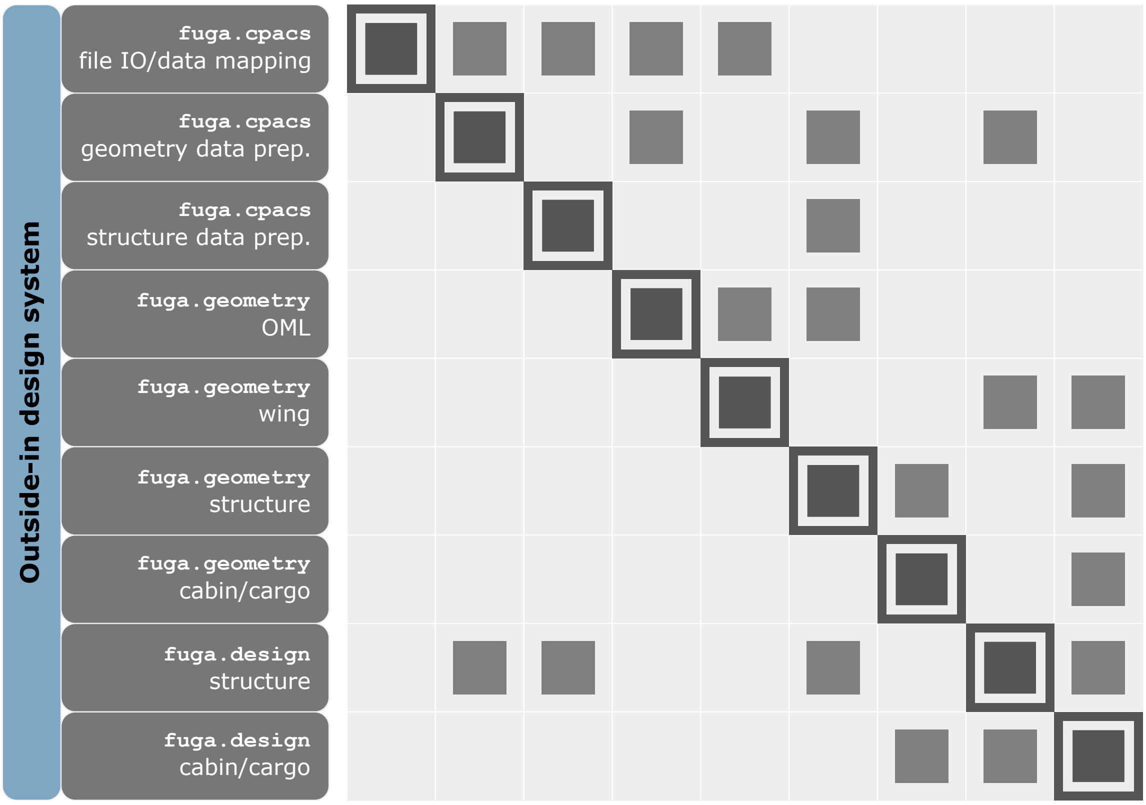


Figure 5.2.1.: N² diagram showing the rule sets and dependencies for a basic outside-in design

the fuselage length is now required, which is discussed in section 6.2.1. Furthermore, a design and modeling rule set for the tanks is required, which is discussed in section 6.3.1. The other rule sets from the outside-in design are retained. Figure 5.2.3 also illustrates that the new rule sets contain references to existing conventional rule sets to determine e.g. the bulkhead and HTP positions. In this way, previously existing knowledge can be applied also in novel contexts.

The additional rule sets can be used to both add new rule repository entries and replace existing ones. When looking for a rule to compute a given data repository entry based on a user request, the inference engine will respect a rule resolution order, where a rule placed in a lower rule set shown in figure 5.2.3 will take precedence over a rule providing the same value in any of the higher data sets. This provides a mechanism for *overriding*[3] existing rules with modified implementations from new, specialized rule sets. In this way, existing components from the default outside-in rule set can potentially be *mocked*[4] as required for new architectures, while descendants of these rules are applied a usual, thus further enabling knowledge transfer across different architectural approaches.

5.3. Discussion

The graph-based KBE system implementation discussed in this chapter yields a number of advantages compared to more conventional imperative and object-oriented approaches. The overall approach shares many similarities with functional programming languages, such as Lisp, Haskell or Scala, thus inheriting many of their advantages, which have been outlined e.g. by Chiusano and Bjarnason [CB14] and Lipovaca [Lip18]. In this respect it is comparable to the approach of La Rocca [LaR11], who

[3] The term *overriding* is borrowed from OOP, where it refers to a method of a child class replacing the functionality of the equally named method of its parent class [Lut10].

[4] The term *mocking* is borrowed from software testing, where it refers to simplified code, which mimics the behavior of the real software [Lut10].

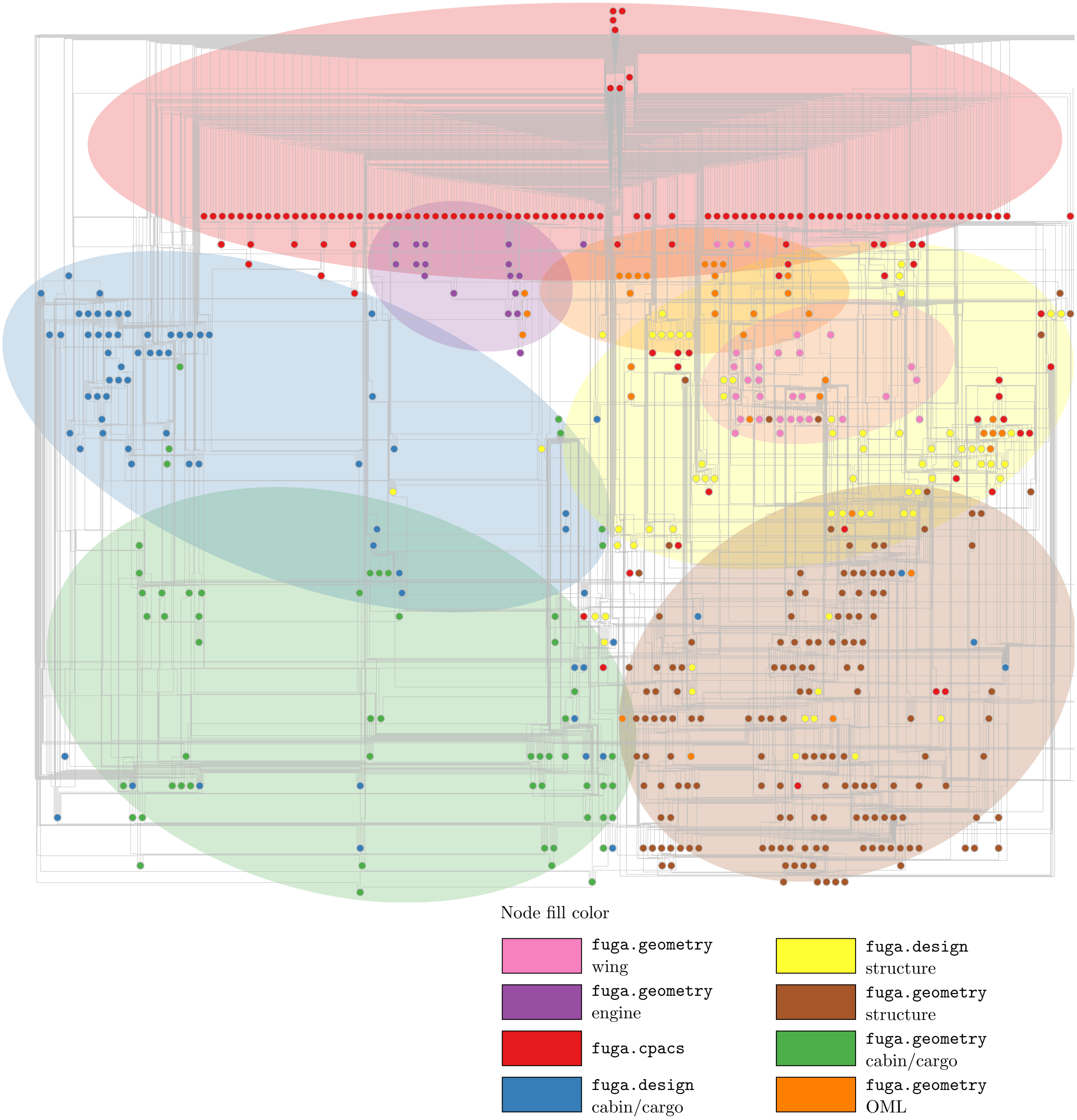


Figure 5.2.2.: Maximal connectivity graph (MCG) of the design and modeling rules colored by rule sets

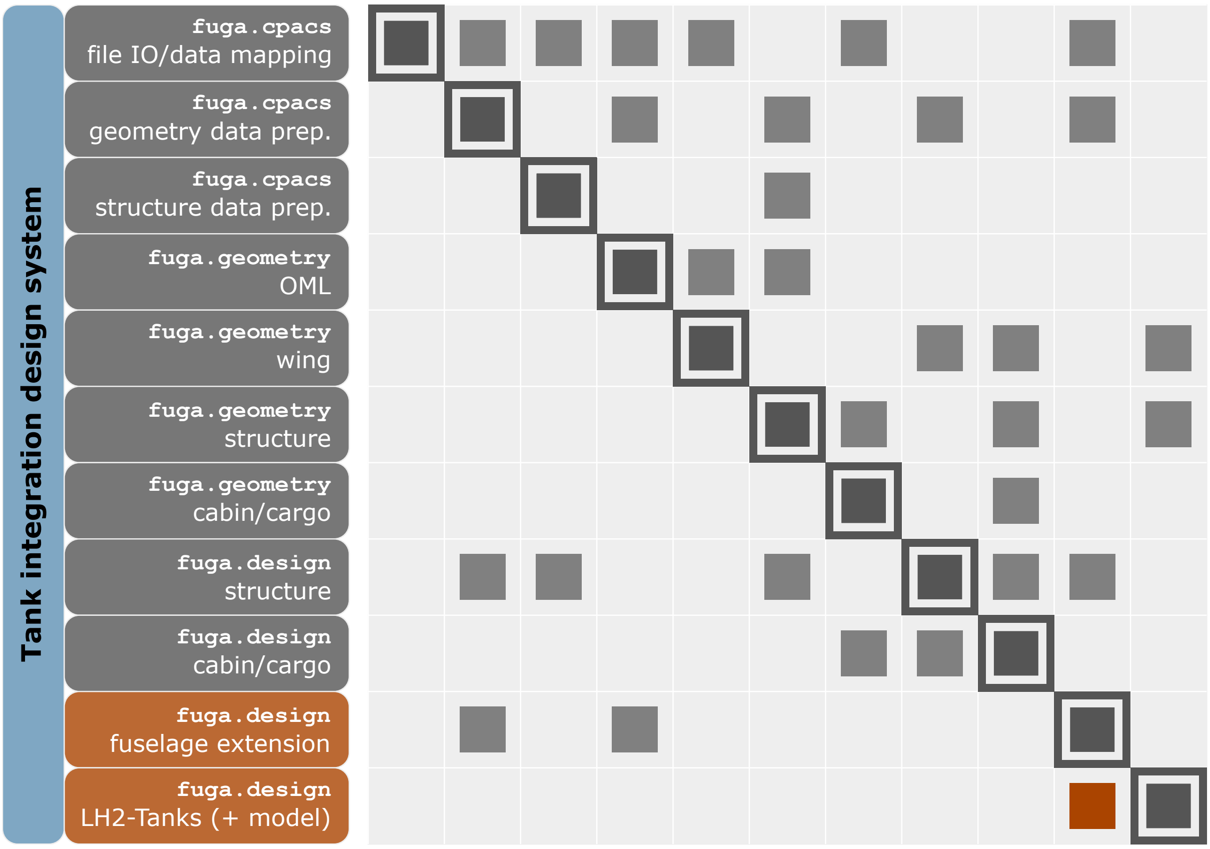


Figure 5.2.3.: Extended N^2 chart for an LH_2 tank rear fuselage integration problem

employed the Lisp-based ICAD design language.

A key property of the functional programming paradigm is that a function is not allowed to have a *side effect*, i.e. when called it only computes and provides the return value without further altering the state of the overall program. This is respected by the function protocol of FUGA introduced in section 5.1.2, since it is only allowed to change the repository value specified in the `provides` attribute. The lack of side effects facilitates the application of *parallel computing*, to run processes on a very large scale [Mar13a]. On the other hand, the function also does not depend on the overall state of the program, but only on the inputs provided. This means that, given the same inputs, the function will always return the same value and is referred to as *referential transparency*. It makes it easy to test individual functions even in a complex software system. The purely functional programming paradigm is a specialization of *declarative* programming, which aims to tell the computer *what things are* rather than *what to do*. The approach enables *lazy evaluation*, which implies that a value is computed only when it is actually needed by the program. Since some model generation operations, such as Boolean operations in CAD, can be comparatively slow, this is particularly valuable, since it enables fast evaluation of lower-fidelity models without any additional programming effort.

On an engineering level, the graph based approach also provides some practical advantages. A valuable capability of the KBE system is result or cache invalidation, which provides the means to maintain consistency of a design. Furthermore, the KBE system is highly useful to react to missing values. Due to the multidisciplinary and multi-fidelity nature of CPACS, it is very common that not all details are available in a given data set. However, which details are missing exactly, usually varies from one data set to another, depending on the specific analysis histories. Therefore, it is necessary to be able to react dynamically to whatever information is available and trigger design rules as required due to the inputs. This step is performed automatically and implicitly in FUGA, when extracting the FPG.

A possible drawback of the approach is its higher initial complexity compared to an imperative implementation. Implementing rules according to the rule protocol requires a basic understanding of the system and advanced Python language features. Also, the implementation tends to be more verbose than a procedural or object oriented implementation, which can harm code clarity and maintenance. However, it must be assumed that an imperative system to address the complex problem of fuselage design will be at least equally hard to maintain. Furthermore, the growth in complexity when scaling the system can be manged in a very elegant fashion for the declarative system using the rule set approach.

The choice of using the Python programming language and the graph capabilities of the NetworkX graph library to mimic the behavior of existing functional programming languages can also be brought into question. Certainly the rule formulation and system execution could have been implemented in a more compact and efficient way using one of the existing languages for KBE. However, in FUGA, the reasoning on the system only takes a very small fraction of the run time compared to the execution of the actual rules e.g. the model generation. Therefore, spending effort on run time optimization of the inference engine is unlikely to yield a substantial benefit. On the other hand, rule development speed is an important factor for the growth of the system. Here, Python is an obvious choice due to its reputation for very quick development cycles, which comes at the expense of program performance [Ros98]. As a dynamically typed language, Python also does not require knowledge engineers to worry about declaring data types right away and thus offers a very high flexibility for rule formulation. Another advantage of Python is that it is well-established in the engineering and research community, which makes it likely more easy to learn for newcomers. Furthermore, the language provides advanced scientific computing capabilities and bindings to a vast amount of libraries also from different languages, which can all be leveraged when implementing the rules.

As described in section 5.2, the methodological approach also allows for modular formulation, distribution and deployment of knowledge. This means that rules corresponding to a given domain can be stored in a dedicated domain rule set, which can be managed more easily by a domain expert. On the other hand, the approach enables ad-hoc assembly of custom design systems, tailored to a given design task and target product architecture, as illustrated by the LH_2 tank integration example. In this way, the system can be augmented with new rules and is therefore open to support unconventional architectures.

In conclusion, the implementation discussed in this chapter demonstrates the fundamental feasibility of the approach to describe aircraft fuselages in a knowledge graph stated in **working hypothesis 1**. The specific applicability to fuselage detail design and modeling for analysis alluded to by **working hypotheses 2 und 3** is, however, yet to be demonstrated. Similarly, the aforementioned capacity for architecture modification, which is mentioned in **working hypothesis 4**, is to be demonstrated in an example. Towards this end, the implementation of a knowledge-based system for fuselage design and modeling using the methodology described in this chapter, which can also be extended for new architectural requirements, is presented in the following in chapter 6. The corresponding application examples are found in chapter 7.

6. A KBE system for the design and geometry generation of aircraft fuselages and cabins

In this chapter, the design and modeling rules for the fuselage and cabin, which comprise the KBE system in FUGA, are discussed. They are distributed in the `fuga.geometry` and `fuga.design` subpackages. The interactions of rules of the different subpackages with the data repository are given in figure 6.0.1.

On the one hand, the `fuga.geometry` subpackage is the KBE-driven implementation of a parameter engine, providing rules to convert parametric data from CPACS into geometry models using OCCT. This capability is necessary to validate **working hypothesis 2**. The rules are discussed in section 6.1.

The `fuga.design` subpackage, on the other hand, contains all rules used to generate new product data based on existing product information, including data from CPACS and the available geometry, or simplified parametric inputs. These rules are introduced in section 6.2 and provide the capabilities described in **working hypothesis 3**.

The default rule sets in FUGA, contained in the aforementioned subpackages, provide design and modeling capabilities for tube-and-wing configurations with conventional propulsion. In section 6.3 it is shown how the architecture of the product can be manipulated by introducing additional sets of rules, in order to extend the capabilities of FUGA towards novel configurations as required in **working hypothesis 4**.

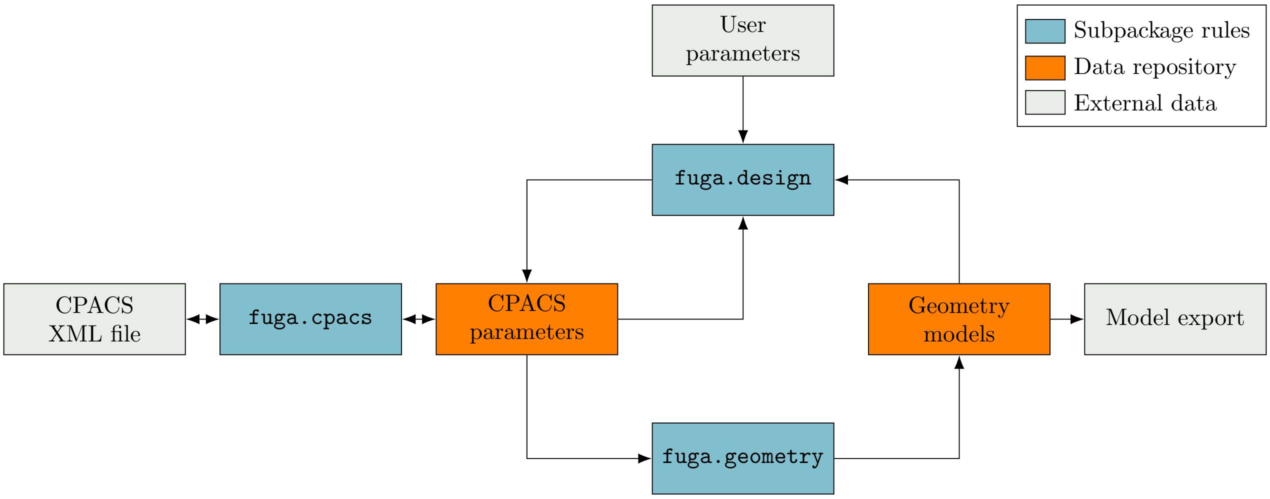


Figure 6.0.1.: Interactions of subpackage rules and data repository in FUGA

A short discussion of the findings in this chapter is given in section 6.4.

6.1. Knowledge-based parametric geometry modeling engine

It has been established in section 4.1.3 that a data-centric product description format such as CPACS needs to be paired with a parametric modeling engine to assure unambiguous geometric interpretation of the parametric data. It has furthermore been established that existing solutions, such as TiGL, are insufficient for the generation of detailed fuselage and cabin models, which are required to fulfill the requirements of typical fuselage analyses discussed in section 4.2. Consequently, FUGA provides its own parametric modeling capabilities, which have been implemented using the OCCT CAD kernel and a generalization of the feature-tree approach, which has been built using the KBE methodology discussed in chapter 5. This is in line with the approach proposed in **working hypothesis 1**. The baseline modeling capabilities for conventional aircraft are deployed in four separate rule sets, which are presented in the following as a contribution towards the validation of **working hypothesis 2**.

The `loft` rule set, described in section 6.1.1, provides all the necessary relationships to generate a CAD representation of the overall aircraft loft from CPACS. This includes both the fuselages and the wings. Based on the results, the `structure` rule set, introduced in section 6.1.2, can be applied to derive more detailed geometry models of the fuselage structure, including stiffeners and floor structures. Some aspects of the `wing` rule set are discussed as well, as it provides additional wing details including structures and movables, which are necessary inputs for the subsequent fuselage structure design (s. section 6.2). The `deck` rule set introduces capabilities to generate cabin models based on CPACS deck definitions. It is presented in section 6.1.3. The model generation steps are illustrated using the D240 configuration, a single aisle design for 240 passengers, as an example.

In section 6.1.4, it is discussed how the KBE approach to geometry generation based on these sets of rules can be exploited to manage different geometry fidelity levels and meet the multi-fidelity geometry requirements to support fuselage and cabin MDAO processes.

6.1.1. Overall aircraft loft generation rules

As discussed in section 4.1.3, the generation of OML geometry from CPACS data sets is supported by all available parametric modeling engines as well as several disciplinary model generation tools. Walther, Petsch, and Kohlgrüber [WPK17] discover significant potential for performance improvement for specific applications, e.g. the determination of frame/stringer intersection points for GFEM generation, compared to the prevalent TiGL library, justifying the implementation of an independent modeling engine. Meanwhile, integrating the OML generation into the FUGA KBE system also holds the promise of benefiting from all the advantages listed in section 5.3, along with additional control to tailor the outputs to the requirements of subsequent modeling rule sets.

A diagram derived from the connections of the loft generation rule set is given in figure 6.1.1. It describes the FPG to build a full CAD representation of the configuration as shown in figure 6.1.5a from the basic fuselage information provided in the CPACS nodes. For improved clarity, an abbreviated version of the XPath notation introduced in section 5.1.1 is used to label the nodes in the graph, where only the last path component is shown. Moreover, the collapsed rule notation as shown in figure 5.1.6b is used.

The nodes shown in the diagram are sufficient to model the complete configuration, since each node implicitly processes all instances of a given type. For instance, the vertical tailplane (VTP) and the horizontal tailplane (HTP) are modeled in the same way as wings in CPACS, which is why they are implicitly included in the left branch of the diagram. Conversely, the right branch could, in theory, include multiple fuselage instances to support multi-fuselage layouts. In figure 6.1.5a, the landing gear is modeled using additional fuselage instances. An additional node for the engine model including nacelle, rotor and pylon is also given. However, the engine model generation details are truncated, since they lie beyond the scope of this thesis.

6.1.1.1. Coordinate system and unit conventions in CPACS

Many of the node names in CPACS contain references to an underlying coordinate system, via the axis names x, y, and z. For the global coordinate system these axes denote the longitudinal, transverse, and vertical axes of the aircraft. The origin of the system is placed in the symmetry plane of the aircraft at the longitudinal position of the nose. For the vertical position, the height of the centroid of the constant section is usually selected. While it is possible, in principle, to define bodies in different axis systems in CPACS, it is an established good practice to adhere to this convention. It is therefore assumed in FUGA that all inputs are provided in this manner.

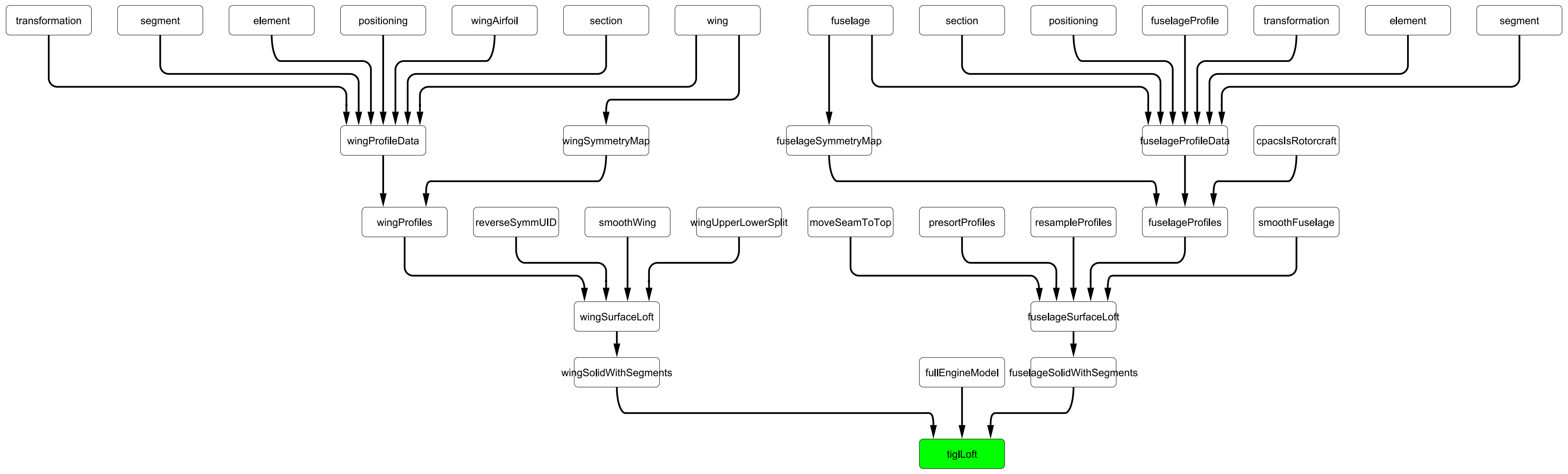


Figure 6.1.1.: FPG extracted from the loft generation rule set to generate the OML loft as shown in figure 6.1.5a

Furthermore, all numerical data in CPACS must be given in a consistent unit system. FUGA is compliant with CPACS convention, by which the SI unit system is to be used. This must be taken into consideration, e.g. when working with the certification specification, where data is commonly given in imperial units.

6.1.1.2. CPACS aircraft body description

As described by Liersch and Hepperle [LH11], CPACS provides a unified way to describe the geometry of primary aircraft components such as fuselages and wings, based on a sequence of cross-section profiles. In the following, the modeling approach will be outlined briefly using the fuselage as an example. Nevertheless, the same procedure can be applied for the wing with minimal adaptations.

Fuselages in CPACS are defined by instances of `fuselageType`, the structure of which is shown in figure 6.1.2. The nodes, which are relevant for the body description, are called `segments`, `sections` and `positionings`. The sections provide the starting point of the fuselage geometry generation. They describe an unsorted collection of section profiles to represent the body shape. According to Liersch and Hepperle, each section may consist of multiple section elements, which allows for e.g. the description of a wing profile and the corresponding movable profiles in a single section. However, for the fuselage a section typically consists of a single section element. This is also true for more exotic cases such as twin fuselage configurations, where each fuselage would be represented by a separate fuselage instance.

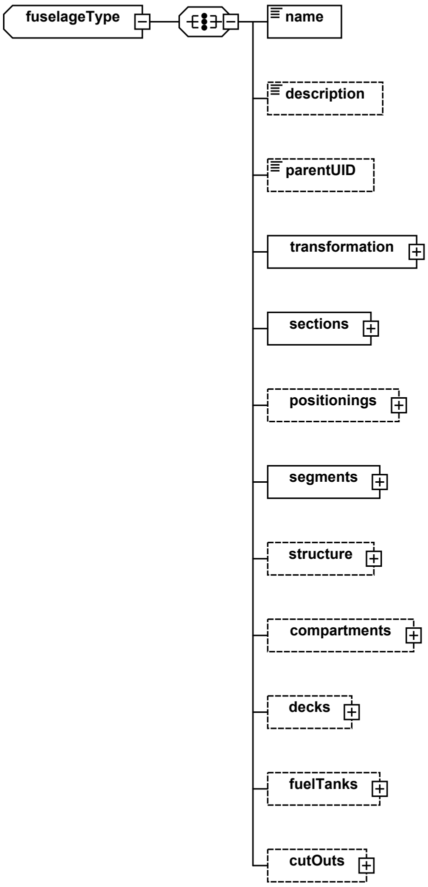


Figure 6.1.2.: XSD diagram representation of `fuselageType` in CPACS

A section element contains a reference to a profile definition. As of CPACS version 3.4, a profile can be described using a point list, CST curves, ellipses or rounded rectangles. By convention, these profiles are provided in the scale from −1 to 1, which means a scaling factor must be applied to achieve the proper scale required for the fuselage. For this, a transformation node is applied to each section element, which can also be used to manage the relative positioning if multiple section elements are present. A transformation node in CPACS provides descriptions of basic affine transformations in 3D, i.e. scaling, rotation (as Euler angles) and translation. The scaling is always applied first. Then, the `refType` attribute of the translation definition determines, whether the translation is applied before (`absLocal`) or after the rotation (`absGlobal`).

Once the section elements are properly scaled and aligned, the section must be placed in 3D space. One possible way to accomplish this is to modify the transformation node of the entire section. Unlike the section element nodes, this transformation is applied on all section elements of the section. In order to position the sections relative to one another, this means that individual transformations must be computed for each section, when assembling the data for the export of a CPACS file. Notably, neither the transformations for the section elements nor for the entire sections are listed in figure 6.1.1. Instead, they are automatically read from CPACS along with the element or section definition data and stored together in a single data structure.

All relevant geometries can be represented using the transformation approach. However, for wing models in particular, engineers are more familiar with concepts such as sweep and dihedral angles to describe the relative positioning of profiles. Therefore, the `positionings` node is provided in CPACS to provide an additional means of positioning profiles relative to one another in terms of a distance l_{pos}, dihedral Γ and sweep angle Λ, as illustrated by figure 6.1.3. Using these parameters, a transformation in 3D space can be computed, which is applied at the origins of the respective local coordinate systems of the profiles. Since the profile origin is typically placed at the leading edge, the angles can be understood as being related to the leading edge, in accordance with the CPACS convention.

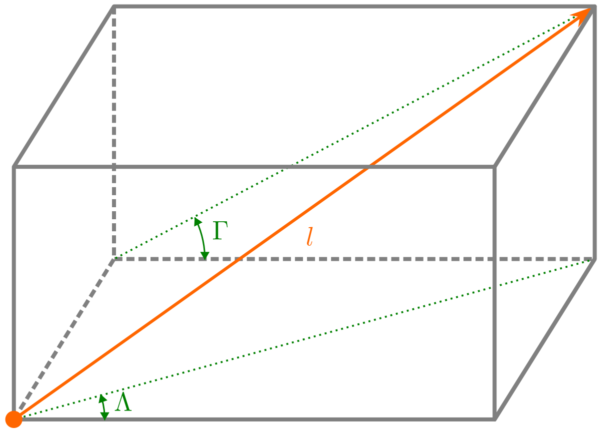


Figure 6.1.3.: Parameters of CPACS positionings definition

At this point, the data from CPACS has been turned into an unsorted collection of section profiles, which are properly scaled and positioned in the global 3D space. Depending on the definition, they are either available as a point cloud or a curve (CST, superellipse).

6.1.1.3. Surface construction

Assuming a point-based profile definition, which is by far the most common curve description type for fuselages in CPACS, a point cloud as shown in figure 6.1.4a can be built by merging the above information. Several techniques exist to construct a CAD surface from this data. If all profiles have the same number of sample points, the easiest and most efficient solution is to apply bivariate point grid interpolation, introduced in section 4.1.1.2, as described by Walther, Petsch, and Kohlgrüber [WPK17]. The circumferential direction is given by the order of the points in the profile definitions and must be controlled, as it has an effect on the normal directions of the final surfaces. Typically, profile curves in CPACS should be ordered in a mathematically positive sense. To detect faulty user inputs, it is, however, possible to compute the enclosed area of the profile using the trapezoidal rule. If the order of the profile is incorrect, the area will be negative and thus the profile order is reversed.

Nonetheless, further information is required for an unambiguous description of the sequence of the profiles in longitudinal direction, which is provided in the segments node in CPACS. A segment describes a stretch of surface from one section to another, by giving a `fromSectionUID` and a `toSectionUID`. A digraph can be assembled from this information, which can be used to identify paths (there is usually only one) and provide a section sequence using a topological sort.

Based on this information, the surface can be constructed as shown in figure 6.1.4c. In both TiGL and FUGA, cubic B-Spline interpolation is traditionally applied in circumferential direction, whereas a linear ansatz is selected for the longitudinal direction. Walther, Petsch, and Kohlgrüber [WPK17] also show that a higher order interpolation can be used in longitudinal direction to achieve a smooth surface. However, they also show the significant effect of the choice of interpolation parameter computation method (i.e. centripetal, chord length or uniform; s. section 4.1.1.1) where the centripetal method should generally be favored.

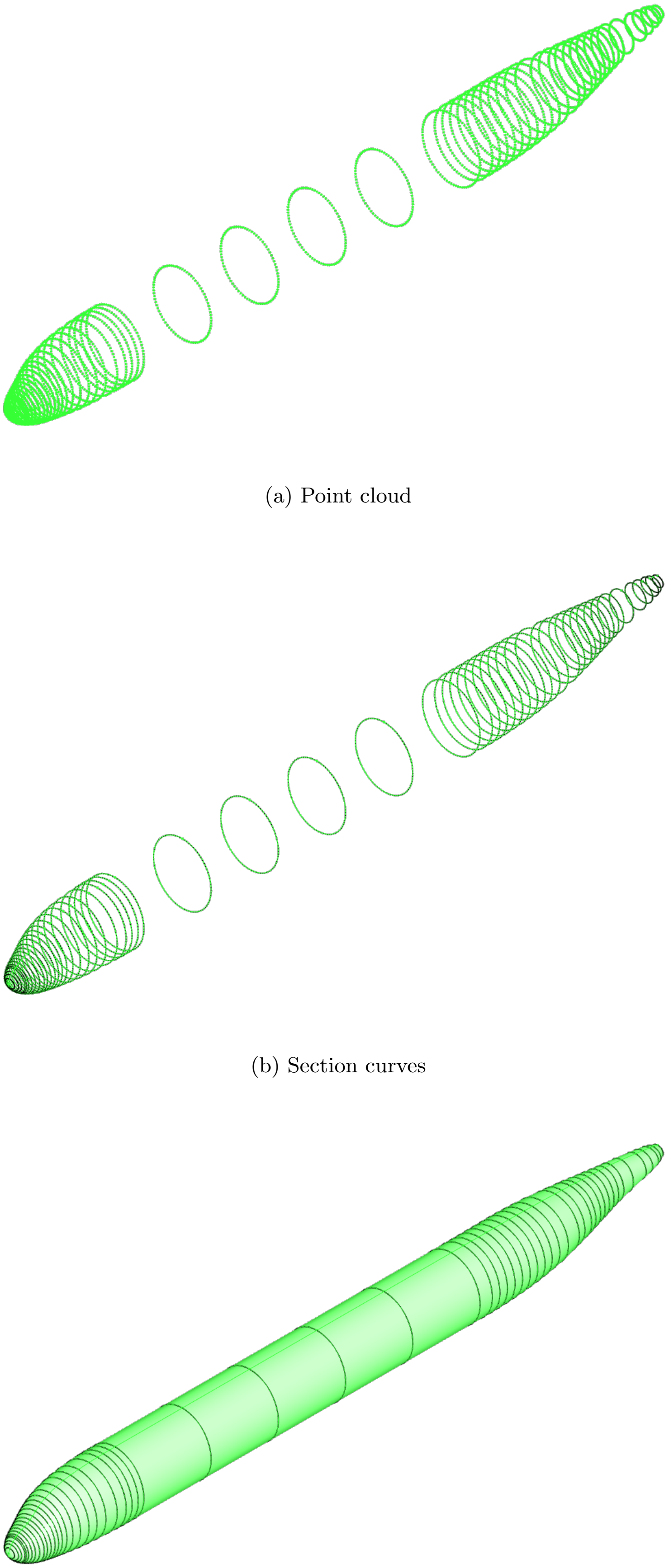
If the constituting profiles of the body contain different numbers of points, the bivariate interpolation cannot be applied, since the points no longer form a regular grid, which can be interpolated easily. In this case, a B-Spline curve must first be computed for each individual profile using curve interpolation. Then, the skinned surface algorithm is applied to construct a surface by interpolating the curves in the sequence, which is still provided via the segment definition. In this way, the other parametric profile descriptions, i.e. CST, ellipse or rectangle can also be implemented as long as NURBS-representations of the curve types are available, which can be used in the skinned surface algorithm. This may, however, not always be possible to do exactly. For instance, the difficulty to describe CST curves using NURBS is discussed in section 4.1.1.1. Similar problems arise for superellipses. In such cases it may be necessary to provide an approximation of the curve instead, e.g. using the least-squares approach given in equation 4.1.13.

A comparison of the bivariate interpolation and skinned surface algorithms is provided in section A.2. In general, bivariate interpolation yields simpler surfaces with fewer control points than the skinned surface algorithm, since the latter requires all profile curves to be compatible, i.e. share the same degree and knot vector. This requires the application of degree elevation and knot insertion techniques, which usually results in the circumferential knot vector being the union of all profile curve knot vectors and thus more complex than any of the individual knot vectors. On the other hand, if very simple profile definitions such as ellipses are applied, it is possible to build surfaces with accurate cross-sections with significantly fewer control points than using an equivalent point list definition. As a result, both algorithms have been implemented in FUGA, though bivariate interpolation is used whenever possible.

6.1.1.4. Configuration assembly

Once the body is built, it can be integrated into a configuration assembly. To this end, some additional information is provided in the type definition. First of all, another `transformation` node is given in CPACS, which serves to place the body as a whole in the global coordinate system. Unlike the `transformation` nodes related to the sections, this one is provided explicitly in figure 6.1.1. An optional `parentUID` can be applied to manipulate the frame of reference. In this way, the position of the HTP in a T-tail configuration can, for example, be expressed in terms of an offset w.r.t. the coordinate system of the VTP, which is more easily understood, than a position in the global coordinate system. Furthermore, the type definition provides a `symmetry` attribute, which allows for mirroring bodies on one of the coordinate planes. The feature is commonly used for wings.

For practical reasons, the transformations are applied to the profiles right away in FUGA when computing the value for the `fuselageProfileData` variable. The order prescribed by the segment definition is already applied. Then the symmetry, which is stored as a mapping in the `fuselageSymmetryMap` variable is applied. Finally, the surface is constructed from the points using one of the above techniques.



(a) Point cloud

(b) Section curves

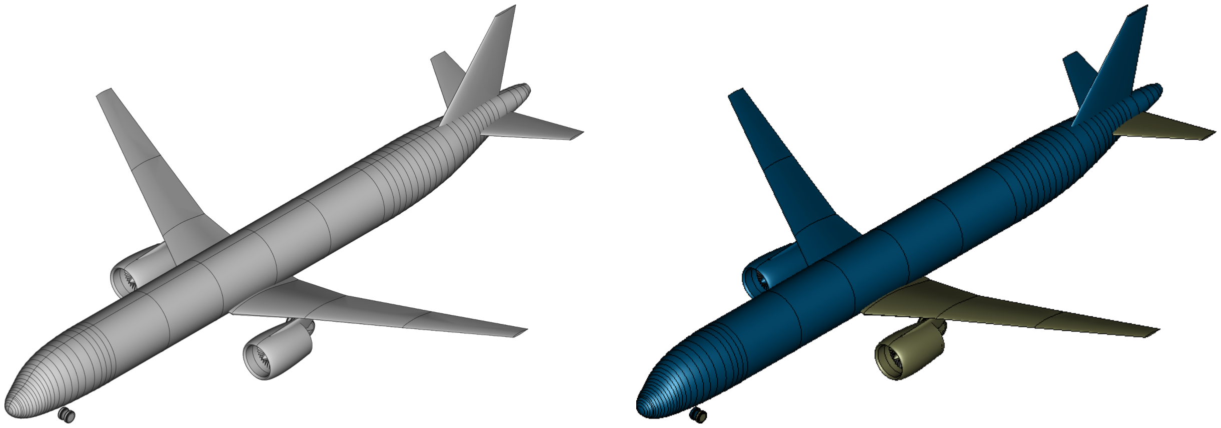
(c) Skinned fuselage surface

Figure 6.1.4.: CPACS fuselage surface generation steps

For volume computations and Boolean operations, it is necessary to make the main aircraft components available as solid bodies in OCCT. As described in section 4.1.1.3, solids can be built from closed shells. To obtain a closed shell from the surface built from the CPACS data, it must first be determined, whether the surface is open or closed. Fuselage surfaces are usually closed in circumferential direction and open in longitudinal direction. Therefore it is only necessary to close the end profiles using flat faces bounded by the outer profiles. In the case of wings, it is possible to define open trailing edges for the profiles to reflect a certain trailing edge thickness. In this case, the shell must first be closed in circumferential direction before closing the outer profiles in spanwise direction. To this end a swept surface can be constructed between the isoparametric curves at the beginning and the end of the wing surface.

It has become a tradition in the CPACS tool landscape and openAD in particular, to model the geometry of components that lack a proper representation in the schema as fuselages. Therefore a given data set may contain any number of fuselage definitions to represent e.g. landing gears, engines or aerodynamic fairings [Gu17; WH+18]. Meanwhile, even the novel designs discussed in section 2.1.3 still feature a single fuselage for payload accommodation. Therefore, the one fuselage to be considered in the subsequent steps can be specified by providing a `structuralFuselageUID` in FUGA. That said, it is not guaranteed that an explicitly defined fuselage body is present in CPACS. This special case is considered more closely in section 6.3.2.

Figure 6.1.5a shows the outcome of the `fusedLoft` modeling rule, which computes a Boolean union of the solid representations of the structural fuselage and all wings. In addition, engine geometry is also provided. For comparison, the TiGL representation is also provided in figure 6.1.5b.


(a) KBE system output (shown in FreeCAD)

(b) TiGL reference geometry (shown in TiGL Viewer)

Figure 6.1.5.: Comparison of the OML output from different parametric modeling engines

It can be seen that the FUGA outer loft modeling rule set can provide equivalent geometric representations. However, there are also a few differences to be highlighted. FUGA provides the option to reposition the longitudinal seam of the fuselage in circumferential direction. Changes in position of the seam can have positive effect on the stability of Boolean operations in OCCT. Furthermore, since the polynomial degree for the interpolation between the section curves can be chosen freely, smooth fuselage surfaces can be created based on sections only. Smooth surfaces once again provide an advantage in terms of stability, but it also means that additional assumptions go into the generation of the surface, which cannot be communicated via CPACS.

Another observation is that no belly-fairing to smooth the transition between the wing and the fuselage is present. This is common practice for CPACS data sets that are used to derive structural analysis models (s. also [SK+13]), since the fairing is assumed not to be load-bearing.

6.1.1.5. Generation of smooth surfaces from CPACS definitions

To avoid assumptions during the geometry generation, only linear interpolations between sections are performed in TiGL, unless additional information on guide curves is provided [SK+19]. The guide curve approach relates more closely to other preliminary design tools discussed in section 2.1.2, which use belly, side and crown curves to describe the fuselage, than the purely section-based definition commonly used in CPACS. However, many current synthesis tools, including openAD, do not provide this information.

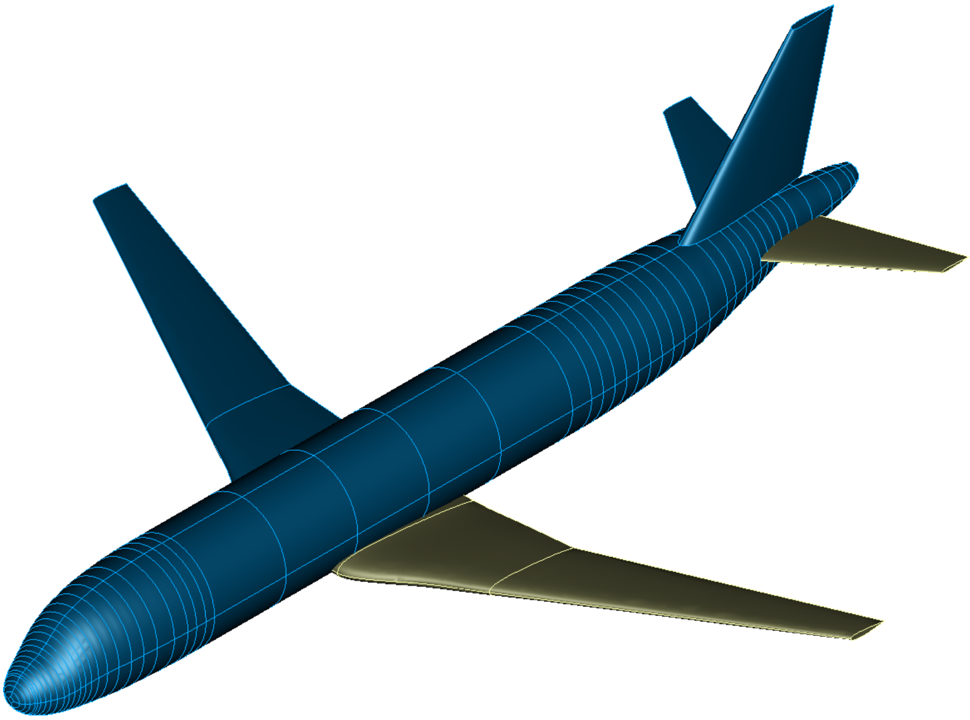


Figure 6.1.6.: TiGL representation of a CPACS configuration with guide curves

The guide curves are highlighted in figure 6.1.6 along with the previously used section curves. Shown are the crown curve on the upper side of the fuselage and the port side curve, analogously to the definition used by Jonge [Jon17] shown in figure 2.1.2. Furthermore, a leading edge guide curve is given for the main wing, which prescribes a tangency of the leading edge of the inner wing segment to the center wing box segment. Together, the guide curves and section curves form a curve network. A B-spline or NURBS surface can be derived from a curve network using the Gordon surface algorithm as described by Siggel et al. [SK+19]. As shown by the wing example in particular, this leads to a smoother transition between the two segment surfaces.

The guide curve approach offers much potential to simplify the definition of fuselage surfaces in particular by substantially reducing the number of section profiles used and simultaneously improving the geometric quality and efficiency of the surfaces. However, it has not been implemented in FUGA yet, due to the current lack of support from preliminary design tools, such as openAD.

6.1.2. Structural model generation rules

Following the OML, the model generation of the structural components is considered. Unlike the OML, this capability is not typically found in CPACS geometry libraries or if so, at a comparatively low level of detail.

The list of structural component types commonly considered in structural sizing tools for the fuselage given in table 4.2.1 provides a reference, which structural components ought to be taken into account. The corresponding modeling capabilities in FUGA are based on the description of the fuselage structure in CPACS introduced by Scherer and Kohlgrüber [SK16], which is found in the optional `structure` node of the aircraft definition in CPACS shown in figure 6.1.2. The diagram for the CPACS XML type of the `structure` node is given in figure 6.1.7. The description primarily assumes a semi-monocoque structural design, where a thin walled skin is stiffened by additional structural members [Niu88]. This design paradigm is common to all modern transport aircraft. More recently, some more generic structure types such as `generalStructuralMembers`, `walls` or `interfaceDefinitions` have been introduced to CPACS to better connect to military applications and related topics such as onboard system space allocation (s. e.g. [TD+21]). These types are, however, not relevant in the scope of this thesis, where the focus is on large passenger aircraft. That said, some typical structure parts, such as the forward landing gear bay, are still missing from the CPACS definition. In the scope of this thesis, which focuses on the integration of the fuselage structure and the cabin, this is not a critical omission.

In addition, details of the wing structure are required e.g. to determine the location of the wingbox, which influences the frame distribution, the design of wing/fuselage intersection areas and the exit placement. The wing structure definition in CPACS, which has been used, can be traced back to Dorbath, Nagel, and Gollnick [DNG11].

Both Scherer and Kohlgrüber and Dorbath, Nagel, and Gollnick subsequently provide implementations of GFEM model generators based on their respective proposed CPACS definitions. The underlying CAD model is restricted to the representation of the OML in TiGL. All further modeling is done at FEM model level. Even though both go on to show some multi-fidelity (Scherer and Kohlgrüber show a fuselage crash model, s. also Schwinn [Sch15]) or multi-disciplinary (Dorbath, Nagel, and Gollnick [DNG13] discuss the generation of corresponding meshes for aerodynamic loads computation on the wing using the vortex lattice method) capabilities, the approach in both cases is essentially to implement a separate tool, which returns consistent output if the same CPACS file is provided as input. FUGA on the other hand also provides the structural component models based on the CPACS definitions at the geometry level, making consistent geometric representations of all the components at different levels of fidelity available to any connected analysis tool as required.

The graph representation of the model generation rule set MCG is given in figure 6.1.8. The system is a subgraph of the outside-in design MCG shown in figure 5.2.2. Visibly, the rule set MCG graph is very complex compared to the loft generation FPG given in figure 6.1.1. On the one hand, this is because it contains all possible rules in the rule set as opposed to only the rules relevant to the problem solution. On the other hand, the structural model generation task is inherently more complex, due to the higher diversity of component types, as shown in the following.

The node color indicates the rule set, which contains the rule that provides the node value. The color schema from figure 5.2.2 is retained. By analyzing the data given for the root nodes, dependencies from other rule sets can be identified. On the one hand, it can be seen that the rule set contains a number of references to the subpackage providing the CPACS Interface. This implies that data is used, which is taken directly from a CPACS data set. On the other hand, some outputs of the loft rule set in the OML model generation subpackage described in the previous section are also used. This shows that, in addition to the CPACS data, geometric information of the OML is also required, which is generated from other CPACS data, via the loft rules. The same applies to some wing geometry details provided vial the wing model generation rule set.

Due to the complexity of the rule set graph, no description of a node-by-node traversal as found in the preceding section is given. Instead the general model generation approach for the relevant types of structural components is explained in the following, starting with the primary structure in section 6.1.2.1 and floor structure in section 6.1.2.2. Then, wing-fuselage intersection areas, including the wing-boxes, are considered in section 6.1.2.3 before ending the section with the structural cutouts in section 6.1.2.4. Nevertheless, all the descriptions are reflections of the rules represented in the rule set graph given by figure 6.1.8. Consequently, any intermediate value given in the figure can be queried, resulting in the extraction of an FPG to determine all unknown ancestors, analogously to the loft generation shown in figure 6.1.1.

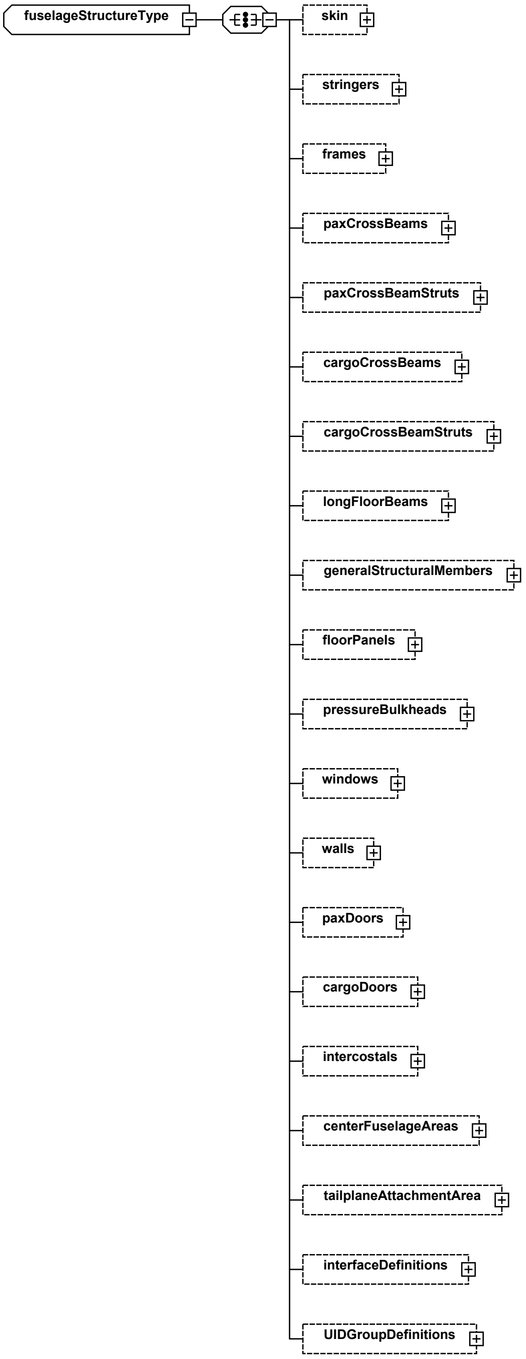


Figure 6.1.7.: Diagram of `fuselageStructureType` in CPACS

6.1.2.1. Primary structure

Following the grouping of components given by Österheld, the primary structure of the fuselage is composed of the entries for the paneling and the frames according to table 4.2.1. The component description in CPACS by Scherer and Kohlgrüber assumes a slightly different grouping, but essentially provides all relevant component data. In the following, the model generation approach for all CPACS component definitions is given, which fall under the umbrella of the primary structure according to Österheld.

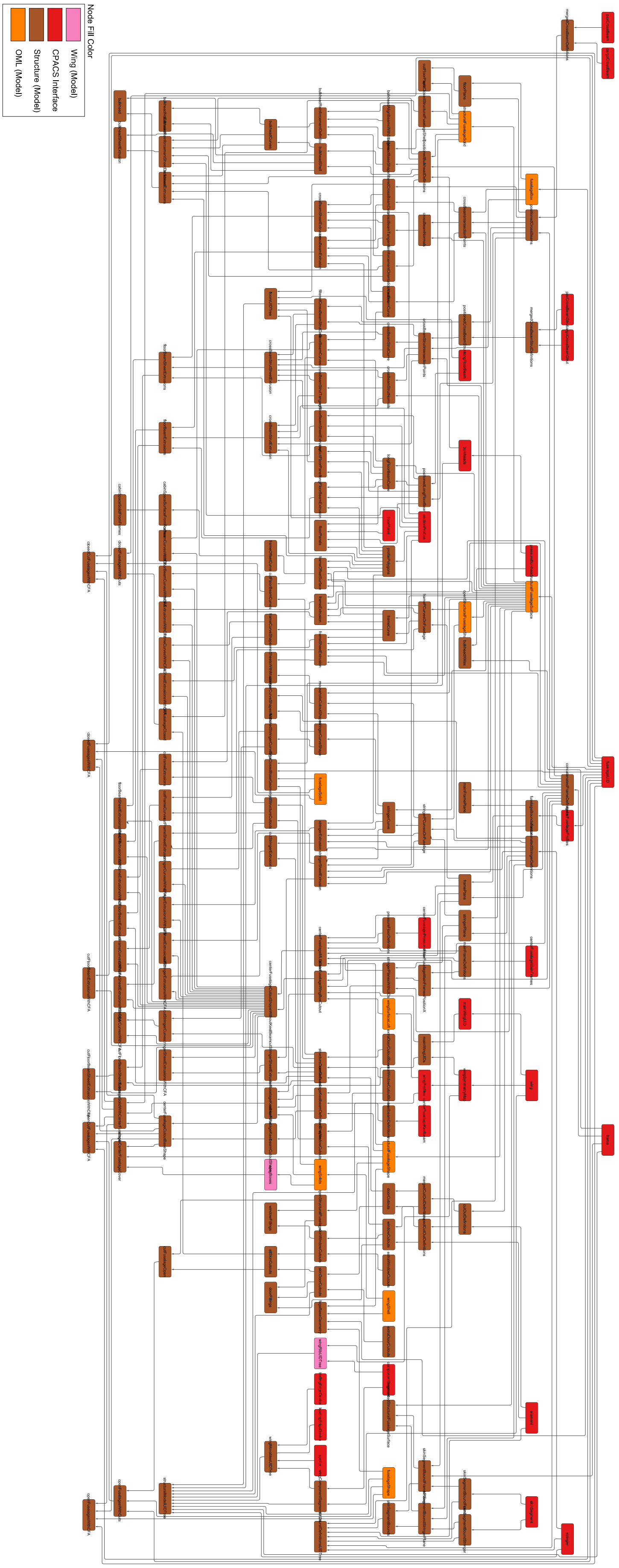


Figure 6.1.8.: MCG of the structural model generation rule set

Frame and stringer grid The primary structure grid in CPACS is given by the distributions of the structural stiffeners, i.e. the frames and stringers. The circumferential frames serve to maintain the shape of the fuselage cross-section [Niu88] and carry loads due to the interior pressure [Öst03]. Longitudinal stringers primarily carry axial loads resulting from the bending moment in the fuselage [Niu88].

The structure grid plays a fundamental role in the description of the fuselage structure in CPACS as many of the structural components considered in the subsequent sections are defined via references to its members. Both frames and stringers are described as topologically 1D geometric entities, i.e. curves, which are assigned cross-sectional properties via a profile-based structural element definition, analogously to beams in structure mechanics. The curves are constructed from definition surfaces, which are described using the `stringerFramePositionType` in CPACS, shown in figure 6.1.9.

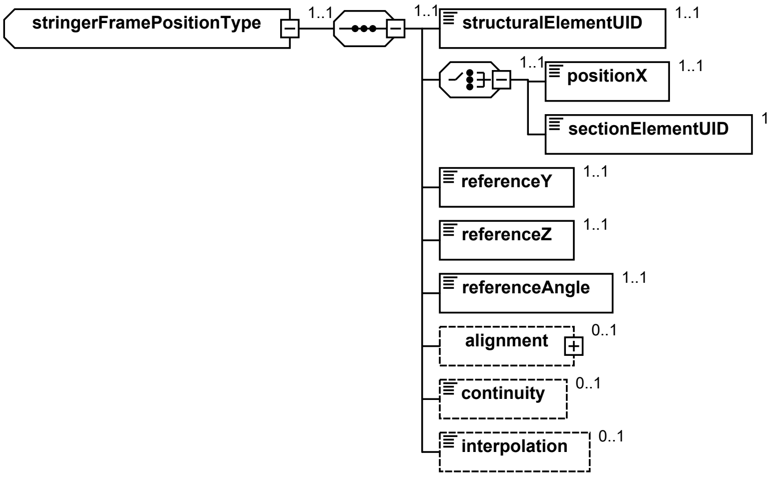
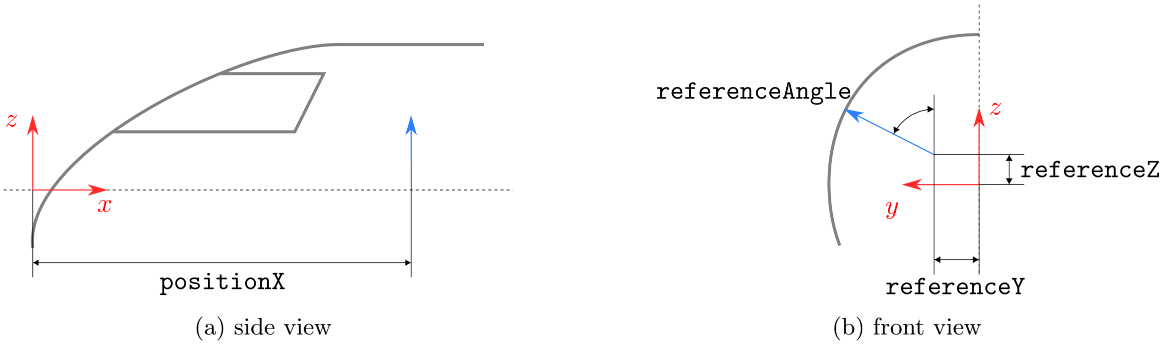


Figure 6.1.9.: Diagram of the stringer and frame position definition on CPACS

Each stringer or frame position defines an intersection vector, which describes a point on the fuselage surface as illustrated in figure 6.1.10. The origin of the vector is given by the `positionX`, `referenceY` and `referenceZ` entries. Instead of providing the longitudinal position explicitly, a reference to a section of the OML can also be provided. The `referenceAngle` provides the direction of the vector, which is assumed to be perpendicular to the longitudinal axis of the fuselage.



(a) side view

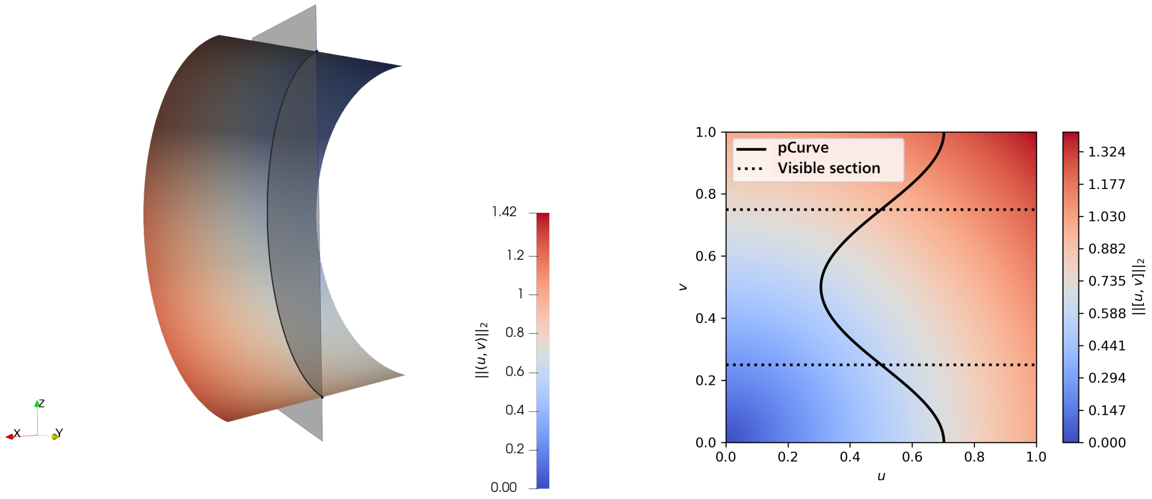
(b) front view

Figure 6.1.10.: Stringer and frame position definition in CPACS (after [WC18])

If only a single position is provided, it is assumed that the vector describes the plane parallel to the yz-plane, in which it lies. This approach is typically used to define frame planes. To describe more complex shapes, a sequence of multiple positions can be given. In this case, the definition surface is built by interpolating between the vectors at the different positions. A further parameter `continuity` is provided to control the interpolation. The FUGA modeling algorithm accepts integer values from

-1 upwards for this parameter. A continuity value of -1 results in the extrusion of the surfaces defined by the vectors along the longitudinal axis, which requires the introduction of discontinuities between the vectors of neighboring positions. Otherwise, a B-Spline interpolation of the corresponding degree is performed between the curves representing the vector.

Intersecting the definition plane and the fuselage OML surface as shown in figure 6.1.11a yields the structural component definition or trajectory curve $\mathbf{T}(v)$. As part of the intersection using OCCT, the projection of the intersection curve onto the parameter or uv-space of the surface $\mathbf{T}_{uv}(v)$ is also computed and can be made available separately in the knowledge repository. This curve is commonly referred to as the p-curve and is shown in figure 6.1.11b.



(a) Cartesian space

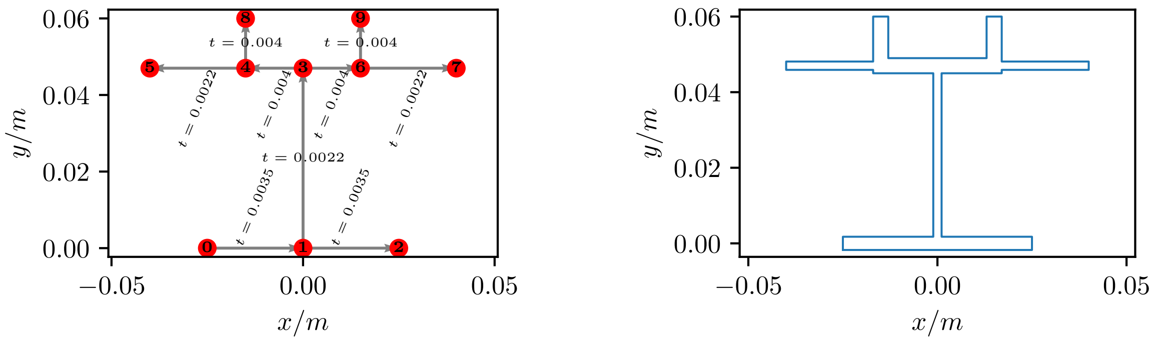
(b) Parameter space

Figure 6.1.11.: Computation of frame curve described by an oblique definition plane in Cartesian and parameter space (from [WH+22a])

To retrieve a more detailed 3D representation of the structural component, the profile-based structural element description must be taken into account. The descriptions are provided separately from the structural component definition in CPACS and can be referenced via the `structuralElementUID` node. A profile-based structural element definition in CPACS consists of a geometric description of the profile cross-section and a reference to a material. One option for modeling the cross-section is a structural profile, which describes the profile as a set of points in 2D, which are connected by sheets, as shown in figure 6.1.12a. In the profile-based structural element definition each sheet is assigned a thickness and a material. This information can be used to construct a polygon representation of the cross-section as illustrated by figure 6.1.12b. The polygon can also be represented by a first-degree B-spline curve.

With both, the component definition curve and the profile curve available, a volumetric representation of the structural component can be constructed using the swept surface algorithm, introduced briefly in section 4.1.1.2. The capacity to provide geometry models of structural components based on CPACS data sets at this level of detail cannot be found in any other parametric modeling engine. Still, such structural details are useful both for the cabin design, as described in section 6.2.3, and for the derivation of detailed models for high-fidelity FEM analysis and immersive visualization. The procedure for constructing the frame geometry using the swept surface algorithm is illustrated in figure 6.1.13 for the example curve shown previously.

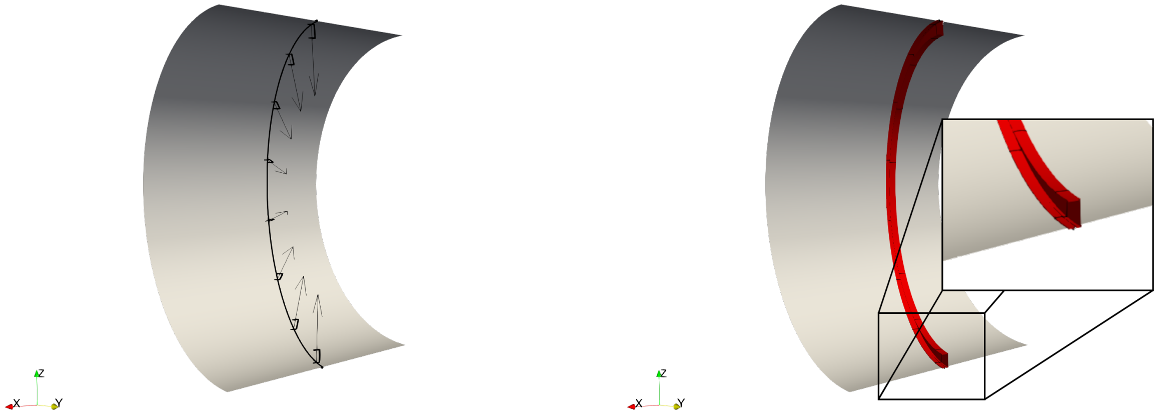
The swept surface algorithm is based on the skinned surface algorithm, which is applied in the loft rule set to generate the surfaces of the OML. However, instead of a sequence of curves, the surface is given by a trajectory curve $\mathbf{T}(v)$, which in this case corresponds to the definition curve of the component, and a section curve $\mathbf{C}(u)$ (the profile curve). Based on this information the surface



(a) Profile definition with sheets and thicknesses t

(b) Profile polygon with modeled thicknesses

Figure 6.1.12.: Cross-section profile description in CPACS



(a) Transformed sections

(b) Extruded surface

Figure 6.1.13.: Extrusion of frame surface using swept surface algorithm (from [WH+22a])

$\mathbf{S}(u, v)$ is given by

$$\mathbf{x}_{swept}(u, v) = \mathbf{T}(v) + \mathbf{A}(v)\,\mathbf{S}(v)\,\mathbf{C}(u) \tag{6.1.1}$$

after Piegl and Tiller [PT96], where $\mathbf{A}(v)$ is a rotation matrix and $\mathbf{S}(v)$ is a scaling matrix. On the one hand, the section curves are assembled using pre-scaled profiles, which means the scaling matrix can be neglected, i.e. $\mathbf{S}(v) = \mathbf{I}$. The determination of the rotation matrix $\mathbf{A}(v)$, on the other hand, is more challenging. For the frames and stringers it can be computed based on the trajectory curve tangent vector $\dot{\mathbf{T}}(v) = \frac{d\mathbf{T}(v)}{dv}$, since the planes of the profiles should be perpendicular to the trajectory curve. If the trajectory curve is a B-spline curve, as is the case in FUGA, the tangent vector is given by evaluating the derivatives of the basis functions. Furthermore, the profile should be aligned normal to the fuselage surface, due to the attachment of the stiffeners to the skin. To determine the normal vector, first the uv-parameter values of the curve point at a given position v on the fuselage surface must be determined by evaluating the corresponding p-curve $[u_{surf}, v_{surf}] = \mathbf{T}_{uv}(v)$. The normal vector of the fuselage surface $\mathbf{x}_{fuse}(u_{surf}, v_{surf})$ can be evaluated at these coordinates. It is given by the cross product of the tangent vectors in both topological directions, i.e.

$$\mathbf{n}_{fuse}(v) = \mathbf{n}_{fuse}(u_{surf}, v_{surf}) = \frac{\partial \mathbf{x}_{fuse}(u_{surf}, v_{surf})}{\partial u} \times \frac{\partial \mathbf{x}_{fuse}(u_{surf}, v_{surf})}{\partial v}. \tag{6.1.2}$$

The two vectors are perpendicular to one another and provide the depth (x, given by $\dot{\mathbf{T}}(v)$) and height (z, given by $\mathbf{n}_{fuse}(v)$) direction of the local profile orientation coordinate system. The width (y) direction is given by the cross product. The normalized representations of three vectors combined yield the rotation matrix $\mathbf{A}(v)$.

This approach is applied e.g. for the stringers. However, in other cases, such as the frames, a different approach to determine $\mathbf{A}(v)$ may be used. For instance, in FUGA, frames must be aligned with the definition plane, rather than be normal to the fuselage surface. Therefore, the frame plane normal is selected for the width direction, as opposed to selecting of the fuselage normal as height direction. The height direction can then be computed analogously via the cross product. Following the rotation, a translation of the profile $\mathbf{C}(u)$ is also performed, which is simply given by the point on the trajectory curve $\mathbf{T}(v)$.

In figure 6.1.13a a series of transformed profiles is shown along with the vectors indicating their height direction. They can be computed, simply by evaluating the above transformations at multiple positions on the trajectory given by the sample coordinate vector $\mathbf{v}$. Based on such a series of sample profiles computed at the sample positions $\mathbf{v}$, an extruded B-spline surface as shown in figure 6.1.13b can be constructed using the skinned surface algorithm.

CPACS contains several additional parameters providing further control of the local alignment of the cross-section profile of frames and stringers, which are given by the `alignment` node in figure 6.1.9. It provides the means to transform the profile in its local coordinate system prior to the extrusion, including offsets in local y and z direction as well as rotation around the local x axis.

Finally, the `interpolation` node provides control over how different section curves are combined. It may take the values 0 or 1 in a sense of True or False. Different sections may occur if a structural component is defined by more than one stringer or frame position, where the profile at the first position is $\mathbf{C}_0(u)$ and the profile at the second position is $\mathbf{C}_1(u)$. If interpolation is turned off, the previously described swept surface approach can be used on each curve segment individually using the respective first profile. Otherwise, a dependency on the trajectory parameter must be added to the profile curve $\mathbf{C}(u)$ in equation 6.1.1. Using a linear interpolation approach between the profile curves at the positions

$$\mathbf{C}(u,v) = \mathbf{C}_0(u) \cdot (1-v) + \mathbf{C}_1(u) \cdot v, \tag{6.1.3}$$

assuming $v \in [0,1]$.

Sheet-based extrusions of the structural components are also provided in FUGA. In contrast to the previously described volumetric extrusion, the sheet thickness of the profile is not modeled. This means that the swept surface extrusion is performed for each sheet individually, instead of a common profile polygon. The approach is useful e.g. for generation of shell-based FEM models as shown by Hesse et al. [HW+23] or more lightweight visualization meshes.

Skin Aside from the grid structure of frames and stringers, the skin is another elementary part of the fuselage primary structure. According to Wiedemann [Wie96], the skin is meant to transfer the loads due to the interior pressure to the frames and stringers and carry most of the shear loads resulting from transverse or torsional loads introduced into the fuselage e.g. during maneuvers.

The structural definition of the skin in CPACS is given in terms of the frame and stringer grid. A skin sheet on the fuselage surface is bounded by a pair of frames and a pair of stringers and is assigned a sheet-based structural element, via the `sheetElementUID` node. The sheet-based structural element in turn contains a reference to a material and a thickness. For all areas of the fuselage surface that are not assigned a skin sheet, a standard sheet element is provided. Analogously to profile-based structural elements for curve-based structural components, sheet-based structural elements are used for any surface-based structural components, analogous to structural shell elements. As shown e.g. by Scherer et al. [SK+13], a skin sheet may span several frame and stringer bays. In this way, larger regions of the fuselage can be assigned the same properties in a way that can also be communicated to the sizing process. This allows for the incorporation of respective manufacturing constraints and shell concepts.

A skin sheet distribution for the example fuselage is given in figure 6.1.14. Visible are four circumferential skin sheets, which are bounded by a total of four stringer curves, and 25 longitudinal sheets, bounded by 26 frame curves. The radome is not assigned to a skin segment and is thus assigned the standard sheet element. The upper skin sheets are defined across the seam of the fuselage, which is also visible in the figure.

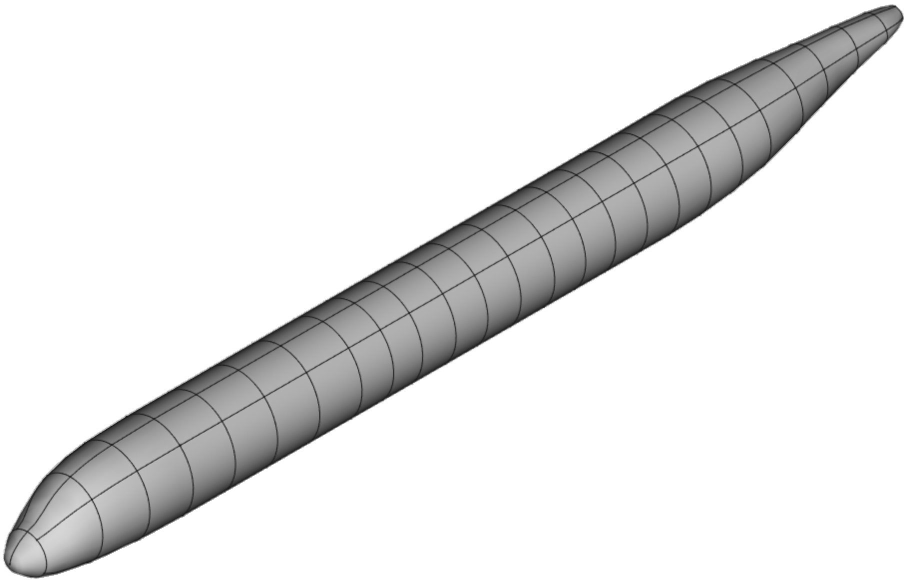


Figure 6.1.14.: Skin sheets of an example fuselage

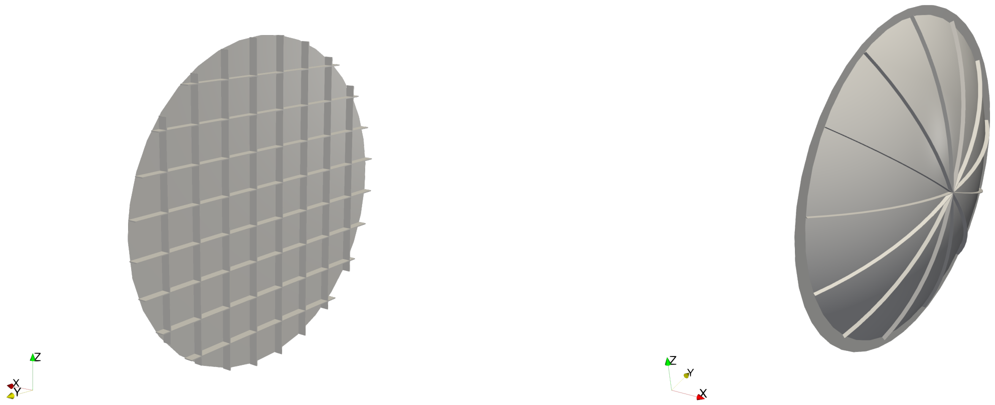
A simple way to generate geometric representations of the skin segments is to recursively perform splitting operations on the B-rep representation of the fuselage surface using all the frame and stringer definition planes listed in the skin segment definition. This procedure will return a shell representation of the fuselage, where each subcomponent represents an individual skin segment. For further processing, e.g. meshing for FEM, it is important that those segment faces, which are split by the fuselage seam, are identified and merged into shells (in a CAD sense). In this way, they are identified as a single component by the mesher and can thus be assigned common sheet properties.

Pressure bulkheads Pressure bulkheads are typically found at both ends of pressurized transport aircraft fuselages. The pressurization must be taken into account e.g. to realistically model the fuselage load for load cases at higher altitudes. In common FEM solvers, pressure can be applied to shell elements as a boundary condition. However, if the representation of the pressure vessel is not watertight, this will result in unwanted and unrealistic reaction forces may occur. This would be the case if the bulkheads were omitted. Therefore, providing a model of the pressure bulkheads is essential.

In CPACS, bulkheads are always placed at a frame position. Consequently, instances of a bulkhead in a given fuselage are always defined by a `frameUID` and a `pressureBulkheadUID`. The latter provides a reference to a detailed definition of the bulkhead, which is stored in the `structuralElements` node, whereas the former prescribes the position in longitudinal direction.

As of version 3.4, CPACS supports two different types of bulkheads, which can both be described using the `pressureBulkheadType`. Both types consist of a base sheet and reinforcements. The base sheet of flat bulkheads can be built simply by filling the plane circumscribed by a frame, whereas spherical bulkheads are characterized by a dome shape, which is better suited to sustain pressure. Flat bulkheads are reinforced by a grid of horizontal and vertical elements, while the stiffeners of the spherical bulkhead are oriented in radial direction. Examples for both types of bulkheads are given in figure 6.1.15.

The generation of the geometry model for the flat bulkhead is straightforward. First, the base sheet is computed. This can be accomplished e.g. by constructing a filling face for the frame curve,



(a) flat

(b) spherical

Figure 6.1.15.: Pressure bulkhead layouts supported in CPACS

or by computing the intersection between the frame definition plane and the solid representation of the fuselage. Then, the positions of the horizontal and vertical reinforcements are determined by computing an even distribution between the horizontal and vertical bounds of the face.

By comparison, the model generation for the spherical bulkhead requires more steps, as the flat base sheet is combined with a spherical dome. The pressure dome shape is defined by its depth t_{dome} and the radius at the frame r_{frame}, both of which are provided in CPACS, via the nodes `maxFlectionDepth` and `bulkheadCalotteRadiusAtFrame` respectively. The information can be used to compute the radius of the base sphere:

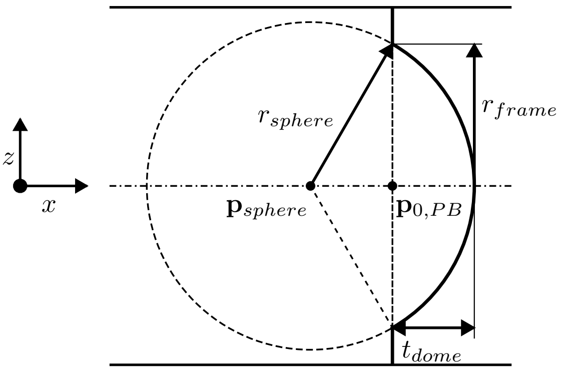


Figure 6.1.16.: Sketch of the relevant parameters of the spherical bulkhead base sheet

$$r_{sphere} = \frac{r_{frame}^2 + t_{dome}^2}{2 \cdot t_{dome}}. \tag{6.1.4}$$

It is assumed that the dome is centered around the frame curve centroid $\mathbf{p}_{0,PB}$. It follows that the center of the sphere for a spherical RPB is given by

$$\mathbf{p}_{sphere} = \mathbf{p}_{0,PB} + \begin{bmatrix} (t_{dome} - r_{sphere}) & 0 & 0 \end{bmatrix}^T. \tag{6.1.5}$$

To assemble the merged base sheet shape, the sphere solid representation is subtracted from the flat base sheet using a CAD Boolean operation. In turn the shell representation of the sphere is intersected with the half space behind the frame plane. The two resulting shapes can be merged to form the base sheet of the spherical bulkhead. The definition planes for the reinforcements are arranged around $\mathbf{p}_{ref}$ in regular angular distance and extend along the x-axis.

To compute extruded representations of the bulkhead reinforcements, the same approach as for the frames and stringers is used, regardless, which type of bulkhead is used.

6.1.2.2. Floor structure

The primary purpose of the floor structure is to provide accommodation for passengers and payload. However, as pointed out e.g. by Niu [Niu88], the components of the floor structure are also integral parts of the overall structural concept of modern aircraft. For instance, the floor crossbeams are attached to the frames to act as tension ties, alleviating the loads on the primary structure due to the interior pressure for non-circular cross-sections. Another aspect driving the design of the floor structure is crash. The importance of this topic is illustrated e.g. by Schatrow and Waimer [SW16], who investigate different design concepts to ensure energy absorption in the event of a crash is sufficient when changing the material of the floor structure from aluminum to CFRP.

In alignment with the component grouping by Österheld in table 4.2.1, the floor structure components in CPACS comprise the floor crossbeams, support struts, longitudinal beams and floor panels. Like the frames and stringers, the crossbeams, support struts and longitudinal beams are defined as profile-based structural elements. Consequently, they are described using a curve and cross-section properties. CPACS distinguishes between cargo and passenger floor structures, which are, however, composed of the same underlying component types. In FUGA, structural components of all floors are therefore combined and processed together in a unified way.

Both crossbeams and crossbeam struts are always placed inside a frame plane. The crossbeams extend in width direction with the height given by the `positionZ` node as illustrated by figure 6.1.17. The support struts are in turn connected to the crossbeam at the absolute y position given by the `positionYAtCrossBeam` node and at an angle around the longitudinal axis of the fuselage given by the `angleX` node. Both the crossbeam and the strut curves are straight lines, limited by the OML of the fuselage, which makes them easy to handle numerically, compared to the primary structure curves.

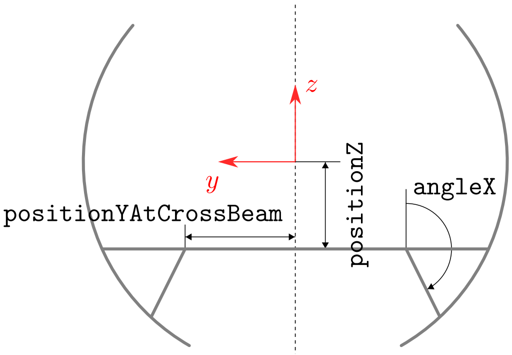
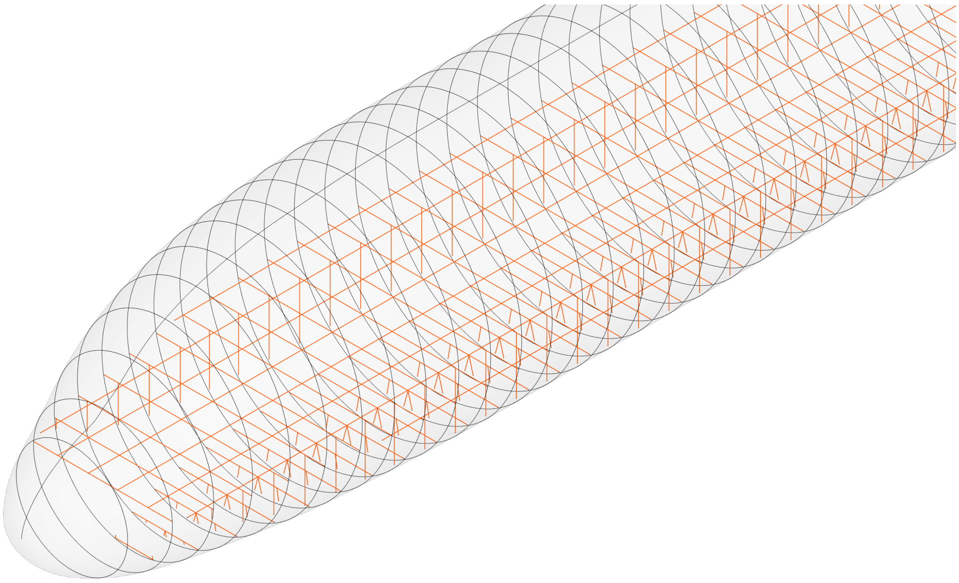


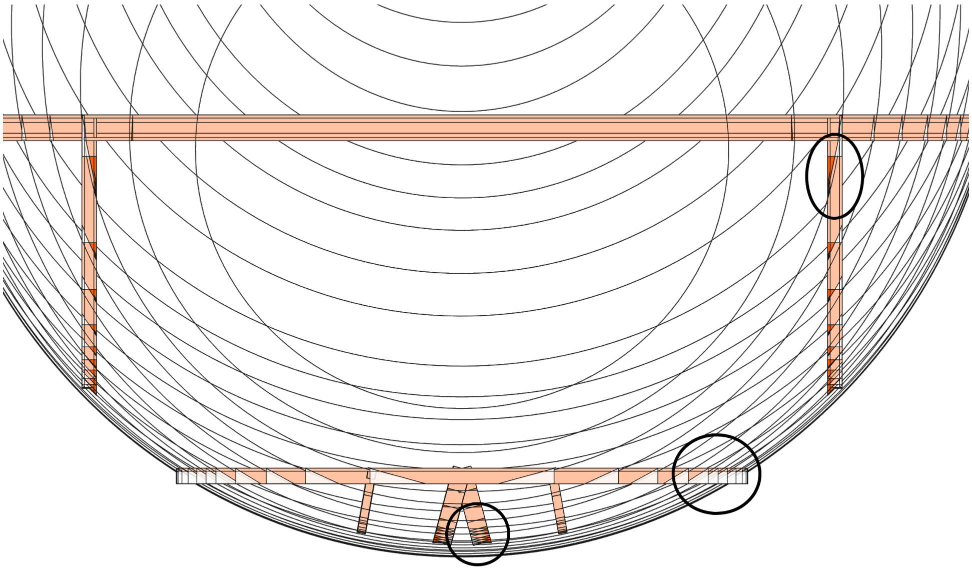
Figure 6.1.17.: Sketch of the floor structure definition within the frame plane (after [WC18])

Longitudinal floor beams, which comprise both seat rails and simple floor support beams, can be defined by providing multiple positions, similarly to stringers. Each position is described by a dedicated `longFloorBeamPosition` node. Similarly to the crossbeam strut, the floor beam position is given at a crossbeam, which is identified via its uID. The position on the beam is given by the `positionY` parameter. In this way, a polyline spanning multiple crossbeams can be described. As for the stringer and frame position definitions, `continuity` and `interpolation` settings are once again available to control the curve interpolation continuity and cross-section shape interpolation respectively. The resulting curves for the example configuration are shown in figure 6.1.18a.

Visibly, the crossbeam curves and the longitudinal beam curves lie in the same plane. This is acceptable or even desirable in some cases, e.g. when building simplified GFEM models of the floor



(a) Floor beam curves



(b) Extruded floor beams with (orange) and without adaptation to the fuselage surface (long beams hidden)

Figure 6.1.18.: Floor geometry generation intermediate steps

structure, but causes problems, once extrusion of the profile cross-sections is performed. The extrusion is again implemented using the swept surface algorithm, as for the frames and stringers. Depending on the definition of the respective cross-sections, the resulting volumetric models are likely to collide. For this reason, the `alignment` node is essential when defining floor structures, as it allows for the definition of vertical offsets for each profile.

Differently to the stringers, the alignment node for floor structure components provides an additional option to define offsets in local x direction, i.e. in curve tangential direction, at both ends of the definition curve. In this way, collisions e.g. with adjacent frames or between struts and crossbeams can be avoided.

An additional option in FUGA is to project the closing profile onto the normal vector of the fuselage at the intersection point with the definition curve. This reduces the penetration of the fuselage surface by parts of the extruded floor structure, as shown in figure 6.1.18b, which would be detrimental e.g. to the immersion in VR-driven human factors analysis. At the same time, the approach using the surface normals is significantly more computationally efficient than computing the intersection between the frame and the solid representation of the fuselage. The approach is only applied to crossbeams and struts, but not to longitudinal beam, which in many cases do not end at the outer fuselage surface. Therefore, they are omitted in figure 6.1.18b.

Finally, the floor panels are added between the longitudinal floor beams. They are the only sheet-based component type of the floor structure. In CPACS, the geometry of a floor panel is described by the `floorPanel` node, which provides uIDs for a pair of longitudinal floor beams as well as a pair of x-positions. In this way, the bounds of the floor panel are given. The `sheetElementUID` node provides a reference to the mechanical properties. An `alignment` node is once again available to determine the offsets from the longitudinal beam definition curves in lateral and vertical direction.

Assuming the longitudinal floor beams to be defined by polylines that lie within the floor plane, the floor panels can be modeled as flat polygons, which can be described by an ordered sequence of points. To determine the corresponding points, both adjacent longitudinal floor beam curves must be evaluated at the given x-positions. This can be done efficiently via linear interpolation of the curve points w.r.t. the x-coordinate. In addition, all curve points lying between the bounding x-positions must be included in the polygon. The offsets defined in the `alignment` node can then be applied to the points. Lateral offsets are applied individually to the points on each longitudinal floor beam curve, whereas the vertical offsets are applied globally. The order of the points on one of the curves must be reversed before joining the two point lists to match the required order of the polygon.

A detailed floor representation including the floor panels and extruded representation of the floor beams is shown in figure 6.1.19.

6.1.2.3. Wing-fuselage intersection areas

The wing-fuselage intersection areas are critical in ensuring a realistic load transfer from the wing to the fuselage in structural analysis [Pet15]. Furthermore, like the bulkheads, they are part of the pressure vessel for pressurized aircraft, separating the pressurized interior e.g. from the unpressurized landing gear bay. For low-wing aircraft, the center fuselage area also introduces an interruption in the cargo floor.

An initial description of wing-fuselage intersection areas in CPACS has been proposed by Scherer and Kohlgrüber [SK16] for conventional low-wing aircraft. They differentiate between the center fuselage area, which comprises the wingbox and the landing gear bay and the tailplane attachment area, which provides the structural connection for the VTP and HTP. Petsch [Pet15] extends these definitions to meet the requirements of SBW configurations, adding support for high-wing center fuselage areas and T-tail attachments. Some fundamental features of the definition for low-wing aircraft given in figure 6.1.20 are supported by FUGA. The choice of the features is primarily motivated by their relevance to fuselage and cabin design.

For instance, the `centerFuselageMainFrames` node provides essential information for the cargo floor design. A total of three frame uIDs are given by the node. The first two uIDs refer to the

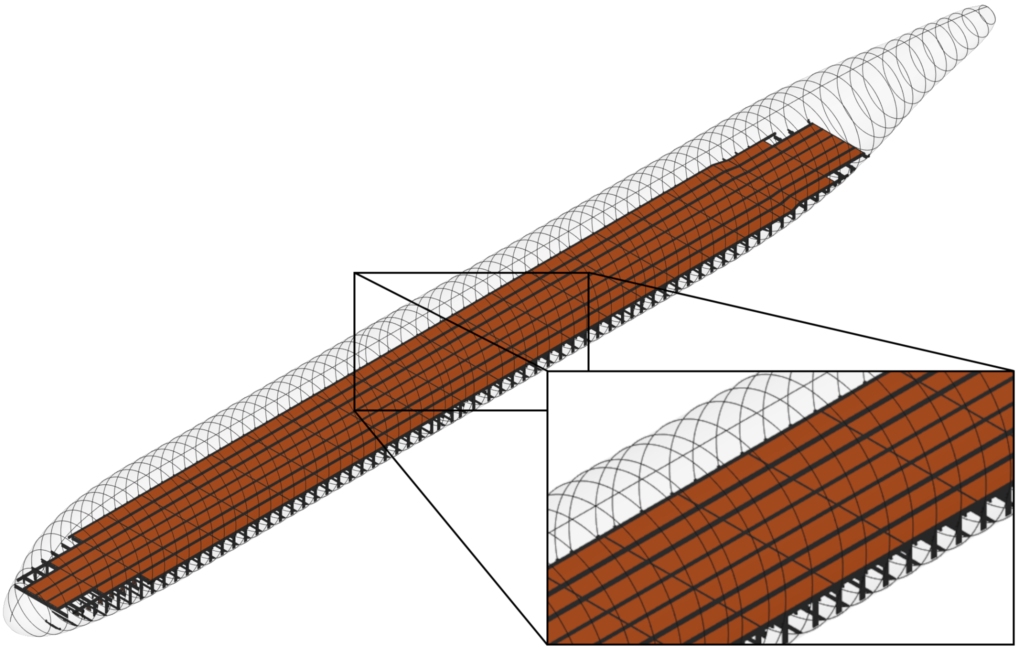


Figure 6.1.19.: Detailed floor structure geometry model

main frames adjacent to the wingbox. It is also assumed that the landing gear bay is located directly adjacent to the wingbox. As such, it is bracketed by the second wingbox mainframe and the third frame given by the definition. If a lower deck with a cargo floor is installed, it usually interferes with the wingbox and the landing gear bay and, as a result, must be interrupted in the space between the frames given in the `centerFuselageMainFrames` node.

The definition of the wingbox geometry in CPACS is provided separately from the center fuselage definition, which means consistency between the frames referenced in the node and the actual geometry must be verified. Any wing structure is defined in the 2D space of the wing midplane defined by the sequence of the lines between the leading edge and the trailing edge center for each profile section. The space is parameterized by the spanwise coordinate η and the chordwise coordinate ξ. The wingbox is bounded by the front spar and the rear spar. As shown in figure 6.1.21a, the geometry of a spar is defined by a sequence of points on the wing midplane, given in $\eta\xi$-coordinates. The resulting line in CARTESIAN space is then extruded in positive and negative z-direction to form the frame definition surface. A solid is constructed between these two surfaces using OCCT, which can then be intersected with the wing solid introduced in section 6.1.1 to give the wingbox shape as shown in figure 6.1.21b. The first two main frame planes from the center fuselage area definition should align with the spar surfaces in the region of the center wing box. The spars are usually supplied by openAD and thus need not be computed by FUGA.

Assuming the wingbox and mainframe positions to be consistent, the upper vertical bound for the region between the first two mainframe uIDs is given by the upper shell of the center wingbox. For the region of the landing gear bay, however, a different way to specify this bound is required, which is given by the `centerFuselagePressureFloor` node. While this node contains several parameters concerning additional details, fundamentally, the height of the pressure floor is given via a reference to a stringer by the `positionZStringerUID` entry. It is implicitly assumed that the stringer will yield a constant z-position. From this, it follows that the fuselage cross-section must be constant at the position of the landing gear bay, which poses a limitation on the applicability of definition. An upper boundary surface for the landing gear bay can be determined by mirroring the trajectory curve of

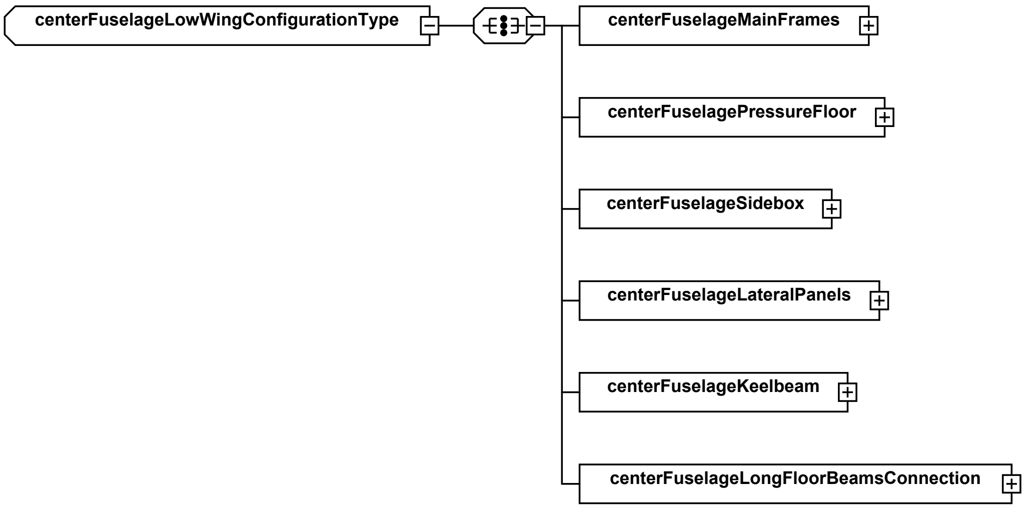

centerFuselageLowWingConfigurationType
centerFuselageMainFrames
centerFuselagePressureFloor
centerFuselageSidebox
centerFuselageLateralPanels
centerFuselageKeelbeam
centerFuselageLongFloorBeamsConnection

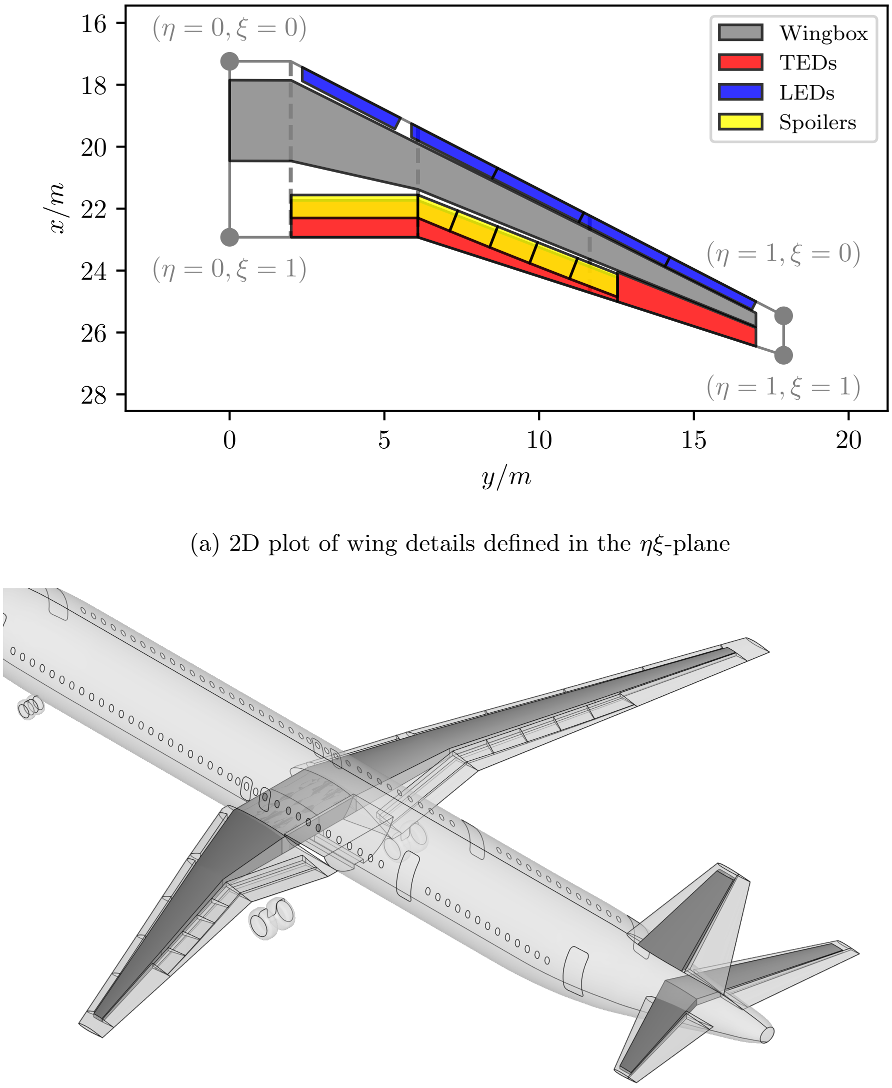
Figure 6.1.20.: CPACS center fuselage area definition for low wing configuration

the stringer specified by the uID on the xz-plane. A surface can then be constructed by computing a skinned surface between the two curves.

Extruding the surface in negative z-direction yields a solid, which can be used subsequently in Boolean operations. To receive a representation of the space claimed by the landing gear bay, the solid can be intersected with the solid defining the space between the two latter mainframes. Analogously, a solid can be created by extruding the upper shell of the wingbox in negative z-direction. The two solids can be merged and subtracted from the fuselage solid representation from section 6.1.1 to yield the cut fuselage shown in figure 6.1.22a. The required surfaces for the bulkheads and pressure floors are created automatically by the solid-based Boolean operation algorithm in OCCT.

From the perspective of structure mechanics, the introduction of a cut in a region, which is subject to significant compressive loads due to the bending moment acting on the fuselage, is highly undesirable. Therefore, a very stiff keelbeam is usually installed, to provide additional structural support [Niu88]. The keelbeam can be described using the `centerFuselageKeelbeam` node. Again, a number of parameters are available, which describe additional details, but the fundamental shape of the keelbeam is once more described using only references to stringers. Similarly to the pressure floor, the upper bound of the keelbeam is given by the `keelHeightStringerUID`. As a result, the upper bound surface can be constructed in the same way. In addition, a second stringer is referenced by the `keelWidthStringerUID`, which provides the width of the keelbeam. This time the stringer curve must be extruded in upward and downward z direction and then again be mirrored on the xz-plane. The definition shape of the keelbeam can then be determined by computing the intersection between the solid defined by the width bounds and the solid provided by the height bound extruded in negative z direction.

Using this definition shape, two ways are available to construct the keelbeam geometry from the fuselage solid. On the one hand, it can be subtracted from the original center fuselage cutout shape, for the keelbeam to be computed automatically as a part of the subtraction of that part from the fuselage solid. This approach has the advantage that the fuselage surface is not interrupted, which is beneficial when generating visualization meshes, since the complete outer skin is found in a single surface. However, the drawback is that the ends of the keelbeam are not closed, leaving openings in the forward and aft bulkheads of the center fuselage area, which must be closed in a separate step. On the other hand, it is desirable for the bulkheads to be meshed as a single closed surface for structural analysis, e.g. to more easily assign element properties. Thus, a second modeling approach is available, where the keelbeam is modeled as a separate feature, by intersecting the definition shape with the



(a) 2D plot of wing details defined in the $\eta\xi$-plane

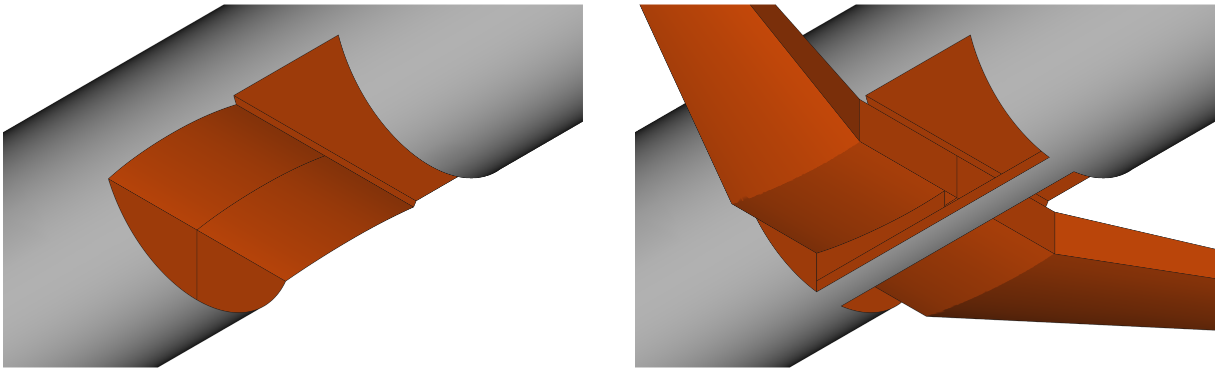
(b) Wingboxes modeled in 3D

Figure 6.1.21.: Wingbox modeling

intersection of the fuselage and the basic center fuselage cutout shape. The result is then combined with the cut fuselage model shown in figure 6.1.22a.

Irrespective of the selected modeling approach, the keelbeam is finally connected to the wingbox using lateral panels, which are simply an upward extension of the keelbeam side wall to the wingbox. They can be computed by intersecting the original sidewall definition planes obtained from extruding the stringer given by the `keelWidthStringerUID` with the space between the first two mainframes and subtracting the space below the keel beam height boundary and above the lower wing shell. The resulting final center fuselage area configuration including the wingbox model is given in figure 6.1.22b. Visibly, the keelbeam has been modeled using the first approach, since the fuselage surface is not interrupted.

The center fuselage area provides a good example of how different components, such as the stringer curves or the mainframes, but also the subsequently created solid bodies, can be re-used in a different context to generate additional details. On the other hand, the strong dependency on frame and stringer definitions means that design parameters for the center fuselage area must also be taken into



(a) Center fuselage area base cutout from fuselage solid

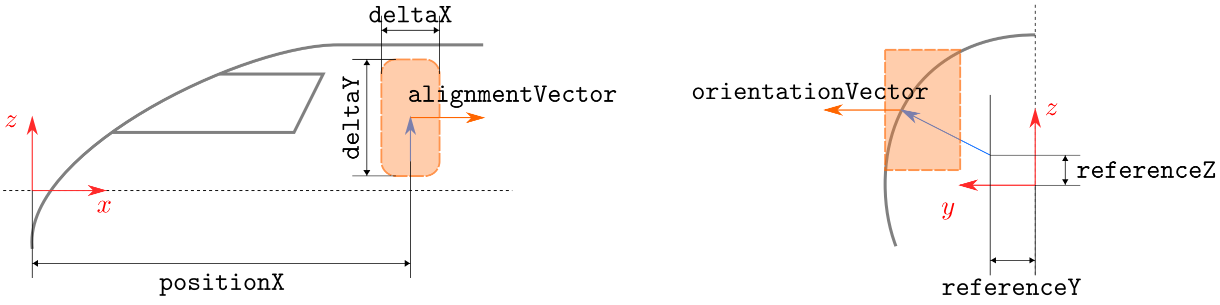
(b) Center fuselage area with keelbeam, panes and wingbox

Figure 6.1.22.: Center fuselage geometry generation steps

account when designing the frame and stringer distributions. This aspect will be discussed further in section 6.2.2.

6.1.2.4. Cutouts

Information on structural openings is essential in fuselage design. For instance, the size and placement of doors directly affects certification, while window positions impact the design of sidewall panels. Several ways to describe structural openings in the fuselage in CPACS have been proposed. Schwinn et al. [SS+13] show detailed FEM models of the door openings and their surrounding structures using a description based on frame and stringer positions. In FUGA, on the other hand, the more general approach to describe structural openings in the fuselage using the `cutOuts` node has been adopted. The geometric definition of a cutout in this node is shown in figure 6.1.23.



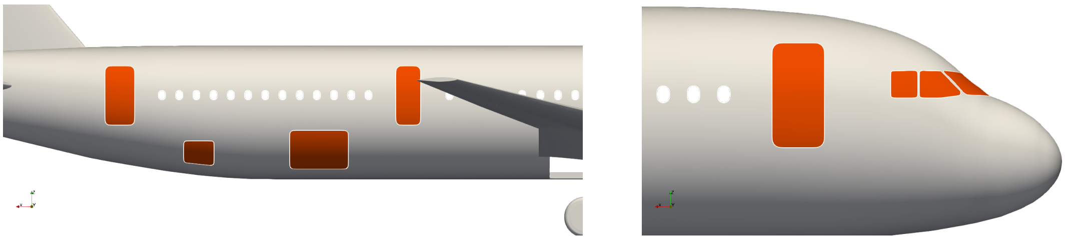
(a) side view

(b) front view

Figure 6.1.23.: Cutout definition (from [WK+22])

Fundamentally, the cutout is described by a solid body, marked in orange in figure 6.1.23, which is to be subtracted from the fuselage shell using BOOLEAN geometry operations. The body is generated by extruding a profile along a vector. The position of the cutout shape is defined by a positioning vector (marked in blue), which determines a point on the fuselage and is defined in the same way as e.g. a stringer position. The origin of the positioning vector is thus again defined by the entries for `positionX`, `referenceY` and `referenceZ`, whereas the direction is given by the `referenceAngle`. At the resulting intersection point on the fuselage a new local coordinate system is created for the cutout profile extrusion. The `orientationVector` describes the local z direction, which is also the extrusion direction. The 2D cutout profile is placed in the xy-plane of the local coordinate system, with the normal projection of the `alignmentVector` prescribing the local x-axis. As of CPACS version 3.4

the profile shape is restricted to a rounded rectangle with the width `deltaX`, the height `deltaY` and the corner radius `filletRadius`. However, different applications in FUGA, such as the description of cockpit windows or non-rectangular bulk cargo doors as shown in figure 6.1.24, require more freedom in profile choice. Therefore, an adaptation of the definition to accept more general profile types, which are also used e.g. for the fuselage shape description, has been proposed.[1]



(a) Rear fuselage area with non-rectangular bulk cargo door

(b) Forward fuselage area with non-rectangular cockpit windows

Figure 6.1.24.: Examples for non-rectangular cutouts

The cutout definition has been adopted in FUGA to describe structural openings, due to its wider range of applications compared to the proposed definition by Schwinn et al. However, since the definition is decoupled from the primary structure grid, KBE rules must be provided to ensure the consistency between the openings and the structure, which are discussed in section 6.2.2.

6.1.3. Cabin model generation rules

The MCG for the cabin model generation rule set is given in figure 6.1.25. Similarly to the structure design MCG, it is also a subgraph of the outside-in design MCG in figure 5.2.2. Once again, any node in the system can be queried, resulting in the extraction of an FPG from the given MCG, which is used to resolve the query.

Once more, the node color indicates the origin of the rules providing the nodes to trace the root node dependencies. The input sources here are visibly less diverse than for the structure, as the description of the cabin is less reliant on the description of the OML. Almost all relevant cabin data is extracted from CPACS. The location of the CPACS file is used to correctly trace references to external files given as relative paths. Apart from this, only the cabin space geometry representation provides a link to the structural layout. This means that the *model generation* for the cabin can essentially be performed independently from the other rule sets. The relationships and constraints to assure consistency of the cabin with the structure are contained implicitly in the CPACS data. Undiscerning manipulation of the values in CPACS can thus easily result in inconsistent designs.

Consequently, the cabin data should be the outcome of a cabin *design* process, as discussed in section 6.2.3, where all the relationships to the structure are provided explicitly.

In the following, the general underlying approaches of the modeling rules implemented in the graph are outlined.

Cabin representation in CPACS As discussed in section 4.1.2, different generations of cabin descriptions within CPACS exist, with the first attempt dating back to Fuchte, Gollnick, and Nagel [FGN13]. In the scope of this thesis, the modernized description by Walther et al. [WH+22a] has instead been adopted, where the cabin is described based on three main columns.

External geometry models constitute the first column of this definition. Unlike the structural model, which relies complex geometric operations to determine the component shape, the cabin definition is based either on these predefined models, which can be provided in a CAD or mesh format, or primitive representations. As a result, the cabin modeling rules are substantially less complex than

[1] A related issue has been created on the CPACS GitHub page [DLR22].

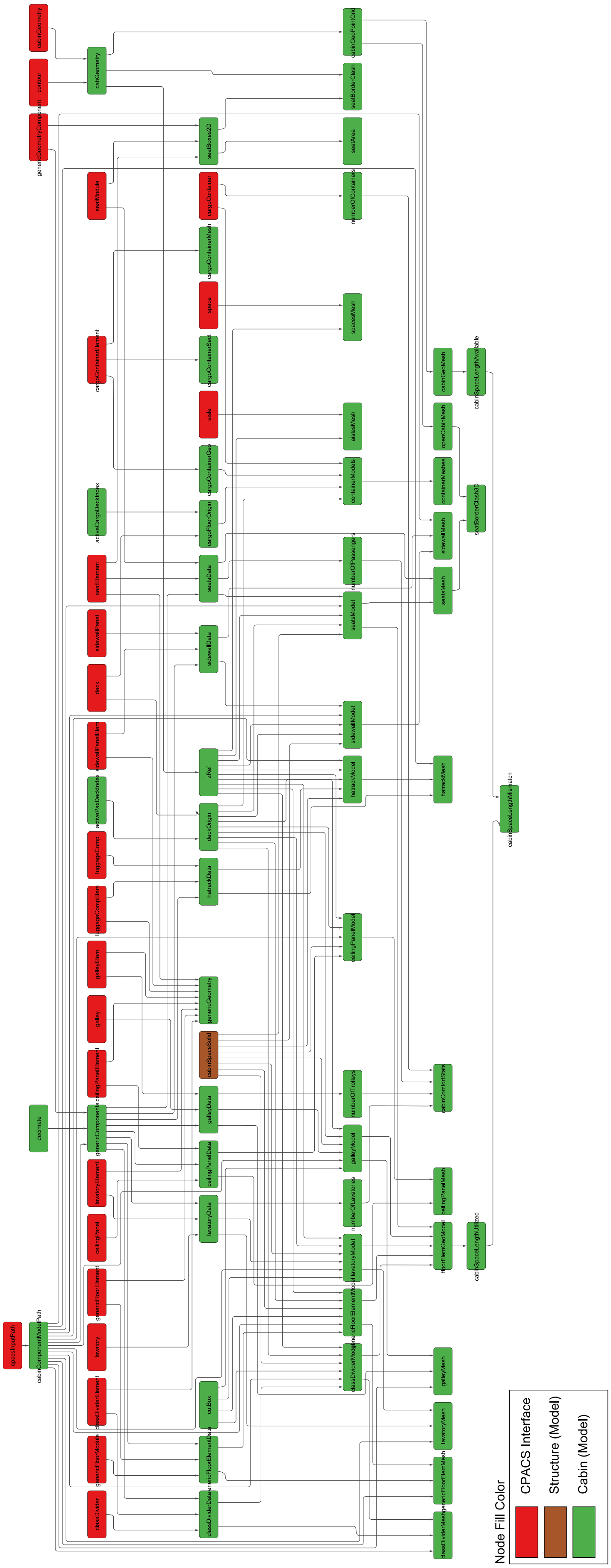


Figure 6.1.25.: MCG for the cabin model generation rule set

their structural counterparts, since the complexity of the component design is encapsulated in the external models.

The second column is the cabin component library node shown in figure 6.1.26, which provides descriptions of all available types of cabin components in CPACS. The supported types include seats and monuments, such as galleys and lavatories, but also elements of the secondary structure, e.g. sidewall panels, ceiling panels and luggage compartments. The geometric shape of the component can be defined either by providing the dimensions of the bounding box, or by referencing the external geometry model. A transformation node can be applied to scale and align the external geometry further. The definition of a broad selection of both floor-based elements and elements of the secondary structure is supported. Floor-based elements include seat modules, galleys, lavatories, class dividers and cargo containers, whereas the secondary structure consists of sidewall panels, overhead stowage compartments, and ceiling panels. Further missing cabin component types can be included using the `genericFloorElements` node, which is used by Walther et al. [WH+22a] to include e.g. bar modules or stairs[2]. Aside from the geometry, the library entries may contain additional metadata relevant to the specific kinds of component. For instance, the entry for a seat module also provides the number of passengers, which can be seated, whereas the definition for a galley contains the number of full-size trolleys, which can be stored.

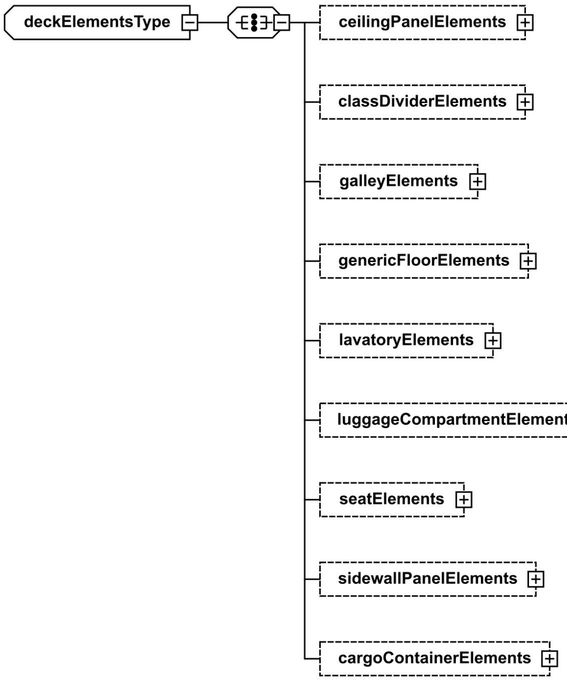


Figure 6.1.26.: XSD diagram of the deck element library

The library node serves to store the details of all the available components in a central location, but the actual cabin layout is then defined in the `deck` node. The node forms the third column of the cabin definition and is given in figure 6.1.27. It is referred to as the instance of the cabin inside the fuselage. Cabin components are placed inside this cabin instance by referencing and transforming entries of the component library. Depending on the type of component, different types of transformations are given. For floor-based components, e.g. seats and monuments, a 2D transformation on the cabin floor plane is applied. Elements of the secondary structure, on the other hand, can be transformed in three

[2] Stairs constitute something of a special case, since they require structural cutouts in floors and may introduce a structural coupling between different floors. Some of these issues are considered in more depth in section 7.1.2. Taking this into account, the introduction of a dedicated stairs type in a future version of CPACS could be of substantial merit.

dimensions to allow for adjustment of the vertical position.

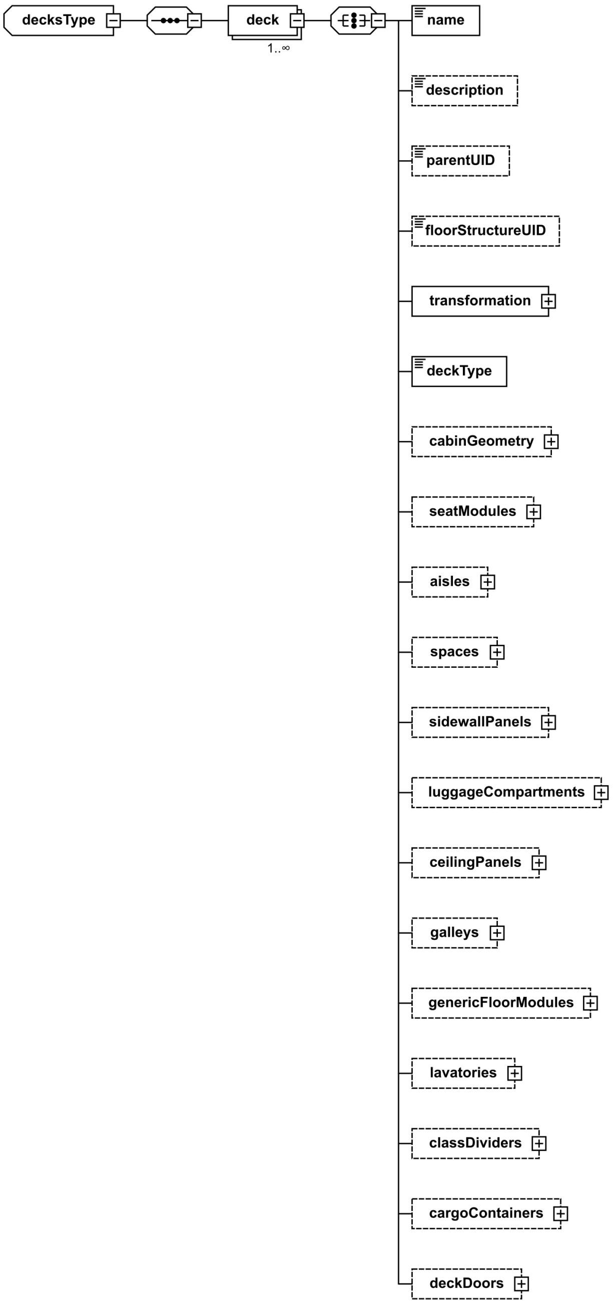


Figure 6.1.27.: XSD diagram of the deck instance in the fuselage

Physical geometry model generation When assembling the cabin model, the first step is to read the external model. Depending on the type of input, this can be accomplished using e.g. the OCCT or VTK library.

If no model is provided, a bounding box is instantiated instead. For some types of models, e.g. monuments, the intersection of the bounding box and the cabin space representation is computed in this case, in order to avoid having the models stick out of the fuselage on the one hand and provide a closer representation of the final component on the other.

A particular challenge during cabin model generation is the consistent handling of CAD geometry and meshes, since it is not clear from the start what will be provided by the user. In either case, the alignment and scaling transformation from the library node must be applied to each component. Then, copies of the components are created and transformed based on the layout description in the deck node. These transformations must be applied consistently to both the CAD and the mesh-based geometry.

Generally, FUGA is capable of providing both an all-CAD and an all-mesh representation of the cabin layout. To this end, all CAD geometry must be converted to triangulated meshes, which can be accomplished e.g. using the Delaunay triangulation algorithm provided by OCCT. Conversely, reliably reconstructing CAD geometry from mesh data is a complex and computationally expensive task. Therefore, only bounding box representations are derived from the meshes, which corresponds to a model fidelity reduction. In figure 6.1.28 both the CAD and mesh representations of the cabin layout for the example configuration are shown. For applications, such as mass analysis, the geometric fidelity can be reduced even further by computing e.g. the centroid to represent the component.

Non-physical geometry model generation In addition to the component definitions, the `deck` definition in CPACS contains non-physical area and volume definitions, which serve different purposes.

The `cabinGeometry` node provides a discretized representation of the available space inside the fuselage. Some authors cite the `cabinGeometry` node as the source for the outer bound of their cabin design activities [FGN13; EDH20]. In FUGA, a different approach is adopted, where the cabin geometry is constructed in CAD based on the structural definitions. This is described in detail in section 6.2.3. In contrast, the `cabinGeometry` node contains a discretized structured grid representation of the space. The parametric description of the grid in CPACS is illustrated by figure 6.1.29. Visibly, the different width values are given for all values in a rectilinear sampling of the xz plane, which are then grouped by height into contours.

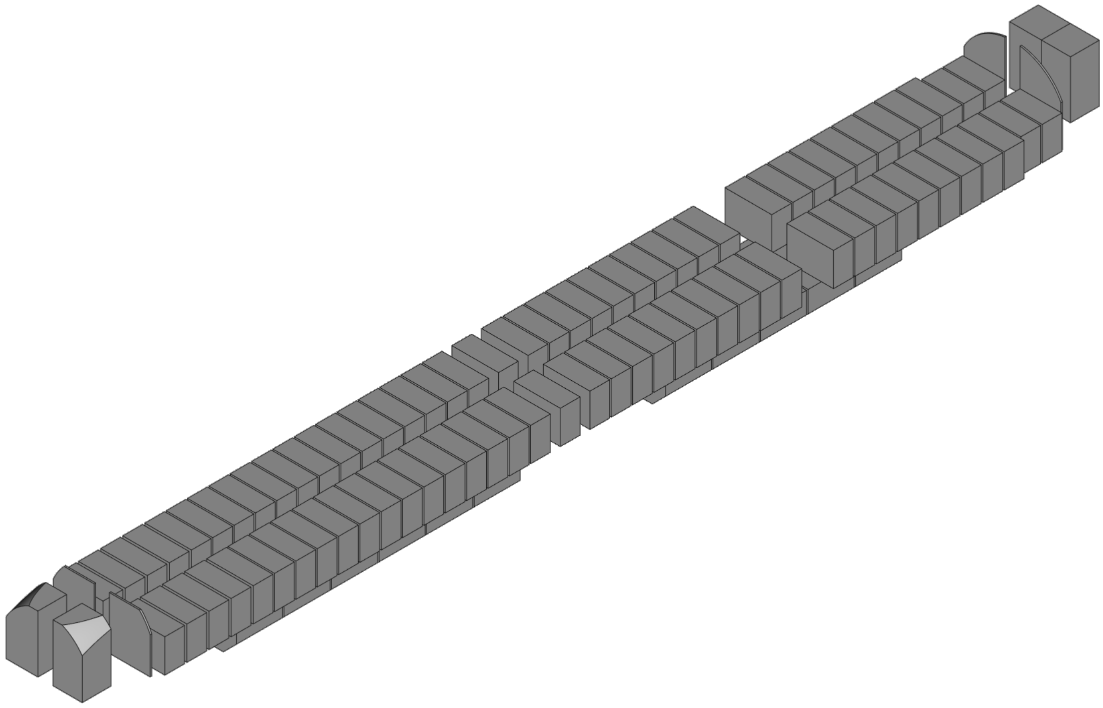
Door definitions are also provided, however, for the updated cabin definition, a design decision was made to make the cutout definitions presented in section 6.1.2 the single source of truth for door geometry. Therefore, the doors given in the deck node simply provide a reference to a cutout. Furthermore, non-geometric information relevant to cabin analysis is provided, e.g. the passenger capacity or the door utilization, which can be one of boarding, cargo, emergency or service.

Other types of non-physical geometry are used to delineate regions of the cabin that should be kept unobstructed, e.g. due to certification requirements. For instance, section CS 25.815 of the certification specification [EAS21] demands a minimum aisle width depending on the passenger number. In section CS 25.813, space requirements are provided to ensure access to the exits e.g. in case of an emergency. Representative control geometries can be defined using the `aisles` and `spaces` nodes in the deck definition respectively, to help assess whether these criteria are fulfilled. Spaces may include e.g. passage ways to exits, cross-aisles or assist spaces, which may be required depending on the exit type. The sketches in figure 6.1.30 illustrate, how the corresponding geometric representations are generated.

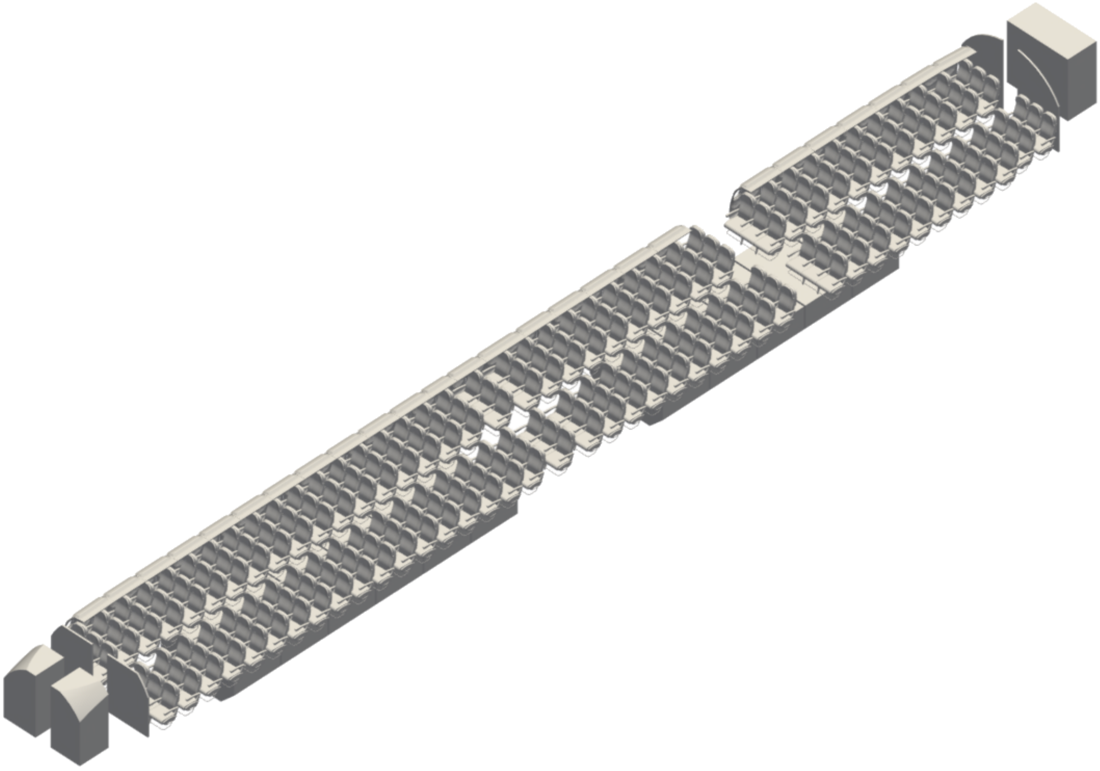
Aisles are represented using a surface on the cabin floor, which is described using a polyline corresponding to the middle of the aisle. The points of the line are defined in xy-coordinates of the cabin coordinate system, implicitly assuming $z = 0$, i.e. the line being at floor level. The width of the aisle in lateral direction, i.e. the y direction of the cabin coordinate system, is given at each point of the polyline.

Similarly, spaces are defined by a closed polygon on the cabin floor. The additional freedom provided by this approach allows for more complex shapes to represent e.g. assist spaces. The polygon is extruded in height direction to form a volume. The length of the extrusion is given by the height parameter.

The non-physical geometry representations are not usually relevant in structural analysis models. They may, however, be required for human factors evaluation e.g. for ingress or egress simulation. These are usually performed in 2D as discussed in section 4.2.3.3. More advanced human-in-the-loop simulations for comfort assessment as shown in section 4.2.3.1 are less likely to require the models, unless very specific clash analyses are to be performed.



(a) CAD



(b) Mesh

Figure 6.1.28.: Cabin representations for the example configuration

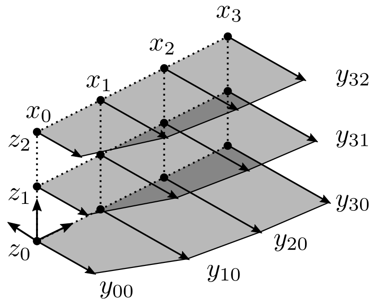
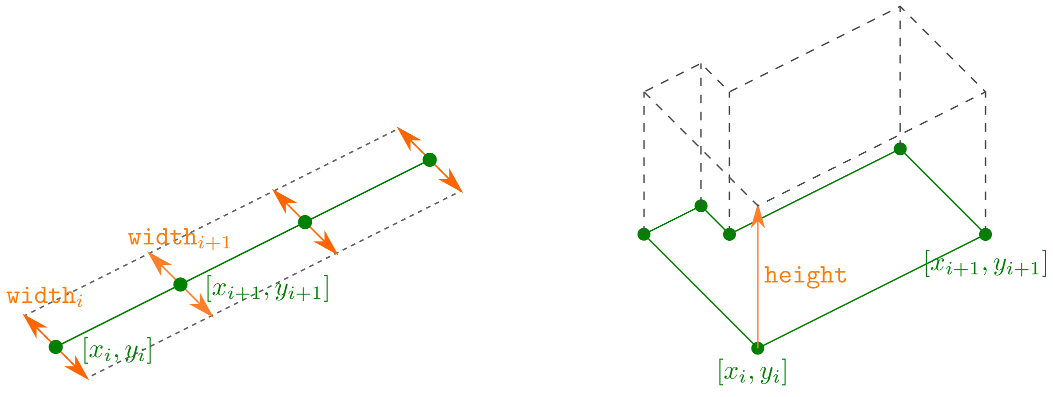


Figure 6.1.29.: Contour lines defining the cabin geometry (from [WH+22a])



(a) Aisle definition

(b) Spaces definition

Figure 6.1.30.: Non-physical geometry definitions in CPACS (from [WH+22a])

Consequently, the greatest value of non-physical geometry is during the design phase. Figure 6.1.31 shows a plot of the LOPA, a standard output of FUGA during the early stages of the design and modeling process, which can be generated very quickly. The plot combines simplified representations of the physical cabin components and structure with the non-physical geometry. The bottom contour of the cabin geometry data is used to provide a cabin boundary polygon. Doors are also marked by red lines on the outer polygon representing the OML. By including representations of the free spaces and aisles in these plots, violations of the aforementioned certification requirements, e.g. due to inconsistent input data or a faulty rule implementation can be detected and corrected early, without having to run the full high-fidelity model generation process.

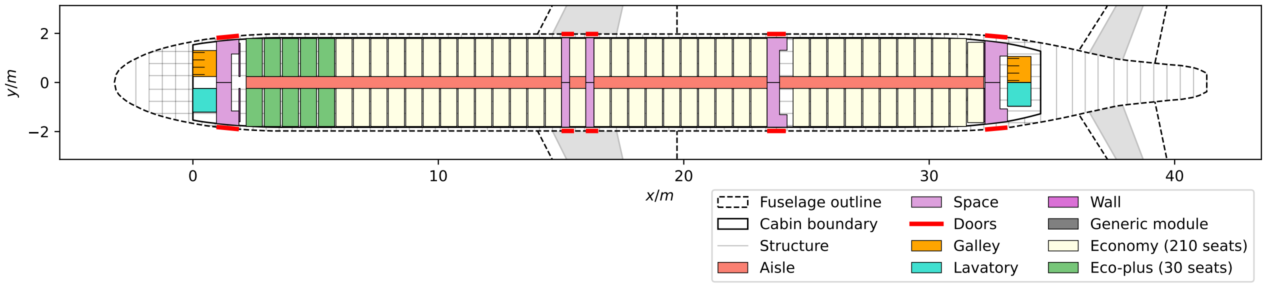


Figure 6.1.31.: LOPA of the example configuration

6.1.4. Management of geometry fidelity levels

As illustrated by the different examples presented in section 4.2 different types of analyses require vastly different levels of fidelity of the geometry model. The levels of fidelity are on the one hand characterized by different levels of detail of the geometric representation of a given component, such as beam-, shell- or volume-based representations of profile-based structures, or the mass of geometry-based representation of cabin components. On the other hand, it may be possible to leave out certain components altogether in certain analyses. The model generation rules discussed in the previous sections provide a basis for handling either of these cases in an efficient manner and without compromising consistency of the models using the knowledge-based methodology implemented in FUGA.

Examples for two ways of building consistent multi-fidelity component representations have been found. On the one hand, for extruded structural components, the models are generated based on available information by adding data. As such, more and more detailed models are created when traveling along the model generation graph. On the other hand, detailed but non-parametric component models may be given. Here, a data reduction is performed instead, e.g. when determining the bounding box and centroid representations. As such, the level of detail is reduced when following the graph.

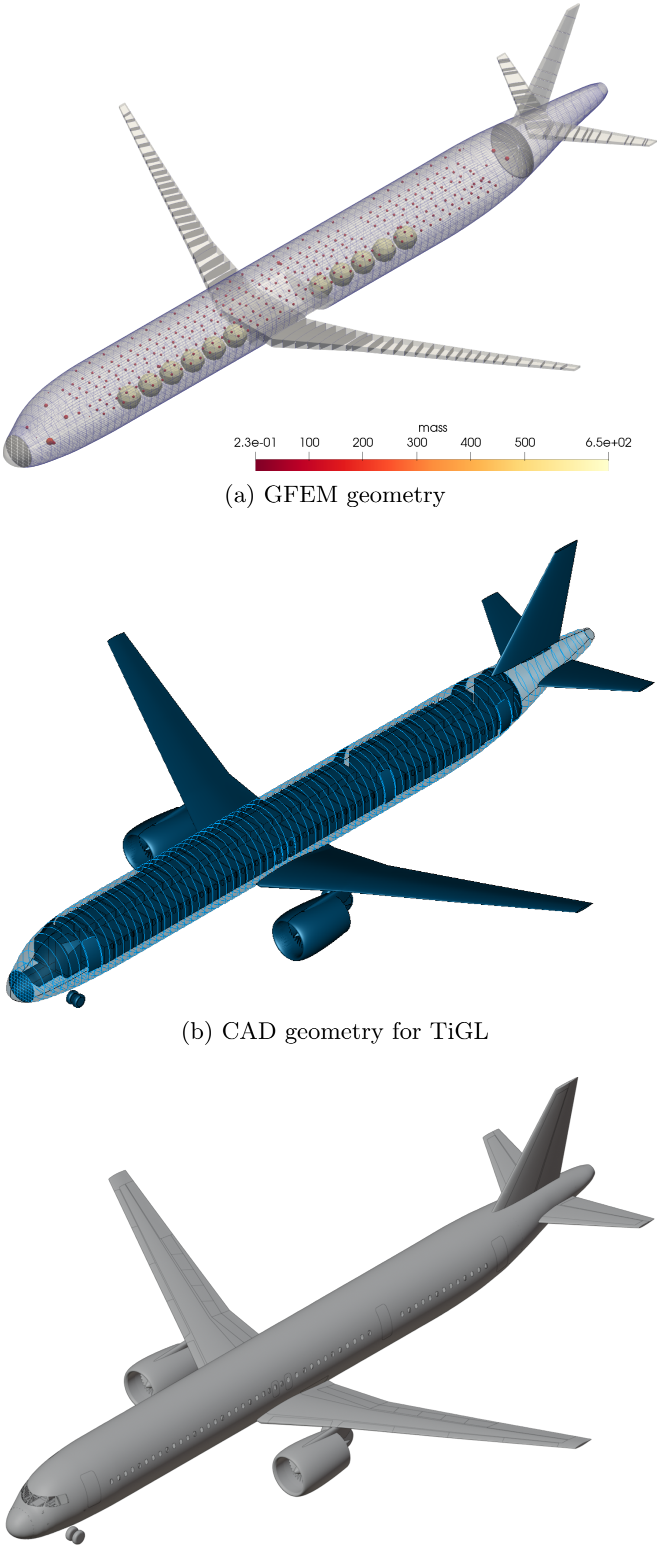
The declarative approach to formulation of the available modeling rules, which enables lazy evaluation, substantially contributes to the efficiency of the model generation, particularly in the data addition case. Any required model can be assembled by requesting and combining the available building blocks in the system, which are provided either by another rule or by the user in an interactive environment. Generation rules for component models, which have not been requested will be skipped, meaning that only those rules are evaluated that are necessary for the computation task at hand. In this way, expensive operations necessary to build highly detailed geometry are performed only if the geometry is explicitly requested.

Another common trait of the different rule sets is that very detailed models are usually built from simpler representations. For instance, detailed solid representations of the primary structure components are built by extruding a section profile along a trajectory curve, which is in turn defined by a plane. With the knowledge-based approach, all intermediate steps of the model generation process can be made available to the user or different rules. Consequently, component geometry models at different levels of detail are provided by the system automatically. Furthermore, due to the caching of results in the data repository, less detailed representations can be retrieved without any significant additional computational effort, once they have been visited during the computation of a more detailed representation.

Three examples for geometric representations of the same design are given in figure 6.1.32. Figure 6.1.32a displays a geometry model which provides the basis for deriving a GFEM. Frames and stringers are given as curves, as is the floor structure. Fuselage cutouts are not modeled explicitly. The wing is represented only by the wingbox and ribs, in correspondence with the approach used e.g. in PrADO or DeLIS, where the influence of the wing stringers is accounted for in the skin shell properties. The cabin is represented using mass points.

In figure 6.1.32b, a CAD model of the configuration for display using the TiGL Viewer is shown. It is intended as a way to make FUGA design results accessible e.g. to aircraft designers, who are familiar with TiGL. Visibly, some configuration details have been added w.r.t. the structural model, e.g. for the wing and engine, which have been introduced to match the established level of detail expected from TiGL. In addition, bounding box representations for the cabin are now included instead of mass points. On the other hand, the structure has been simplified by omitting the stringer curves, leaving only the frame curves for orientation. Door and window cutouts are now shown.

Finally, figure 6.1.32c shows the detailed mesh representation of the same design in Blender. The model as shown is ready to be exported e.g. to Unity for human factors evaluation. All previously discussed structural details are modeled, including frame and stringer cross-sections, floors, cutouts and wing details. A cockpit window cutout has been added to improve the realism of the model, which is, however defined outside CPACS. The cabin is represented using detailed 3D meshes, of both the seats and the secondary structure, which is now included in the model.



(a) GFEM geometry

(b) CAD geometry for TiGL

(c) Triangulated mesh for VR

Figure 6.1.32.: Multi-fidelity geometry outputs for the example configuration

All of the above models have been assembled using the same approach of making queries to the FUGA KBE system to receive the necessary component representations and assembling the results. They provide just a few examples of how the available component representations can be combined into a configuration representation. In addition, the 2D LOPA representation in figure 6.1.31 is simply another representation, derived from the multi-fidelity system in much the same way as the 3D models shown above. In this case, however, instead of OCCT, the matplotlib plotting library [Hun07] is used as geometry library.

The examples show that, using the KBE methodology described in chapter 5 and a suitable set of rules as shown in the preceding sections, geometry models can be provided, which are tailored to the fidelity requirements of specific disciplinary analyses. They therefore partially validate **working hypothesis 2**. However, it is not shown yet that executable analysis models can be built from the geometry data to evaluate the designs. To this end, two example analyses are provided in section 7.3.

6.2. Knowledge-based fuselage design rules

The multi-fidelity modeling capabilities described in section 6.1 can provide the basis for the generation of analysis models for a variety of disciplines. However, as the level of fidelity is increased, the parametric modeling engine relies on ever more detailed information being available in CPACS. This is not necessarily the case in a real life collaborative product design process. Thus, if further product information is requested by the modeling rules, but not available in the system, additional design rules must be deployed in order to enable augmentation of details as stipulated by **working hypothesis 3**.

Dorbath [Dor14] establishes an understanding of a design process as essentially being a mapping of a small number of user friendly input parameters to a substantially larger number of tool specific, or, more generally put, more detailed parameters using a "knowledge base". Consequently, the design capabilities introduced here are based not only on the product information already available in the data repository, which is a requirement for consistent designs, but also a manageable number of user-friendly top level design parameters, which must be provided as additional inputs.

The design rules, which form the knowledge base according to Dorbath, have been collected in the `fuga.design` subpackage and will be discussed in the following. Analogous to section 6.1, discussions of the design rules are provided for the outer fuselage shape in section 6.2.1, the primary structure in section 6.2.2, and the cabin in section 6.2.3. In principle, each rule set in the design subpackage is designed to provide the source nodes to its corresponding generation rule set from the geometry subpackage as sinks. However, the assignment of a given rule to one of the rule sets may be ambiguous due to interdisciplinary dependencies, which are discussed in section 6.2.4.

6.2.1. Fuselage shape design rules/interfaces to OAD

The design of the outer fuselage shape is closely related to activities from OAD. Aside from cabin design considerations, it is influenced by aerodynamics (c_w), operations (e.g. rotation during takeoff), and many more disciplines, which exceed the scope of FUGA at this point. Instead, the goal for the fuselage design rule set is to either provide the basic geometry if none is available, or to make informed changes to existing designs based on more detailed product information. To this end, two different approaches to fuselage design are introduced in the following, which, though not mutually exclusive, are deployed in separate rule sets.

On the one hand, the design of a fuselage shape from scratch based on cross-sections and guide curves, similar to the approach taken by Jonge [Jon17] presented in section 2.1.2, is discussed in section 6.2.1.1. The goal is to generate a CPACS section description as introduced in section 6.1.1 based on a reduced set of parameters. This capability is required to enable true inside-out fuselage design.

On the other hand, in section 6.2.1.2, the options for extending the constant mid-section are discussed. This capability is usually required, if FUGA is applied in tandem with an OAD synthesizer such as openAD. In this case, an initial design is provided by the synthesizer and must only be extended and adapted to suit the requirements of the structure and cabin integration. Not a true outside-in design approach, this procedure can be referred to as a hybrid design.

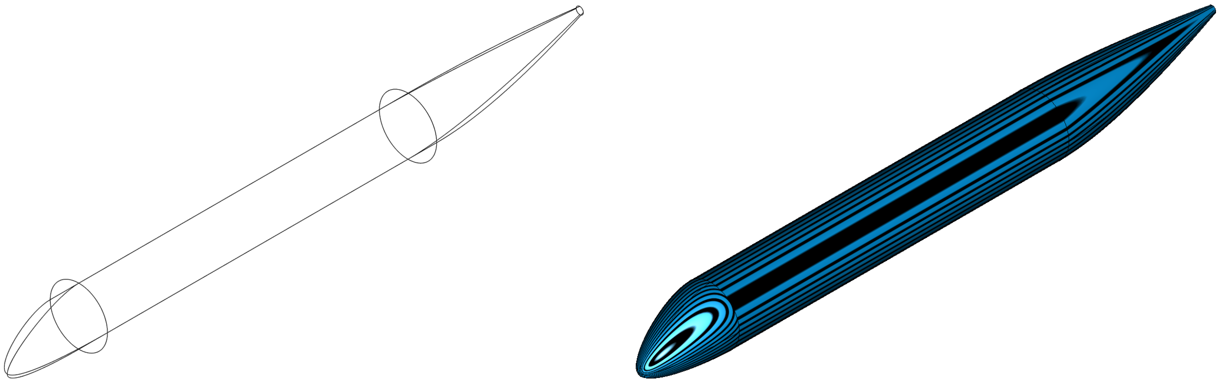
6.2.1.1. Scratch design from cross-section and guide curves

Several aspects must be taken into account when designing the fuselage OML from scratch. As discussed in section 2.1.2, the problem of the payload storage essentially introduces a tradeoff between the design of the main cross-section in the mid-segment and the length of the fuselage. As discussed, the two parameters are coupled via the number of seats abreast. Similarly, the aerodynamic drag contribution can be approximated via the slenderness ratio, which again depends on the cross-section and the length.

Apart from these basic parameters, additional details must eventually be considered, such as the shapes of the cockpit and tail sections, which affect e.g. aerodynamic performance and rotation capability during takeoff, but also the available space inside the fuselage. Furthermore, new designs often deviate from a circular cross-section, which introduces more degrees of freedom to the design.

An example for an implementation of such a detailed inside-out design process can be found in ParaFuse [Jon17]. The process begins with the design of the cross-section. To this end, a low number of reference points for the cabin cross-section layout is computed based on a given or computed number of seats abreast, around which the cross-section profile selected by the user is fitted. The true length of the fuselage is then determined by performing a cabin configuration. Based on the length and a predefined cockpit and tail section shape, four guide curves are determined as illustrated by figure 2.1.2. The fuselage surface is then generated by computing copies of the main cross-section curve at various positions of the fuselage, which are scaled to fit inside the guide curves. The actual fuselage surface is generated from these curves using a skinned surface interpolation algorithm.

A comparable approach to provide a very simple outer surface for conventional fuselages has been implemented using FUGA. Based on the aforementioned segmentation of the fuselage into cockpit, constant and tail segments, the surface is described by a network of the four longitudinal guide curves, also used in ParaFuse, and four section definition curves describing the cross section at the boundaries of the different fuselage segments, as illustrated by figure 6.2.1a. All curves are given as B-spline curves. A smooth surface, as shown in figure 6.2.1b, can be generated from the curve network using the Gordon surface algorithm. Visualization of the isophotes, sometimes also referred to as "zebra stripe" shading, enables the assessment of the surface quality [Poe84]. Aligning isophotes at the transitions between two surface patches indicate tangential continuity across the patch boundary. If the curvature is furthermore continuous, this indicates curvature continuity. Examination of figure 6.2.1b thus reveals that curvature continuity can be achieved in circumferential direction, whereas in longitudinal direction, tangential continuity is reached, even at the transitions between sections.

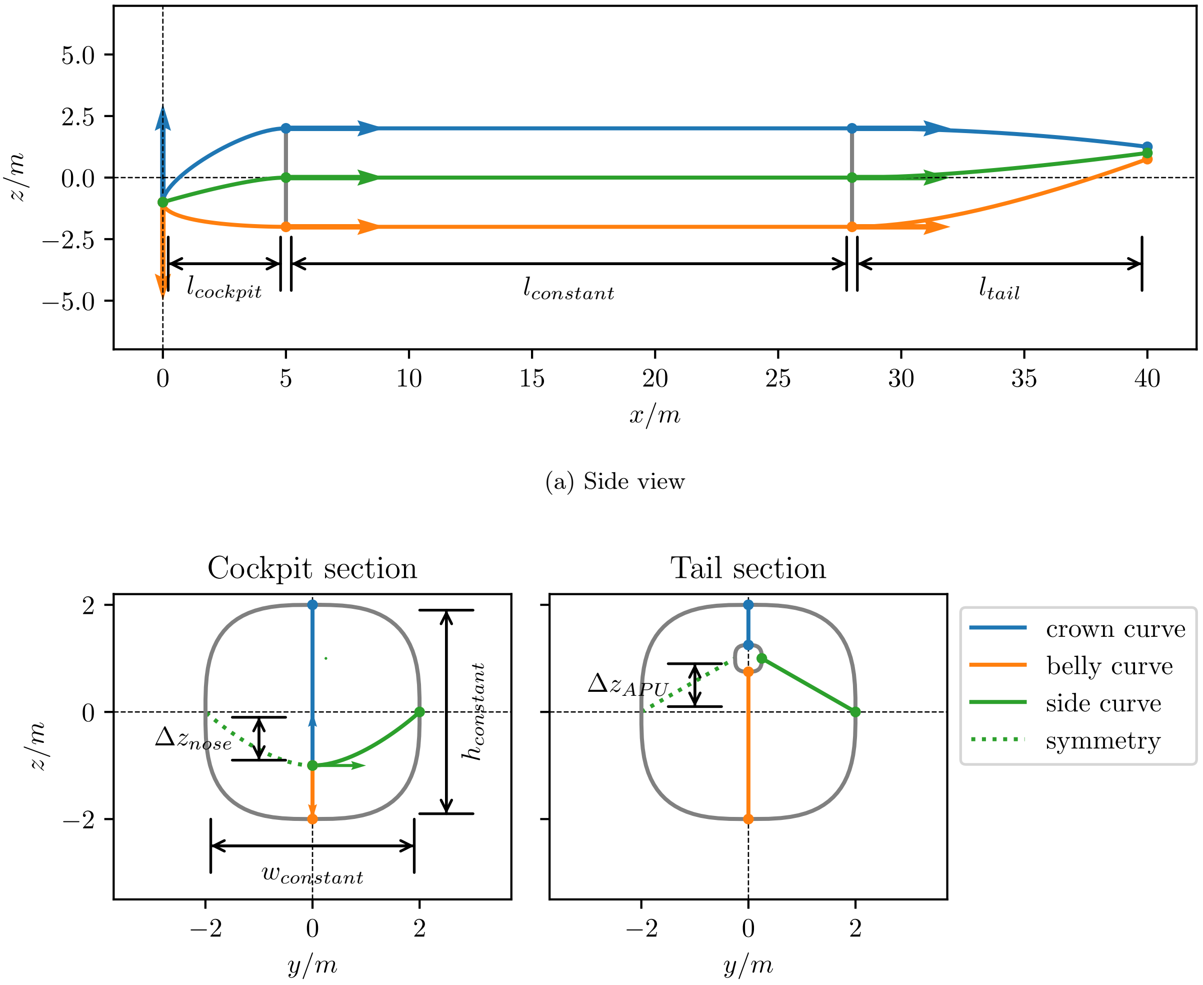


(a) Constituting curve network

(b) Surface with isophotes shown in TiGL Viewer

Figure 6.2.1.: Fuselage surface construction from curve network

The modeling approach provides several options for manipulating the fuselage surface by modifying the members of the constituting curve network. The underlying definition of the curves is illustrated in figure 6.2.2. Visibly, the longitudinal guide curves and the section curves are mutually dependent via a set of common points, which are marked in the figure. The placement of these points is related to a multitude of fuselage parameters. To begin with, the lengths of the different segments $l_{cockpit}$, $l_{constant}$, and l_{tail} must be provided, which, in summation, yield the overall length of the fuselage $l_{fuselage}$. Similarly, the width and height of the constant section $w_{constant}$ and $h_{constant}$ are prescribed. Similarly to the ParaFuse implementation, these parameters can later be coupled to the cabin parameters via FUGA.



(a) Side view

(b) Front view

Figure 6.2.2.: Fuselage surface definition curves

Additional control over the cross-section is available via the choice of the section curve. While the curve must pass through the shared points, its path can be chosen freely in between. To highlight this, a superellipse cross section is shown in figure 6.2.2b, whereas a circular cross section is used in figure 6.2.1a. A limitation, due to the implementation of the GORDON-surface algorithm, is that the curve parameter should not deviate too much from the relative curve length, which precludes the application e.g. of circles described by NURBS curves.

As shown in figure 6.2.2b, the shape of the cockpit and tail sections can furthermore be modified by prescribing an eccentricity Δz_{nose} and Δz_{APU}, respectively. These parameters influence the position of the respective end points w.r.t. the middle axis of the constant section. For the cockpit section, it is furthermore assumed that the curves coalesce in a singular umbilical point. As highlighted in figure 6.2.2, the tangency at the nose is prescribed by a vector for each curve. The length of the vector can be changed to modify the radius of the radome. Moreover, to ensure a smooth transition to the constant section, another set of tangent vectors, all pointing in x-direction, is given at the end of the cockpit section, as shown in figure 6.2.2a.

Analogously, the same tangent vectors are also prescribed at the beginning of the tail section. The tail end, where the auxiliary power unit (APU) is commonly placed in conventional designs, is, however, not represented by an umbilical point, but by a separate section profile. The section is characterized by its own width and height w_{APU} and h_{APU}. Furthermore, unlike for the cockpit section, no tangents are given at the APU section, resulting in a "free" boundary condition. Due to the different types of boundary conditions for the guide curves in the different segments, it is advisable to construct the curves separately for each segment and subsequently join the results.

To avoid loss of accuracy, the resulting surface can be passed directly to FUGA, to be used as the basis for subsequent modeling steps. However, to enable the exchange of the fuselage shape, it must be converted to a CPACS point cloud. Due to the modeling approach based on flat section curves, all circumferential isoparametric curves lie on an offset plane of the yz-plane. Consequently, the profile points needed for the description of the surface in CPACS can be computed simply by evaluating the surface on a grid of longitudinal and circumferential coordinate samples.

In summary, the modeling approach provides a very simple way to build fuselage surfaces of good geometric quality based on a very low number of parameters, which is adequate for preliminary design studies. That said, the examples shown are still substantially different from surfaces of real aircraft. One obvious aspect is the shape of the cockpit segment, which is missing the common kink due to the cockpit window, which is often found in passenger aircraft. This would be possible to implement at the cost of an increased number of input parameters. Different design parameterizations, deployed using different sub-rulesets and made accessible via a categorical parameter, similarly to the solution in ParaFuse, could help provide more versatile modeling capabilities without sacrificing too much accessibility. That said, newer designs for smaller aircraft, e.g. by Embraer or Airbus Canada, illustrate a tendency towards a less pronounced kink and are thus better represented in comparison. Furthermore, the benefit in terms of design accuracy w.r.t. the cabin and structure is likely limited, as long as the cockpit and the neighboring systems are not taken into account in detail in the design process.

Another critical aspect is the design of the cross-section, which is typically a compound of multiple segments in modern aircraft. While such sections can, in principle, be approximated using B-splines, they are more difficult to describe parametrically on the one hand, and the resulting behavior of the Gordon surface generation algorithm may be harder to predict and control on the other.

6.2.1.2. Design modification by extension of the constant mid-section

As mentioned previously, the creation of a fuselage shape from scratch may not always be necessary or desirable. For instance, the entire idea of family concepts, which are commonplace in industry today, is based on multiple configuration sharing the same fuselage cross-section and simply extending or shortening the mid-section, where the cross-section is constant. This has some very practical advantages in terms of manufacturing and reduces the need for redesigns.

In a collaborative environment, it may be necessary, to build a design based on input from another source, where the assumptions are not clear. In this case it is advisable to keep disruptive changes to a minimum, in order to facilitate the iteration of results. An example is the coupling of FUGA with the OAD synthesizer openAD. In openAD, an assumption is made w.r.t. the length of the cabin, as well es cross-sectional details e.g. the number of aisles and seats abreast. The output of the synthesis is then passed to FUGA, where a more detailed design of the cabin is performed. The assumptions pertaining to the given cross-section are respected, and only a new cabin length is computed and reported back to openAD, which must be executed again to account for snowball effects.

This design approach is referred to as a *hybrid* design [Baa15], as it combines elements of outside-in and inside-out fuselage design. It also has the advantage of being very simple to implement using

CPACS. W.r.t. to the mathematical description of design problems, this approach furthermore has the advantage of reducing the design of the fuselage OML to a single variable, i.e. the change in fuselage length $\Delta l_{fuselage}$.

Figure 6.2.3 illustrates the hybrid fuselage extension approach implemented in FUGA. The extension can simply be implemented by adding a translation in longitudinal direction Δx to each section positioning node. The total desired change in fuselage length is given by $\Delta l_{fuselage}$. As shown in the figure, the fuselage can then be divided into three lengthwise segments. In the first segment, no change in section position is made, i.e. $\Delta x_{sec} = 0$. In the mid-section, the section positions are offset by half the length change, i.e. $\Delta x_{sec} = \frac{\Delta l_{fuselage}}{2}$. The rear section is finally offset by the full length $\Delta x_{sec} = \Delta l_{fuselage}$. This procedure yields a relatively smooth distribution of the length change, given that the segments are chosen well. In figure 6.2.3, the mid-section is determined based on the change in cross-section area. In this way, the fuselage can be split into a cockpit section, mid-section and tail section, as established in section 2.1.2. Another feasible approach, which has been described by Walther et al. [WH+22b], is to assume an even split of the fuselage, where the length of each segment is $l_{seg} = \frac{l_{fuselage}}{3}$. Whereas this method is more basic, less input data is required for the segmentation since the section areas need not be computed before the length update, resulting in a less complex system.

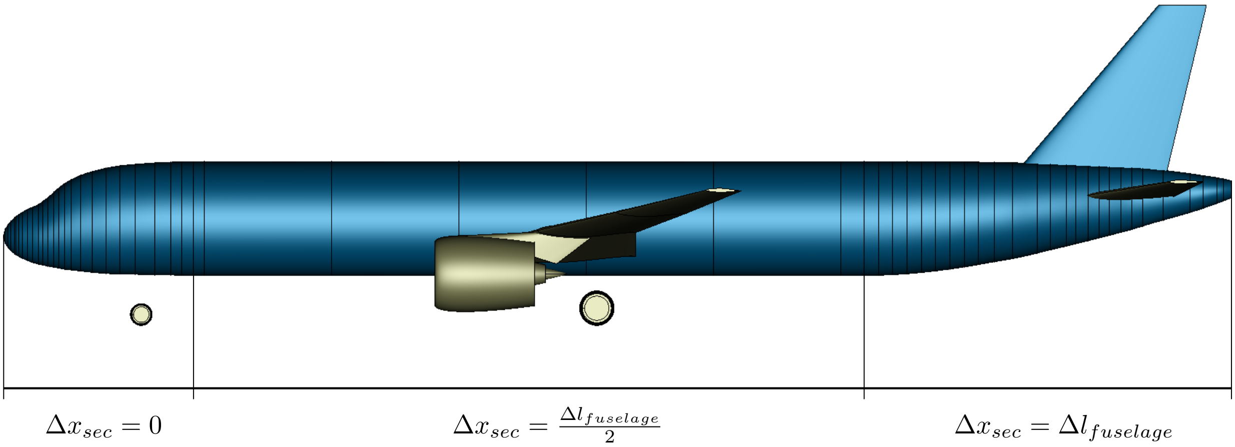

$\Delta x_{sec} = 0$
$\Delta x_{sec} = \frac{\Delta l_{fuselage}}{2}$
$\Delta x_{sec} = \Delta l_{fuselage}$

Figure 6.2.3.: Hybrid fuselage extension

A continuous deformation approach, e.g. $\Delta x_{sec}(x) = \frac{x}{l_{fuselage}} \Delta l_{fuselage}$ was not adopted, because the deformation distribution is also meant to be applied on any other object besides the fuselage, i.e. wings, other fuselages, engines and pylons. To keep sub-configurations, such as the wing including the pylon and engine nacelle, intact, it is necessary for the same displacement to applied to all of their respective member components. Taking the configuration in the figure as an example, a continuous approach would result in a lower displacement for the engine nacelle than for its pylon, because the longitudinal position of the former is lower than that of the latter, thus breaking the relative positioning.

The change in position of the wing and empennage implemented here is largely due to aesthetics. In truth, a change in length of the fuselage, will invalidate the overall design. To achieve a consistent design again, the wing position must be adjusted taking into account the overall center of gravity in several different loading conditions. Similarly, the design of the empennage is affected. In the event of a fuselage length increase, it will likely become smaller due to the increased lever arm [Tor76]. These types of considerations are performed by an OAD synthesizer. For a consistent overall design, an iteration loop should therefore be set up, where the correct cabin length is communicated back to the OAD synthesizer, and the fuselage design process is repeated for the new configuration. For the purposes of this thesis, however, the simplified update is considered sufficient.

6.2.2. Structure design rules

As discussed in section 6.1.2, Scherer and Kohlgrüber [SK16] and Dorbath, Nagel, and Gollnick [DNG13] provide detailed CPACS descriptions of the structural layout of fuselages and wings respectively. In both cases, the geometric interpretation is, however, demonstrated by FEM model generators by the authors, whereas in this work a feature-tree-based CAD-driven approach is proposed. Moreover, a common trait of the FEM model generators is the need for the structural information to be readily available in CPACS. The design capacity of both the TRAFUMO/PANDORA tool by Scherer and Kohlgrüber and the ELWIS tool by Dorbath, Nagel, and Gollnick is limited to a sizing of an existing layout in order to estimate the structural mass. The sizing encompasses the determination of skin thicknesses and potentially some profile cross section parameters using the FSD routines implemented in S_BOT+ in ANSYS [Dor14]. The topology of the structure, e.g. the number or position of components, is, however, largely fixed.

That said, designs in a collaborative process are commonly the output of OAD synthesizers that are not capable of providing such structural topology details. It is therefore necessary to instantiate a structural layout in order to enable the application of any of the above tools. This process step is not addressed by either of the authors, who instead expect the user to provide the necessary input parameters directly in CPACS [Dor14]. In practice, it has, however, become clear that these definitions are too complex to fill manually, especially in coordination with concurrent design activities e.g. for the cabin.

Consequently, Fuchte, Gollnick, and Nagel [FGN13] describe an initial attempt at creating a comprehensive design tool that can initialize structural details of the fuselage along with a cabin layout. Yet, while many promising capabilities are demonstrated, the tool is not sufficiently developed to support automatic generation of executable FEM analysis models. In contrast, Walther and Ciampa [WC18] present a tool for fuselage structure initialization dedicated to providing all necessary information to generate executable FEM models, which is, however, lacking cabin integration capabilities.

In the following, the design rules for the structural layout implemented in FUGA are discussed, which are a further development of the work of Walther and Ciampa. Due to the elevated importance of the primary structure grid in the CPACS fuselage structure definition, the rules for the frame and stringer distribution are presented separately in figures 6.2.13 and 6.2.2.3. In section 6.2.2.4, the placement of the floors based on the primary structure grid is then discussed, followed by the determination of bulkhead and skin properties in section 6.2.2.5.

6.2.2.1. Design inputs

Aside from a CPACS data set to provide the OML, the user needs to provide some additional inputs. On the one hand, an additional XML file must be provided, which contains the simplified *user friendly input parameters*. The XML tree may also be embedded in the original CPACS file, as an entry of the `toolspecific` node. Tool specific namespaces are commonly used in CPACS to provide and document input or control parameters, which are necessary for the application of a given tool, but not relevant to the final design description.

On the other hand, a *donor model* must be provided, which should contain the structural element definitions, i.e. profile and sheet based structural element descriptions as well as material data. The role of the donor model is to provide a library of these elements to be referenced for the new design.

The input files are provided to FUGA during the instantiation of the structure design system.

6.2.2.2. Frame distribution

The frame distribution provides the basis for a significant portion of the structural definitions. As such, parameters and assumptions about many different components must be taken into account during its design on the one hand, which introduces substantial complexity. On the other hand, the frame planes are commonly aligned normal to the longitudinal axis, which facilitates the design.

In FUGA, the detailed frame distribution is designed based on two main inputs. On the one hand, main frame positions $\mathbf{x}_{mfr}$ must be provided where a main frame must be present e.g. for load introduction. On the other hand, a nominal frame pitch $\Delta x_{fr,nom}$ is provided to interpolate frame positions between the main frames. Whereas the latter is an input parameter provided by the design engineer via the input parameter list given in table 6.2.1, information on the former must be assembled from the available data. FUGA places mainframes at

- Fuselage bounds, which are determined from the OML bounding box,
- Bulkhead positions, which are given in the input parameter list or by the VTP front spar (s. also section 6.2.2.5),
- Door bounds, which are provided by the cabin layout (s. section 6.2.3),
- Wing box bounds, which are determined from the wing spar positions usually provided by openAD,
- Landing gear bay, which is offset from the rear spar position by a length parameter in the input list.

Since the frame plane normals are aligned with the x-axis, position of the i-th mainframe can simply be represented by its longitudinal position $x_{mfr,i}$.

The next step is to compute the positions of the regular frames between the main frames. To this end, the required number of frames is computed based on the distance between main frames and the nominal frame pitch using

$$n_{sections,i} = \left\lfloor \frac{x_{mfr,i+1} - x_{mfr,i}}{\Delta x_{fr,nom}} \right\rfloor + 1. \tag{6.2.1}$$

From the number of frames, the position $x_{f,ij}$ of the j-th frame between the i-th pair of main frames is given by

$$x_{f,ij} = x_{mfr,i} + j \cdot \frac{x_{mfr,i+1} - x_{mfr,i}}{n_{sections,i}}, \{j \in \mathbb{N}_0 | 0 \leq j < n_{sections,i}\}. \tag{6.2.2}$$

A distribution resulting from the approach is given by figure 6.2.4.

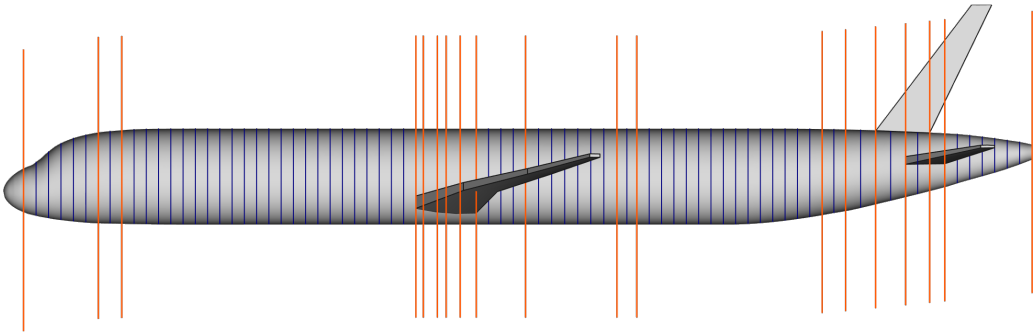


Figure 6.2.4.: Frame distribution generated from mainframe positions

Due to the way the number of sections $n_{sections,i}$ is computed, the nominal frame pitch acts as an upper boundary for the distance between two adjacent frames. It is, however, likely that the true frame distance is actually slightly lower than the pitch. Furthermore, the distance between frames will likely vary between different main frame pairs. That said, the main frame positions are observed exactly by the layout. Different interpolation schemes than the one outlined above are fathomable, such as a forward biased or centered pitch observing scheme as follows:

$$x_{f,ij} = x_{mfr,i} + j \cdot \Delta x_{fr,nom}, \{j \in \mathbb{N}_0 | 0 \leq j < n_{sections,i}\}, \tag{6.2.3}$$

$$x_{f,ij} = x_{mfr,i} + \frac{(x_{mfr,i+1} - x_{mfr,i}) - n_{sections,i} \cdot \Delta x_{fr,nom}}{2} + j \cdot \Delta x_{fr,nom}, \{j \in \mathbb{N}_0 | 0 \leq j < n_{sections,i}\}. \tag{6.2.4}$$

In these cases, the pitch given by $\Delta x_{fr,nom}$ would be observed by all frame pairs, except for some cases in the neighborhood of main frames.

Aside from the frame plane definition, the definition of the profile properties is a key aspect of the frame distribution design. Here, the library of profile based structural elements provided by the donor model is leveraged. Separate cross sections can be assigned to mainframes and regular frames by referring to the uIDs of the desired profile based structural element in the donor data set. The corresponding data is then copied over to the design data set and the uIDs are assigned to the newly created frame instances.

6.2.2.3. Stringer distribution

In some ways, the procedure to compute the stringer distribution is similar to the frame distribution. However, due to the increased geometric complexity of the stringer, several complications must be addressed. To begin with, one stringer plane is usually not sufficient to properly represent the stringer base curve along the entire length of the fuselage. Therefore, the stringer plane distribution is usually split longitudinally e.g. as illustrated by figure 6.2.5a. In the figure, two separate distributions are shown for the forward and the aft part of the fuselage. The confluence points of the planes, which are extruded along the x-axis, lie in the nose and the center of the tail, resulting in a vertical offset. At the same time, both distributions must be compatible, i.e. intersect the fuselage at the same point at the transition position. Furthermore, the placement of seed values around the circumference requires significantly more effort than the 1D-distribution of the frames in longitudinal direction, requiring mapping between curve and Cartesian coordinates.

Computation of reference points based on angles Walther and Ciampa [WC18] present an approach to describe such a distribution based on the number of circumferential stringers $n_{stringer}$, which must be provided in the input file. The orientation of the stringer planes is computed based on the assumption that the stringer curves are arranged in equal angular distance around the centroid $\mathbf{p}_{st,ref}$ of a reference section, which lies in the constant mid-section of the fuselage. The stringer plane angles φ in Radian are thus given by

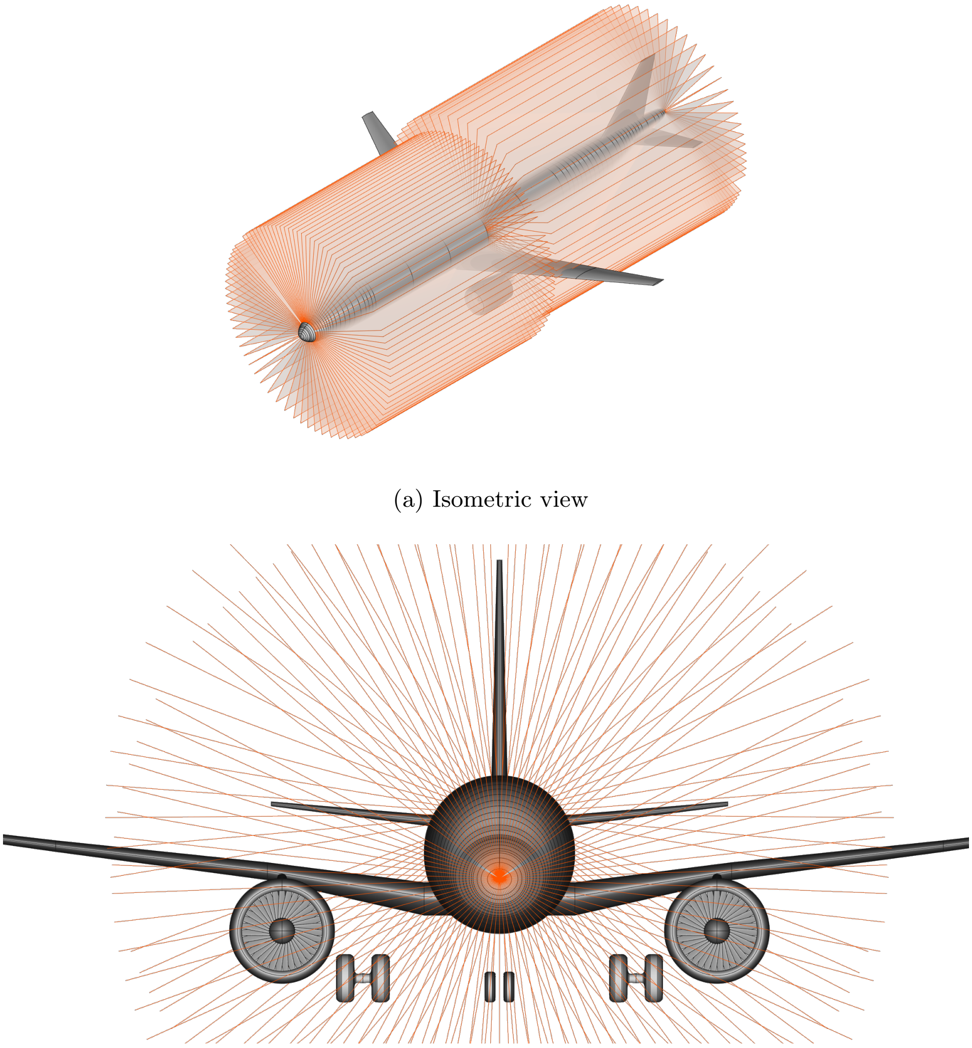
$$\varphi_i = 2\pi \cdot \frac{i}{n_{stringer}}, \{i \in \mathbb{N}_0 | 0 \leq i < n_{stringer}\}. \tag{6.2.5}$$

The i-th stringer curve point $\mathbf{p}_{st,i}$ at the reference section is then given by the intersection of the fuselage surface and the vector defined by $\mathbf{p}_{st,ref}$ and the i-th angle φ_i. In this way, the locations of the stringer curves in the constant section are determined.

If window cutouts are present, it is possible to remove certain stringer curve points that fall in the height range of the window cutouts. This is in line with common design practice [Niu88], where a substantial increase in thickness of the skin around the windows is usually favored over interrupted stringers.

Determination of stringer planes In order to construct the stringer planes, it must furthermore be taken into account that the stringers must meet at confluence points. The points are chosen based on the assumption that the stringers will meet in the first and last section of the fuselage. This results in two segments of the stringer distribution, one from the fuselage nose section to the reference section and a second one from the reference section to the tail section. The assumption of a singular confluence *point* $\mathbf{p}_{con,j}$, i.e. a single point in which all stinger planes meet, holds, if the end sections are approximately circular. This is the case e.g. for aircraft nose cones. As shown in figure 6.2.5b, all stringer planes then have to originate in $\mathbf{p}_{con,j}$ and pass through the curve reference points to maintain consistency. Consequently the actual stringer plane angle φ_{ij} for the i-th stringer and the j-th confluence point is given by

$$\varphi_{ij} = -\arctan2\left(p_{st,i,y} - p_{con,j,y}, p_{st,i,z} - p_{con,j,z}\right). \tag{6.2.6}$$



(a) Isometric view

(b) Front view

Figure 6.2.5.: Stringer planes for piecewise constant interpolation scheme

If the height at the confluence section is significantly different from the width, as is the case e.g. for the tail design of the Boeing 777, the assumption of a singular confluence point no longer holds, as it will lead to uneven spacing of the stringer curves, with a cluster of curves along the longer sides. Therefore, the stringers are instead assumed to end in a confluence line, on which all the confluence points for the individual stringers $\mathbf{p}_{con,ij}$ lie. An ellipse analogy, which maps the angle φ_i to an ellipse representation of the confluence section, is used to determine both the extent of the line and the locations of the individual points on it. The ellipse axis lengths $2a_{ellipse}$ and $2b_{ellipse}$ are given by the section height and width. Therefore, the ellipse is upright if the section height is larger than its width, otherwise it is horizontal. The confluence line is given by the line between the foci of the ellipse. In this way, cases ranging from a circle, i.e. $\frac{a_{ellipse}}{b_{ellipse}} = 1$, to a line, i.e. $\frac{a_{ellipse}}{b_{ellipse}} = \infty$ assuming $a_{ellipse}$ is the major semi axis, can be covered. The distance between the center and a focus is given by the linear eccentricity of the ellipse

$$c_{ellipse} = \sqrt{a_{ellipse}^2 - b_{ellipse}^2}. \tag{6.2.7}$$

The point on the confluence line corresponding to a given angle φ_i is then given by

$$\mathbf{p}_{con,ij} = [p_{con,j,y}, p_{con,j,z} + c_{ellipse} \cdot \cos(\varphi_i)] \qquad \text{for an upright ellipse,} \tag{6.2.8}$$

$$\mathbf{p}_{con,ij} = [p_{con,j,y} - c_{ellipse} \cdot \sin(\varphi_i), p_{con,j,z}] \qquad \text{for a horizontal ellipse.} \tag{6.2.9}$$

Temporary values of φ_i can be computed using equation 6.2.6, taking the profile centroid $\mathbf{p}_{con,j}$ as reference. Once the actual source points $\mathbf{p}_{con,ij}$ are known, the final angles φ_{ij} are computed using the same formula, but with the new point as reference.

The resulting stringer curves for a design using the above approach are shown in figure 6.2.6.

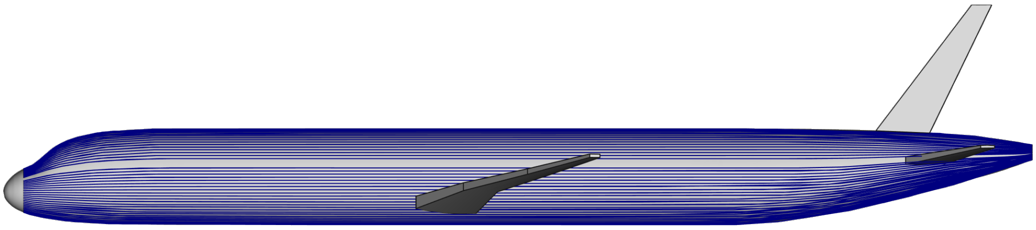


Figure 6.2.6.: Stinger curves from piecewise constant definition

Computation of reference points based on arc length The above approach does not work well, if the reference cross section deviates significantly from a circular shape, which again leads to uneven stringer spacing. Furthermore, it fails to take into account the parameters of the center fuselage area. As explained in section 6.1.2, many key properties of the center fuselage area are given in terms of frame and stringer positions. Therefore, if certain positions, e.g. the passenger floor height, are to be respected, it must be considered during the design of the stringer distribution. The same is true for the height and width of the keelbeam. Furthermore, it is a good idea to consider the floor heights, if the goal is to generate a GFEM of the design. In the GFEM, floor beams may be moved to the nearest frame/stringer intersection point to avoid the introduction of additional nodes and elements [SK+13]. The introduction of these constraints implies that, in the constant mid-section, some stringers are expected at certain vertical positions due to the floors and keelbeam height as well as lateral positions due to the keel beam width.

To incorporate these requirements, the assumption of an even angular distribution is abandoned in favor of a more detailed evaluation of the reference section curve. Since the reference section can freely be chosen within the constant center segment of the fuselage, any of the section curves that make up this segment can be considered. The section curve $\mathbf{C}_{ref}(v) = \left[\begin{array}{ccc} C_{ref,x}(v), & C_{ref,y}(v), & C_{ref,z}(v) \end{array} \right]^T$ is a B-spline curve, which is the outcome from an interpolation of the points of a CPACS section element definition (s. section 6.1.1). To fulfill the above requirements, those points on the curve, where the curve crosses the desired vertical or lateral position, must be determined. This means solving

$$C_{ref,y}(v) = y_{req} \tag{6.2.10}$$

or

$$C_{ref,z}(v) = z_{req} \tag{6.2.11}$$

for the local curve parameter v for all lateral constraints y_{req} and all vertical constraints z_{req}. The problem can be solved e.g. using a quasi-NEWTON gradient-based solution method, such as the secant method. To ensure symmetry, it is a good idea to compute the parameter for a given z_{req} on both sides of the fuselage by splitting the allowable solution range for v. The resulting parameter values $\mathbf{v}^*$ take on the role of seed values for the stringer distribution, much in the same way the mainframe positions do for the frame distribution.

To determine the distance between the seed points, the arc length of the curve segments between each pair of coordinates given by

$$s_{12} = \int_{v_1}^{v_2} \left\| \dot{\mathbf{C}}_{ref}(v) \right\|_2 dt \tag{6.2.12}$$

as shown e.g. by Farin [Far01]. The integral can be approximated using e.g. the trapezoidal rule, i.e. summing chord lengths between a sufficiently high number of sample points, or GAUSSIAN quadrature. For the purposes of the present application, the accuracy of the trapezoidal rule is usually adequate.

Since the total number of stringers is given by $n_{stringers}$, they must now be distributed across the curve segments proportionately to the segment arc lengths. To this end, a target stringer bay arc length is computed by dividing the length of the entire reference profile curve s_{ref} by the number of stringers:

$$\Delta s_{st,nom} = \frac{s_{ref}}{n_{stringers}}. \tag{6.2.13}$$

Using $\Delta s_{st,nom}$ and the known curve segment lengths, the number of stringers per segment can be estimated analogously to the frames using equation 6.2.1.

To compute the equidistant Cartesian curve points w.r.t. the arc length, given the section curve $\mathbf{C}_{ref}(v)$ with $v \in [v_{i,1}, v_{i,2}]$ and the number of stringers on the i-th curve segment $n_{stringers,i}$, a *reparametrized* representation of $\mathbf{C}_{ref}$ can be used. A reparametrization $\tau = \tau(v)$ allows for changing the parameters of points on the curve, without changing its shape [Far01]. In the present case, a reparametrization by the arc length given in equation 6.2.12 is required, i.e.

$$s(v) = \int_{v_1}^{v} \left\| \dot{\mathbf{C}}_{ref}(v) \right\|_2 dv. \tag{6.2.14}$$

The relationship can be inverted to provide the parameter of the original curve based on the new parametrization $t = t(s)$. The parameter s_{ij} for the i-th stringer in the j-th curve segment is then simply given by

$$s_{ij} = i \cdot \frac{s_j}{n_{stringers,j}}, \{j \in \mathbb{N}_0 | 0 \leq j < n_{sections,i}\}. \tag{6.2.15}$$

Based on the resulting curve points and the section centroid, the procedure for the determination of the stringer planes can be applied as above.

A drawback of this second approach is that the user-requested stringer number may no longer be matched exactly in all cases, due to rounding errors. On the other hand, it introduces the opportunity for the user to prescribe the target bay arc length directly, which may be a more meaningful parameter from a structural analysis perspective (s. e.g. Bruhn [Bru73], where many analysis methods based on the bay arc length/height are provided). The arc-length approach furthermore translates better to non-circular profiles.

Incorporation of cabin constraints for transport aircraft The above approach is sufficient to generate structural designs, which can be evaluated and sized using FEM software as shown by Walther and Ciampa [WC18]. However, as more details of the cabin are introduced, it becomes obvious that the approach still has some shortcomings. For instance, figure 6.2.7 shows the structural layout around a door cutout in the rear section, which has stringers bending upward and passing through the door cutout. This is undesirable for typical door surround structure designs as shown e.g. by Niu [Niu88] and Schmidt et al. [SK+15].

As a solution, a modified stringer plane arrangement can be adopted, where a third set of stringer planes is added in order to force stringers to run in parallel to the passenger floor in the area of the cabin. Such a stringer plane arrangement, assembled manually based on drawings of the original A320-200 configuration [Sch11], is found in the D150 configuration used e.g. by Scherer et al. [SK+13].

This approach can be automated in FUGA, taking into account the available cabin geometry data introduced in section 6.1.3, which provides information about the vertical position and extent of the cabin as well as key positions in the fuselage, e.g. of the cockpit rear wall, the rear bulkhead and the exits. Based on this information, the forward and aft boundary of the new stringer definition plane segment can be determined. Typically, an offset from the forward limit of the most forward exit pair and the rear bulkhead are good choices for the limits. In this way, it can be avoided that the middle segments extend too far in the non-constant sections of the fuselage, which could cause undesirable discontinuities in the stringer definition curves.

With the limits established, the stringer reference points are computed analogously to the previously described approach, based on a reference section in the constant part of the fuselage. Differently from

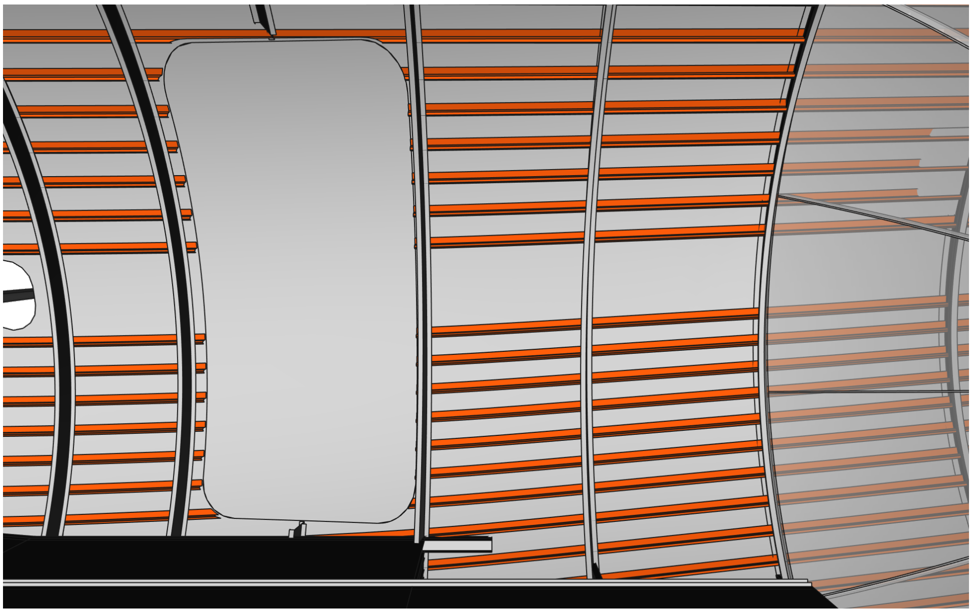
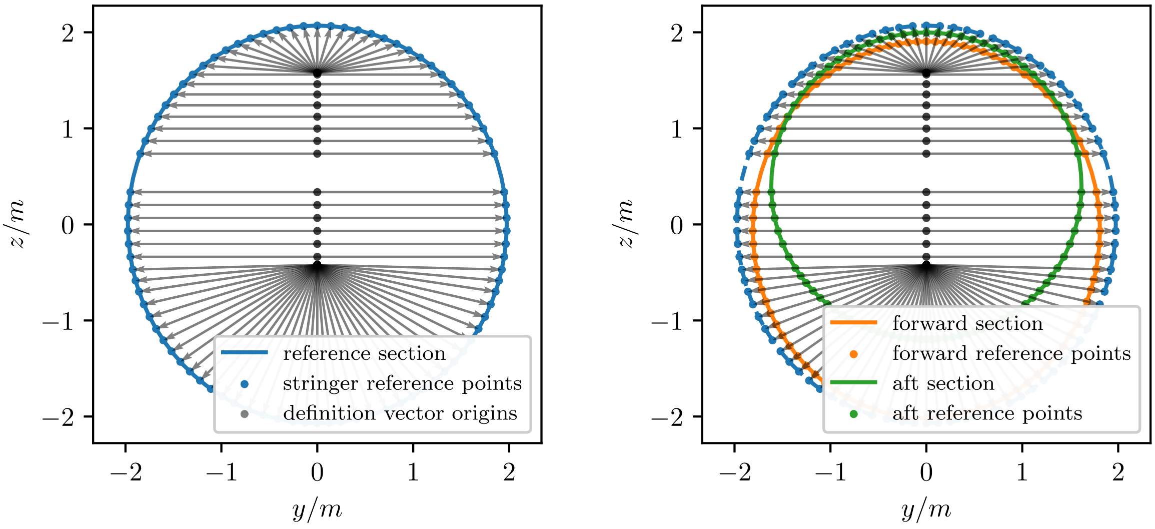


Figure 6.2.7.: Faulty stringer distribution surrounding the rear door (cf. figure 6.2.10)

the previous method, the definition vector distribution is then computed for the middle stringer planes first. An example distribution is shown in figure 6.2.8a. The choice of origins for the vectors merits particular attention. In the area of the cabin, which is located between the vertical position of the cabin floor and a height limit, based on the cabin boundary geometry, the z-position of the origins is aligned with the reference point on the circumference, resulting in a vector, and thus a plane, which runs in parallel to the floor. In this way, it can be achieved that the stringers always run in parallel to the passenger floor. Outside of the cabin region, the z-position of the vector origins is restricted to the minimum or maximum of the cabin region, resulting in a hub and spoke distribution similar to the previous examples.



(a) Baseline reference points and definition vectors

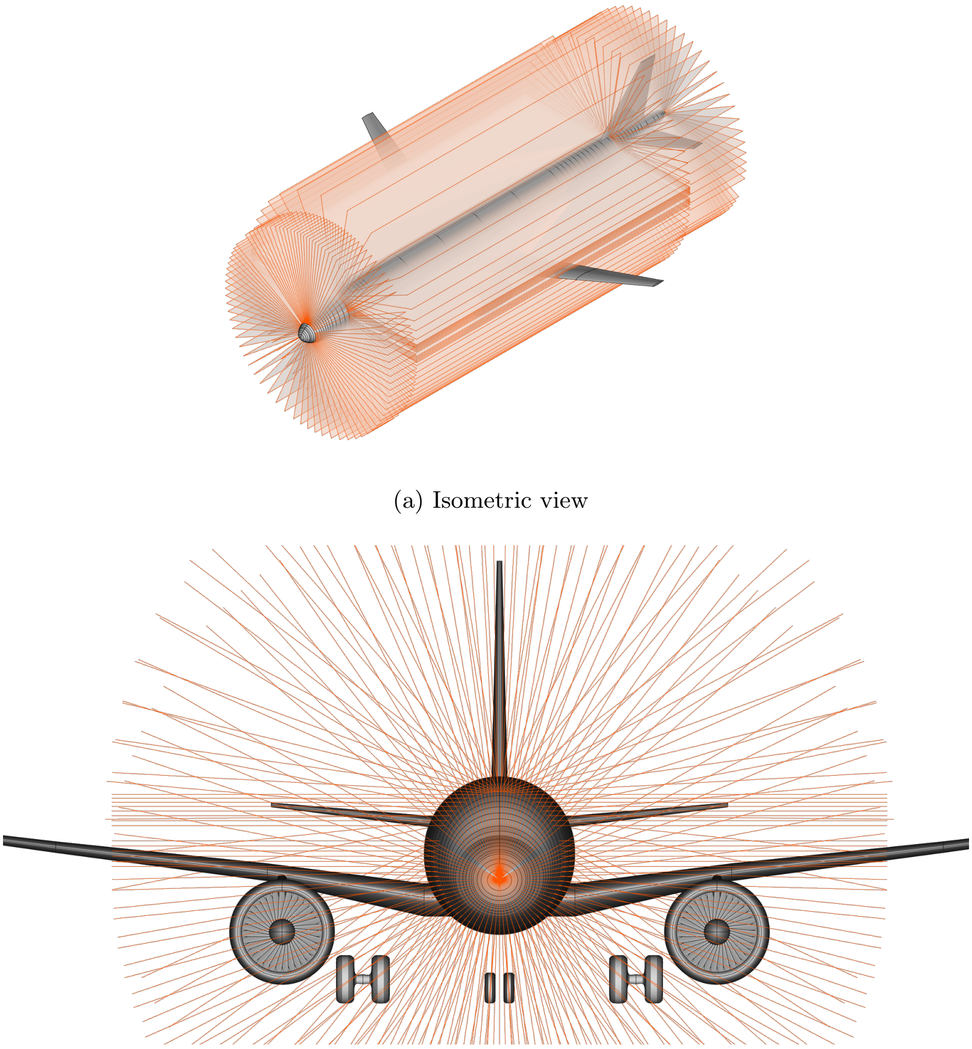
(b) Updated outer reference points

Figure 6.2.8.: Determination of the stringer definition planes in the middle segment and updated outer reference points

Next, the distributions of the forward and aft stringer definition plane sets must be determined. This can again be accomplished using the confluence point projection approach described by equation

6.2.6. However, instead of the reference points on the main reference section, the projected reference points on the fuselage surface at the forward and aft limit of the middle stringer definition plane set must be computed. The points can be computed by intersecting the fuselage surface with copies of the middle stringer definition vectors translated to the forward and aft limit of the definition space. The resulting points are given in figure 6.2.8b.

Applying the projection method to the new reference sections and combining the three sets of stringer segment definition planes yields the distribution in figure 6.2.9a. Compared to figure 6.2.5a, the third definition range is clearly visible. The front view in figure 6.2.9b reveals the horizontal definition planes in the cabin area.



(a) Isometric view

(b) Front view

Figure 6.2.9.: Stringer planes including cabin

An updated view of the door area from figure 6.2.7 is given in figure 6.2.10. It shows that the new design approach including the cabin data is clearly beneficial for a more structured stringer arrangement around the doors. The improvement in the door areas comes at the cost of noticeable kinks in the transition areas between different stringer curve segments.

As mentioned previously, the design method described here depends on inputs from the cabin design rule set. In some cases, such as strong deviations from a classical passenger transport aircraft configuration, this can be a disadvantage over the methods described earlier, which depend only on the outer geometry. As such, it is desirable to retain both methods in the design system in a way, where the advanced method is used if the cabin design and modeling rules are applicable, and the

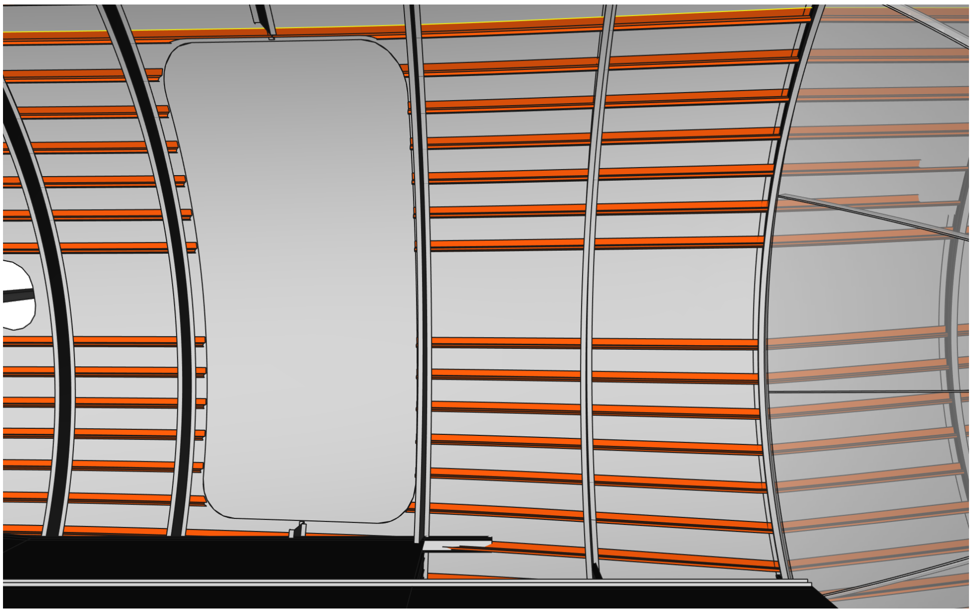


Figure 6.2.10.: Corrected stringer distribution surrounding the rear door (cf. figure 6.2.7)

basic method is used as a fallback if this is not the case. A simple way to accomplish this is to place the basic design rules in the structural design rule set and the advanced rules in the cabin design rule set. In this way, the rules in the former will be overloaded by the rules in the latter, if it is included in the overall design system.

6.2.2.4. Floor element placement

As described in section 6.1.2.2, the floors in CPACS are initially defined in terms of the crossbeams, which in turn reference the frame definitions. Consequently, a frame distribution must be available at the beginning of the floor design. In addition to this, only very few additional parameters must be provided in the user input deck.

To determine the vertical position of the floor, a z-position is given for the crossbeam. To begin with, a crossbeam distribution across the full length of the pressurized region of the fuselage, i.e. between the first and last bulkhead is assumed. Based on a side projection of the fuselage OML, it is then determined, whether a given crossbeam curve lies outside the fuselage, in which case it is eliminated. This occurs commonly e.g. for a lower cargo deck. A profile-based structural element uID from the donor model must be provided along with the z-position, to determine the cross-section properties of the crossbeam.

Along the length of the crossbeams, an arbitrary number of crossbeam struts can be generated by giving a y-position on the crossbeam and an orientation angle. Based on a line representation of the crossbeam, it is verified that the desired y-position of the crossbeam is within the fuselage bounds, in which case the strut is added to the design. Again, a profile-based structural element definition must be provided for each combination of y-position and angle. This process is repeated for all crossbeams associated with a given floor.

Two different ways are available in FUGA to determine the longitudinal floor beams. Walther and Ciampa [WC18] simply provide a list of y-positions and corresponding cross-section uIDs, based on which a line is generated. The line extends across all crossbeams as long as its y-position is within the fuselage bounds.

That said, the path of the longitudinal beams is closely related to the cabin layout. For instance, seats are usually mounted in seatrails, and support beams are usually placed on both sides of the aisles as well as on the outer bounds of the cabin. Therefore, designing the longitudinal floor beams without consideration of the cabin layout is bound to result in inconsistent designs.

Jonge [Jon17] implements a seat placement based on the seatrail paths, which are assumed to be offsets of the OML for the outer seat blocks and straight lines for the inner blocks. This leads to curved seatrails, which introduce difficulties in manufacturing and are harder to model in CPACS compared to their polyline counterparts. Therefore, a different approach is implemented in FUGA, where the seatrails are generated based on an existing seat distribution. The generation of such a distribution is discussed in section 6.2.3.4. The seat models can be augmented with metadata, allowing for definition of axes, where the longitudinal beams are expected to pass the seat, as shown in figure 6.2.11. The inner axes represent seatrails, whereas the outer axes denote regular support beams.

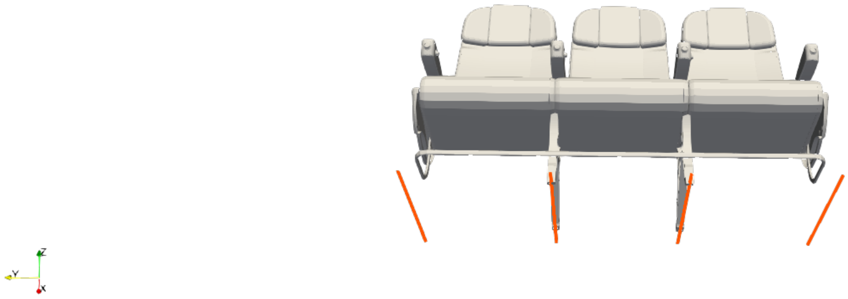


Figure 6.2.11.: Floor beam axes defined for a seat model

The axes for each seat are combined into a set of polylines for each seat block, which must then be resampled at the frame x-positions corresponding to the crossbeams to give the longitudinal beam positions required for CPACS. Outside the bounds of the seat blocks, the line can be extrapolated to determine the required lateral positions. The final result is shown in figure 6.2.12. Due to the limitation of CPACS that longitudinal beams samples can only be placed at crossbeams, the beam curves stored in the format cannot always match the axes provided by the model metadata exactly. Nonetheless, reasonable consistency between the structure and the cabin layout can be achieved.

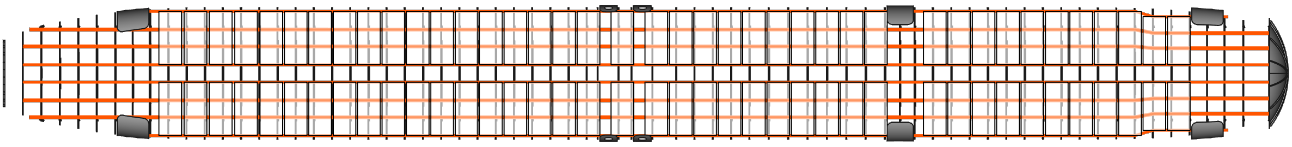


Figure 6.2.12.: Seat rails (orange) generated from seat layout

Floor panels are defined between adjacent longitudinal floor beams and bounded by longitudinal positions, as explained in section 6.1.2.2. In the structural design rule set, it is assumed that a floor panel is placed between each pair of neighboring beams created in the previous step. The computation of the x-positions, which are assumed to be the same for all pairs of floor beams, is based on the frame positions. To this end, the user can specify the number of frames $n_{fr,panel}$ spanned by each floor panel as an input parameter. Thus, assuming $n_{fr,span} = 3$, a break between panels would be introduced at every third frame position, counting from the first crossbeam, where a longitudinal beam is present. If the number of frames in the range is not evenly divisible by $n_{fr,span}$, the last panel will be shortened. The properties of the panels are given by a uID referencing a sheet-based structural element definition in the donor model.

A final important aspect of the floor design in FUGA is the introduction of offsets. As explained in section 6.1.2.2, offsets can be used to manipulate the cross-section profile of a curve-based structural component. In the preceding procedure, only a single z-position was provided, namely the vertical position of the crossbeam. However, it is easy to understand that the physical counterpart e.g. of the crossbeam and a longitudinal beam cannot be placed at the same position. Therefore, offsets must be determined to arrange the various components in such a way as to avoid collisions. According to the convention adopted in FUGA, the given z-position is interpreted as the upper bound of the crossbeam. Considering the crossbeam profile with the points $\mathbf{p}_{prof}$, the necessary downward offset in z-direction is given by the maximal profile z-coordinate $\Delta z_{CB} = -\max(\mathbf{p}_{prof,CB,z})$. Similarly, the z-position also represents the lower bound of any longitudinal beams and the offset is given by the minimal profile z-coordinate $\Delta z_{LB} = \min(\mathbf{p}_{prof,LB,z})$. An offset along the longitudinal axis may be applied e.g. to avoid overlaps between the crossbeam and the frames or between the struts and the crossbeams. For the floor panels, the vertical offset is assumed to be the height of the lowest longitudinal beam profile.

The input deck allows the specification of an arbitrary number of floors. In order to specify, whether e.g. crossbeams be exported to the `paxCrossbeams` or `cargoCrossBeams` node of the fuselage in CPACS, an additional attribute `is_cargo_floor` is provided.

6.2.2.5. Bulkheads and skin segments

As mentioned in section 6.2.2.2, bulkhead positions are essential inputs for the frame distribution when creating the structural layout for a modern transport aircraft. The approach described by Walther and Ciampa [WC18] implemented in FUGA allows the user to specify the longitudinal positions of the outer bulkheads, either in absolute or relative fuselage coordinates, or in terms of an offset from the longitudinal fuselage bounds. Furthermore, the number of bulkheads can be specified, to introduce additional bulkheads in an evenly spaced distribution between the outer bulkheads. However, this feature is not needed for conventional transport aircraft. If information on the VTP spar positions is available, an additional rule can be introduced in FUGA, which places the RPB at the VTP front spar, as is common in passenger aircraft designs.

In addition, the bulkhead properties must be provided. It is possible to either provide a single property to use globally, or to provide a vector, where the properties of each bulkhead are specified explicitly. Both types of bulkheads introduced in section 6.1.2.1 are supported, and all CPACS parameters can be set directly by the user via the input deck.

The designation of skin segments is necessary to assign mechanical properties to the fuselage skin. It also provides a way to identify e.g. sizing regions for structural sizing, where all skin panels within a skin segment share the same properties. In this way, constraints, e.g. due to manufacturing, which prohibit each panel thickness to be sized individually, can be taken into account. For the skin thickness sizing, this implies that all member panels of the skin segment are sized by the most critical panel in the segment.

As described in section 6.1.2.1, the skin segments are bounded by stringers in circumferential direction and by frames in longitudinal direction. For the design, the number of lengthwise and circumferential skin segments can be specified by the user. The design algorithm will then attempt to split the number of frame and stringer bays in longitudinal and circumferential direction as evenly as possible. For the frames the starting point for the distribution is the front, whereas for the stringers it is the crown curve. If an even split of the frame or stringer distributions is not possible, the segments at the end and the bottom are adjusted respectively. Furthermore, an option is available to center the upper circumferential segment around the crown curve instead of it beginning there. In this way, the sizing regions can better be tailored towards the common load regions in the fuselage, i.e. tension at the top, compression on the bottom and shear on the side.

As the design capabilities implemented in the `fuga.design` subpackage are not intended to provide a sized structure, but a structural topology that is a basis for subsequent structural sizing tools, the description of the skin thickness distribution is kept simple compared to those commonly found in flying aircraft. Revisiting the idea of the major load regions, a thickness can be assigned by the user to each circumferential segment, which is then applied along the fuselage length. A global material is provided via a material uID, referencing the donor model.

6.2.2.6. Cutouts

FUGA automatically determines the cutouts for windows, passenger doors and main cargo doors. The passenger and cargo door cutouts are derived from the cabin and cargo floor layout respectively and provide necessary inputs for the frame and stringer distributions. Conversely, the window cutouts are introduced only once the frame distribution is complete.

The passenger door cutout positions are a result of the exit distribution from the cabin layout, the design of which is described in detail in section 6.2.3.3. The x-positions depend on the distribution of seats and monuments, whereas the z-positions rely on the cabin floor position and a possible sill height. The cutout dimensions, i.e. the width w_{door} and height h_{door} as well as the corner radius r_{corner}, are prescribed by the exit type, which is selected during cabin design based on the required passenger capacity.

As discussed in section 6.1.2.4, the position of the cutout is given by a point on the fuselage surface, described in terms of a reference point, given by the entries of `positionX`, `referenceY` and `referenceZ`, and an angle, given by `referenceAngle`. To facilitate the description, the angle is chosen so that the intersection vector is parallel to the y-axis, i.e. 90° for the left side of the fuselage in flight direction and -90° for the right side. This means that the x and z-coordinates of the intersection point can reliably be defined using the `positionX` and `referenceZ` nodes. Since the reference point provides the center of the cutout, the values of the nodes are given by

$$x_{ref,door} = x_{min,door} + \frac{w_{door}}{2} \tag{6.2.16}$$

and

$$z_{ref,door} = z_{min,door} + \frac{h_{door}}{2}, \tag{6.2.17}$$

where $\begin{bmatrix} x_{min,door}, & z_{min,door} \end{bmatrix}$ is the side view position of the lower forward corner of the door notwithstanding the corner radius, which is applied subsequently.

Contrary to the passenger doors, no certification requirements exist w.r.t. the size of cargo doors. Nevertheless, a cargo container, for which the aircraft is designed should comfortably fit through the opening. Consequently, the size of the door is computed in FUGA based on the dimensions of the intended container type and a tolerance margin. Both the type and the margin are specified by the user, along with a corner radius.

With the dimensions established, the position of the cargo door is determined based on the cargo floor layout, which is considered in section 6.2.3.6. The lower bound of the cutout is aligned with the floor height to allow for smooth passage of the container during loading. Furthermore, the door should be placed at one of the ends of each cargo floor segment. Multiple segments may exist due to an interruption of the cargo floor by the center fuselage area. To ensure safe operations, the container position, which is furthest from the wing is selected for each section. An example of the design approach is given in figure 6.2.13.

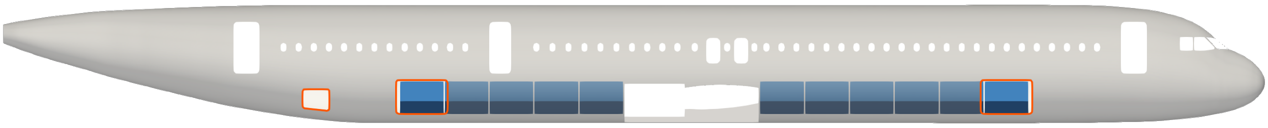


Figure 6.2.13.: Cargo door placement due to container distribution

Finally, the windows are modeled based on the frame distribution. The dimensions of the window cutouts are provided by the user as well as the height above the floor. The longitudinal position is selected in such a way that a window is always centered between two neighboring frames. A window is always placed, unless

- a boarding door is present (windows in emergency exit doors are allowed),

- a monument is present in the adjacent space (s. section 6.2.3.5),
- the frame bay is outside the cabin boundaries.

No design capability for cockpit window cutouts is available at this point, as non-rectangular cutout profiles are not supported by CPACS, yet. However, interfaces are provided ion FUGA to supply the cutout shape as an external input.

6.2.2.7. Summary of structural design parameters

The collected simplified input parameters for the structural design in FUGA can be found in table 6.2.1. Also listed is the corresponding input source from section 6.2.2.1, e.g. the user input deck or the donor model. Parameters marked with the source `decks`, rely on input from the cabin layout. This mainly affects the longitudinal beams, which rely on the seat layout and the shape and position of the door cutouts, which are determined by the exit layout. Some further, indirect dependencies exist, e.g. between the frame positions and the door cutouts or the advanced stringer generation and the cabin boundaries, which can also be traced back to the cabin. However, these dependencies do not appear in the list, since they are encapsulated in the design rule set and thus not exposed to the user.

The table shows that the number of necessary design inputs can be reduced substantially w.r.t. to the overall number of parameters necessary to describe the structural layout in CPACS. On the one hand, this is achieved by describing distributions of components e.g. frames or stringers, which are all essentially similar, using generalized formulations. This means that less control over the placement of each individual part is given to the user, as the task of creating part instances is taken over by the design rules. Since the distribution of the components of the given types can be well described algorithmically, this is a desirable behavior.

On the other hand, the number of parameters can be reduced by referencing libraries of predefined knowledge via the donor model. As shown by table 6.2.1, this is primarily leveraged for the profile-based structural element definitions and for the materials. Here, the user only needs to provide a uID in the input deck, which references an entry in the donor model. In this way, the complexity of describing the section or material properties is detached from the design process. In addition, the limitation to a library of profiles, which can be understood as semi-finished products, or materials is in line with common engineering practice, where designer is often limited to a certain set of options, due to cost or certification constraints.

The list of parameters given in table 6.2.1 covers the structural design capabilities described in this section. To determine the required cabin inputs, which, if missing, could prevent a successful structural design run, a set of complementary cabin design rules is introduced in the following.

6.2.3. Cabin design rules

Analogously to the structure design rules in section 6.2.2, the rules for the generation of a cabin layout are presented in this chapter. The goal is to provide cabin data according to the CPACS schema by Walther et al. [WH+22a], which can be translated into a geometric model using the rule set discussed in section 6.1.3. Similarly to the ambition described by Fuchte, Gollnick, and Nagel [FGN13], the goal is to link the design of the structure and the cabin for a more integrated fuselage design, albeit using the KBE approach discussed in chapter 5, to support the generation of cabin models at multiple levels of fidelity.

Following a short introduction of the necessary design inputs in section 6.2.3.1, the design rules for the determination of the available cabin space based on the structural layout are discussed in section 6.2.3.2. The design of the exit layout is then described in section 6.2.3.3. Subsequently, the positioning of the floor-based components, i.e. the seats and the monuments, is discussed in section 6.2.3.4 and 6.2.3.5, before proceeding to the cargo floor definition in section 6.2.3.6. Finally, the secondary structure component layout design is addressed in section 6.2.3.7.

Table 6.2.1.: List of simplified structure design inputs in FUGA

Component	**Parameter/data item**	**Symbol**	**Source**
Frames	nominal frame pitch	$\Delta x_{fr,nom}$	user input deck
	main/regular frame structural element uID	-	user input deck
	main/regular frame structural element	-	donor model
Stringers	number of stringers	$n_{stringers}$	user input deck
	stringer structural element uID	-	user input deck
	stringer structural element	-	donor model
Floors	crossbeam z-position	z_{cb}	user input deck
	crossbeam profile uID	-	user input deck
	crossbeam structural element	-	donor model
	crossbeam strut y-positions	y_{strut}	user input deck
	crossbeam strut angles	φ_{strut}	user input deck
	crossbeam strut structural element uID	-	user input deck
	crossbeam strut structural element	-	donor model
	longitudinal beam y-positions	y_{long}	`decks`
	longitudinal beam structural element uID	-	`decks`
	longitudinal beam structural element	-	donor model
	floor panel number of frame	$n_{frames,panel}$	user input deck
	floor panel thickness	t_{panel}	user input deck
	floor panel material uID	-	user input deck
	floor panel material definition	-	donor model
Bulkheads	number of bulkheads	n_{bulk}	user input deck
	outer bulkhead x-positions	$x_{bulk,i}$	user input deck
	spherical bulkhead radius at frame	r_{frame}	user input deck
	spherical bulkhead depth	t_{dome}	user input deck
	number of radial reinforcements	$n_{reinf,r}$	user input deck
	number of horizontal reinforcements	$n_{reinf,h}$	user input deck
	number of vertical reinforcements	$n_{reinf,v}$	user input deck
	bulkhead sheet thickness	$t_{sheet,bulk}$	user input deck
	bulkheads sheet material uID	-	user input deck
	bulkheads sheet material definition	-	donor model
	bulkhead reinforcement structural element uID	-	user input deck
	bulkhead reinforcement structural element	-	donor model
Skin segments	circumferential skin thicknesses	$\mathbf{t}_{skin}$	user input deck
	material uID list	-	user input deck
	material definitions	-	donor model
	start at zero degree switch	-	user input deck
Cutouts	door widths	w_{door}	`decks`
	door heights	h_{door}	`decks`
	door corner radii	r_{door}	`decks`
	door positions	$x_{ref,door}, y_{ref,door}$	`decks`
	window width	w_{window}	user input deck
	window height	h_{window}	user input deck
	window corner radius	r_{window}	user input deck
	window height above cabin floor	$z_{cab,window}$	user input deck

6.2.3.1. Design inputs

Fundamentally, cabin design is a trade-off between airline revenue, passenger comfort and safety requirements within the bounds of the OML and the structure, which are driven by flight performance considerations. Consequently all these fields must be considered during the layout generation. Once again, this is accomplished via both information provided by rules from other rule sets and additional information provided in a user input deck.

That said, the cabin design rules in FUGA are predominantly concerned with the design of the cabin layout, as opposed to the components in the cabin, which are provided in the form of “dead” geometry as part of the input deck. The geometry source may be provided as a CAD model, or as a triangulated 3D mesh. A selection of models for various components including seats and secondary structure is given in figure 6.2.14. In principle, it is also possible to incorporate 3D models retrieved by laser-scanning physical components as described by Rauscher et al. [RB+21], even though the comparatively high number of faces typically makes them more difficult to handle than meshes which have been generated manually or based on CAD geometry.

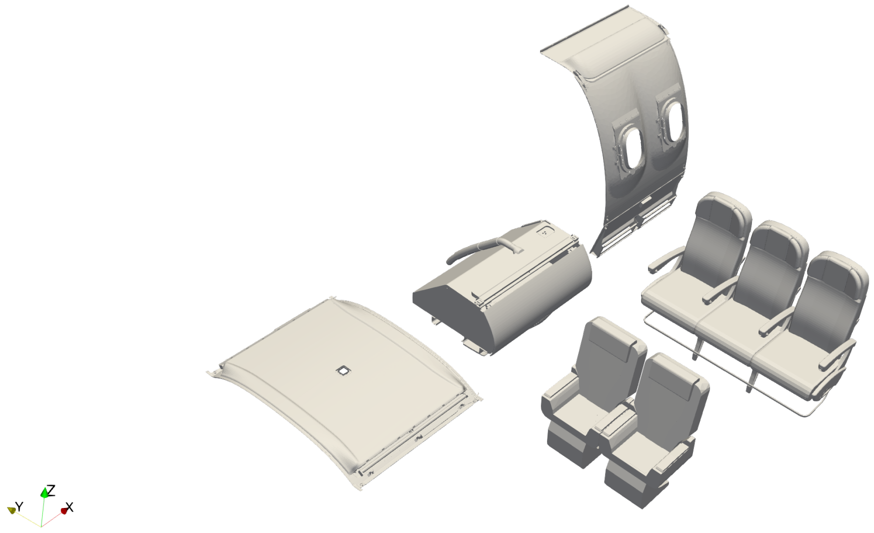


Figure 6.2.14.: Mesh-based models of different cabin components

6.2.3.2. Cabin space

The cabin space plays an important role in the cabin design as it provides the boundary for the cabin. As explained in section 6.1.3, a discretized representation of the cabin space is provided in CPACS. In addition, Walther et al. [WK+22] propose an ad-hoc computation of the cabin space based on the structural layout, which provides higher quality surfaces to be used e.g. in the creation of monument models.

To this end, first, the available space within the bounds of the frames is determined, based on the frame tip curves. To compute the tip curves, simplified frame surfaces are constructed analogously to the frame surfaces discussed in section 6.1.2.1, but using a simplified section curve, which represents the frame height. From these surfaces, the tip curves can be determined by evaluating the isoparametric curve at the endpoint of the section curve. Using the skinned surface algorithm, a surface passing through the tip curves as shown in figure 6.2.15 can be created. An OCCT solid body is built from this surface by filling the surfaces circumscribed by the end tip curves.

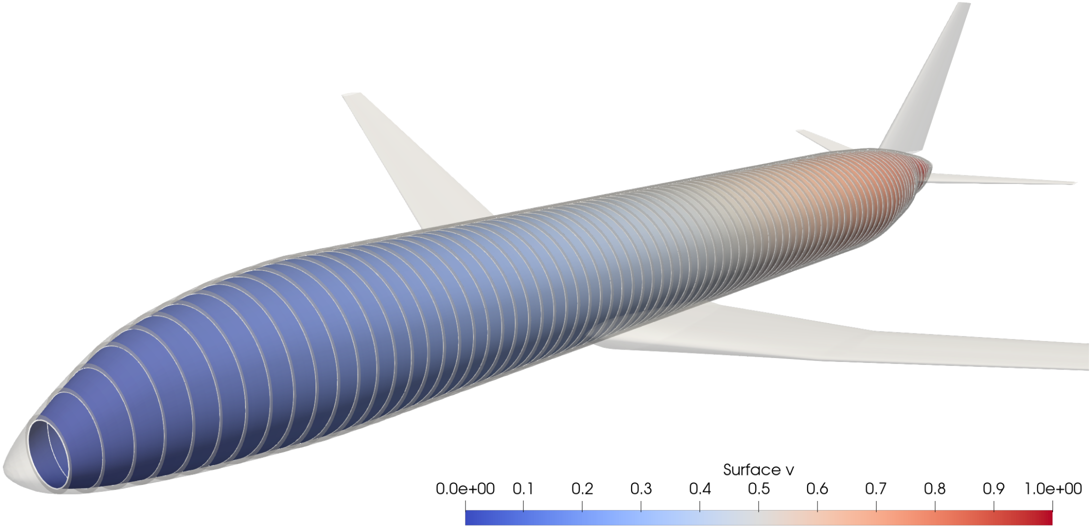


Figure 6.2.15.: Surface to represent the space circumscribed by the frames

Aside from the frames, the cabin is furthermore bounded by the floor, the cockpit rear wall and the RPB. The vertical position of the floor is given by a position in z-direction, which can be computed from the floor structure data taking into account the crossbeam z-positions and the offset due to the longitudinal beam height. On the other hand, the cockpit wall and the rear bulkhead provide the bounds in x-direction. Whereas the rear bulkhead position is given by the structural definitions for an outside-in design scenario, the cockpit length is a user-defined parameter. Typical values lie in the range between $3m$ and $5m$. Incidentally, the position of the cockpit rear wall and the floor height also indicate the position of the cabin origin, along with the center of the fuselage bounding box in width direction.

To determine the cabin space, a box is generated, whose bounds in longitudinal direction correspond to the longitudinal bounds of the cabin, whereas the lower bound in vertical direction is given by the floor height. The upper bound, i.e. the height of the cabin space, is given by the user, whereas the bounds in lateral direction are chosen to lie outside the bounding box of the fuselage surface. Now, the CAD representation of the cabin space can be determined by computing the intersection of the space bounded by the frames and the box as shown in figure 6.2.16.

To compute a discretized representation suitable for exchange using CPACS, a list of samples in x- and z-direction is required. To this end, the user can select the number of samples to evaluate in longitudinal and vertical direction respectively. The length and height of the cabin space are then divided into evenly spaced segments correspondingly. To determine the missing width of the space at each combination of xz-coordinates, the intersection point between a ray with its origin at the coordinate and pointing in positive y-direction with the cabin space is computed. The y-coordinate of the intersection point corresponds to the width of the cabin space for that sample. CPACS implicitly assumes the cabin space to be symmetric, which is why the width is evaluated only on one side. No verification of the assumption of symmetry is performed in FUGA, since the discretized cabin space is only an output, which lies downstream of any design activity.

6.2.3.3. Exits

The exit layout is among the first properties of the cabin to be determined in a FUGA design run. The certification specification provides various exit types in section CS 25.807, which are characterized by their size and prescribe the corresponding passenger capacity. An overview is given in table A.3.1. It follows that the choice of exit types largely depends on the overall number of passengers required by

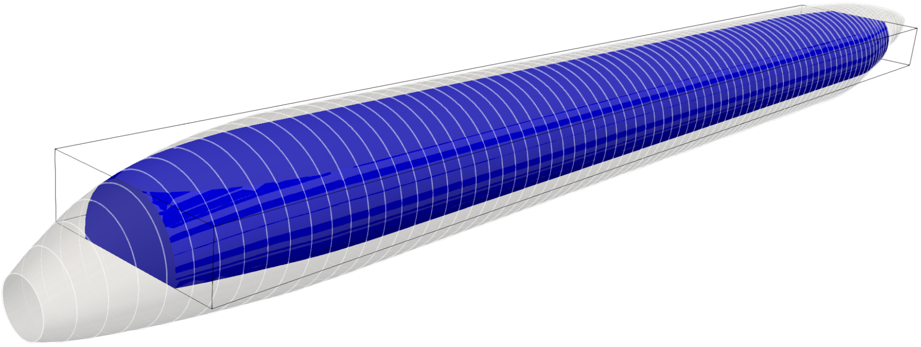


Figure 6.2.16.: Cabin space by intersection of the frame bounds and the box representing the known cabin bounds

the TLAR.

Two possibilities are provided to determine the exit types in FUGA. On the one hand, it is possible for the user to manually specify the sequence of exits from front to rear, by providing an ordered list of exit types. Types are always assigned to pairs of opposite exist, which means four types must be specified if four exit pairs are provisioned. This approach has the drawback that it is up to the user to ensure that the capacity of the exits is sufficient w.r.t. the passenger number. Therefore, on the other hand, a second, automatic approach leveraging mixed integer optimization has been implemented.

However, before explaining the automatic approach, the design of passageways and cross-aisles must be considered. Certification requires in section CS 25.813 that exits be reachable via passageways from the nearest aisle to the exit or cross-aisles between neighboring aisles. The necessary widths are listed in table A.3.1 for the different exit types. In addition, larger exits of types A, B and C require assist spaces to allow flight attendants to support passengers during an evacuation. For smaller exits, it is sufficient if a cross-aisle is found in the vicinity of the exit, whereas for larger exits, it is required for the cross-aisle to fully overlap with the passageway.

From the above information, a total cabin length penalty for an exit type can be determined, i.e. how much the length of the cabin is increased due to the exit. The resulting optimization problem is then to find a combination of exit types that fulfills the capacity requirements with a minimal length penalty w_{exits}.

To this end, the vector of the number of exit pairs per exit type $\mathbf{n}_{exittype}$ as defined in table A.3.1 is optimized, where each entry corresponds to the number of exits in the corresponding row of the table. Correspondingly, the width penalty per exit type is given by $\mathbf{w}_{exittype}$ and the capacity by $\mathbf{c}_{PAX,exit}$. The cabin length penalty for each exit given by the width of the passageway and the adjacent assist spaces. Consequently the length penalty for exit of the type described in the i-th row is given by

$$w_{exittype,i} = w_{passageway,i} + n_{assist,i} \cdot w_{assist,i}. \tag{6.2.18}$$

Based on this, the optimization problem can be stated as follows:

$$\begin{aligned}
\underset{\mathbf{n}_{exittype}}{\text{minimize}} \quad & \mathbf{n}_{exittype}^T \cdot \mathbf{w}_{exittype} \\
\text{subject to} \quad & n_{exittype,i} \geq 0, \qquad\qquad\qquad i = 1, \ldots, n_{exittypes}, \\
& \mathbf{n}_{exittype}^T \cdot \mathbf{c}_{PAX,exit} \geq n_{PAX}, \\
& \sum_{1}^{n_{exittypes}} n_{exittype,i} \geq 2.
\end{aligned} \tag{6.2.19}$$

Since the entries of $\mathbf{n}_{exittype}$ can only be integer values, the above is classified as a mixed-integer linear programming (MILP). The minimization is restricted by several constraints. The first constraint states that the exit type count must be a non-negative integer. In the second constraint, the total capacity of the exits is compared to the number of passengers demanded by the TLAR. It ensures that sufficient exit capacity is provided. Finally, the sum of all exit type counts, i.e. the overall number of exit pairs, irrespective of the type, is set to be at least two. This is required by CS 25.807 for aircraft where $n_{PAX} \geq 20$.

More specific constraints may be introduced to the optimization problem in equation 6.2.19 as required for the analysis at hand. For instance, the CS 25.813 poses some further limitations on the exit layout for single-aisle configurations, where Type A and B exists must not be installed at the end of the fuselage to allow for passenger flow from the front and aft. Conversely, CS 25.807 states that if only two exits are present, they need to be situated near the ends of the cabin. In practice, this means that, if a Type A exit is installed in a single-aisle configuration, at least two smaller exits must also be installed. This relationship could be expressed in an additional constraint as follows:

$$\sum_{1}^{n_{exittypes}} n_{exittype,i} - (n_{typeA} + n_{typeB}) \geq 2. \tag{6.2.20}$$

Furthermore, exit types can be eliminated from the table as required. For instance, the certification process for the higher capacity A+ and C+ type exits is still ongoing, which is why they should be eliminated from the list of eligible exits, e.g. when considering legacy designs for validation purposes.

Even though the automatic approach to determine the exit layout is very powerful, it is sometimes possible to use multiple combinations of exit types to achieve the same length penalty. Furthermore, the results stay the same irrespective of the order of exits. It is impossible to tell in advance, which result the optimizer will return, which gives the method an element of randomness, which may be undesirable to the design engineer. Consequently, the optimization is not used as default design method in FUGA. It is, however, executed in the background as a means to validate the user inputs. If a non-optimal solution is chosen, a warning will be raised proposing an optimal solution instead. Then, it is the responsibility of the design engineer to adapt the exit sequence.

6.2.3.4. Seat distribution

The design of the seat distribution in FUGA is based on the concept of seat blocks. A seat block contains a an arbitrary number of seats of the same type, which are arranged one behind the other in longitudinal at a given seat pitch Δx_{seat}. To this end, references to external seat models can be combined with a seat pitch to formulate class definitions in the user input deck. The seat pitch Δx_{seat} is then one parameter to describe the level of comfort in the cabin.

A seat block can then be generated by referencing a class and giving a number of seats in the block. As illustrated by figure 6.2.17, each block is bounded in longitudinal direction by another seat block, or an exit zone (s. section 6.2.3.5). In lateral direction, it is bounded by an aisle or the cabin boundary. The relevant cabin boundary is given by the cabin space at floor level. The aisle width is a user input parameter. However, certification requires a minimum aisle width based on the passenger number in section CS 25.815. Within the lateral bounds, seats can either be centered, or be adjusted to the left or the right, i.e. the wall or aisle respectively for the leftmost block. The exit zones are positioned based on the seat block lengths from front to back. Due to e.g. cross-aisle overlap criteria, it is still possible, for some space to remain. In this case the seat block remains shifted towards the front. Nevertheless, the positions of the seats can be manipulated using an initial gap parameter.

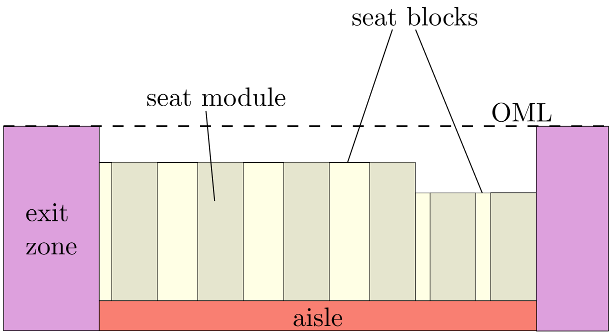


Figure 6.2.17.: Placement and bounds of seat blocks consisting of different seat module types

To allow for multiple classes, multiple seat blocks can be placed back-to-back between a given pair of exits. Transitions e.g. from triple to double seat modules to account for reduced space in the tail section can be implemented in the same way. Here, restrictions w.r.t. the number of seats abreast must be observed. For a single-aisle configuration section CS 25.817 restricts the number of seats abreast per row to three on each side of the aisle. For configurations with multiple aisles, up to five seats abreast are possible between two aisles.

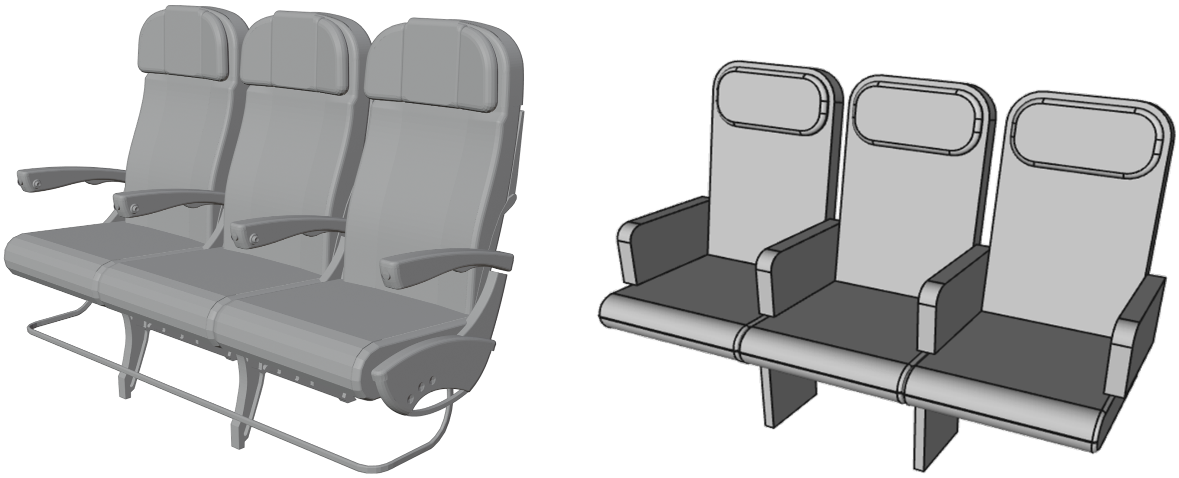
If the seat block distribution is given explicitly, the passenger number is given by the sum of passenger numbers in the seats. However, it is often more desirable to determine a seat layout based on a passenger number without having to assign block sizes manually. To this end, the design problem could again be formulated as an optimization problem. Compared to equation 6.2.19, the problem is, however, significantly more complex, since it combines mixed-integer design variables and black box constraint functions. As such, automation of the distribution design is not considered as part of this thesis. Instead, constraints are formulated via FUGA rules to ensure that some basic requirements are met. These include logical checks, e.g. that the number of seats shall be equal to the requested number of passengers, but also constraints due to certification requirements. An example for the latter is the maximum exit distance, which is limited to $\Delta x_{exit} \leq 60ft$ by CS 25.807 (e) (4), since the distance between the exits is actually driven by the sizes of the intermittent seat blocks.

The above methods assume that a non-parametric or “dead” seat geometry model as shown in figure 6.2.18a is available, leaving only the seat pitch Δx_{seat} as a means of manipulating the passenger comfort. However, many more parameters are available when considering the seat at component level. In figure 6.2.18b, a simplified seat CAD model, also implemented using FUGA based on a parametric description provided by Torenbeek [Tor76], is shown. The model introduces further design variables, such as seat width or recline, which also have an effect on the perceived comfort of the passenger. The integration of the detailed design of the component model into the overall design is, however, not pursued as part of this thesis.

6.2.3.5. Monuments

Monuments are large floor based cabin components, such as galleys and lavatories. Similarly to the cabin pitch, the availability of lavatories and service trolleys, the latter of which depends on the available galley space, is indicative of the cabin comfort level.

In FUGA, all monuments are placed in the neighborhood of exit spaces creating exit zones. Similarly to the seats, the geometry can be provided in terms of a model or dimensions. A system has been devised to identify positions next to exits using three parameter. First, the exit index specifies next to which exit pair the monument is placed. The index is zero-based and runs from front to back. As



(a) Triangle mesh

(b) Basic parametric CAD model

Figure 6.2.18.: Seat model types

such, the last exit pair in a configuration with a total of 5 exit pairs would have the index 4, while the second one would have the index 1. Similarly, an index is given for the lateral position. Going from left to right in flight direction, each seat block area or aisle between seat block areas is assigned an index. A twin aisle configuration would therefore have indices running from zero to four, with one and three indicating aisle positions. Finally, a Boolean parameter specifies whether the monument is placed in front of (0) or behind the passageway or cross-aisle (1).

Different models for monuments can be provided by the user. To place the model in the fuselage, the three position indices must be given. If the model, e.g. a lavatory module is used more than once, multiple sets of indices can be given. If two models are placed in the same position, FUGA will raise an input error. Otherwise, if the bounding boxes of two models in neighboring positions collide, the tool will attempt to resolve the collision by moving the monuments outward.

The monument models can furthermore be enriched with metadata. For instance, a lavatory module may contain multiple separate cabins, which is indicated by the number of lavatories n_{lav}. On the other hand, the number of full-size trolleys that can be held in a galley can be approximated by dividing the galley width (FUGA uses the larger of the x and y dimensions of the galley) by a given full-size trolley width w_{FST}. As a default value, $w_{FST} = 0.3m$ is assumed. The overall number of trolleys n_{FST} can be computed by summing up the contributions from all galleys.

Based on these numbers, the utilization factors $\frac{n_{PAX}}{n_{lav}}$ and $\frac{n_{PAX}}{n_{FST}}$ can be determined, which can be used as further comfort indicators. As with the passenger number for the seats, typically only the utilization factors will be provided as a required input in the TLAR, whereas the exact distribution of the monuments is of secondary concern.

6.2.3.6. Cargo floor

The design of the cargo floor in the cabin rule set is mainly concerned with the placement of the cargo containers, which in turn influences e.g. the placement of cargo doors as stated in section 6.2.2.6. In CPACS, cargo containers are considered to be floor-based components similar to seats and monuments. A notable difference is, however, that no model is given explicitly. Instead the standardized geometry of the cargo container is indicated via its type as listed in table A.3.3. The type used for the design is given as a user input.

Compared to seat layout, the only additional difficulty when computing the cargo floor layout is the presence of the center wing box for low-wing configurations as well as the landing gear bay. Here, information provided in the center fuselage area description must be taken into account, splitting the cargo floor into two separate segments. The user can provide an overall number of containers, to be distributed on the cargo floor via the input deck. The distribution between the two floor segments is then performed automatically, based on the length ratio. Several options are then available for the longitudinal alignment of the container within the segment, e.g. centered, front or rear. In addition it is possible to align the containers towards the fuselage center, i.e. towards the rear for the first segment and towards the front for the second segment.

Similarly to the monuments, a utilization factor for cargo containers can be determined, e.g. $\frac{n_{pax}}{V_{cargo}}$, which can be applied for comfort evaluation. For a more precise utilization factor, additional cargo, which does not belong to the passengers, could be taken into account as well.

6.2.3.7. Secondary structure

Out of all cabin components supported by FUGA, the secondary structure is the most closely coupled to the structure. In this thesis, the term secondary structure is applied in a different sense, than in an industrial context, where it often refers to a subset of ATA-chapter 53, e.g. brackets or fittings. Instead, the term is used here to refer to the subset of those cabin components, which are mounted to the walls and ceiling of the cabin. Hence, the secondary structure comprises the sidewall panels, which in turn include dado panels and cowl panels, the overhead stowage compartments (OHSCs) and the ceiling panels. Similarly to the seats, the design once again relies on external models to supply the component geometry.

The architecture of the secondary structure typically depends on the number of aisles. A sidewall panel is always found at the outermost position, on top of which a OHSC is installed. In a single-aisle configuration, the two compartments are then connected by a ceiling panel. If more than one aisle is present, a ceiling panel is usually installed above each aisle, with a two-sided OHSC element connecting two adjacent ceiling panels.

Determination of component positions Components of the secondary structure are commonly designed to align with the frame distribution. In most cases, the components span multiple frame bays. For instance, in an Airbus A320 family cabin, as shown in figure 6.2.19, a sidewall panel or ceiling panel usually spans two frame bays, whereas an OHSC spans four frame bays. Walther et al. [WK+22] propose a procedure for placing the component models based on the known frame positions and cabin boundary, which is applied in FUGA and outlined in the following. As a simplification it is assumed in FUGA that OHSCs, too, span only two frames, which means that a section of sidewall panels, OHSCs, and ceiling panel can be considered independently.

To start off the design process, the intersection points between the frame planes and the cabin boundary curve are computed. These points provide the reference for the positioning of the sidewall panels. Whereas Walther et al. rely exclusively on the bounding boxes of the components and lateral overlap parameters for the positioning, it is also possible for the user to identify structural attachment points on the original model in FUGA. Taking the sidewall panel model as an example, four attachment points are given. The two lower attachment points denote connections to the frame and floor intersection points, whereas the upper two attachment points provide the connection to the OHSC. In turn, the OHSC provides a pair of connection points connecting to the sidewall panel and another pair for the ceiling panel etc.

Based on the relative positions of the points, or the axes defined by a pair of points, the corresponding transformation matrices for the components can be determined. The matrices must determine a scaling on the one hand, and a translation and possibly a rotation on the other. All of these operations can be expressed through a single transformation matrix in homogeneous coordinates $\mathbf{H}$, where the traditional rotation matrix, e.g. in 2D, which describes a rotation by an angle γ and a scaling by a scaling factor S_{glob} is expanded by an additional row and column to describe the translation by the vector $\left[\begin{array}{cc} d_x, & d_y \end{array}\right]^T$:

$$\mathbf{H} = \begin{bmatrix} S_{glob} \cdot \cos(\gamma) & \sin(\gamma) & d_x \\ -\sin(\gamma) & S_{glob} \cdot \cos(\gamma) & d_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.2.21}$$

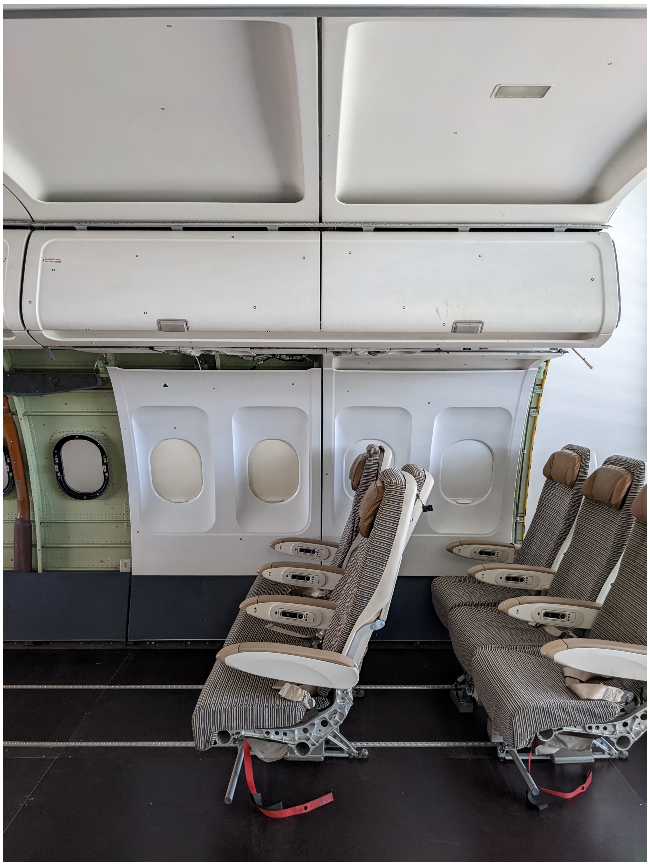


Figure 6.2.19.: Section of OHSC and two sidewall and ceiling panels (cabin mock-up at the DLR Institute of System Architectures in Aeronautics)

To determine the position of a point after the transformation, the point must also be converted to homogeneous coordinates, i.e. $\mathbf{q} = \begin{bmatrix} x, & y, & 1 \end{bmatrix}^T$. The transformed vector in homogeneous coordinates is then given by

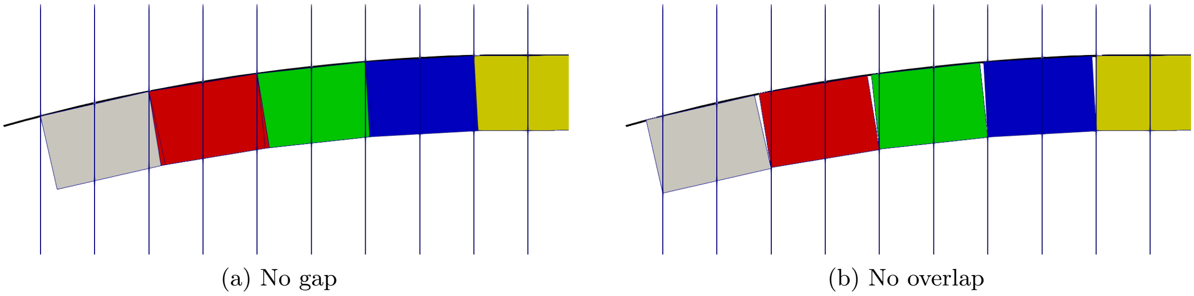
$$\mathbf{q}' = \mathbf{H} \cdot \mathbf{q}. \tag{6.2.22}$$

While the approach is described for the 2D case here for simplicity, it works analogously in 3D.

To determine the transformed coordinates of multiple vectors $\mathbf{Q}'$, an $n \times n_{dim} + 1$ matrix $\mathbf{Q}$ can also be multiplied with $\mathbf{H}$, instead of the vector $\mathbf{q}$. To solve equation 6.2.22 for the transformation matrix $\mathbf{H}$ by multiplying from the right by $\mathbf{Q}^{-1}$, $n \overset{!}{=} n_{dim} + 1$, i.e. the number of points must correspond to the number of homogeneous coordinate dimensions. In the case of a 3D model, where $\mathbf{H}$ is a 4×4 matrix, this results in 4 points. The points in $\mathbf{Q}$ must furthermore be linearly independent.

Typically, an alignment axis given by a pair of attachment points is given in FUGA to position e.g. a sidewall panel. To determine the two additional points, the origin of the axis is offset by the unit vector in z-direction $\mathbf{e}_z$, providing an additional point. This ensures that the component stays upright. Furthermore, the inward facing unit vector must be computed. The direction of the vector is given by the cross product of the alignment axis and $\mathbf{e}_z$. However, the system is not necessarily right handed, which means the orientation of the vector must be reversed on one side for it to be pointing inward. Again, the fourth point is given by the sum of the origin and the normalized inward facing vector.

As the default geometry models for the secondary structure used in FUGA are designed for the constant mid-section, the problem of positioning the components can be solved as described above without problems for this part of the fuselage. Difficulties arise in the forward and aft sections, where the secondary structure needs to follow the curvature of the fuselage. The problem is illustrated for the sidewall panels by figure 6.2.20. If the attachment points are placed at the frame positions along the cabin edge, the paneling will be closed, but the models will overlap going inward. This problem is amplified if OHSCs are installed at their attachment points. Therefore, another approach is to determine the axes defined by the attachment points and shift the component position along the relevant axis, so that the innermost points are at the frame positions. In this way, gaps appear in the paneling, but no overlap appears.


(a) No gap

(b) No overlap

Figure 6.2.20.: Sidewall panel positioning strategies for non-constant fuselage sections (from [WK+22])

Finally, scaling and placing the rectangular ceiling panel in a way that the attachment points are aligned is impossible if the neighboring components are installed at an angle. A simple solution is therefore to compute a transformation matrix, which minimizes the distance between the neighboring attachment points and the ceiling panel attachment points in a least-squares sense.

Adaptation of models in non-cylindrical fuselage segments The example of the ceiling panel shows that the external models provided are not always well-suited in a specific geometric context during the design. Therefore, some component design capabilities must be introduced at this point, which go beyond simple scaling of models. With no parametric CAD component models available, the FFD method described in section 2.2.3 can be applied instead to manipulate the geometry mesh. In FUGA, FFD is deployed e.g. to create tapered ceiling panels as shown in figure 6.2.21.

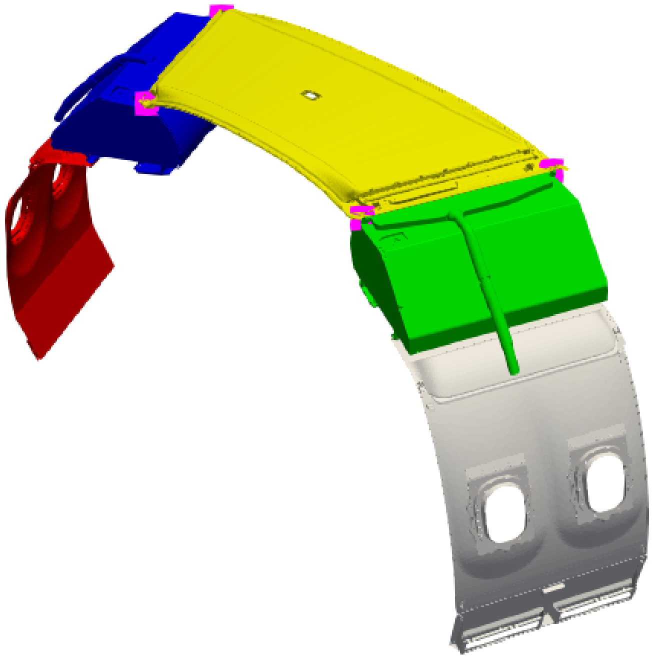


Figure 6.2.21.: Secondary structure segment with deformed ceiling panel and attachment points marked

To this end, first a FFD lattice is placed around the undeformed panel based on the bounding box. Due to the simple nature of the expected deformation, no subdivision of the lattice is required. A box is a special case of a parallelepiped and can thus be described via its origin $\mathbf{p}_{0,bbox}$ and three edge vectors $\mathbf{v}_{u,bbox}$, $\mathbf{v}_{v,bbox}$, and $\mathbf{v}_{w,bbox}$ pointing away from the origin. Based on this description, the local coordinates $\mathbf{u}_i = \begin{bmatrix} u_i, & v_i, & w_i \end{bmatrix}$ of the i-th panel attachment point $\mathbf{p}_{attach,i}$ are given by

$$\mathbf{u}_i = \begin{bmatrix} \mathbf{v}_{u,bbox} \\ \mathbf{v}_{v,bbox} \\ \mathbf{v}_{w,bbox} \end{bmatrix}^{-1} \cdot \left(\mathbf{p}_{attach,i} - \mathbf{p}_{0,bbox}\right). \tag{6.2.23}$$

Now a deformation of the lattice points must be found so that $\mathbf{p}(\mathbf{u}_i) = \mathbf{p}_{target,i}$, where $\mathbf{p}_{target,i}$ is the attachment point on the luggage compartment, corresponding to $\mathbf{p}_{attach,i}$. This can be accomplished by making the lattice points the control points $\mathbf{c}$ of a piecewise polynomial tensor product volume as described in section 4.1.1.2. In the case of the ceiling panel, due to the simplicity of the intended tapering deformation, a linear Bézier volume is sufficient to deform the bounding box of the panel.

The new positions of the Bézier volume control points to align the panel attachment points with the adjacent luggage compartment attachment points are computed by Bézier volume interpolation with prescribed parameters, which is the three-dimensional variant of bivariate point grid interpolation. Since the bounding box volume is defined by eight control points, eight pairs of source and target points must be given. However, as illustrated by figure 6.2.21, only four pairs of attachment points are given. Therefore it is necessary to introduce an additional constraint. In this case, it is assumed that the height of the original shape remains unchanged. The assumption can be taken into account by including copies of the original points, to all of which a small offset in z-direction is applied.

To compute the deformed mesh, the local coordinates of all mesh points in the undeformed FFD lattice are determined using equation 6.2.23. Then the deformed FFD lattice is evaluated at these coordinates, providing the updated mesh point positions.

Notably, the resulting deformation of the mesh cannot be expressed using CPACS, e.g. using a transformation node. Therefore, the deformed mesh must be stored separately from the original mesh as a new external model with its own entry in the `deckElements` node.

6.2.3.8. Summary: List of cabin design parameters

In some ways, the rationale behind the choice of design input parameters for the cabin can be compared to the structural design discussed in section 6.2.2.7. On the one hand, a library of components is required, which is provided via the donor model in the case of the structure. In the case of the cabin, such a donor model was not available during early development. Therefore, the component descriptions are provided via the cabin design input file instead, using the parameters given in table 6.2.2. The list contains descriptions of seat modules, floor-based components and paneling components, which are very closely aligned with the CPACS cabin definition proposed by Walther et al. [WH+22a] and described in section 6.1.3. That said, some additional metadata is given, i.e. the placement axes for the longitudinal beams in the case of the seat modules and the required relative positions of the paneling elements including OHSCs, which can be determined via the locations of the attachment points. Furthermore, each component is given a tag, which serves as an identifier and thus the basis for the subsequent determination of the component uIDs in CPACS.

In addition to the component descriptions, table 6.2.2 contains the seat block class definitions. As described in section 6.2.3.4, these classes can be assigned to the individual seat blocks to determine the type of seat, the seat pitch, and the longitudinal and lateral alignment of the seats. The seat component tags are used to reference the desired seat model. A class type is given to provide a descriptor of the intended type of cabin class, e.g. first, business or economy.

Similarly to the CPACS definition, the deck instance can now be specified based on the components via the parameters listed in table 6.2.3. Four groups of parameters are identified. First, the overall deck definition parameters give some relevant inputs e.g. for the determination of the cabin origin to

Table 6.2.2.: Component/class type definitions

Component	Parameter/data item	Symbol	Source
Seat modules	tag	-	user input deck
	number of passengers	n_{pax}	user input deck
	model geometry	-	external model
	model scale	$\begin{bmatrix} S_x, & S_y, & S_z \end{bmatrix}$	user input deck
	long beam placement axes	$\begin{bmatrix} y_{long}, & z_{long} \end{bmatrix}$	user input deck
Floor element	tag	-	user input deck
	type	-	user input deck
	model geometry (optional)	-	external model
	dimensions	$\begin{bmatrix} l_{fe}, & w_{fe}, & h_{fe} \end{bmatrix}$	user input deck
	number	e.g. n_{lav}	user input deck
	trolley width (galley only)	w_{FST}	user input deck
Paneling	tag	-	user input deck
	model geometry	-	external model
	attachment point locations	$\mathbf{p}_{attach,i}$	user input deck
Seat block classes	tag	-	user input deck
	class type	-	user input deck
	seat tag	-	user input deck
	seat pitch	Δx_{seat}	user input deck

be used e.g. in the cabin space determination described in section 6.2.3.2. Furthermore, the intended number of passengers and aisles are given, which do not drive the design, but are used in order to validate the inputs given in the subsequent groups.

Next, the exit definition is given, where the different exit pairs are given via a list of their respective classifications, e.g. A or C, from front to rear. The boarding flag parameter indicates for each exit, whether it is meant to be a boarding exit, which will result in increased exit dimensions w.r.t. the minimum requirements formulated in the CS-25 (s. table A.3.1). For symmetric single aisle configurations it is furthermore possible to define the exit position via a cumulative passenger share. This means that an exit can be defined to lie e.g. behind 50% of all the passengers in the cabin. This substantially facilitates the layout of the seat blocks.

The monument placement parameters are given next, which reflect the placement strategy described in section 6.2.3.5. Similarly, the seat block definition parameters are given in correspondence to section 6.2.3.4. Notably, multiple block instances with the same longitudinal and lateral index can be given to describe composed seat blocks as shown in figure 6.2.17. If a cumulative passenger share can be applied, a single composed seat block is sufficient for the entire cabin, otherwise the blocks must be separated to reflect the exit locations.

It becomes apparent by considering the sources given in tables 6.2.2 and table 6.2.3 that the cabin design is controlled almost entirely via the user input deck. The only additional input, aside from the CPACS data set of the base configuration layout, are the external component geometry models. In the current version of FUGA, the user input deck is provided as a Java Script Object Notation (JSON) (Java Script Object Notation) file [Ecm17]. The format integrates well with established Python data structures and can be modified manually with low effort, which makes it interesting for the tool development phase. That said, for the deployment of FUGA in collaborative tool chains, it would be desirable to provide the inputs via a `toolspecific` namespace within CPACS, analogously to the structural design inputs. Furthermore, it might be expedient to assemble some kind of design template library, which combines the relevant inputs from the structural donor model and cabin component library elements. In this way, data from various projects could be collected and leveraged in subsequent activities.

Table 6.2.3.: Deck instance definition

Component	Parameter/data item	Symbol	Source
Overall deck	tag	-	user input deck
	vertical floor position	z_{floor}	user input deck
	cockpit length	$\Delta x_{cockpit}$	user input deck
	number of passengers	n_{PAX}	user input deck
	number of aisles	n_{aisles}	user input deck
	aisle width	w_{aisle}	user input deck
Exits	type list	-	user input deck
	approximate cumulative passenger share (optional)	$n_{PAX,i,target}$	user input deck
	boarding flag	-	user input deck
Monuments	component tag	-	user input deck
	exit index	i_{exit}	user input deck
	before exit flag	-	user input deck
	width position index	i_{width}	user input deck
Seat blocks	block class	-	user input deck
	lateral index	i_{lat}	user input deck
	longitudinal index	i_{long}	user input deck
	number of rows	$n_{rows,i}$	user input deck
	lateral alignment	-	user input deck
	longitudinal alignment	-	user input deck

6.2.4. Interdisciplinary connections in fuselage design rule sets

Although the design of the OML, the structural layout and the cabin configuration are presented as separate sets of rules in this chapter, it quickly becomes apparent that no individual task can be performed adequately without taking into account contributions from the other. Interdisciplinary connections can be found in any of the three activities, making them mutually reliant.

Several examples can be found in the above rule sets. The most obvious nexus of disciplines is the cabin space, discussed in section 6.2.3.2. It represents an essential input to the cabin design, both as a limit for the floor layout and as a reference for the design of the secondary structure. Meanwhile, both the OML of the fuselage and the structural layout must be considered when building the cabin space. Contributions from several structural components are required, i.e. frame heights, floor position and the rear bulkhead position.

Conversely, the design of the structural layout also depends on decisions taken during cabin design. For instance, the positions of the mainframes depend on the door cutout positions, which are generated based on the exit layout for the cabin. Another example is given by the longitudinal floor beams, which are designed using the seat layout as an input.

These connections are also found in the N^2-chart illustration of the outside-in design system in figure 5.2.1. It shows both feed-forward connections from the structural design to the cabin design and feedback connections in the opposite direction. It can also be seen that the cabin design rule sets provides outputs only to the cabin model generation rule set, whereas the structural design rule set only provides outputs to the structural model generation rule set. For the latter, it must, however, be noted that some of the outputs are provided in an unprocessed form and must thus pass through one of the CPACS data preparation rule sets, before being fed to the model generation.

When introducing even the hybrid design rule set for modifying the fuselage length, even more connections are introduced. For conventional configurations, it can make sense to introduce a rule to align the position of the rear bulkhead with the front spar position of the VTP. In this way, the cabin space can be manipulated by adapting the change in fuselage length. The position of the rear bulkhead can, in turn, be prescribed to the required cabin length determined from the floor element positions and dimensions. In this way, the OML can be adjusted to fit a prescribed cabin layout exactly.

The automated handling if the interdisciplinary relationships, using the KBE approach implemented in FUGA, makes it possible to not only assemble separate disciplinary design rule sets, but also to dynamically combine contributions from disciplines as required. Thus, missing data can be augmented based on the combined design knowledge from all disciplines involved. This is a key contribution towards truly multidisciplinary and integrated cabin design as required by **working hypothesis 3**. To further illustrate the validity of the hypothesis, two cabin design examples are given in section 7.1.

6.3. Rule sets for architecture modification

The previously discussed rule sets provide multi-fidelity modeling and inside-out and outside-in design capabilities for conventional passenger transport aircraft. However, as discussed in section 2.1.3, novel system architectures, which radically deviate from the traditional layout, must be considered as an alternative to incrementally improved conventional designs to address the current challenges in aviation. According to **working hypothesis 4**, this should be reflected in the KBE system by providing the possibility to include further knowledge or rules to describe architectural patterns that are new and unanticipated by the original system. To this end, it is proposed in section 5.2.2 to introduce additional rule sets, leveraging the modular approach to structuring and deploying knowledge in FUGA, in order to adapt the KBE system to novel architectures, while retaining relevant aspects of the baseline knowledge.

Two examples for additional rule sets are introduced in the following. On the one hand, a basic rule set for the geometric design of a fuselage-integrated LH_2 tank is proposed in section 6.3.1. The rule set is intended to be applied for integration studies into existing designs leveraging a hybrid approach as discussed in section 6.2.1.2. On the other hand, the determination of the cabin space for an unconventional BWB configuration to enable cabin design is addressed in section 6.3.2.

6.3.1. Cryogenic tank design rules

The introduction of a new large structural component, such as an LH_2 storage tank, effects far reaching changes in the overall architecture of the design. However, as shown by Walther et al. [WH+22b], an integration into a KBE tool such as FUGA, is feasible using a relatively small set of rules. On the one hand, the rules must provide a parametric description of the tank, on the other hand a link to the remaining components of the fuselage is required. The rule set described in the following is an attempt to combine the two requirements into an intuitive description of the tank. More in-depth works on tank design have been provided e.g. by Höhne [Höh22] for metallic tanks and byBiermann [Bie22] for tanks made from CFRP.

The fundamental design of the tank is illustrated by figure 6.3.1. It consists of an inner tank, which is surrounded by an insulation layer. The inner wall is made from aluminum and must be designed to withstand the interior pressure of the tank. The outer insulation layer is added to prevent excessive heat ingress to the LH_2, which has an initial temperature of around $21.5K$ [BC+21]. Typically, the tank is designed as a Dewar, where the inner vessel is separated from a second outer vessel by a vacuum to minimize heat ingress by conduction. More detailed examples for different approaches to designing the insulation are provided by Brewer [Bre91]. However, for the purpose of integrating the tank into the fuselage, the overall thickness t_{wall} is sufficient as a parameter.

In the present rule set, the tank is defined by its outer layer. The inner layers are computed based on the outer layer as offset surfaces, where the offsets are given by the respective thicknesses. The innermost layer encloses the available tank volume, which is an important design parameter. The required fuel volume is usually determined by other aspects of the design, such as the design mission and the fuel efficiency of the engines. Using OCCT, the volume of the tank can be determined using the included Gaussian integration method.

The outer surface is described based on a truncated eccentric cone, defined by two circles with

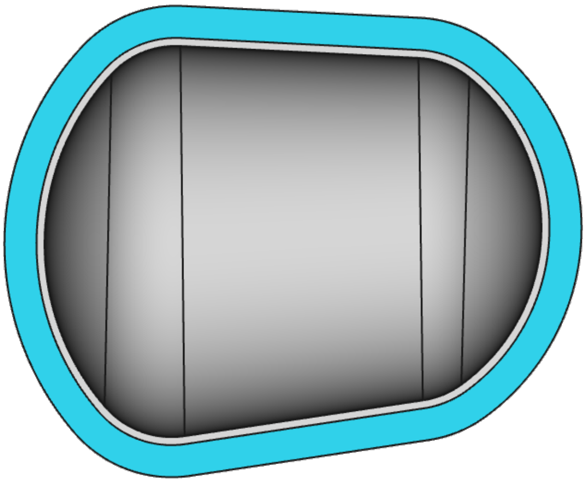
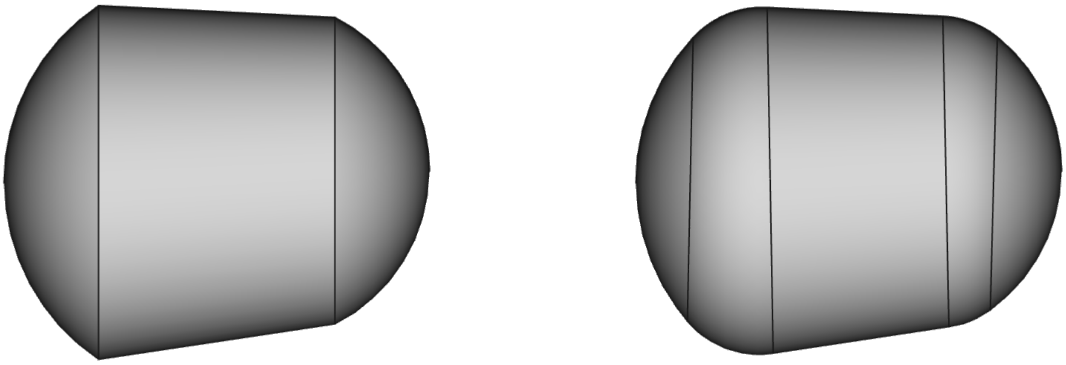


Figure 6.3.1.: Section view of an LH_2 tank with insulation (blue) and wall (gray) (from [WH+22b])

individual centers $\mathbf{c}_{cap,1}$ and $\mathbf{c}_{cap,2}$ and radii $r_{cap,1}$ and $r_{cap,2}$. A spherical closing cap with the depth $l_{cap,1}$ and $l_{cap,2}$ closes the shape on either side, as shown in figure 6.3.2a. To achieve a smoother shape, which is more suited to withstand the interior pressure, the transition between the cone and the caps is rounded out using a fillet with the radius r_{fillet}. The resulting shape, which is closely related to a classical torispherical head shape, is displayed in figure 6.3.2b.



(a) Base primitives

(b) Fillets applied

Figure 6.3.2.: Tank outer shape modeling steps (from [WH+22b])

To assure its consistency with the overall fuselage design, the parameter values of the tank are computed based in the space enclosed by the frames, which also provides the basis for the determination of the cabin space (s. section 6.2.3.2). For a given longitudinal position x_i, the center $\mathbf{c}_i$ and radius r_i are given by the inner circle of the frame space section at that position, as shown in figure 6.3.3.

To determine the longitudinal positions x_i, several additional parameters must be taken into account. First of all, the number of tanks n_{tanks} can be specified by the user, which also determines the total number of sections. Moreover, the overall space for the placement of the tanks is bounded by the rear bulkhead and the HTP wing box in forward and backward longitudinal direction respectively, which, like the frame space, can be determined using rule sets from the `fuga.design` and `fuga.geometry` subpackages. Finally, the lengths of the conic midsections can be computed taking into account the given cap depths $l_{cap,i}$.

As a result of this description, the volume of the tank can be modified, by extending the fuselage

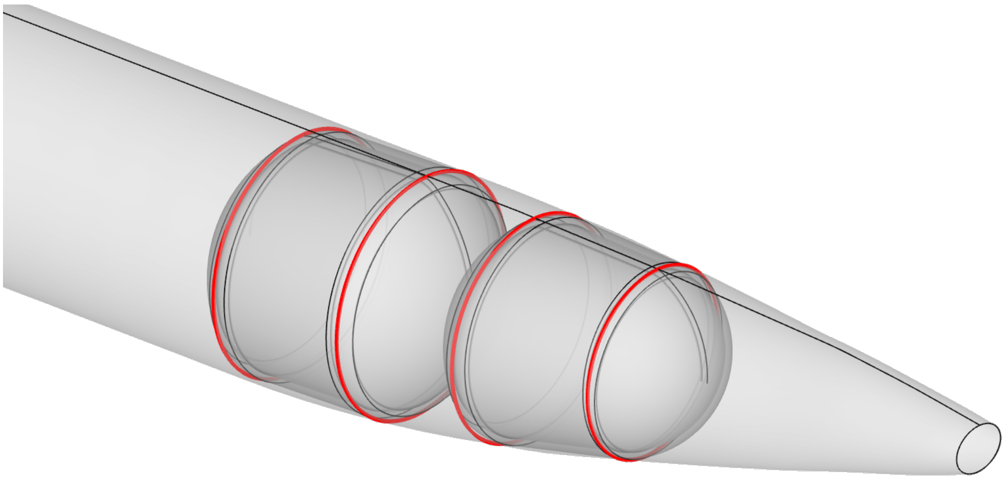


Figure 6.3.3.: Determination of circle parameters from cabin space surface (from [WH+22b])

length using the hybrid OML design rule set described in section 6.2.1.2. As implemented in the rule set, the deformation will also be applied to the HTP and thus the wingbox, whereas the position of the bulkhead remains unchanged, if it is defined in absolute coordinates. In this way, the tank volume can be expressed as a function of the length change of the fuselage $\Delta l_{fuselage}$ and the number of tanks as well as the tank parameters:

$$V_{tanks} = V_{tanks}\left(\Delta l_{fuselage}, l_{cap,1}, l_{cap,2}, r_{cap,1}, r_{cap,2}, r_{fillet}, \mathbf{c}_{cap,1}, \mathbf{c}_{cap,2}, t_{wall}, n_{tanks}\right). \tag{6.3.1}$$

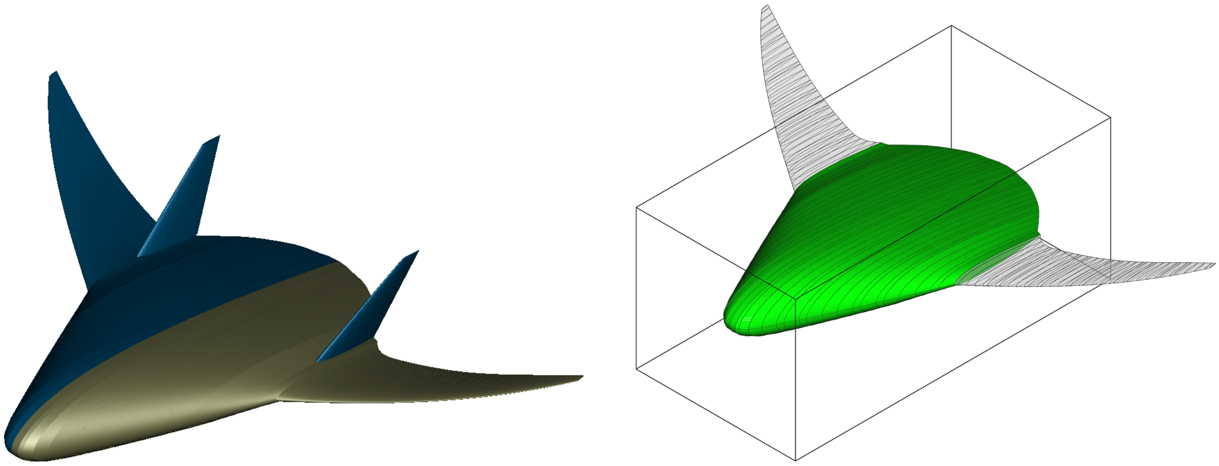
This relationship can be exploited to design for a specific tank volume $\hat{V}_{tanks}$ as described in section 7.2.1.

6.3.2. BWB cabin design interfaces

The determination of the cabin design interfaces for a BWB is another example for an adaptation of the KBE system for novel architectures. The application case is complementary to the tank integration example discussed in section 6.3.1 for several reasons. On the one hand, the BWB represents a more radical departure from conventional architecture, than the configuration with the integrated LH_2 tank. This has an effect also on CPACS modeling practice. Almost all publications on BWB configurations using CPACS, only employ the wing description to define the configuration. This means that no fuselage OML definition is available, which is, however, required for significant portions of the baseline design rules. Consequently, a way must be found to provide an equivalent representation.

On the other hand, the structural design of a BWB is, on its own, still a matter of ongoing research, as discussed in section 2.1.3. Hence, the assembly of a corresponding rule set lies outside the scope of this thesis. Nevertheless, authors such as Lee [Lee03] and Baan [Baa15] have been able to show detailed cabin concepts for a BWB, which could be emulated using FUGA. Therefore, the rule set described in this section is introduced to bypass the structural design by mocking the necessary interfaces to the cabin design discussed in section 6.2.4 using simplified assumptions instead.

Since no fuselage is available in the model, one of the first steps towards mocking a fuselage structure design for a BWB is to define the region of the aircraft, which qualifies as the fuselage, i.e. the area holding the payload. A simple way to accomplish this is to assign boundaries in spanwise direction. In figure 6.3.4, the solid representation of an example BWB configuration from the AGILE project as described by Shiva Prakasha et al. [SD+18] is shown, where the fuselage segments are extracted from the main wing using a box which models the spanwise limits. The resulting shape can be used to provide the closed shell and solid representations of the fuselage described in section 6.1.1.



(a) BWB configuration shown in TiGL Viewer

(b) Fuselage space determination from spanwise boundaries

Figure 6.3.4.: Determination of fuselage space for BWB example

Based on the fuselage shape, the cabin space shape is then to be determined. To this end, a replacement of space enclosed by the frames must be found. A simple approach to accomplish this is to assume a global frame height and compute an offset surface of the fuselage using the corresponding OCCT algorithm. With the solid defined by the offset surface available, the cabin space can be determined by intersecting with a box of cabin boundaries in the same way as described in section 6.2.3.2. A key difference here is that the floor height and the rear bulkhead position are not given since no structural layout is provided. Consequently, the values must be provided by the user as inputs. Figure 6.3.5 illustrates the determination of the cabin space for the above example.

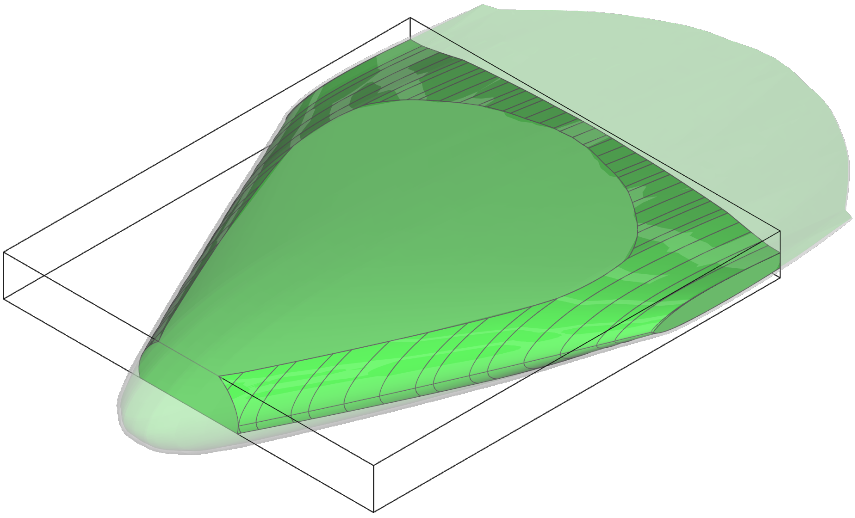


Figure 6.3.5.: Cabin space determination for the BWB example

Aside from the determination of the cabin space, all cabin component positioning rules, which rely on the structural layout are affected. This mainly concerns the secondary structure, as the components are aligned based on the frame positions. However, since no frame positions are given, the secondary structure is omitted. Similarly, the placement of the window cutouts is also based on the frames. If necessary, the distribution can instead be approximated similarly to the frame distribution by providing a target window pitch. That said, the omission of the secondary structure would greatly impair the immersion in an interactive visualization, effectively ruling out such applications of the model. Thus, the issue of the window cutouts is not pursued further, since the addition of the windows does not provide a significant contribution to the remaining types of analyses.

Furthermore, no design of the cargo floor is performed, as the available space also relies on structural details, such as the design of the passenger floor. Due to the unusually high width of the cabin area in the BWB, it can be assumed that conventional design concepts are not applicable and a substantially increased amount of support structure will be required.

Despite these omissions, the BWB example nevertheless shows, how the interfaces of the structural design to the cabin design can be *mocked* by applying simpler modeling techniques deployed in an additional rule set. This enables a cabin design for a BWB, even without having a structural design available. The results of a cabin design applying this rule set are presented and discussed section 7.2.2. Nevertheless, the lack of a well-grounded structural description naturally compromises the validity of the cabin layout, which means that fundamental changes to the design will most likely be required, as more details of the structure become available.

6.4. Discussion

In the preceding sections of this chapter, implementations of different rule sets have been introduced to assemble a KBE system capable of providing multi-fidelity geometry models of aircraft fuselages and cabins, augmenting missing design data, as required, through design capabilities, and adapting to novel architectural requirements. Therefore, this chapter makes a significant contribution towards validating the **research hypothesis** of this thesis.

In section 6.1, the implementation of a parametric modeling engine for CPACS using KBE methodology is presented. One the one hand, this approach to implementing the parametric modeling engine ensures consistency between different disciplinary models, since all disciplinary details are derived from the same basic geometry objects. On the other hand, it also enables fine-grained tailoring of model fidelity by combining different component models at different levels of detail. This fulfills, to a large extent, the requirements formulated in **working hypothesis 2**. However, it is yet to be shown that the geometry models can be used to derive actual analysis models, enabling analysis of the design. To this end, two application cases, selected from the analysis methods for the fuselage listed in section 4.2, are evaluated in section 7.3.

In section 6.2, different sets of design rules are introduced for the OML, fuselage structure and cabin. The primary motivation is to introduce additional design details, which are missing from the initial product data set, to enable higher fidelity analysis. Consistency with the existing results is a key requirement, as formulated in **working hypothesis 3**. It has already been discussed in chapter 5, how technical features of the declarative KBE approach, such as cache invalidation, can help enable consistency. In addition, two fuselage and cabin design synthesis examples are provided in section 7.1 to illustrate, how the rules provided in this section can be deployed to solve actual cabin design problems, as required by **working hypothesis 3**.

In section 6.3, two examples for system adaptations for modified architectures are considered in partial fulfillment of **working hypothesis 4**. The corresponding necessary changes to the baseline outside-in design system and the additional required rule sets are introduced. In the following, it must, however, still be shown that the updated rule sets can be used for design studies in a similar way as the baseline system. To this end, example studies are provided for both novel architectures and compared to the conventional reference design in section 7.2.

7. Application studies

In chapter 6 a collection of rules for modeling the geometry of aircraft fuselages and to determine the necessary design details including structure and cabin has been described. That said, the goal for implementing these rules is to provide an initial set of consistent configuration details for MDAO processes, as stated in the **research hypothesis**. The applicability and effectiveness of the rule sets to provide the necessary product data at various stages of the MDAO-driven design process is therefore demonstrated in the following in a series of application studies.

To begin with, the capability to create consistent fuselage structure and cabin layouts at the preliminary design stage in an outside-in or hybrid inside-out/outside-in sense is demonstrated in section 7.1, in order to confirm **working hypothesis 3**. First, a single aisle design is discussed in order to establish a reference case. In a second step, the versatility of the rule sets for conventional configurations is then showcased by applying the same methods to a twin-aisle and multi-deck configuration.

The extensibility of the design approach via rule sets to support unconventional architectures, as required by **working hypothesis 3**, is discussed in the following. Applying the additional rule sets described in 6.3.1 and 6.3.2 respectively, first an LH_2 tank integration study is performed on the baseline configuration before applying the cabin layout generation capabilities to a BWB configuration.

Finally, the capacity to produce consistent multi-disciplinary analysis models, as described by **working hypothesis 2**, is illustrated in section 7.3. The selection of the example cases is based on the established computational analysis and design methods outlined in section 4.2. On the one hand, the data generation for a GFEM model for fuselage structure sizing is discussed. On the other hand, a very detailed model for immersive cabin visualization is considered.

The findings in the different use cases are then discussed in section 7.4

7.1. Fuselage design synthesis

In **working hypothesis 3**, the need to include design functionality in order to determine component modeling parameters and enable the geometric modeling is stated. An overview of the rules implemented in the `fuga.design` package is given in section 5.1. The applicability of these rules is evaluated in this section by applying them to different use cases.

Even though the design and geometry generation are executed concurrently as required by the graph-based reasoner, the design synthesis can typically be considered the first step in a run of the KBE application. Consequently, a single-aisle baseline configuration is established first in section 7.1.1. This baseline will appear again as input in some of the subsequent studies. Furthermore, the synthesis problem is expanded to a configuration with multiple aisles and passenger decks in section 7.1.2.

7.1.1. Single-aisle baseline

For the single-aisle baseline a stretched variant of the D150 configuration is used. The D150 configuration, a replica of the Airbus A320 configuration, which was initially introduced in the VAMP project [ZCN12] and has since been used as a reference by many authors including Scherer et al. [SK+13] and Klimmek et al. [KS+19]. The stretched configuration is designed to carry 240 passengers and is thus referred to as the D240 configuration in the following. The starting point for the fuselage design synthesis is an output from the preliminary design synthesizer openAD (cf. table 2.1.1), provided in CPACS format. A similar configuration has been described e.g. by Wöhler, Walther, and Grimme [WWG22]. Since the configuration has already been used for illustration in previous chapters, the visualization of the initial CPACS geometry is already provided in figure 6.1.5. Aside from the OML and engine details shown in the figure, the data set also contains a basic wingbox definition and movables layout. As discussed in section 6.2.2, the wingbox in particular must be taken into account for the fuselage design.

7.1.1.1. Design inputs

The D240 configuration is meant to seat 240 passengers and perform short to mid-range missions with a design range around $2000nm$. The cabin design roughly corresponds to the high-density variant of the Airbus A321neo ACF (Airbus Cabin Flex) layout [Alc17]. Table 7.1.1 provides an overview of the relevant input variables.

Discipline	Parameter/data item	Symbol	Unit	Value
OAD	Number of passengers	n_{PAX}	-	240
	Design range	R	nm	2000
Cabin/cargo	Aisle width	w_{aisle}	in	19
	Exit layout	-	-	2×C+, 2×III, 1×C
	Class layout	-	-	30 eco-plus, 210 economy
	Seat pitch	$\Delta x_{seat,eco+}$	in	29 (eco-plus)
		$\Delta x_{seat,eco}$	in	28 (economy)
	Seat type	-	-	economy
	Number of galleys	$n_{galleys}$	-	2
	Number of lavatories	$n_{lavatories}$	-	2
	Cargo container type	-	-	LD3-45
	Number of containers	$n_{container}$	-	11
Structure	Frame pitch	$\Delta x_{fr,nom}$	in	21
	Number of stringers	$n_{stringer}$	-	87

Table 7.1.1.: List of inputs for the D240 configuration

The overall design is targeted on an efficient realization of the transportation task. As a result, some compromises in terms of comfort are acceptable, if seating capacity can be improved. Nevertheless, a two-class layout is adopted, which includes an eco-plus class with 30 seats installed at a slightly higher seat pitch compared to the the remaining seats in regular economy class. Irrespective of the class the same seat types are used. The aisle width is set to $w_{aisle} = 19in$. This is in accordance with the aisle width requirements of the CS-25 given in table A.3.2 as the arm rest, i.e. the widest part of the seat is located at a height below $25in$. Consequently, the $20in$ width limit for $z_{cab} > 25in$ is not violated.

In keeping with the low comfort requirements only two galleys and lavatories are included respectively, which are placed at both ends of the cabin. The number of lavatories can be translated to a lavatory utilization factor $UF_{lav} = \frac{n_{PAX}}{n_{lav}} = 120$, which is used as a further indicator for the cabin comfort. The number is high, compared to values from literature [Tor76; Gob15], where a the upper bound for utilization is commonly found to be around 60. However, the higher value is deemed acceptable, due to the overall design goal. That said, ongoing developments for a dual rear lavatory could help reduce the number to $UF_{lav} = 80$. For the galleys, the utilization factor is commonly computed based on the number of FSTs. However, with only the number of galleys given, the value cannot be computed until more details on the galleys are made available.

Due to the high number of passengers, the known exit layout from the A320 or A321 must be adapted by introducing an additional type C door behind the wing. Furthermore, the rating of the outermost exits is increased to C+, which requires an additional flight attendant as well as improved lighting and emergency slides [EAS14]. The changed rating results in a passenger capacity of 65 compared to the original 55. The double type III over-wing exit is retained. This results in a C+-III-III-C-C+ exit layout, which can be certified for up to 250 passengers. The mixed-integer exit optimization described in section 6.2.3.3 confirms that this is an optimal solution for seating 240 passengers in a single-aisle layout in terms of length penalty.

It is worth noting that it is also possible to replace the type C exit with an exit of type I. The change would have no effect on the cabin length, but decrease the exit capacity to 240 passengers. Whereas this might restrict further increase of the cabin density, it may present a weight saving opportunity, due to the smaller structural cutout size. Such considerations are, however, not taken into account in the optimization at this point.

A total of 11 LD3-45 cargo containers, which are used e.g. in the A320 family, are placed in the cargo floor. Taking into account the information from table A.3.3, this results in an overall cargo container volume $V_{cargo} = 40.80m^3$. The container volume utilization then amounts to $UF_{cargo} = \frac{n_{PAX}}{V_{cargo}} = 5.88\frac{PAX}{m^3}$. Again, this number appears to be very high compared e.g. to Torenbeek, who assumes a volume utilization of $1.695\frac{PAX}{m^3}$. However, it must be noted that V_{cargo} does not take into account carry-on luggage, which is the preferred option for many passengers nowadays, especially on short to mid-range flights.

For the structural design, a maximum frame pitch $\Delta x_{fr,nom} = 21in$ and a total of 87 stringers around the circumference are adopted.

7.1.1.2. Outside-in design

Based on these inputs, first an outside-in design is performed, using the corresponding basic design system shown in figure 5.2.1. A LOPA, generated from the design outputs in CPACS, is given in figure 6.1.31.

The illustration shows the floor layout as well as the cabin space and the fuselage outline. Key structural components are included as well. In this way, the design inputs provided in section 7.1.1.1 can be retraced. The two seating classes are visible, as are the cabin monuments and the exits.

Furthermore, some new details have been added automatically, such as the exit spaces, or the trolleys in the galleys. The determination of the trolleys is based on the galley dimensions and a standard trolley width $w_{FST} = 0.3m$. This information can be used to determine the trolley utilization, to further quantify the cabin comfort level. Figure 6.1.31 shows a total of 6 trolleys, resulting in a utilization $UF_{galleys} = \frac{n_{PAX}}{n_{FST}} = 40$. Once again, this reflects the overall low comfort level of the configuration.

Finally, the visualization allows for the evaluation of different space requirements from the CS-25, such as aisle width and evacuation spaces around the doors. As mentioned in section 6.1.3, these non-physical geometries can also be exchanged via CPACS, along with the cabin layout.

In figure 7.1.1, a section view of the layout at the longitudinal position $x = 12m$ is shown. The position lies in the constant section, away from any unusual components such as doors or the center wingbox, which makes it a suitable reference. Aside from the seats, which are also included in figure 6.1.31, the secondary structure including sidewall panels, OHSCs and ceiling panels is shown as well. Differently from the LOPA, more geometric detail is required for an expressive visualization, especially for concave components, such as the sidewall. Consequently, the mesh representations are plotted, which introduces a very high level of detail. This allows for the evaluation of the attachment points as well as the assessments of overlaps. In the case of figure 7.1.1, it can be concluded that all secondary structure components are properly connected and that no collisions with the seats occur.

Details of the primary structure are also contained in the figure, including stringers, floor beams and struts. Based on this information, the consistency between the structural design and the cabin design can be inspected. For instance, with the longitudinal floor beams shown, it can be seen that the legs of the seats align with the seat rails. For the cargo floors, it can be determined that no collision occurs between the LD3-45 containers and the components of the floor structure and that sufficient floor width is available to fit the base area of the container. Since the stringers are plotted as well, it can furthermore be assessed, whether the gap in the stringers due to the window row aligns with the window funnel of the sidewall model.

For further assessment of the links between the structure and the cabin, a more detailed 3D geometry model can be generated, which can be used to derive a cut side view as shown in figure 7.1.2. Once

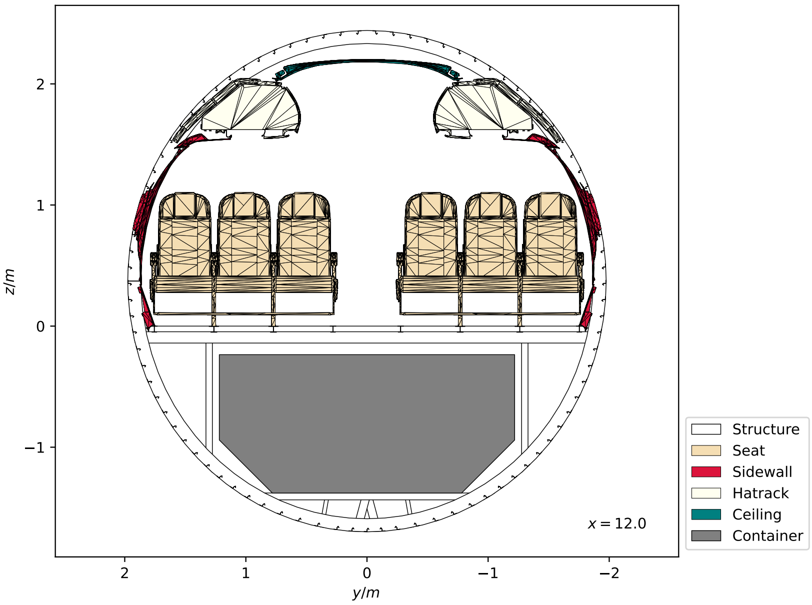


Figure 7.1.1.: Section view at $x = 12m$

again, the components of the secondary cabin structure are shown as well as the floor-based components and the containers. In addition, the door openings are visible with the exception of the emergency exits, which are covered by the sidewall panels. This is by design, as the regular sidewall panel distribution pattern is not typically interrupted for emergency exits. Instead, custom-made sidewall panel components with appropriate cutouts are used, which are, however not modeled in FUGA. On the structural side, the frame and stringer distributions are shown as well as the floor structure. Moreover, additional details are provided including the bulkheads and the wing boxes, but also the center fuselage area.

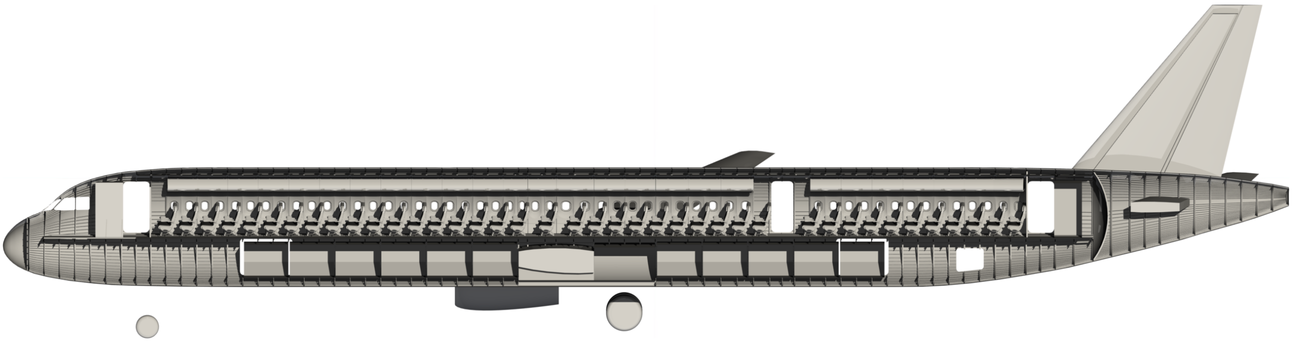


Figure 7.1.2.: Cut side view of the detailed D240 geometry model

The side view further highlights the interconnections between the structure distribution, particularly the frames, with the cabin layout. To begin with, all the cabin door openings are bracketed by a pair of frames, as are the wing boxes. The bulkheads also lie at frame positions. The stringers run parallel to the floor in the full cabin region from the first door to the rear bulkhead. In addition, all the pictured components of the secondary structure are aligned with the frame layout, in correspondence with the “no gap” positioning strategy described in figure 6.2.20. This strategy is well-suited e.g. for subsequent interactive visualization.

In contrast, the cargo floor openings are not aligned with the overall structure. They are, however, aligned with the containers of the container distribution, which in turn respects the boundaries of the center wing box and the landing gear bay.

However, figure 7.1.2 also reveals the reliance of the design, especially of the secondary structure, on the availability of suitable component models. In regular seating areas, the correct component models for the sidewall panels and the OHSCs are available and thus added to the layout. The same is not the case in the neighborhood of the doors, where, as a result, no secondary structure can be placed and the underlying stringers and frames are revealed instead. Unlike the floor monuments, which can easily be replaced by boxes in CAD, to provide at least a basic representation of the component, the more complex geometry of the secondary structure requires more detailed CAD-based component descriptions based on more complex sets of parameters. Whereas a fundamental approach to address this problem using the graph-based methodology is proposed for the example of a seat in section 6.2.3.4, it is not pursued further for the secondary structures within this thesis.

Aside from this, there are further issues to be found with the design based on figure 6.1.31. For instance, the seats in the last row have a decreased width, compared to the other rows. This is necessary due to the reduced fuselage cross section area in the tail section of the fuselage, which results in a reduced cabin space. FUGA provides rules to detect the seats for collisions with the cabin space, including a detailed mesh-based implementation based on the VTK library. The result for a cabin layout without the slimmer last row is given in figure 7.1.3. As discussed in section 6.2.3, it is currently not possible to have FUGA update the design automatically, based on this information. However, a warning can be issued to the user, who can then make the necessary changes in the design inputs.

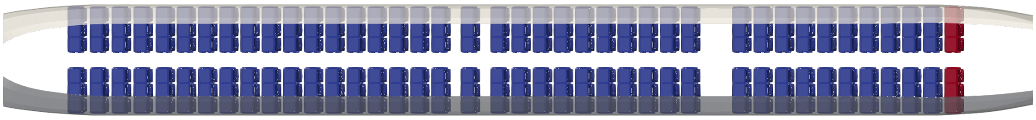


Figure 7.1.3.: Mesh-based detection of collisions between seats and cabin geometry

Another problem is the gap in longitudinal direction between the RPB and the rear floor monuments. This effectively means that the available cabin space is not utilized completely. The problem results from two main factors: On the one hand, due to a rule from the cabin design rule set, the rear end of the cabin space is determined by the RPB, which in turn, according to the structural design ruleset, depends on the wingbox of the VTP, as outlined in section 6.2.2.5. Since the wingbox boundaries are already provided by openAD, the RPB position is thus determined before the FUGA design process even begins. On the other hand, the cabin is assembled from front to rear without introducing any unnecessary gaps. No additional degree of freedom is available to make use of the additional available space.

It follows that the problem can be resolved by providing an additional degree of freedom either to the cabin design, or to the fuselage OML design. The latter approach changes the type of problem from an outside-in to a hybrid design problem, which requires an additional rule set as described in section 6.2.1.2. The hybrid design problem is discussed in section 7.1.1.3.

Consequently, if the fuselage outline is not to be touched in an outside-in design, an additional cabin parameter must be introduced to provide the required degree of freedom. Ideally, the additional space should be used to increase the payload. However, the length of the gap $\Delta l_{cabin} = 0.3916m$ is smaller than the seat pitch, which prohibits the addition of an extra row of seats. Instead, a global factor on the seat pitch $f_{pitch,glob}$ is selected to utilize the space to the benefit of the passenger comfort. That said, the outcome is unlikely to to be put into practice, due to the limited number of discrete options to connect a seat to a seatrail in modern aircraft. Nevertheless, it provides a simple academic example to explore the capabilities of the FUGA modeling approach.

In order to formulate the design problem, a rule to compute the difference between the available and the utilized cabin space Δl_{cabin} is provided by FUGA. The structural design rules and the geometric modeling rules from the `fuga.geometry` package are employed to determine the rear bulkhead position and the maximum x-coordinate of all floor element boundaries respectively. Furthermore, the additional input parameter $f_{pitch,glob}$ is introduced to the cabin design system and applied to the all specified class seat pitches during the design.

This enables the formulation of the functional relationship $\Delta l_{cabin}\left(f_{pitch,glob}\right)$ using FUGA, by extracting the subgraph between the two nodes from the MCG. Aside from the two nodes themselves, it contains the intersection of the ancestors of the node representing Δl_{cabin} and the descendants of $f_{pitch,glob}$. These are the dependent variables, which must be re-evaluated for each function call. All other ancestors of Δl_{cabin} must be computed only once and do not change if $f_{pitch,glob}$ is changed.

In a practical implementation, this can be accomplished by solving the system once for Δl_{cabin} using a default pitch factor e.g. $f_{pitch,glob} = 1$. Then $f_{pitch,glob}$ is updated, discarding all of its descendants. In this way, all independent variables are set. A query for Δl_{cabin} will then automatically assemble the necessary subgraph and recompute all dependent variables. The above highlights an important benefit of the graph-based formulation of the KBE system: By extracting the subgraph, only those rules are reevaluated, which are absolutely necessary for the solution of the problem at hand. All other data is retained, thus avoiding unnecessary computational overhead, which would typically occur in a global iteration loop used in tools like PrADO.

Based on this functional relationship, the problem of removing the gap can be approached numerically, as described in section 5.1.3.3, by formulating the scalar root finding problem:

$$\Delta l_{cabin}\left(f_{pitch,glob}\right) = 0. \tag{7.1.1}$$

This type of problem can be solved using e.g. the secant method, which offers the benefits of gradient descent techniques such as Newton's method without the need to provide derivatives of the function. An implementation is provided e.g. by the SciPy package [VG+20].

As shown in figure 7.1.4, the solution of the above problem can also be expressed in XDSM notation, highlighting the close relationship between the KBE-driven design problem and a Gauss-Seidel MDA problem as presented in section 2.2 (cf. figure 2.2.2). The previously computed independent variables $\mathbf{x}_j$ are provided to the system from the beginning and remain unchanged. Meanwhile, the dependent variables $\mathbf{y}_{i,j}$ are updated for each iteration. Departing from the original specification of XDSM by Lambe and Martins [LM12], the stacked analysis block is used not to represent analysis blocks, which are executed in parallel, but simply the evaluation of a set of rules using FUGA. Similarly to Pate, Gray, and German [PGG13] and Gent [Gen19a], the sequence of the rules as well as their connections via the dependent variables $\mathbf{y}_{i,j}$ in the upper triangular are handled in detail by the graph-based reasoner, which reduces the complexity of the problem formulation considerably. The output of the secant method is then the root of the function, i.e. the value $f^*_{pitch,glob}$, where equation 7.1.1 holds.

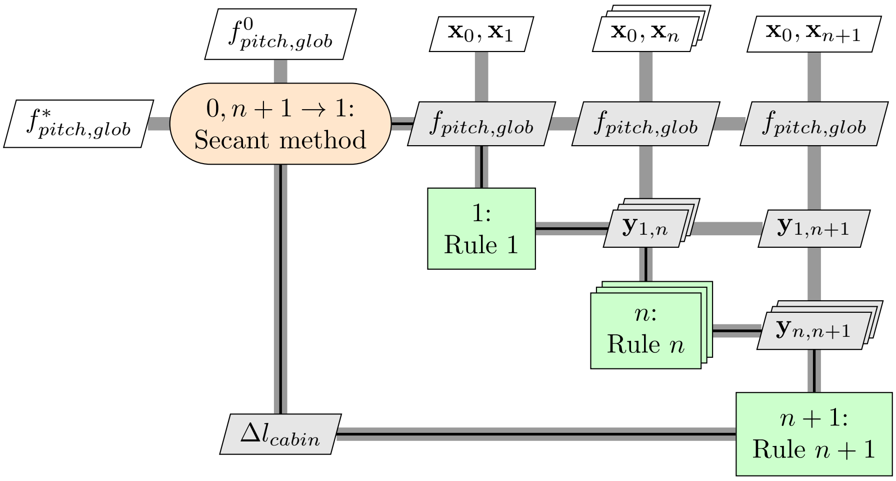


Figure 7.1.4.: XDSM representation of the solution to the root finding problem

Solving equation 7.1.1 for the D240 configuration yields the updated layout shown in figure 7.1.5. While the 2D plot for the updated result suggests that the second to last row has become a risk for clash with the cabin bounds, 3D collision analysis reveals that the design is feasible. Visibly, the gap between the rear bulkhead and the monuments behind the last exit has disappeared. The secant method shows that a global seat pitch factor $f^*_{pitch,glob} = 1.0124$ is required to achieve this. The result corresponds to a seat pitch increase of slightly more than 1% and a gain in leg room per row in the range of $0.01m$.

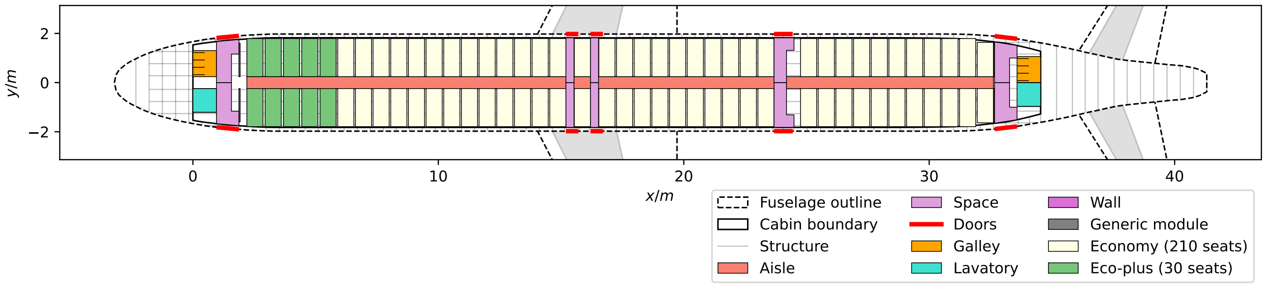


Figure 7.1.5.: Updated layout of the D240 for $f_{pitch,glob} = 1.0124$

The assessment of the effects of these results in terms of passenger comfort is the objective of subsequent human factors evaluations using e.g. the human model approach. To this end, both designs could be shared with disciplinary experts in a collaborative workflow, either as CPACS files or in the form of 3D models, as discussed further in section 7.3.2.

7.1.1.3. Hybrid design

Whereas an adaptation of the seat pitch may be an interesting option for true outside-in use cases, such as retrofitting scenarios, the choice of pitch in a preliminary design context is typically informed by experience from airliners and earlier designs and can thus be considered a hard requirement. Consequently, instead of increasing the seat pitch it is usually more interesting to reduce the length of the fuselage, in order to reduce the mass of the overall design and improve the efficiency.

To this end the basic outside-in design system is extended using the hybrid loft design rule set described in section 6.2.1.2, as mentioned in section 7.1.1.2. The new rule set adds the possibility to manipulate the length of the constant section of the fuselage using a length change parameter $\Delta l_{fuselage}$. As a result, the root finding problem formulated in equation 7.1.1 can be updated to

$$\Delta l_{cabin}\left(\Delta l_{fuselage}\right) = 0. \tag{7.1.2}$$

The solution to this problem is typically trivial, i.e. $\Delta l_{fuselage} = -\Delta l_{cabin}\left(0.\right)$, which implies that a change in fuselage length results in an equivalent change in cabin length. This is a reasonable assumption, especially for small displacements, since the cabin is not moved sufficiently deep into the tail section to effect any substantial changes to the layout.

However, a new challenge, which arises from the introduction of the loft design system, is the presence of cycles in the MCG of the design system. The cycles appear as a result of the necessary updates of the fuselage geometry. Updates inherently use and change available data, resulting in system nodes, which eventually reference themselves. This poses a problem to the topological sorting algorithm used in the graph-based approach approach, since it only works on a DAG. That said, in principle, cycles in the MCG do not preclude the system solution, unless they are propagated to the FPG, in which case the determination of the PSG using the topological sorting algorithm will fail. Consequently, it is possible to perform the analysis if the system is queried in several steps in such a way that the cycles are broken up, i.e. have clearly defined starting and end points.

As an example, the fuselage positioning node is considered, which contains the distances between the fuselage profiles that must be modified in order to adjust the fuselage profile positions for the length update. To resolve the cyclic connectivity graph, the following steps can be taken, which are shown in figure 7.1.6:

- *Step 0:* During the initialization of the system, the positioning values are read from the CPACS input file and stored in the data repository and stored in the node `/cpacs/vehicles/{aircraft}/model/fuselages/fuselage/positionings/positioning`.
- *Step 1:* A query is made for a the node `/fuga/design/initialFuselagePositionings`, resulting in the creation of a copy of the contents of the positioning node in a separate node.
- *Step 2:* An update for the value of the node `/cpacs/vehicles/{aircraft}/model/fuselages/fuselage/positionings/positioning` is computed, based on the copy of the original positionings stored in `/fuga/design/initialFuselagePositionings` and the prescribed fuselage length change $\Delta l_{fuselage}$, which is provided externally via the `/fuga/design/fuselageDeltaX` node. As an intermediate step, the function for the local length change at each position x due to $\Delta l_{fuselage}$ must be determined. As described in section 6.2.1.2, this computation is based on the boundaries of the constant section of the fuselage. In order to circumvent the caching mechanism and re-compute the updated contents of the positioning node, its original contents must be discarded before performing this step.
- *Step 3:* A query for a downstream node, e.g. `/fuga/geometry/fuselageSurfaceLoft`, can be used to retrieve any further desired system outputs based on the updated fuselage data.

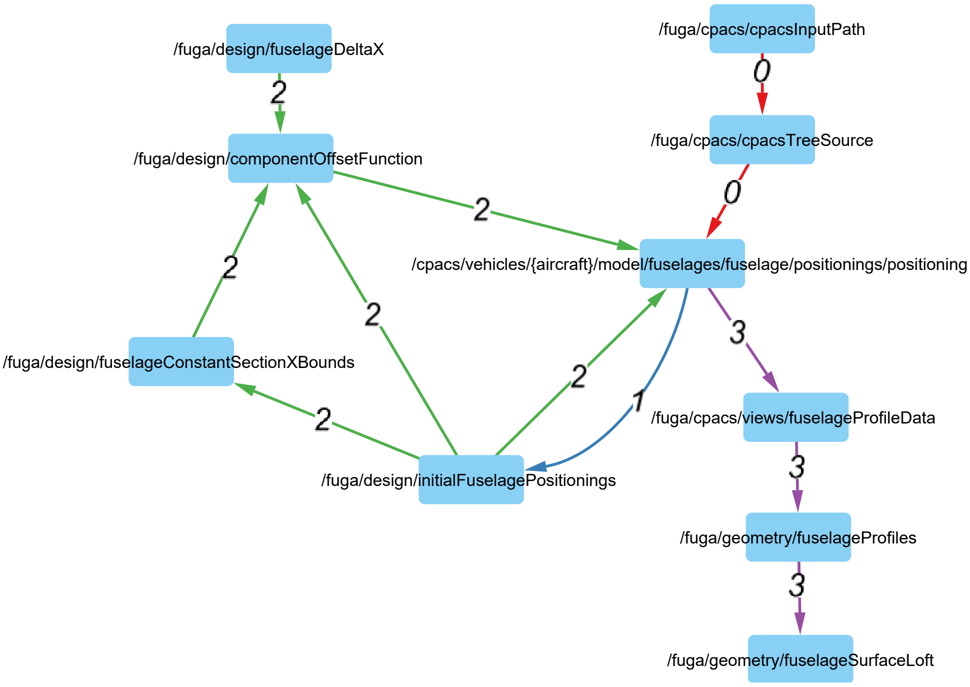

/fuga/cpacs/cpacsInputPath
/fuga/design/fuselageDeltaX
0
2
/fuga/cpacs/cpacsTreeSource
/fuga/design/componentOffsetFunction
0
2
/cpacs/vehicles/{aircraft}/model/fuselages/fuselage/positionings/positioning
2
2
3
/fuga/design/fuselageConstantSectionXBounds
2
1
/fuga/cpacs/views/fuselageProfileData
2
/fuga/design/initialFuselagePositionings
3
/fuga/geometry/fuselageProfiles
3
/fuga/geometry/fuselageSurfaceLoft

Figure 7.1.6.: Solution steps for the cyclic fuselage update system

Based on the figure, it can be determined that there are in fact three cycles in the graph, which are however resolved simultaneously, by exploiting shared edges.The traversal of the cycle is implemented by formulating one query (*step1*), which leads into the cycle from a known starting node. A second query (*step 2*) then leads out of the cycle again and back to the starting node. In this way, the correct updated profile positionings based on the given $\Delta l_{fuselage}$ can be determined.

For the iterative solution of equation 7.1.2 $\Delta l_{fuselage}$ is updated by modifying the contents of the `/fuga/design/fuselageDeltaX` node. Since the connections of *steps 0* and *1* do not lie on the path between this node and the final downstream node `/fuga/geometry/fuselageSurfaceLoft`, only *steps 2* and *3*, including the discarding of the old positioning values, must be repeated during the iteration.

Applying the approach to the D240 confirms the above supposition that $\Delta l_{fuselage} = -\Delta l_{cabin}$ (0.), with a computed necessary change in fuselage length $\Delta l_{fuselage} = -0.3916m$. Due to the relatively small length change compared to the overall fuselage length, the resulting layout in figure 7.1.7 does not differ substantially from the result with the adapted seat pitch in figure 7.1.5. Once again, the gap disappears and the 3D collision analysis shows that the second to last row does not collide with the cabin bound.

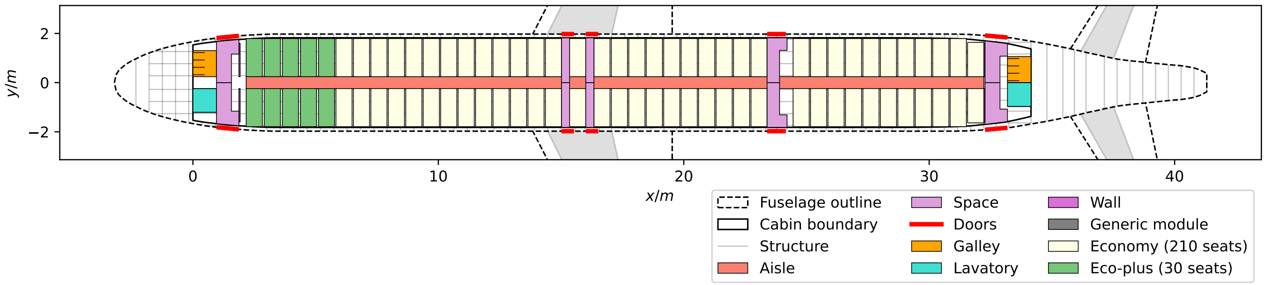


Figure 7.1.7.: Updated layout of the D240 with $\Delta l_{fuselage} = -0.3916m$

Meanwhile, the shortened fuselage holds the promise of gains in efficiency and performance of the configuration. Similarly to the previous outside-in design example, where a downstream human factors analysis is necessary to assess the impact of the changed seat pitch, the fuselage length change determined here can be communicated back to the OAD synthesizer. Here, the overall design can be updated and the impact on the overall configuration including snowball effects can be assessed. This step is advisable in real-life design campaigns, as it also ensures the consistency of the overall design, taking into account factors, which have explicitly been neglected in the outer geometry update described in section 6.2.1.2. These factors include the wing positioning and empennage scaling, but also aspects like landing gear placement and rotation capability during takeoff and landing. Once again, techniques from collaborative MDAO can be leveraged to set up an iteration loop and automate the exchange.

Another key finding of the hybrid design example is that good system insight is necessary to determine the evaluation steps to resolve cyclic systems as shown in figure 7.1.6. This step can currently not be performed automatically, which means the end user must build a script to prescribe the correct sequence of queries. In this context, the explanation subsystem features described in section 5.1.4 as well as the possibility to interact with the system via Jupyter Notebooks have proven to be valuable support tools. Nevertheless, as system complexity grows with more and more rules being included, the proper documentation of the available knowledge will become essential to avoid an insurmountably steep learning curve for new users.

7.1.1.4. Knowledge graph exploration analysis

To further investigate the benefit of the KBE methodology for the efficient execution of the design task, the exploration of the knowledge graph, i.e. an analysis which rules were executed during the process, is provided for the outside-in design case without seat pitch modification in figure 7.1.8. The layout and node locations of the graph correspond to the MCG for the outside-in system given in figure 5.2.2. This means that the nodes are once again arranged according to their topological layers starting from the top. The ellipses to represent the rule set regions are also retained. The coloring of the nodes is adapted to reflect the various execution stages of the system. Nodes marked in white have not been evaluated during the design synthesis, since they do not lie on the path to any of the the queried nodes.

The execution stages reflect the requests made to the system by the user and are listed in table 7.1.2. Visibly, the solution of the present design problem requires only two steps. First, the system is initialized upon instantiation. This step entails the readout of the incoming CPACS data set and the addition of the corresponding entries to the data repository. As illustrated by figure 5.1.1 initialization is always the first step preformed. The corresponding rules are thus found predominantly in the

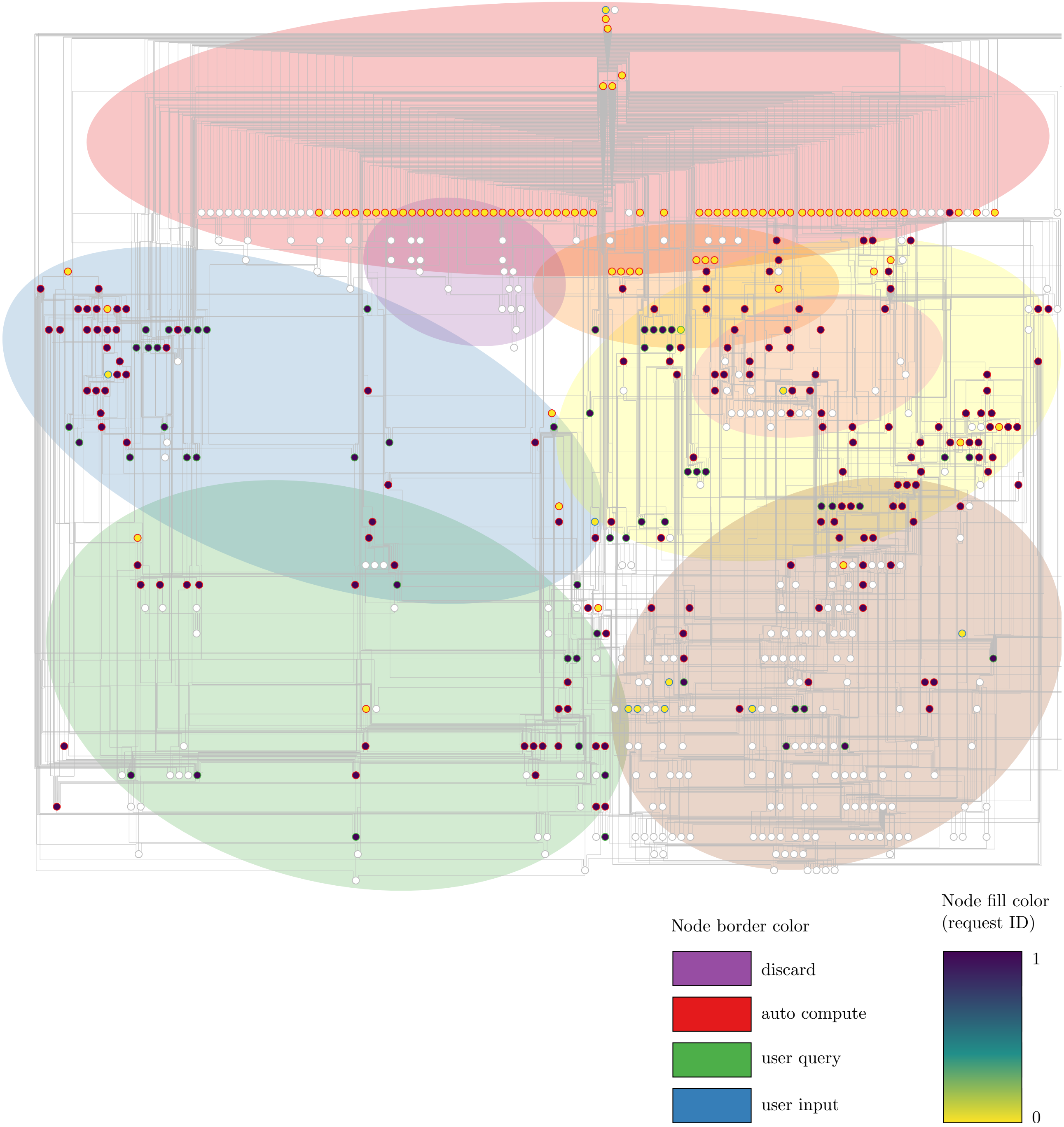


Figure 7.1.8.: Knowledge graph execution chart for design synthesis

topmost topological layers, as these entries provide the necessary inputs for the subsequent steps, i.e. the design synthesis in this example. In addition, some further user-defined control parameters can be found scattered throughout the system which are also set during the initialization.

Table 7.1.2.: System requests and performance data for design problem

Request ID i	**Description**	**Rule count**	**Timing** t_i/s	**Cumulative timing** $\sum_0^i t_i/s$
0	Initialize	120	0.46	0.46
1	Compute design rule set sinks	208	15.33	15.79

The synthesis is then executed by making a query for the union of the sinks of both the structure and cabin design rule set graphs $V_{des,str}$ and $V_{des,cab}$, i.e. $V_{query} = \{v \in V_{des,str} \cup V_{des,cab} | \delta^+ (v) = 1\}$. The specific queries are also reflected in figure 7.1.8 by the coloring of the node borders. Whereas nodes that have been requested explicitly by the user are marked in green, intermediate nodes, which have been called by the reasoner, are marked in red. Nodes, which have been set before starting the query, e.g. the CPACS input path, are marked in blue. It can be seen that all automatically determined nodes are eventually followed by a requested node in a subsequent topological layer. This illustrates, how it can be assured that all rules in the set are evaluated, thus providing all possible parameters, by querying all the sinks of a rule set.

Visibly the nodes visited during the synthesis step are located mostly in the middle third of the graph. In contrast, none of the nodes in the last three topological layers have been visited. This is comprehensible since, as shown by figure 5.2.2, the entries in these layers are all associated with the subsequent model generation tasks, which have not been requested in this example.

Based on the profiling data, which is also provided in table 7.1.2, it can be seen that almost twice as many rules were evaluated during the synthesis step compared to the initialization step. Conversely, the execution time for the synthesis is approximately 30 times higher than for the initialization. This can be explained, because, on the one hand, the run time of the initialization is mainly determined by the file input operation of reading the CPACS XML file from disk. On the other hand, the design rule set requires some computationally expensive geometric operations, e.g. the determination of the frame curves on the fuselage, which are necessary to determine the cabin space, as discussed in section 6.2.3.2. Furthermore, hard drive access once again has an impact, as the available external component models must be read to determine e.g. the bounding box dimensions. Nevertheless, the cumulative run time, which is in the order of tens of seconds, is sufficiently low to allow for quick creation and exploration of different design configurations, either in an interactive environment, or in small automated parameter studies.

7.1.2. Twin-aisle and multi-deck configuration

Building upon the capabilities demonstrated for the reference configuration, the more complex application case of the D380 configuration is considered in this section. Based on the Airbus A380 design, the D380 contains two passenger decks, each with a twin aisle cabin. The configuration has been selected previously by Walther et al. [WH+22a] to highlight the versatile modeling capabilities of the updated CPACS cabin description introduced in version 3.4. Expanding on those findings, the goal of this chapter is to investigate the applicability of the conventional outside-in rule set to different types of configurations and thus determine the extent of the design space, which can be supported. This also includes novel cabin concepts. Once again, the starting point for the design is a OAD synthesis using openAD, as shown in figure 7.1.9, provided as a CPACS data set.

7.1.2.1. Cabin layout

In contrast to the single aisle configuration discussed previously, which is focused on high density, the cabin layout for the D380 is derived from a configuration presented by Rudolph et al. [RR+21],

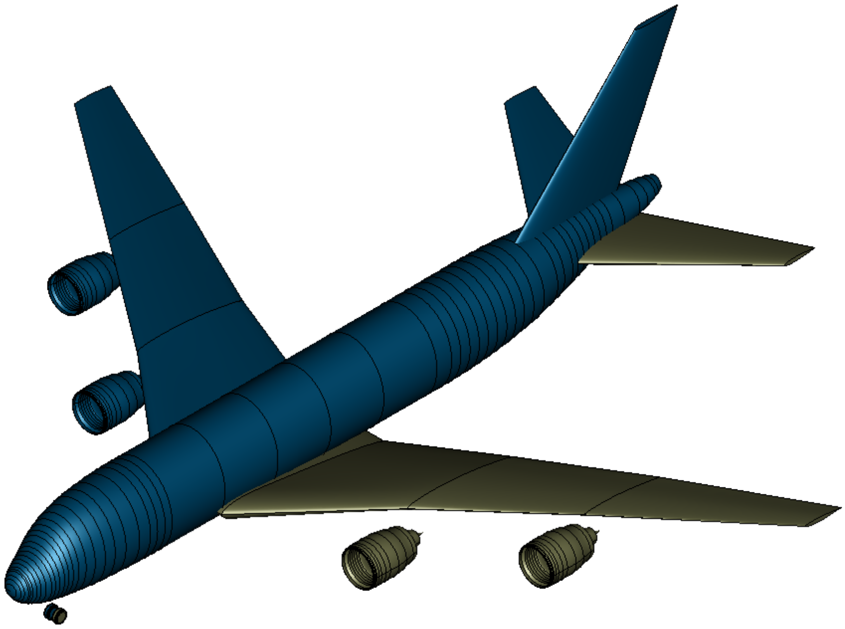
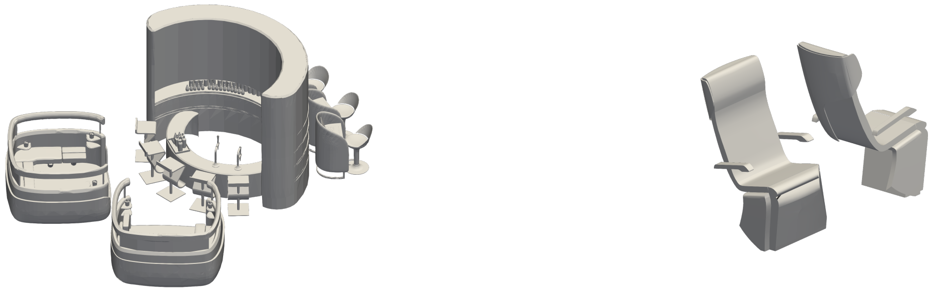


Figure 7.1.9.: D380 geometry as provided by openAD visualized in TiGL

where the design is predominantly concerned with meeting passenger requirements identified via a design thinking approach [RR+20]. This is achieved through both a more spatial and varied cabin arrangement as well as the introduction of new types of cabin components. Key features of the cabin include a central entrance area with a welcome desk and two sets of stairs leading to the second deck on either side. A bar area is placed in the center region of the second deck between the stairs. The corresponding component model is shown in figure 7.1.10a. To appeal to individual travelers, a block of so-called "zig-zag" seats installed at a 45° angle in alternating direction, as shown in figure 7.1.10b, is furthermore placed in economy class.



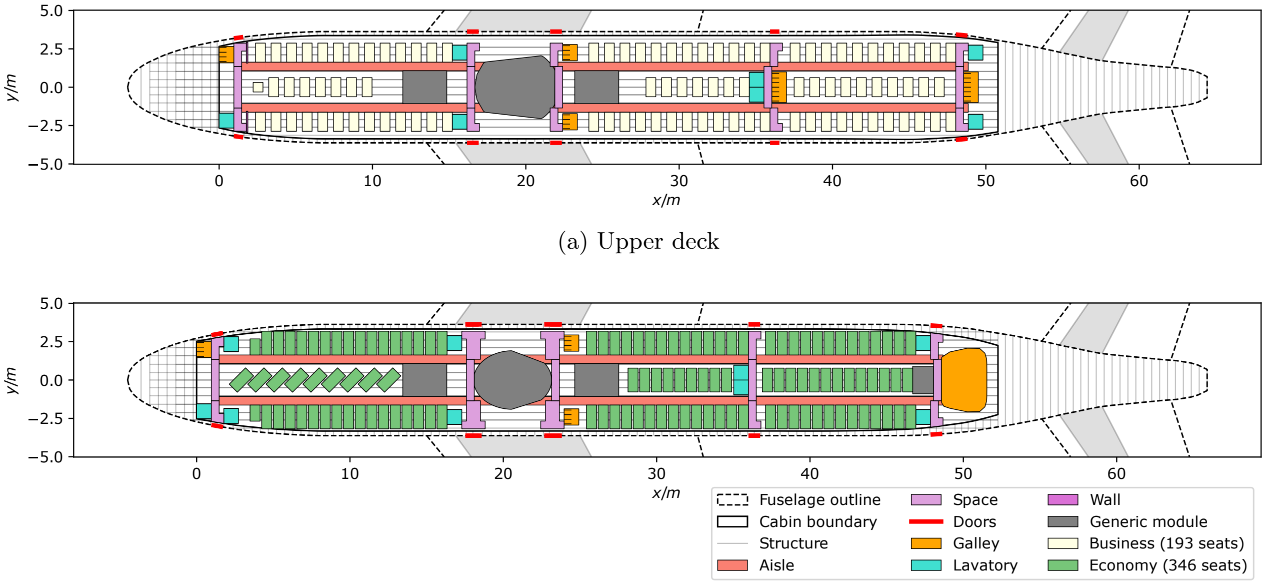
(a) Bar area model

(b) Zig-zag seat model

Figure 7.1.10.: Novel cabin component models [RR+21]

In order to meet the ensuing space requirements, Rudolph et al. propose a newly designed aircraft configuration, built from scratch around the cabin. The configuration design is, however, not substantiated by any kind of preliminary design synthesis or structural layout. In order to enable the assessment of the economical and ecological footprint of the comfort oriented cabin configuration using available design and analysis tools, a replica of the configuration is therefore created based on the D380 configuration using FUGA.

The resulting LOPAs for the main deck and the upper deck are given in figure 7.1.11. The upper deck is reserved to the business class, whereas the economy class is found in the main deck. Following the original design from Rudolph et al., a twin-aisle layout with three blocks of three-abreast seat modules at a seat pitch $\Delta x_{seat,eco} = 30in$ is adopted for most of the economy class, with only the block of “zig-zag” seats forming an exception. In the business class in the upper deck, a layout of three double seats with a pitch $\Delta x_{seat,biz} = 60in$ is adopted.



(a) Upper deck

(b) Main deck

Figure 7.1.11.: LOPA of the adapted innovative cabin for the D380 configuration

The cabin comfort indicators are given in table 7.1.3, both for the individual decks and for the overall configuration. The computation can be performed analogously to the single-aisle example. A minor difference is the presence of a detailed component model for the main deck rear galley. Evaluating the model reveals that it can hold 21 FSTs, which must be taken into account in addition to the other FSTs visible in the LOPA in figure 7.1.11, which are determined based on the same assumptions as for the reference configuration. With the upper deck holding the business class, unsurprisingly, the level of comfort is significantly higher than for the main deck. Nevertheless, the comfort indicator values for the main deck, too, clearly exceed those for the single aisle configuration. This is expected for a long range configuration, such as the D380, and even more so for a design driven by passenger requirements. In fact, the values in table 7.1.3 correspond much better to the reference values from literature. The only exception is the cargo volume utilization, which is still very high. Once again, only the volume available in the 13 LD6 cargo containers is considered. For comparison, the original Airbus A380 can carry 32 LD3 containers [ANA22], which can approximately be understood as half of an LD6 container (s. also table A.3.3). This means that 6 LD3 containers are missing in the present layout, which are placed in the middle of the fuselage in the region of the landing gear bay cutout in the A380 [ANA22]. This type of design is, however, foreseen by neither the design rules for the center fuselage area nor the underlying CPACS definition and can thus not be generated without profound modification of the rules sets.

The chosen exit layouts of C-A-A-C-C for the main deck and I-C-C-I-C in the upper deck both provide sufficient capacity for the given number of passengers. The corresponding free spaces are shown in figure 7.1.11 as well. In addition to the passageways including assist spaces, which are also included in the single-aisle layout, CS-25 furthermore requires cross-aisles for twin-aisle cabin layouts. Depending on the type of exit, the cross-aisles must either overlap with the passageway of the adjacent exit or be situated in close vicinity thereof. In FUGA, the vague latter formulation is interpreted with a maximum allowable distance criterion of $\Delta x_{exit,CA} \leq 1m$. However, for the majority of the exits in the D380 configuration, the overlap rule applies, with the type I exits in the upper deck providing the exception.

Table 7.1.3.: Cabin comfort indicators for the D380 configuration

Deck	upper	main	total
n_{PAX}	193	346	539
n_{lav}	7	8	15
n_{FST}	21	30	51
$n_{container}$			13
$UF_{lav}/\frac{PAX}{lavatory}$	27.57	43.25	35.93
$UF_{galley}/\frac{PAX}{FST}$	9.19	11.53	10.57
$UF_{cargo}/\frac{PAX}{m^3}$			4.63

When creating the replica layout, it needs to be taken into account that the D380 configuration is designed to be boarded primarily via the lower deck. This introduces a fundamental difference to the layout shown by Rudolph et al., who place the welcome area in the upper deck. Consequently, the decks from the reference are swapped in the D380 configuration, i.e. the main entrance area is placed in the lower deck with stairs leading to the business class and bar area in the upper deck.

Both the bar and the welcome area models as well as the stairs, are represented by "generic" modules. In the CPACS cabin description, this type makes it possible to reference external models of nondescript type, which have no additional particular properties associated to them [WH+22a]. In the LOPA plot, these models can then be represented by their convex boundary polygon instead of their bounding box, in order to convey a more accurate impression of the component.

The above shows, how the design rules for the floor layout can successfully be translated to the twin aisle configuration. The same is not the case for the secondary structure as illustrated by figure 7.1.12. Here, it becomes apparent that the component models of the sidewall panels in particular have been designed specifically to fit the cross section of the single-aisle fuselage. As a result, the shapes do not align with the more elliptic cross-section, especially for the narrower upper floor, revealing the limitations of the approach using "dead" geometry and highlighting the need for a more versatile parametric component description. Nevertheless, it is also apparent that the more complex secondary structure arrangement with the additional row of OHSCs in the middle can be represented in principle.

The presence of multiple passenger floors entails some additional complications in the design process. Initially, the cabin design is performed independently for each deck using two separate design systems in FUGA. However, it is necessary to subsequently ensure the compatibility of the designs. This becomes especially apparent when considering the component representing the stairs, which obviously needs to be at the same position in both layouts. This can be accomplished by synchronizing the positional data between the two systems, which must be implemented by the design engineer.

7.1.2.2. Structural design

A major problem for the structural design resulting from the separated design approach for the passenger decks is a lack of completeness in the cabin information in either of the two systems, which results in faulty structural layouts. For instance, each system only considers the exits of the respective deck, which means the exits from the other deck are not accounted for e.g. when determining the mainframe positions. This error proliferates as more structural details, e.g. window positions, are determined, which leads to two separate and incompatible structural layouts. The stringer distributions are affected in a similar way. On the one hand, both rows of windows must be considered to determine the stringer seed points as described in section 6.2.2.3 to avoid stringers passing through windows. Furthermore, the vertical extent where parallel stringer planes are placed in the cabin region when including the cabin constraints into the stringer distribution must be expanded to include both decks.

In order to resolve this issue, a number of key parameters provided by the cabin design, which are subsequently used in the structural design, must be manipulated before proceeding. As with

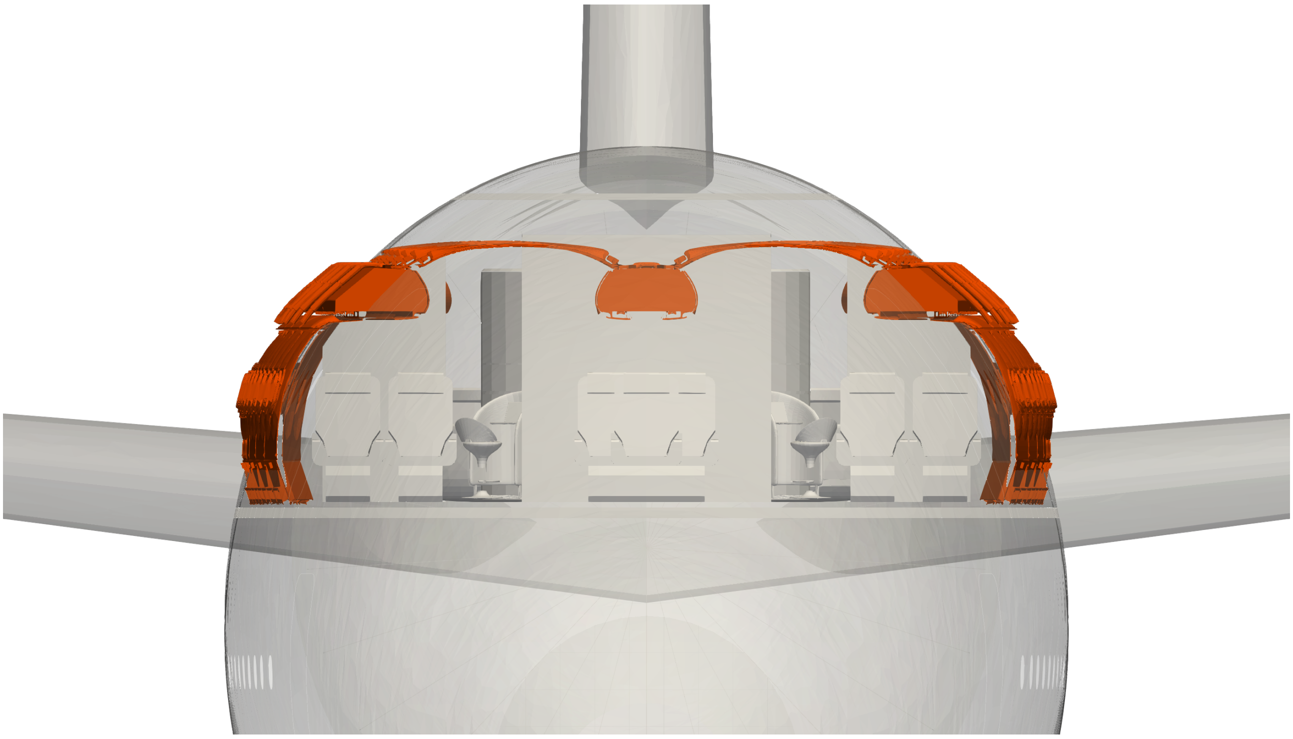


Figure 7.1.12.: Mismatch of upper floor secondary structure and OML

the cabin layout, this requires some intermittent synchronization steps, which are implemented by partially evaluating the systems for both decks, merging the results and updating the system values. It follows from the above that the mainframe positions and the stringer seed points on the reference section are among the most critical inputs for the structural layout generation. Since the mainframe positions are defined by the passenger door positions, among other things, the corresponding cutout definitions are first evaluated for both the upper deck and the main deck. Then, the definitions are combined and fed back to both systems. Based on the combined door cutout data the mainframe positions can then be computed consistently based on the complete available door data. To proceed with the computation of further deck details e.g. the window positioning, the cutout data should be reset, to consider these details separately for each deck. Consistency is maintained nevertheless via the common frame distribution. The cutout data must be merged again before building the geometry model.

For the determination of the stringer seed points, the vertical positions of both passenger floors must be considered. Furthermore, the vertical position and height of both window rows must be known for the removal. In addition, the z-range for the distribution of parallel planes is given by the position of the main deck $z_{floor,main}$ and the cabin height $h_{cabin} = z_{floor,upper} - z_{floor,main} + h_{cab,upper}$ for both decks. The longitudinal extent of the parallel stringer planes is chosen to be from the most forward door to the rear bulkhead.

The resulting basic structural layout for the D380 configuration is shown in figure 7.1.13. It shows both passenger floors as well as the cargo floor with the corresponding structural openings. A fourth set of crossbeams is placed above the upper deck. Whereas the lower two floors have downward facing support struts, the struts of the uppermost crossbeams point in upward direction. The upper deck floor is not supported by any struts as to not diminish the space for the main deck cabin. The frame distribution visibly follows the door cutouts of both decks very well. One exception is the second exit, where the mainframe distance tolerance criterion is triggered. Whereas the cargo doors are again not considered in the frame distribution, their size has been updated to reflect the change from LD3-45 to LD6 type containers (s. table A.3.3). The window cutouts of both floors follow the frame distribution exactly. The window positions are in turn respected by the stringer distribution, as a stringer is left out for each window row. Visible kinks in the stingers at the mainframe before the first main deck exit and at the RPB also show that the parallel stringer planes have been placed as required.

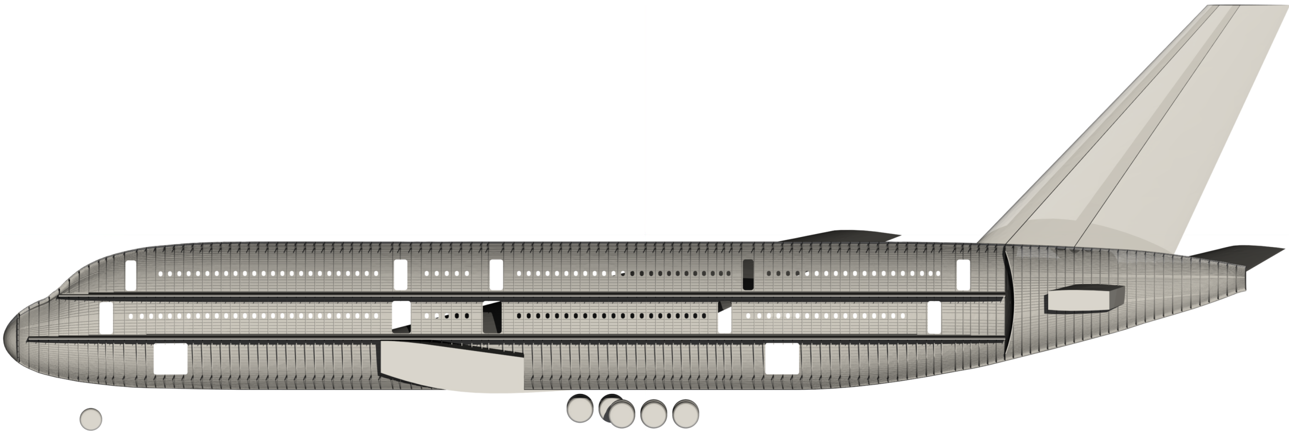


Figure 7.1.13.: Structural layout for the D380 configuration

In contrast, it can also be observed that some aspects are missing compared to the structural model of the single-aisle baseline. To begin with, no detailed center fuselage area is shown. This is due to the much higher angle of attack of the wing root, which causes the center wing box to transgress the lower boundary of the fuselage. This case is not yet supported by the geometric construction rules for the center fuselage described in section 6.1.2.3. Nevertheless, the principal information relating to the center fuselage area is available in the system, e.g. the length of the landing gear bay, which is an important input for the cargo container distribution.

Another difference is the absence of cockpit window and bulk cargo door cutouts. Since these types of cutouts cannot be communicated via CPACS at this point, they have been created manually for the baseline configuration but are not part of the standard design process.

Finally, the structural layout does not take into account the mechanical interconnection of the two floors e.g. via stairs, which are modeled as cabin components only. No floor cutouts or structural connections are provided by the model, which are, however essential to a reliable structural assessment. Thus, additional rules are required to improve the level of detail of the model in this respect. A simple implementation for the cutouts in the upper floor, where the bounding box of the stair component is simply subtracted from the floor model using Boolean geometry operations, is used in the following to provide an enhanced version of the structural model.

7.1.2.3. Full model

Based on this enhanced structural model, the full 3D model of the D380 configuration can be assembled by combining the structural model and the deck component models. The result is shown in figure 7.1.14. The central entrance area with the main deck welcome desk and the upper deck bar area is represented in detail due to the inclusion of external models. The stairs leading to the upper deck are shown as well and align with the floor cutouts. Furthermore, the "zig-zag" seats are visible in the forward section of the main deck, whereas the detailed galley model is found in the rear.

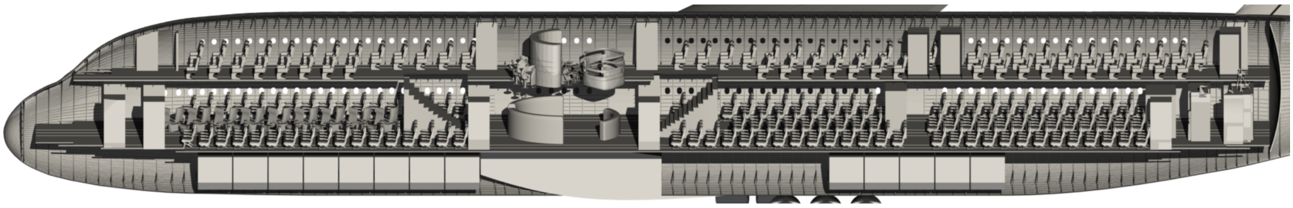


Figure 7.1.14.: Full fuselage 3D model of the D380 configuration

Also included in the model are the aforementioned LD6 cargo containers. The distribution of the containers between forward and aft fuselage aligns with the distribution of the side-by-side LD3 containers in the A380 cargo floor. The gap between the wingbox of the main wing represents the space for the landing gear bay, which is, however, not modeled explicitly.

7.1.2.4. Discussion

Given the current challenges w.r.t. sustainability of air transportation, the value of a configuration like the D380, presented in this section, can be presumed to be extremely limited. In fact, Wöhler, Walther, and Grimme [WWG22] propose a "people mover" aircraft based on the twin-engine Airbus A350, which is capable of carrying more passengers than the D380 by translating the high density cabin concept from single-aisle to wide-body configurations.

Nevertheless, the example of the D380 configuration serves to highlight several additional features of the fuselage design KBE system w.r.t. the single-aisle baseline. To begin with, the flexibility of the cabin design rule set is showcased, as it can also be used to model configurations with multiple aisles, taking into account the corresponding certification rules e.g. for cross-aisles. Furthermore, the capacity to support novel cabin concepts by integrating generic external component models is showcased.

Secondly, the versatility of the structural design capabilities is on display, as well. For example, the interior structure of the D380 configuration is described using a total of four floor instances, also including the upper crossbeams. Both of these points serve to further corroborate **working hypothesis 3**, by showing that the KBE methodology can indeed be leveraged to provide fuselage design details for a variety of conventional aircraft configurations.

That said, the example also reveals some shortcomings of the approach. Since the cabin design rule set was initially intended for configurations with a single passenger and cargo deck, some interventions of the design engineer are required to produce a consistent layout, e.g. the synchronization of the stair or mainframe positions between the two systems. This step can currently not be performed automatically and identification of the necessary parameters requires a profound understanding of the KBE system, which can not always be expected from a design engineer. This shows that a given rule set may be handled with relative ease when used within its designated design space, but will require more and more user intervention and "creative" utilization of the existing capabilities when approaching the limits of this space. A possible consequence, which must be considered, is that a bias towards the well-known is introduced, when deploying the tool in design campaigns, putting novel and unusual concepts at a disadvantage once again. That said, the knowledge captured during studies, such as the D380, can also inform further developments to permanently expand the scope of both FUGA and CPACS.

Finally, while the simple inclusion of the novel components highlights the advantages of the application of external component geometry models, the example of the secondary structure components also reveals the risks associated with the approach. It shows that, while the external models are a good way to add details without complicating the parametric design process, their lack of adaptability, as exemplified by the sidewall panels and the cross-section, makes them very difficult to rely on when exploring a broad range of configurations.

A possible path towards overcoming these limitations is the introduction of further knowledge to the system in the form of additional rule sets. By providing more rules either for parametric component design, or updating existing design rules, e.g. for the structure, both more details and support for new product architectures can be added to the system. Applications of the approach for architecture modification are discussed in section 7.2.

7.2. Architecture modification

The examples in section 7.1 illustrate the capacity of the knowledge-based methodology to produce consistent fuselage structure and cabin designs for conventional aircraft configurations. However, according to **working hypothesis 4**, the approach should also be easily applicable to novel architectures. It has been discussed in section 6.3, how this may be accomplished by creating additional rule sets, which can be used for pinpointed modification of the knowledge-based system to adapt to different design tasks.

Applying the rule sets introduced in that section, first, the integration of an LH_2 tank into the D240 baseline configuration using the rule sets presented in section 7.1.1 is investigated in 7.2.1. Secondly, the cabin layout generation for a BWB configuration leveraging the rule set proposed in section 6.3.2 is shown in section 7.2.2.

7.2.1. Liquid hydrogen tank integration

As anticipated by figure 5.2.3, the problem of integrating the LH_2 tank in the rear fuselage is addressed using an expanded design system, which is once again based on the baseline outside-in system. Different integration options, such as on top of the fuselage or in wing pods, are also fathomable, as outlined by Silberhorn et al. [SA+19]. The rear integration is selected in this example due to its relatively low aerodynamic penalty as well as the large effect on the fuselage design compared e.g. to pods.

Two additional aspects must be considered for the rear integration w.r.t. the baseline. On the one hand, the capability for increasing the available space in the fuselage, which has already been demonstrated in section 7.1.1.3, is again provided via the fuselage extension rule set described in section 6.2.1.2. On the other hand, the tank geometry is modeled based on the surrounding structure using the rule set described in section 6.3.1.

As formulated in equation 6.3.1, this makes it possible to compute the available fuel volume in the tanks V_{tanks} based on a change in length of the fuselage $\Delta l_{fuselage}$, a list of detailed tank geometry parameters, and a given tank count n_{tanks}. However, it is also stated that the required fuel volume $\hat{V}_{tanks}$ is typically prescribed prior to the fuselage design. Based on equation 6.3.1 and the simplifying assumption that all detailed tank geometry parameters be constant, this problem can then be described by

$$V_{tanks}\left(\Delta l_{fuselage}\right) = \hat{V}_{tanks}. \tag{7.2.1}$$

In order to determine the necessary change in fuselage length to meet the volume requirement, equation 7.2.1 can be rewritten as a scalar root finding problem, where

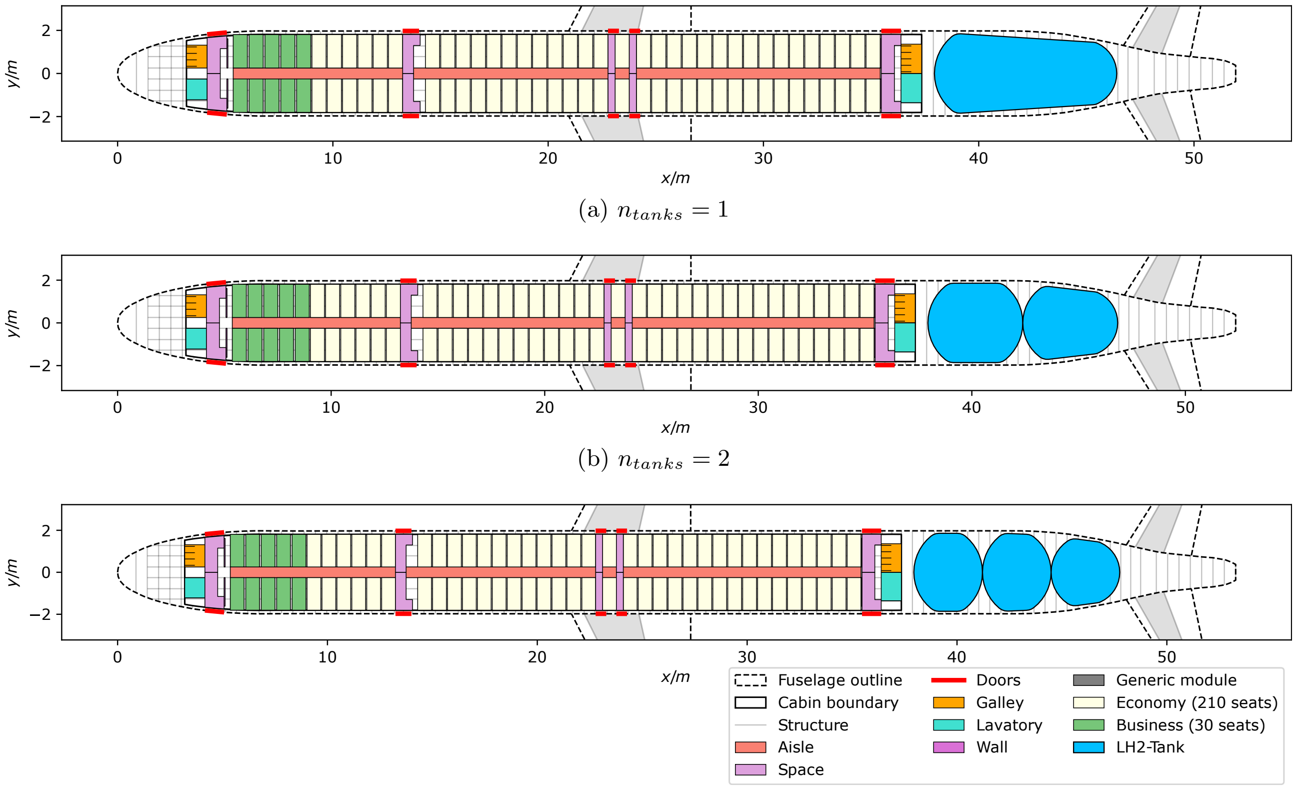
$$g\left(\Delta l_{fuselage}\right) = V_{tanks}\left(\Delta l_{fuselage}\right) - \hat{V}_{tanks} = 0. \tag{7.2.2}$$

Analogously to equation 7.1.2, the problem can be solved numerically by extracting the FUGA subgraph representing the relationship $V_{tanks}\left(\Delta l_{fuselage}\right)$. Differently to the above case, a trivial solution is not expected, due to the nonlinearity of the relationship between the available volume in the tail section and the fuselage length change.

In the following, the integration problem is solved for the D240 baseline configuration discussed in section 7.1.1, as described by Walther et al. [WH+22b]. The target tank volume is set to $\hat{V}_{tanks} = 55m^3$ for a hypothetical short-range design mission, based on reference data by Brewer [Bre91] and Burschyk et al. [BC+21]. The analysis is performed for different tank count values n_{tanks} in the range of one to three. The resulting fuselage layouts are given in figure 7.2.1. Visibly, layouts of the structure and cabin are included and considered for the tank design, leveraging the design capabilities of the baseline outside-in system.

Based on the results in figure 7.2.1, figure 7.2.2 provides a breakdown of the volume and length of the individual tanks. It shows that the accumulated volumes for each tank number amount to the prescribed $55m^3$. However, the distribution of the volume among the tanks is biased towards the front, if multiple tanks are installed. This is due to a design assumption that the length of each individual tank $\Delta x_{tank,i}$ can determined by splitting the available space Δl_{tanks} between the rear bulkhead and the HTP box evenly, i.e.

$$\Delta x_{tank,i} = \frac{\Delta l_{tanks}}{n_{tanks}}. \tag{7.2.3}$$



(a) $n_{tanks} = 1$

(b) $n_{tanks} = 2$

(c) $n_{tanks} = 3$

Figure 7.2.1.: Detailed geometry output for layouts with integrated LH_2 tank.

Hence, as the cross-sectional area decreases towards the rear of the fuselage, so does the tank volume.

With the overall volume is fixed, the accumulated lengths vary with the number of tanks used. As more tanks are added, more space is required by the tank structure due to the additional intermediate caps. It can thus be concluded that the number of tanks should be kept as low as possible to keep the fuselage length low. That said, a minimum of two tanks should be installed to comply with redundancy requirements [NH22].

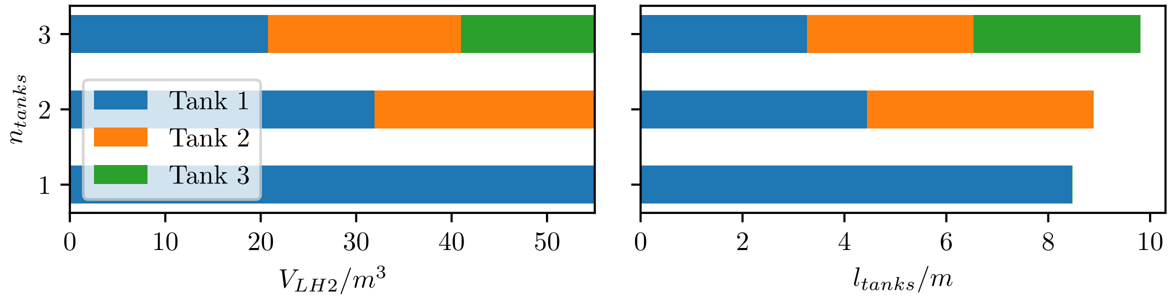


Figure 7.2.2.: Comparison of tank volume and length for equal length distribution (from [WH+22b])

Based on said redundancy requirement, it may also be deduced that the goal for the design of multi-tank layouts should be to achieve an even distribution of the volume among the tanks rather the length. This case has not been discussed by Walther et al. [WH+22b] and adds further complexity to the design process. In addition to the root finding problem in equation 7.2.2, a nested optimization problem is introduced, where the inner bounds between the tanks are shifted to achieve an equal volume distribution.

To enable the implementation of the optimization, several additional parameters for the tank design must be exposed in the knowledge-based system, to enable manipulation of the individual tank lengths. To begin with the outer bounds of the design space for the tank $x_{tanks,min}$ and $x_{tanks,max}$, and the resulting available tank space $\Delta l_{tanks} = x_{tanks,max} - x_{tanks,min}$ are determined from the RPB and the HTP box respectively, based on the length change $\Delta l_{fuselage}$. Notably, the connection between the rear bulkhead and the VTP front spar is separated and the bulkhead is instead placed directly at the end of the cabin by overriding the corresponding rule. This provides the boundaries for the distribution of the individual tank lengths, which are given by the vector $\Delta \mathbf{x}_{tank}$ of length n_{tanks} and initialized assuming the equal length distribution given in equation 7.2.3.

The individual tank boundary positions necessary to build the tank volumes can then be computed from the cumulative sum of $\Delta \mathbf{x}_{tank}$ and the outer bounds. Since the fuselage length change $\Delta l_{fuselage}$ remains constant throughout each iteration of the scalar root finding problem, this is also true for the outer tank bounds. Consequently, the individual tank volumes in the vector $\mathbf{V}_{tank}$ can be expressed as a function of the individual tank length as given by

$$V_{tank,i} = V_{tank,i}(\Delta x_{tank,i}).$$

Based on this expression, which can once again be implemented using FUGA by resetting the values for $\Delta \mathbf{x}_{tank}$ and making a query for $\mathbf{V}_{tank}$, the following optimization problem can be formulated, to determine the tank lengths for equal volumes:

$$\begin{aligned}
\underset{\Delta \mathbf{x}_{tank}}{\text{minimize}} \quad & \Delta V\left(\Delta \mathbf{x}_{tank}\right) = \sqrt{\sum_{i=1}^{n_{tanks}} \left(V_{tank,i}(\Delta x_{tank,i}) - \bar{\mathbf{V}}_{tank}\right)^2} & \\
\text{subject to} \quad & \Delta x_{tank,i} \geq 0, & i = 1, \ldots, n_{tanks}, \\
& \sum_{i=1}^{n_{tanks}} \Delta x_{tank,i} = \Delta l_{tanks}. &
\end{aligned} \tag{7.2.4}$$

For the objective function the ℓ^2-norm of the deviations of the individual volumes from the mean volume $\bar{\mathbf{V}}_{tank}$ is selected. This ensures that all resulting volumes are equal. Furthermore, an equality constraint is introduced to assert that the sum of the individual tank lengths corresponds to the overall available length Δl_{tanks}. In addition, a constraint is introduced, which ensures that all lengths are nonzero.

Similarly to the original root finding problem in section 7.1.1.3, the updated root finding problem with the nested optimization can be described using XDSM notation, as shown in figure 7.2.3. The notation illustrates the nested loops very well via the process flow line. The outer loop remains controlled by the solution for the fuselage length, whereas the inner loop is controlled by the optimizer. The corresponding rules executed at different stages are also shown. This makes it easy to follow which rules are executed how often during the overall solution. Clearly, the rules 1 to n as well as the rule $n+m+2$ are visited once for every iteration of the root finding problem solution. Thus, the total number of calls for each rule would be $n_{iter,root}$. Similarly, the optimization is performed once per iteration of the secant method. However, the rules $n+2$ to $n+m$, which are contained within the optimization loop, are called once for every iteration of the optimization loop. Thus, they are called $n_{iter,root} \cdot n_{iter,opt}$ times overall. This highlights the particular need for efficiency of the rules called within the optimization loop to avoid excessive run times. In this context, the graph-based methodology is helpful to determine the exact set of rules, which is necessary for the evaluation of the objective function, to help avoid computational overhead due to unnecessary reevaluations of parameters. That said, if possible, an especially high standard of efficiency should also be applied to the implementation of rules themselves, if they participate in the optimization loop. In the example at hand, the tank volume reevaluation takes a matter of seconds owing to efficient underlying implementations, e.g. of the volume integration, in OCCT.

The updated breakdown of the volume and length of the individual tanks for the equal volume design problem is given in figure 7.2.4. In comparison to figure 7.2.2, the requirement that the overall

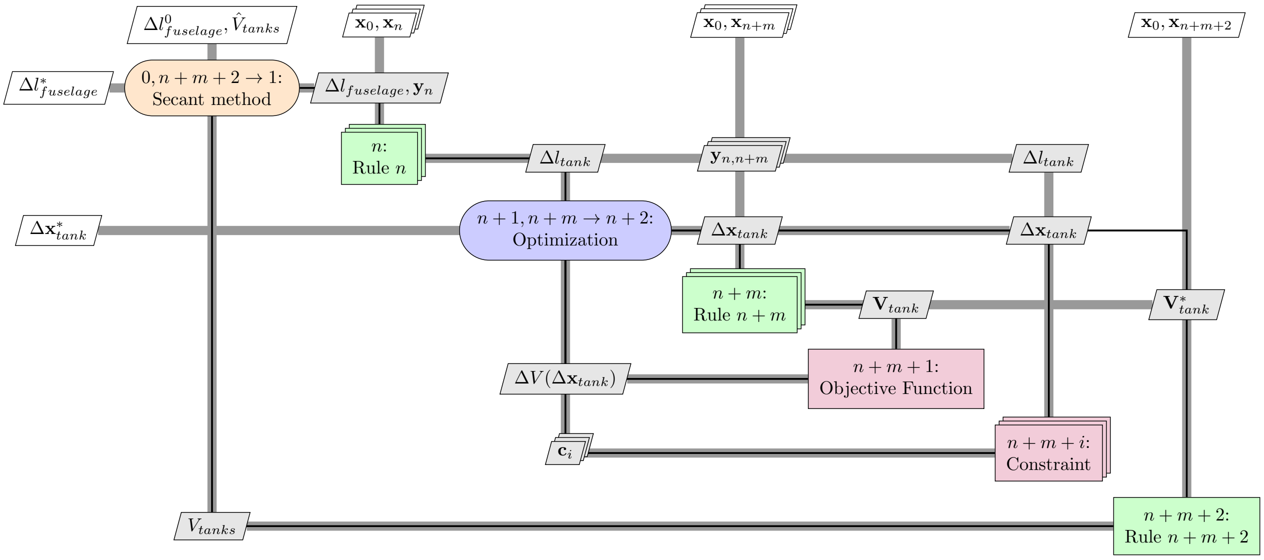

$\Delta l^0_{fuselage}, \hat{V}_{tanks}$
$\mathbf{x}_0, \mathbf{x}_n$
$\mathbf{x}_0, \mathbf{x}_{n+m}$
$\mathbf{x}_0, \mathbf{x}_{n+m+2}$
$\Delta l^*_{fuselage}$
$0, n+m+2 \to 1$: Secant method
$\Delta l_{fuselage}, \mathbf{y}_n$
n: Rule n
Δl_{tank}
$\mathbf{y}_{n,n+m}$
Δl_{tank}
$\Delta \mathbf{x}^*_{tank}$
$n+1, n+m \to n+2$: Optimization
$\Delta \mathbf{x}_{tank}$
$\Delta \mathbf{x}_{tank}$
$n+m$: Rule $n+m$
$\mathbf{V}_{tank}$
$\mathbf{V}^*_{tank}$
$\Delta V(\Delta \mathbf{x}_{tank})$
$n+m+1$: Objective Function
$\mathbf{c}_i$
$n+m+i$: Constraint
V_{tanks}
$n+m+2$: Rule $n+m+2$

Figure 7.2.3.: XDSM representation of the solution to the root finding problem with the nested optimization problem

volume be divided evenly between the tanks is now satisfied. In return, the lengths of the tanks now increase towards the rear.

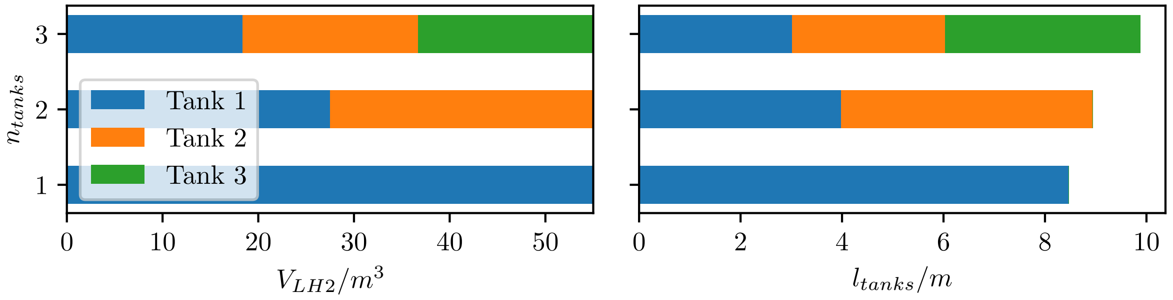
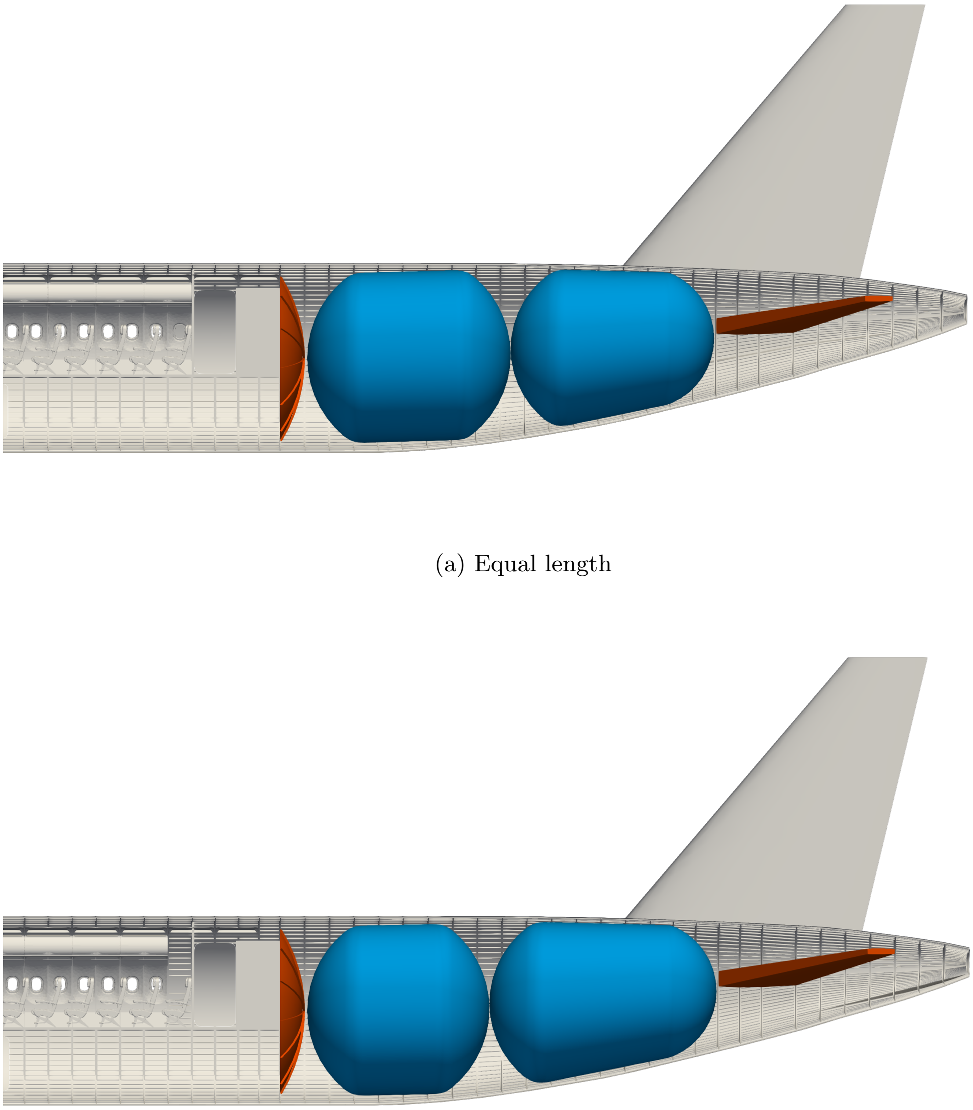


Figure 7.2.4.: Comparison of tank volume and length for equal volume distribution

In figure 7.2.5, the detailed geometry models derived from the integration results for $n_{tanks} = 2$ for both design paradigms are given. Both models include the detailed geometry of the structure and the cabin, including the detailed RPB and the HTP box. The availability of these details of the bounding components is useful for the assessment of the validity of the designs. Visibly, the tanks are fitted properly in both cases. The detailed model furthermore provides a way to determine any clashes with the fuselage structure, as the extruded representations of the stiffeners are available. Comparing the two models, the observations from figures 7.2.2 and 7.2.4 pertaining to the length distribution are confirmed.

A noteworthy difference between the two layouts in figure 7.2.5 is the gap in the sidewall panels in front of the rear cabin wall, which is found in the result equal volume design strategy, but not for its equal length counterpart. This is caused by a change in the number of frame bays between the two designs. Whereas for the equal length strategy an even number of bays is returned, the number is odd for the equal volume approach. The reason for this is the small length difference between the two designs, which results in a slightly different position of the center wing box. Since the center wing box is, however, an input to the frame distribution, as described in section 6.2.2.2, even relatively small changes may have amplified effects further downstream in the design process, as exemplified by the



(a) Equal length

(b) Equal volume

Figure 7.2.5.: Detailed geometry output for tank integration with $n_{tanks} = 2$

different sidewall arrangements.

The above example shows that, as required by **working hypothesis 4,** the knowledge-based design system can be adapted with relatively little effort to support novel and and unanticipated design problems by incorporating additional rule sets. In this case, the additional rule set effects an *extension* of an existing system. As evidenced by the cabin and structural layout generation, large parts of the baseline rule set can still be deployed, whereas the additional rule sets provide the means for targeted manipulation of the system behavior. This results in design data sets, which can match the very high level of detail of the baseline configuration. That said, the tank integration use case clearly benefits from the fact that the basic configuration is still a tube-and-wing configuration, which is an important condition for the application of many of the baseline rule sets. To investigate a configuration, which is a more radical departure from the tube-and-wing configuration, the BWB cabin is considered in section 7.2.2.

Similarly to the findings in section 7.1.1.3 it is also shown that the efficient implementation of such additional rule sets once again requires deep insight in the available baseline system, not only to avoid duplicate implementations, which can possibly break consistency, but also to ensure that updated or overloaded data repository entries, as exemplified by the rear bulkhead position, are returned to the correct nodes and in the correct format. This further underlines the importance of the explanation subsystem and rule documentation.

7.2.2. BWB cabin layout

Whereas the modification of the product architecture for the LH_2 tank integration in section 7.2.1 is implemented primarily by *extending* the knowledge-based system by adding further rules to design additional components, the system for the BWB cabin design use case predominantly requires *mocking* of existing rules. This is supported by the description of the additional design rules for the use case in section 6.3.2, which essentially serve to mimic necessary structural inputs for the cabin design. The mocking rules are deployed by overriding the corresponding rules in the conventional design system.

By including the updated rules, the BWB cabin layout generation can be considered an extension of the twin-aisle design capability demonstrated in section 7.1.2, where the number of aisles is simply increased further. An additional requirement is the possibility to define empty blocks between aisles and exits to account for the substantially decreasing fuselage cross-section towards the front.

In the following, the BWB configuration from the AGILE project, used previously for illustration in section 6.3.2, is again considered. To determine the targeted level of detail, the BWB cabin layouts created using the Cabin Configurator by Baan [Baa15] and shown in figure 2.1.4 are used as a reference. As a result, the cabin layout generation is restricted to the floor layout. Any structural details are bypassed and no paneling is included. The design range for the configuration is specified at $R = 8300nm$ by Shiva Prakasha et al. [SD+18], which means a cabin layout with a long-range level of comfort is required.

Figure 7.2.6 shows the LOPA for the AGILE configuration. The single-class layout can provide seating for 538 passengers. Analogously to the majority of the layouts proposed by Baan, the maximum number of seats abreast per block is limited to four, which results in seven aisles at the widest part of the cabin behind the third exit.

A total of 12 lavatories are installed, eight of which are a part of dual lavatory modules. This results in a lavatory utilization $UF_{lav} = 44.833\frac{PAX}{lavatory}$. Furthermore, six galleys, providing space for a total of 36 FSTs are present, which yields a trolley utilization $UF_{galleys} = 14.944\frac{PAX}{trolley}$. These figures are only slightly inferior to the figures for the economy main deck in the D380 configuration in table 7.1.3 and are therefore consistent with the long-range design mission of the configuration. That said, a very low seat pitch $\Delta x_{seat,eco} = 28in$ is retained, which may be justified e.g. by more compact modern seat designs with a reduced space requirement.

For the exit layout four type A+ exits are envisaged. Following the example of Liebeck [Lie04], three pairs of exits are distributed across the forward part of the fuselage, which is not obstructed by

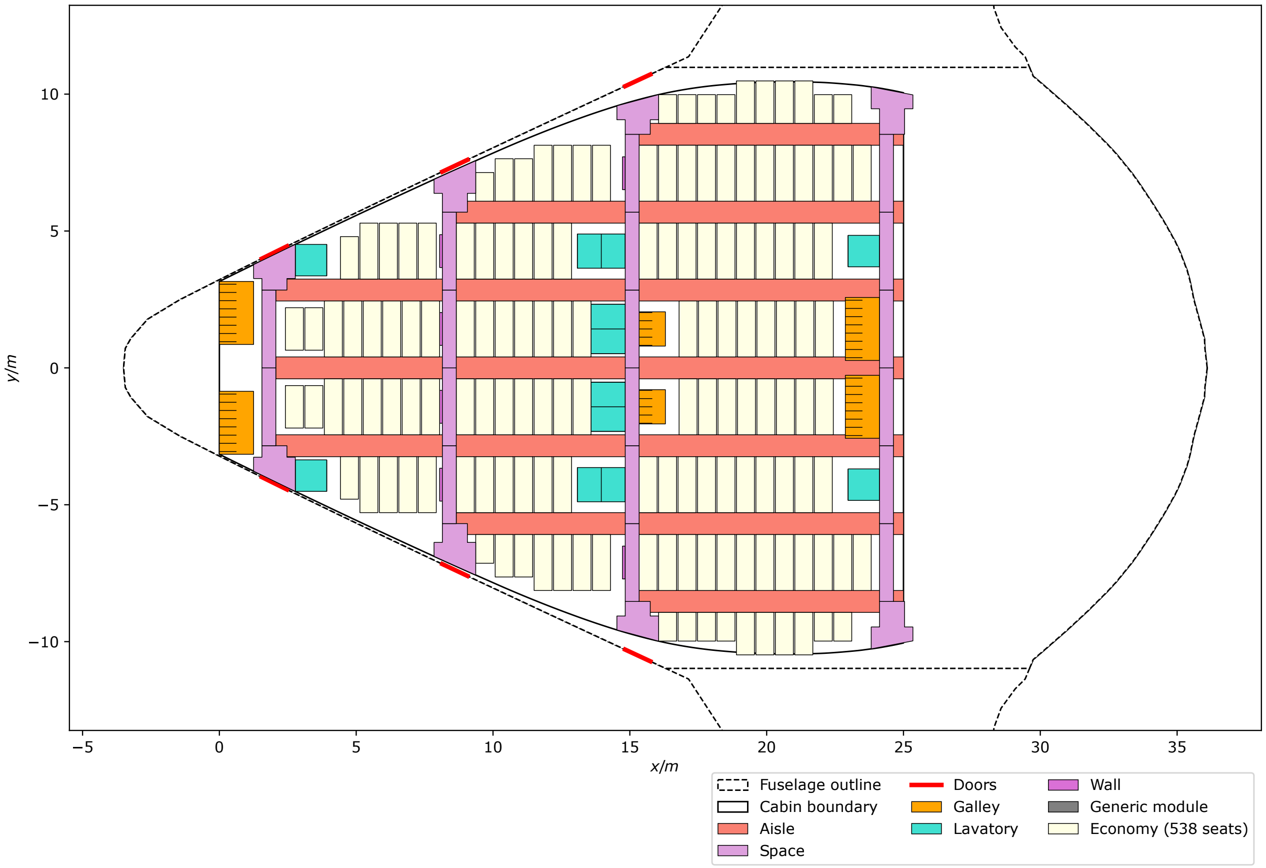


Figure 7.2.6.: LOPA for AGILE BWB configuration

the wing. In contrast, the rear exit, which is required by CS 25.807(f), is modeled as a virtual exit and does not lead to a door opening.

The resulting capacity from the exit layout remains beneath the total seating capacity. However, adding further exits in the forward part will lead to a very uneven distribution of block sizes, with the forward blocks having approximately 4 rows each, compared to 12 rows in the rear block. This is aggravated further by the increasing number of lateral blocks toward the rear of the cabin. Thus, the effectiveness of the additional exit would be severely diminished, since only a small number of passengers would be able to reach it quickly. Moreover, it is not possible to move back the third exit pair to improve the distribution of the rows, since it would cause another collision with the wing. To overcome these limitations, alternatives to the commonly used side exits, such as ventral exits need to be considered. However, those solutions present entirely new sets of problems e.g. the possibility to evacuate during a ditching event.

These examples highlight the limitations to the exploitation of the additional available space in a BWB configuration. Besides, some additional issues could be worth investigating, e.g. whether the cross-aisle width of $20in$ is sufficient in an egress event if more than two aisles are present. To this end, the details provided in figure 7.2.6 could already be sufficient for a basic boarding or egress simulation as shown in section 4.2.3.3. Furthermore, due to the availability of designated aisles and exit spaces as well as a more fine-grained control over the floor layout, especially w.r.t the monuments, the design shown actually surpasses the BWB reference layouts from the Cabin Configurator in terms of fidelity.

To further support this, a 3D view of the layout is given in figure 7.2.7, which provides an even further increased level of detail. Again, the 3D seat component models can be used in the overall model. The cabin monuments are once more represented by blocks. The cabin boundary retrieved from the updated rule set is used to clip the boxes, as in the baseline system. Furthermore, the lower plane of the boundary is used to represent the cabin floor.

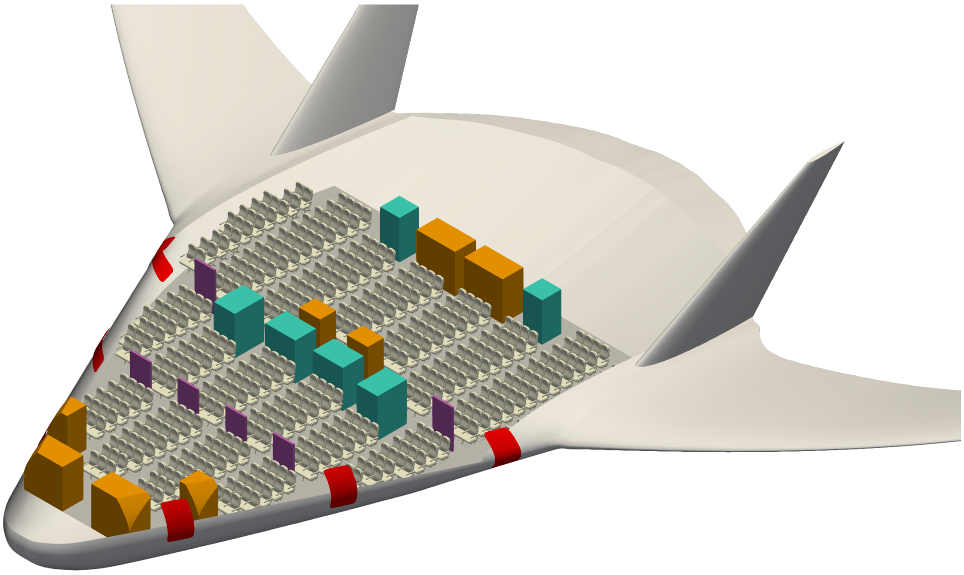


Figure 7.2.7.: 3D view of the BWB cabin

The cutouts for the exits in the outer surface are also modeled. It becomes apparent, especially for the exit right in front of the wing root that the intersections result in rather complex shapes. This might pose a challenge to door design and manufacturing. Furthermore, it becomes clear once again that it is impossible to realize a pair of side exits at the end of the cabin due to the presence of the wing, highlighting the need for different solutions to fulfill the certification requirements.

In conclusion, the level of maturity of the BWB cabin design is clearly not on the same level as for the previous conventional configurations. This is a reflection of the available knowledge in the aerospace community, where BWB cabin or structural design are still rarely considered in detail, as shown in section 2.1.3. Similarly, there are still unresolved safety and certification issues, partly because no OEM has attempted to certify a large passenger BWB to date.

Nonetheless, the example shows that design capabilities for the novel configuration can be implemented by including simplified assumptions as part of an updated rule set. In this case, the detailed structural design rules could be bypassed to leverage the available cabin design rules described in section 6.2.3 and provide a cabin layout at a level of detail comparable to the layout presented in section 7.1.2 for the more conventional long range configuration. In conjunction with the extending rule sets presented in section 7.2.1, this example illustrates the range of options for adapting the knowledge-based system to support novel architectures as required by **working hypothesis 4**.

Comparing the results shown in figures 7.2.5 and 7.2.7 it can, however, be observed that the mocking approach, unlike the extension approach, may have adverse effects on the level of detail of the resulting geometry models. In the BWB example this is obvious as all the structural details are omitted. Since the goal of **working hypothesis 4** is to support fundamental decisions on different system architectures, which are informed by disciplinary analysis, the potential loss of fidelity resulting from mocking rule sets must consequently be considered a risk. Missing critical information e.g. the structural information may result in inability to perform some disciplinary analyses, e.g. the structural sizing or mass estimation, impeding a fair comparison of these aspects of the different architectures. Thus, while the technique may be useful during design exploration and rule development, an extending rule set, which does not simply bypass knowledge provided in the baseline system should usually be preferred.

7.3. Fuselage analysis models

Returning to the D240 baseline design discussed in section 7.1.1, this section is meant to showcase capabilities enabled by the graph-based methodology implemented in FUGA to provide analysis models and provide a comparison to the state of the art in the corresponding field as established in section 4.2. The goal is to show that the approach can be used to create models at a level of detail, which would allow a disciplinary expert in the field to perform meaningful analyses on the design, to conclusively prove **working hypothesis 2**.

From the analysis and design methods listed in section 4.2, the GFEM model for structure sizing presented in section 4.2.2 and the fuselage model for human factors analysis introduced in section 4.2.3.1 are selected as examples. On the one hand, the selected analyses cover a wide range of geometric detail, with the GFEM requiring a high degree of abstraction and the human factors model relying on a high level of geometric detail. On the other hand, the two use cases complement each other very well in terms of which components are in focus, since the GFEM analysis is primarily driven by the structure, whereas the human factors analysis is focused on the cabin.

7.3.1. GFEM for fuselage structure sizing

The review of structural model generators for the FEM-driven sizing of fuselage structures in section 4.2.2 illustrates a clear trend towards parameter driven model generation based on CPACS since the first publications appeared on the format. The structural definitions provided by the format are a good foundation for the description of most of the relevant structural components in GFEMs listed in table 4.2.1, which, for the most part, adhere to the template provided by PrADO. Out of the tools listed in the table, PANDORA (formerly TRAFUMO) provides the most complete coverage of the CPACS structure definition for the fuselage and is therefore selected as the reference in the following. Applications of PANDORA, which serve as a reference in the following, have been published e.g. by Scherer et al. [SK+13], who consider the D150 configuration.

As discussed previously, a common weakness of most GFEM generation tools based on CPACS is the lack of flexible initialization capabilities for structural layouts for a given data set from OAD. As a result, the sizing capabilities are typically focused on improving individual structural component properties e.g. skin thicknesses, but do not readily support more fundamental changes in topology. As shown in section 6.2.2, FUGA is capable of providing modeling capabilities to support this type of analyses.

Consequently, the possibility to combine the design initialization and geometry generation rules to generate GFEMs, which are useful for structural sizing application, described in **working hypothesis 2**, is demonstrated in this section. The single-aisle configuration described in section 7.1.1 is used as input to the model generation, being the most similar configuration to the D150 configuration evaluated by Scherer et al.

7.3.1.1. Base structure grid generation from CPACS

To showcase the GFEM generation capabilities of FUGA, the objective in the following is to assemble an automated process chain, which can provide an FEM analysis solution based on an empty fuselage surface in CPACS without any structural definitions. This capability was first demonstrated by Walther and Ciampa [WC18], whose design and model generation tool provides the basis for many fundamental design capabilities in FUGA. As a result, comparable models can still be provided using FUGA. Figure 7.3.1 provides an example of a GFEM of the D240 configuration at the level of detail established by Walther and Ciampa.

Visibly, the model contains only a subset of the components, which are typically accounted for by reference tools such as PANDORA. Nonetheless, many essential parts are already present, including most entries from the paneling, frames and floor structure component groups listed in table 4.2.1. The model components can roughly be divided into three different groups:

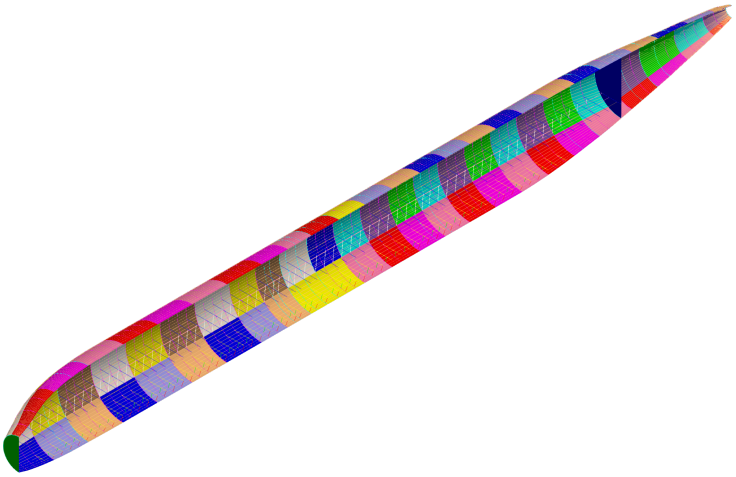


Figure 7.3.1.: Basic fuselage structure FEM mesh after [WC18] (colors indicate element property IDs)

Base grid The frames and stringers are modeled as beam elements whereas the skin panels, which they surround, are represented by a single shell element. Together, they form the base grid. The nodes, which are computed by sampling the fuselage surface using vectors defined by the intersection of the frame and stringer definition planes, as described e.g. by Walther, Petsch, and Kohlgrüber [WPK17], form the corner points of the skin panels. The individual skin panel elements are grouped into skin segments, which share common element property IDs.

Floor structure Floor components including crossbeams, struts and longitudinal beams are modeled using beam elements. In a basic GFEM modeling approach, the vertical positions of the crossbeams defined in CPACS must be adjusted based on a distance criterion to ensure that the beams align with a node in the base grid. This is a source of inconsistencies, which is why the position of the crossbeams is considered during the generation of the stringer distribution in FUGA to keep this correction minimal.

Flat bulkheads Only a simplified flat bulkhead representation is implemented, exclusively using shell elements. The basic shape is given by a polygon, which is extracted from the base grid at the frame position of each bulkhead. The meshes are then generated from the polygons using Gmsh [GR09]. Whereas the bulkheads are needed to create a closed pressure vessel in the model, the simplified modeling approach illustrates the need for more advanced geometric modeling capabilities, which have since been made available through FUGA.

Aside from the bulkheads, another notable simplification is the lack of a center fuselage area, which is required to realistically model the introduction of the wing loads as stated e.g. by Scherer et al. Once again, the omission can be attributed to a lack of advanced geometric modeling capabilities.

In order to show that the knowledge-based model generation approach can support the generations of models, which are equivalent to those provided by PANDORA, it is necessary to introduce these additional details into the above mesh. This is discussed in the following section.

7.3.1.2. Application of FUGA to integrate of additional model details

To augment this basic model with more detailed component models, which correspond more closely to the standard set by tools such as PANDORA, the more advanced modeling and design capabilities provided by FUGA and described in 6.1.2 and 6.2.2 respectively can be leveraged. In total, three additional entries in the list of components are taken into consideration here: The detailed bulkheads, the center fuselage area composed of the center wing box and the landing gear bay, and the payload.

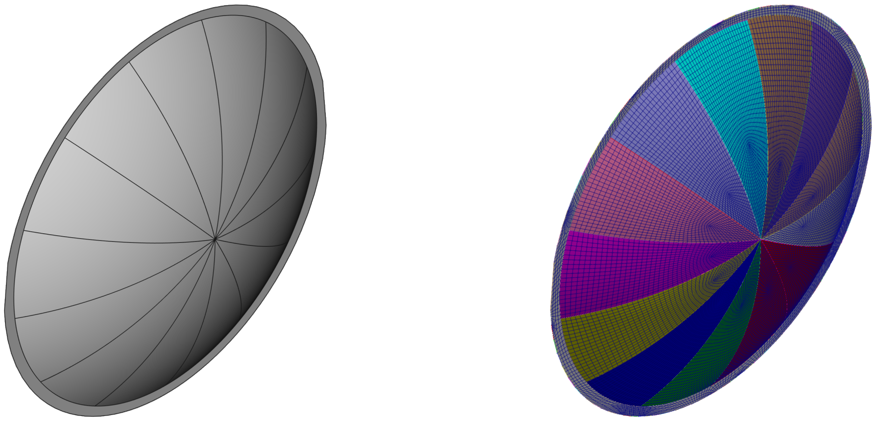
Detailed bulkhead model The bulkheads are an essential part of the model, as they are required to form a closed pressure vessel. A closed vessel surface is necessary to avoid unrealistic reaction forces when applying the loads due to the interior pressure. A detailed model, especially of the spherical rear bulkhead, which is expressly shaped to alleviate pressure loads, can contribute to a more accurate assessment of the effects of the interior pressure compared to its flat counterpart.

To build the mesh, the geometric rules for the detailed model generation provided by FUGA can be combined with geometric information gathered from the base grid. In particular, the frame polygons are required, which also provide the basis for the generation of the flat bulkhead meshes, are now used to provide the base surface for the subsequent geometry generation in FUGA. Since the meshing algorithm used is guaranteed to place a node on every corner of the polygon, it can be assured in this way that the nodes of the base mesh are also present in the component mesh to enable subsequent node-based connection of the two. The surfaces retrieved from the polygons are used to replace the base sheets introduced in the description of the bulkhead model generation rules in section 6.1.2.1.

A drawback of this modeling approach is the possibility for hanging node on the polygon edge due to the substantially lower element size in the bulkhead connector ring compared to the GFEM grid. In the scope of this study, this is tolerated due to the significantly higher stiffness of the bulkhead compared to the typical fuselage structure. A possible remedy could be the introduction of RBE2 constraint elements across the polygon edge.

Once the base sheet has been updated, the subsequent geometry generation steps for the detailed bulkhead with reinforcement curves can be traversed as before. For the example of the spherical RPB, this yields the geometry model shown in figure 7.3.2a. The multi-fidelity modeling capabilities enabled by the graph-based system formulation are taken advantage of, as the reinforcements are provided as curves, omitting the extra step of extruding the cross-section profiles of the reinforcements.

Next, a quadrilateral mesh generation algorithm provided by Gmsh is applied to the new geometry model, which results in the mesh shown in figure 7.3.2b. Conveniently, Gmsh not only provides the mesh nodes and cell connectivity as outputs, but also returns the corresponding edge or face indices for each 1D- or 2D-element respectively, which are represented by the coloring of the elements in the figure. Having this information available makes it possible to trace each element back to its original CPACS definition, which enables the proper assignment of element properties according to CPACS.


(a) CAD geometry (FUGA)

(b) Quadrilateral mesh (Gmsh)

Figure 7.3.2.: Rear bulkhead model

With the properties assigned, the bulkhead mesh can then be merged with the base mesh with the common nodes from the base polygon providing the connections. A drawback of this approach is that the resolution of the bulkhead mesh is substantially higher than that of the base mesh, as points are added by the mesher on the boundary between the initial polygon points, which are not coupled to the base mesh. For the purposes of this thesis, this is deemed uncritical, as the stiffness of the bulkhead is substantially higher than the surrounding base structure. Consequently, the effect of the possible deformations is considered minor, compared to the more coarsely resolved but less stiff base grid.

Center fuselage area Whereas the rules from FUGA can be leveraged quite easily for the generation of the bulkhead model, the situation is slightly more complex for the center fuselage area model. Fundamentally, the same approach is used as before, i.e. a CAD component in FUGA is mocked using geometry extracted from the mesh. However, the extraction of the correct geometry is somewhat more complicated than simply extracting a frame polygon from the mesh.

Since the center fuselage modeling rules are all built upon the fuselage solid model, this is the geometric component to be mocked. The base grid in the GFEM should, however, only be replaced in the region of the center fuselage cutouts, retaining the remaining mesh. By converting all mesh nodes to OCCT vertices, Boolean geometry operations can be applied to determine those nodes, which lie within the center fuselage cutout volume. The volume is determined as described in section 6.1.2.3. The elements adjacent to the identified nodes are removed from the GFEM base mesh. Then, the free edges of the updated mesh, i.e. edges, which are only adjacent to one element, are determined in order to find the boundary of the cutout. This boundary polygon provides the basis from which to build the mocked fuselage solid for the center fuselage mesh generation.

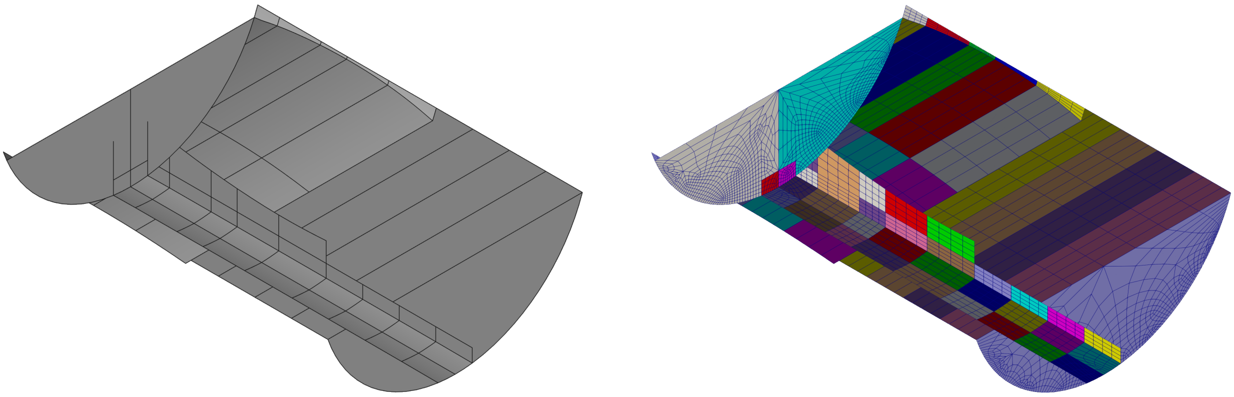
To build the closed solid, the boundary polygon is first split in half at the yz-plane, resulting in two open polygonal wires. The curves are then rearranged to make sure they are in the same order and an open shell is built using the OCCT section interpolation algorithm. The algorithm is similar to the skinned surface interpolation shown in section 4.1.1.2, but can operate on B-rep objects such as wires. In this way, the faces of the center fuselage cutout, which lie inside the fuselage, can be reconstructed. In addition, the outer surface must be rebuilt in such a way that it fits the edges of the boundary polygon. To this end, a closing edge is added to each of the two open wires and a filling algorithm is applied. The filling algorithm is also provided by OCCT and provides similar functionality to the point cloud approximation, also described in section 4.1.1.2. As such, the algorithm attempts to build a face from the boundary by fitting a NURBS surface. While the accuracy of the resulting surface w.r.t. the original surface is reduced significantly, the boundary remains compatible with the free edges from the base mesh, which means that, like the detailed bulkhead model, the meshed representation of the center fuselage area will contain shared nodes with the base mesh, which can be used to connect the two. Combining the faces provided by the filling algorithm with the inner shells yields a closed shell, which can be converted to a solid.

The newly created solid geometry can now be used to mock the fuselage solid geometry input for the center fuselage area model generation rules in FUGA. The resulting CAD geometry is shown in figure 7.3.3a. Visibly, the contributions from the outer surface are limited to the bottom of the keelbeam and the skin connecting the wingbox to the neighboring stringers. In either case the resulting patches are small, which means a deviation due to the surface reconstruction can be accepted if it leads to compatible edge nodes.

The mesh built from the geometry is shown in figure 7.3.3b. As indicated by the cell coloring, the face indices necessary for mapping the mesh cells to their original CPACS definitions are once again available.

Payload and cabin masses The capability of FUGA to generate both structural and cabin details consistently and make them available via CPACS can also be leveraged in the GFEM model generation. As shown by table 4.2.1 the payload in GFEM models is typically represented by mass points. To this end, the cabin layout and thus the expected positions of the payload masses as well as major cabin component masses can be determined by FUGA. Leveraging the multi-fidelity capabilities for data reduction, a simple approximation of the positions is given by the geometric centers of the bounding boxes of the component models, as illustrated by figure 7.3.4. A collection of methods for estimating the mass values has been provided e.g. by Fuchte [Fuc14].

To provide connections to the structure, distributing coupling elements (e.g. RBE3 in Nastran) are introduced. To determine the structural attachment points, a simple approach, which selects the n



(a) CAD geometry (FUGA)

(b) Quadrilateral mesh (Gmsh)

Figure 7.3.3.: Center fuselage model

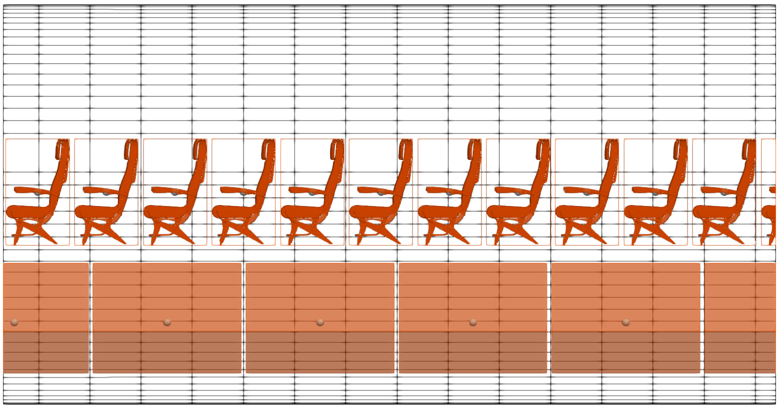


Figure 7.3.4.: Cabin mass point determination from component meshes

nearest structural nodes has been implemented. The problem of finding the neighbors can be solved very efficiently using spatial decomposition trees [MM02; VG+20]. The more difficult issue is to determine the correct subset of structural nodes, which are eligible for the connection. To this end, each type of mass node, corresponding to a certain type of cabin component, can be mapped to one of several node sets. On the one hand, a node set is provided for each separate floor. In this way, floor-based cabin components, such as seats and monuments, can be linked to the passenger floor, whereas cargo container masses can be linked to the cargo floor. On the other hand, the base grid is available as a set as well. This set is used to connect the components of the secondary structure, such as sidewall panels or OHSCs. The resulting mass modes as well as the corresponding RBE3 connections are shown in figure 7.3.5.

Merging these component models with the basic structural mesh from figure 7.3.1 yields the more detailed overall fuselage model shown in figure 7.3.6. It shows that, by leveraging the geometric modeling capabilities of FUGA in combination with a mesh generation tool such as Gmsh, the level of detail of the model can be increased with relatively little effort to approach the fidelity of state of the art tools such as PANDORA. That said, several details such as the tailplane attachments or the doors and windows are not modeled in detail yet. Nevertheless, the doors and windows are already taken into account in the design of the frame and stringer distribution, and the corresponding elements can easily be identified and assigned dedicated properties if necessary.

It may be put into question, whether the approach adopted here, i.e. to begin with a very simple base mesh that is successively augmented by detailed component meshes, is the best possible solution. Another option could be to assemble a single BRep-model and pass it to Gmsh as a whole. If the model is built in the correct way, Gmsh will automatically ensure the compatibility of the meshes

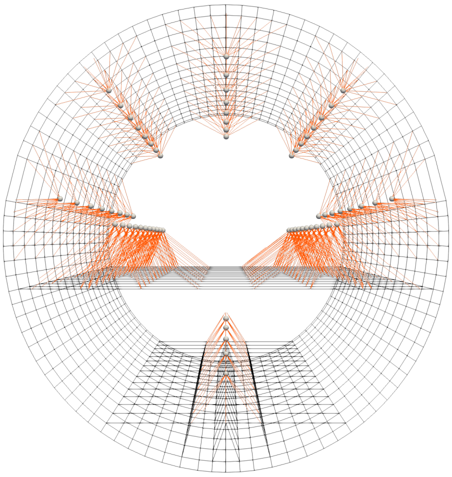


Figure 7.3.5.: Cabin mass points and structural attachments

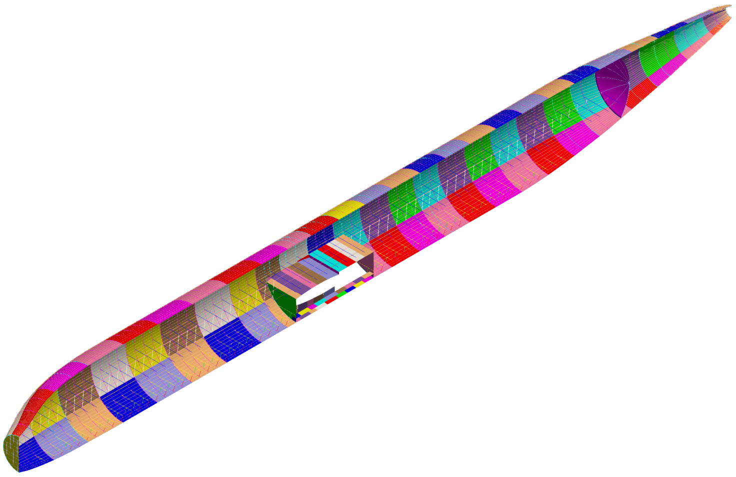


Figure 7.3.6.: Augmented fuselage structure FEM mesh (colors indicate element property IDs)

of neighboring components, which results in a well-connected and consistent mesh. However, this approach requires a single mesh generation technique, including control parameters such as element length, to be applied across the entire model. Meanwhile, the component-wise meshing approach allows each component mesh to be discretized using the best possible algorithm and settings. Furthermore, it is currently not possible to build the base grid using Gmsh, as each face is subdivided at least once. The knowledge-based model generation approach implemented in FUGA also complements this approach, as it provides fine-grained access to components and fidelity levels, substantially reducing the effort necessary to build the individual component models.

7.3.1.3. Interfaces for load application

Aside from a suitable mesh, a set of loads to be applied to the model, which covers the entire flight envelope, is essential to enable reliable structural sizing [KO+16]. Whereas the determination of the loads themselves is beyond the scope of this thesis, the provision of the proper interfaces to apply the loads to the mesh is a crucial part of the model generation. Several types of loads are commonly considered at the GFEM analysis stage. On the one hand, loads due to the flight state, e.g. maneuver or gust loads, must be taken into account, which are usually the output of an aerodynamic analysis. Inertial forces, which occur due to the acceleration prescribed by the flight state must be included as well. On the other hand, the pressure difference between the cabin interior and the surroundings results in pressure loads, acting on the fuselage shell.

The flight loads may be supplied in several ways. Commonly in early design, a set of cut loads is given at a series of discrete points on the load reference axis of the fuselage [KO+16]. An example of loads for a fictitious test case is given in figure 7.3.7b. For this type of load input, the load application points must be available in the GFEM and coupled to the structure. Similarly to the cabin masses, the coupling may be implemented using RBE3-type elements. In a collaborative design process, the load reference axis points are usually provided by loads experts via CPACS, along with the loads themselves. However, FUGA also provides simple rules to determine the load reference axis points for the fuselage and the wings as shown in figure 7.3.7a. The loads process usually includes a simplified mass model, which means that contributions due to inertia are already accounted for.

Another option is to map the pressure loads from the surface mesh of an aerodynamic simulation directly to the surface nodes of the structural model. This is typically achieved using mesh-free techniques, where forces on one set of points (the aerodynamic surface mesh points) are interpolated onto another set of points (the structural mesh points) [BW01; QMM05]. When applying these techniques, it is critical for the two meshes to be compatible, i.e. to avoid large gaps between the aerodynamic surface mesh and the structural mesh, which would result in exaggerated forces. Consequently, it is important that the mapping is performed only between nodes of the aerodynamic surface mesh and the corresponding wetted surface nodes of the structural mesh. These nodes must therefore be identified during the model generation by combining the nodes of the base grid with the nodes of the center fuselage model surfaces, which correspond to the fuselage surface. The node IDs are stored in a set and can be extracted from the model as shown in figure 7.3.8a. Differently from the cut loads, the inertial forces are not included in the surface pressure distribution, which means they must be included separately. Typically, this can be achieved by applying a global acceleration to the model. To receive correct inertial loads in this way, it is important for the model to accurately represent the mass distribution of the aircraft. In this context, the capacity to include detailed distributions of cabin masses provided by FUGA makes a valuable contribution.

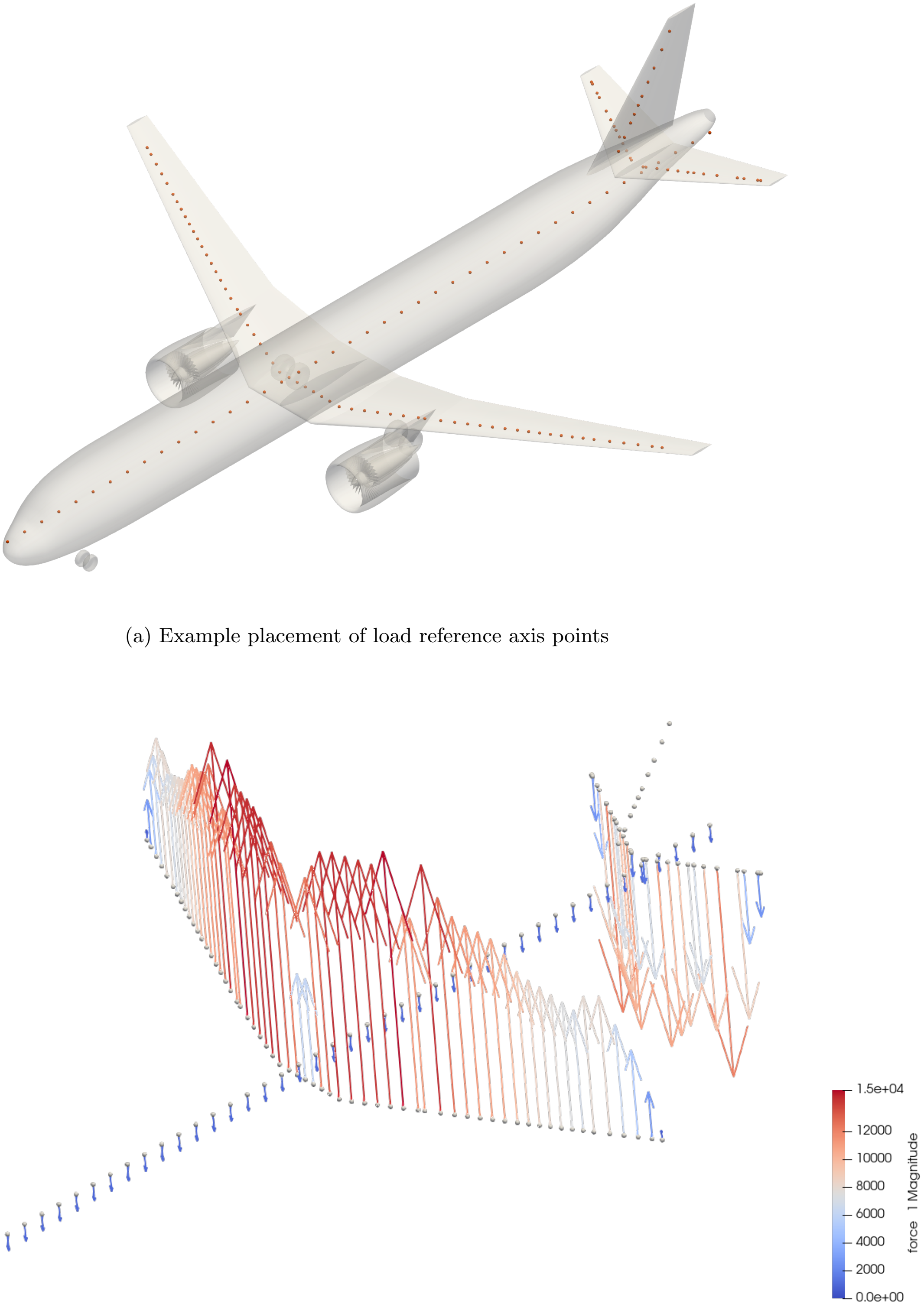
In order to properly apply the loads due to interior pressure, the pressurized regions of the fuselage must furthermore be identified. Unlike the aerodynamic loads, which are applied as discrete force or moment vectors at the nodes, the interior pressure loads can be applied directly on the shell element surfaces. Consequently, the pressurized region can be described by a set of shell elements, which is composed from those elements of the base grid that lie between the two pressure bulkheads, the bulkhead shells, and the shells generated on the pressurized surfaces of the center fuselage area. The resulting element set is illustrated in figure 7.3.8b.

The list of load cases covered here is not exhaustive and omits several types of load cases, which are in fact essential for sizing real aircraft. This includes landing shocks and crash or ditching scenarios. However, these cases are typically too complex to be assessed reliably using a GFEM, instead requiring higher fidelity models and nonlinear FEM solvers, as outlined in section 4.2.2.2. Nevertheless, some of the aspects discussed here may also be of interest for such higher fidelity analyses. Once again, the availability of detailed cabin masses must be highlighted, which is particularly interesting for crash simulations, since the high accelerations result in substantial inertial loads, which the floor structure must be capable of withstanding. More accurate knowledge of the masses therefore results in more realistic loads.

7.3.1.4. Wing and empennage models

For improved accuracy when modeling the introduction of the wing loads into the fuselage, it is advisable to take into account the wing structure in detail as well. As shown in section 6.1, some wing modeling capabilities are provided in FUGA. It is therefore possible to provide wing meshes in addition to the fuselage mesh using the established structural component mesh generation approach, based on the wing geometry model. The resulting mesh for the main wing is given in figure 7.3.9.

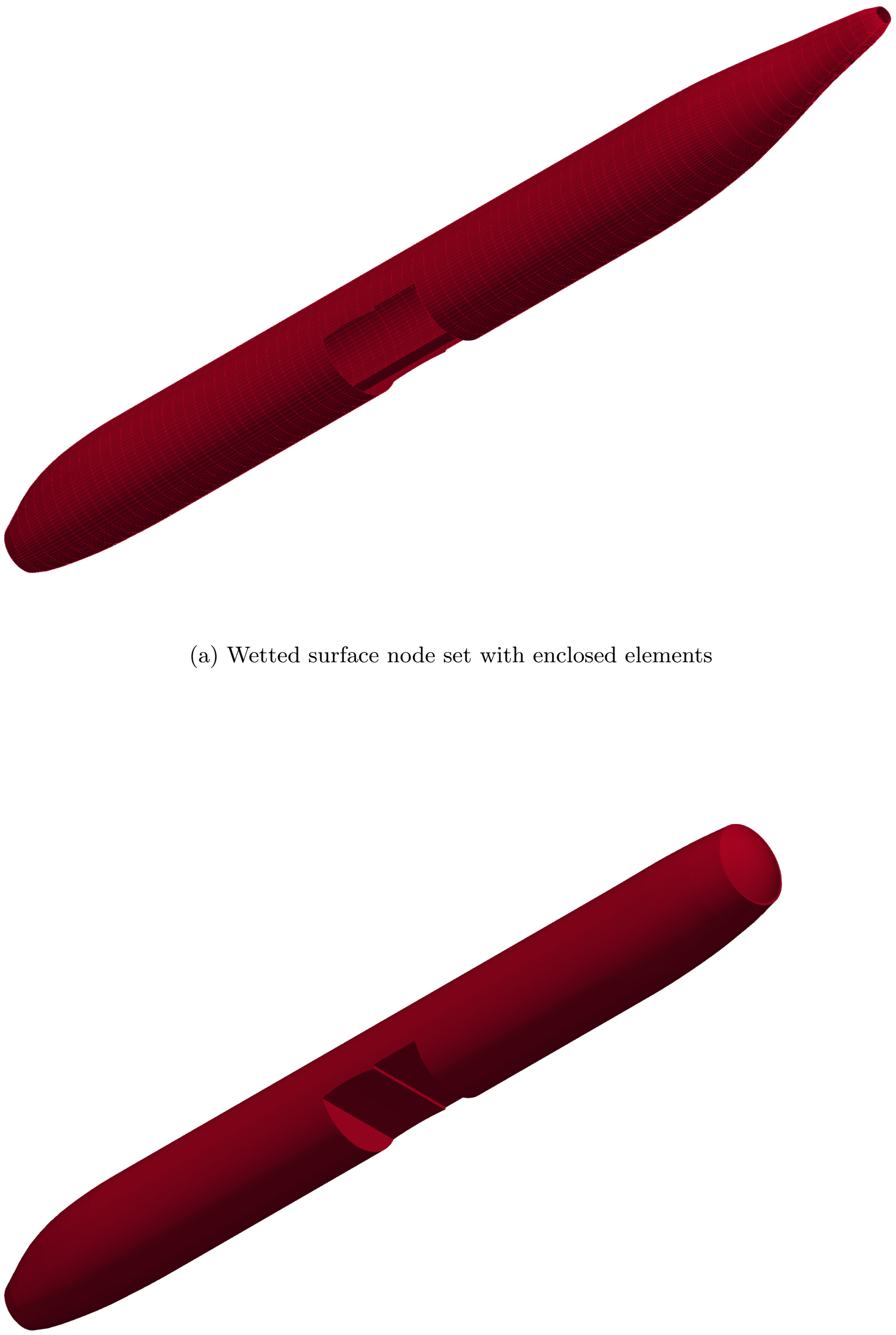
Visibly, the model contains structural details of the wing including spars and ribs as well as stringers, which are represented by beam elements. The properties are assigned according to CPACS specification. Once again, the forces and moments at the load reference axis points can be introduced at the



(a) Example placement of load reference axis points

(b) Example cut loads at reference axis points (F/N)

Figure 7.3.7.: Cut load application via load reference axis points



(a) Wetted surface node set with enclosed elements

(b) Pressure vessel element set

Figure 7.3.8.: Model subsets for load application

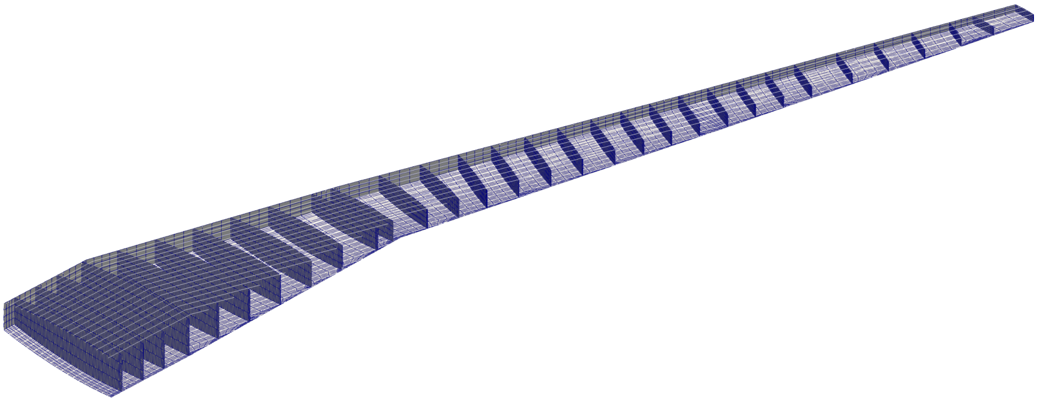


Figure 7.3.9.: Wing mesh view (with backface culling applied)

nearest rib via RBE3 elements. The meshes for the empennage are created analogously.

Rigid coupling (RBE2) elements are used to couple the wings to the fuselage mesh. To this end, the 150 closest pairs of nodes between the fuselage and respective wing meshes are identified and their translatory degrees of freedom coupled.

7.3.1.5. Full configuration analysis

Combining the component models described in the previous sections yields a GFEM of the overall aircraft. Applying the cut loads shown in figure 7.3.7b and a pressure difference $\Delta p = 60.165 Pa$, the model can be solved using the MSC Nastran solver. Differently from Scherer et al. [SK+13], who apply single point constraints to suppress rigid body motion, inertia relief is applied here. This has the advantage that reaction forces at the fixation are avoided. The stress and displacement results of the linear static analysis run of the full model loads are given in figure 7.3.10.

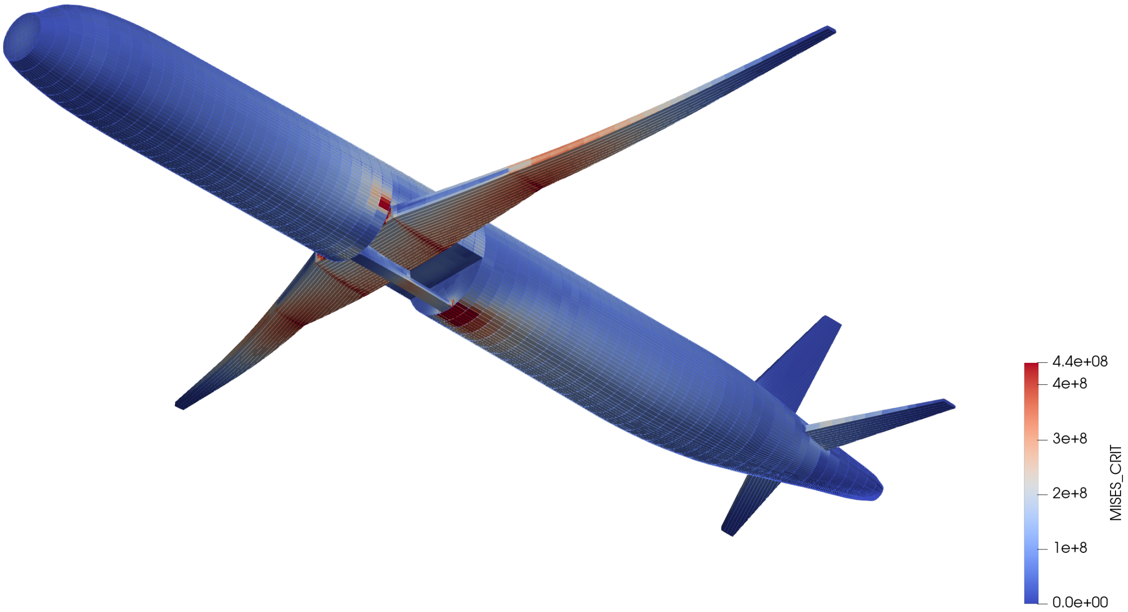


Figure 7.3.10.: Deformation and critical von Mises stresses for full configuration model ($\sigma_{mises,crit} / \frac{N}{m^2}$)

The results shown are computed using an unsized structure based on imaginary loads. Nonetheless, the overall response is qualitatively comparable to the analysis results presented by Scherer et al. [SK+13] for a 2.5g load case, also given in figure 4.2.8. Notable differences occur at the transition of the center wing box and the keelbeam to the base structure. This could be due to a larger difference in stiffness resulting from different material thickness assumptions for the keelbeam, or a smoother transition between the components in the model by Scherer et al. due to the introduction of additional structural details to the model such as stiffeners for the bulkheads, lateral panels on the landing gear bay, or triangular panels at the end of the keelbeam. In addition, the deformation of the HTP is notably larger in figure 7.3.10, which suggests a higher loading and thus a higher bending moment at the fuselage center. Finally, the assumption of a constant load distribution along the fuselage length may lead to an exaggerated influence of the forward and aft regions on the bending moment, as well.

To enable a static sizing of the skin panels, the different failure criteria for each stringer bay must be evaluated to compute utilization factors $UF_i = \frac{\sigma_i}{\sigma_{allow,i}}$. To this end, the failure criteria for static strength based on the VON MISES stress and single panel buckling after Bruhn [Bru73] are evaluated. A global yield strength for aluminum $R_{p0.2} = 441.216MPa$ is assumed. No safety factors are taken into account. The resulting critical utilization factors are given in figure 7.3.11. The results corroborate the previous observation that the loading around the center fuselage area is high compared to the results presented by Scherer et al., which leads to a large region where $UF_i > 1$. Furthermore, some of the window elements towards the center of the fuselage with a low aspect ratio due to left-out stringers have a critically high UF.

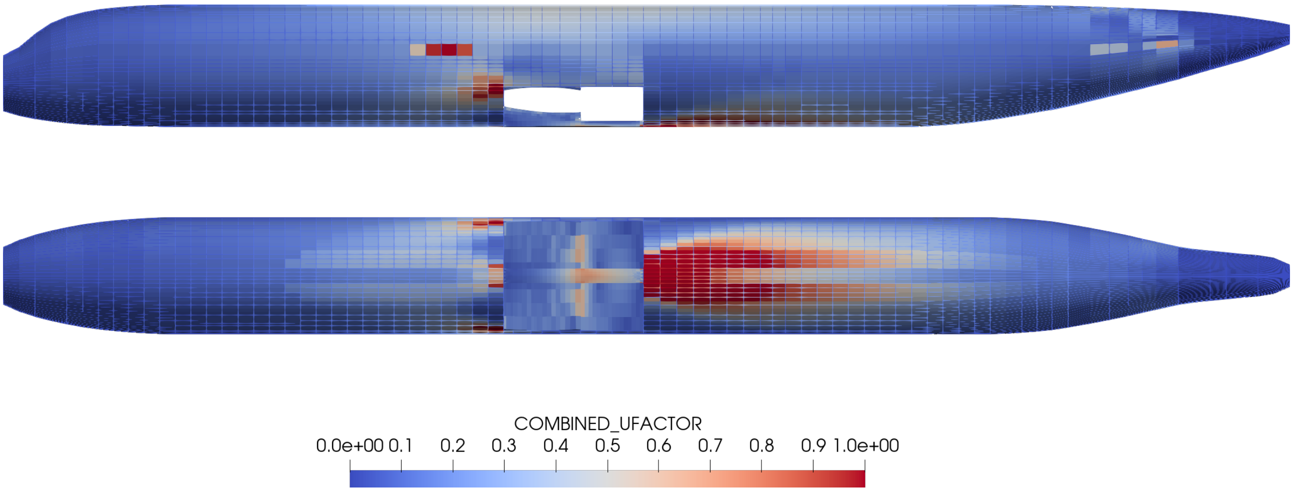


Figure 7.3.11.: Critical utilization factor for stress solution

To further support the understanding of the results, it is useful to also analyze, which failure criterion yielded the critical UF for a given skin panel. This information is provided in figure 7.3.12. As expected, the lower part of the fuselage is primarily driven by buckling, whereas static strength is critical in the upper part. That said, buckling is also critical in some of the skin panels in the window row, which further highlights the adverse effect of the decreased aspect ratio.

The above shows that the GFEMs built from the structure designed and modeled using FUGA can be used to assess the structure in order to inform sizing decisions. However, as only a single load case is considered, the available information is not sufficient, to reliably size the fuselage. Instead a variety of load sets should be available, in order to cover the entire flight envelope of the design. In a collaborative design process, this information is typically made available by dedicated loads experts. Thus, it is nevertheless shown that, as required by **working hypothesis 2**, suitable geometry models for structural sizing using GFEM can be generated using the KBE-driven approach.

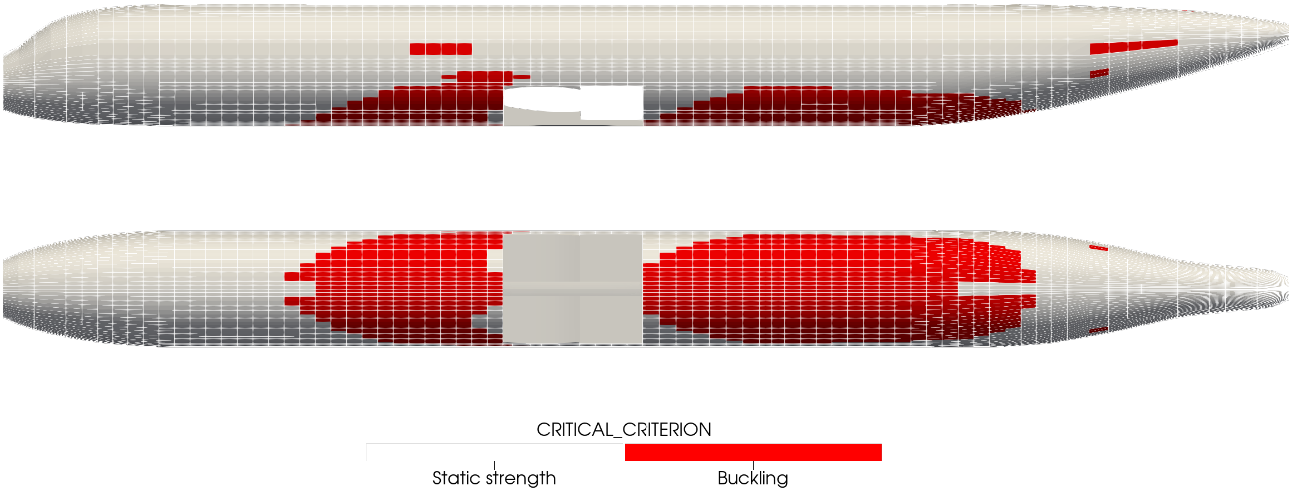


Figure 7.3.12.: Critical failure criterion for stress solution

7.3.1.6. Knowledge graph exploration analysis

Analogously to section 7.1.1.4, the coverage of the knowledge graph is considered once again based on the graph chart given in figure 7.3.13. Again, the node positions and layout are retained from the original outside-in system MCG in figure 5.2.2. Compared to the design process chart in figure 7.3.13 the density of nodes visited in the middle third representing the design rules is reduced. This is because only those design rules are triggered, which provide details required for the GFEM generation. The determination of cabin mass points is omitted for this example, since the determination of the mass values surpasses the capabilities provided by the baseline outside-in system. As a result, only very few rules on the left side of the graph, which contains the cabin design and model generation rules, have been evaluated. In contrast, the number of rules evaluated in the bottom right corner, which represents the structural geometry generation is increased significantly.

Given the staggered nature of the GFEM model generation process described above, more system evaluation steps are required than in the previous example. They are given in table 7.3.1. Notably, aside from the mass points, the wing model generation is skipped as well. Since no rib data is available in the baseline CPACS file, these modeling steps, too, would require additional design rule sets, which are, however, not included in the baseline outside-in design system considered here.

Table 7.3.1.: System requests and performance data for GFEM problem

Request ID	**Description**	**Rule count**	**Timing**	**Cumulative timing**
i			t_i/s	$\sum_0^i t_i/s$
0	Initialize	120	0.46	0.46
1	Build base grid	118	15.11	15.56
2	Build floor grid	19	0.54	16.10
3	Compute center fuselage cutouts	15	1.42	17.53
4	Build detailed bulkhead geometry	7	0.97	18.50
5	Build center fuselage geometry	5	8.46	26.96

The process again begins with the system initialization. The rule count remains unchanged compared to the design example, since the same input data is used. Consequently, the execution time is very similar as well.

Next, the base grid is built. This is the most time consuming step, since it includes the computation of the intersection points with the frame and stringer intersection definition vectors and the fuselage surface. Aside from the evaluation of the Boolean operation itself, this also requires leveraging a substantial subset of the design rules to determine e.g. the frame and stringer distributions. Further-

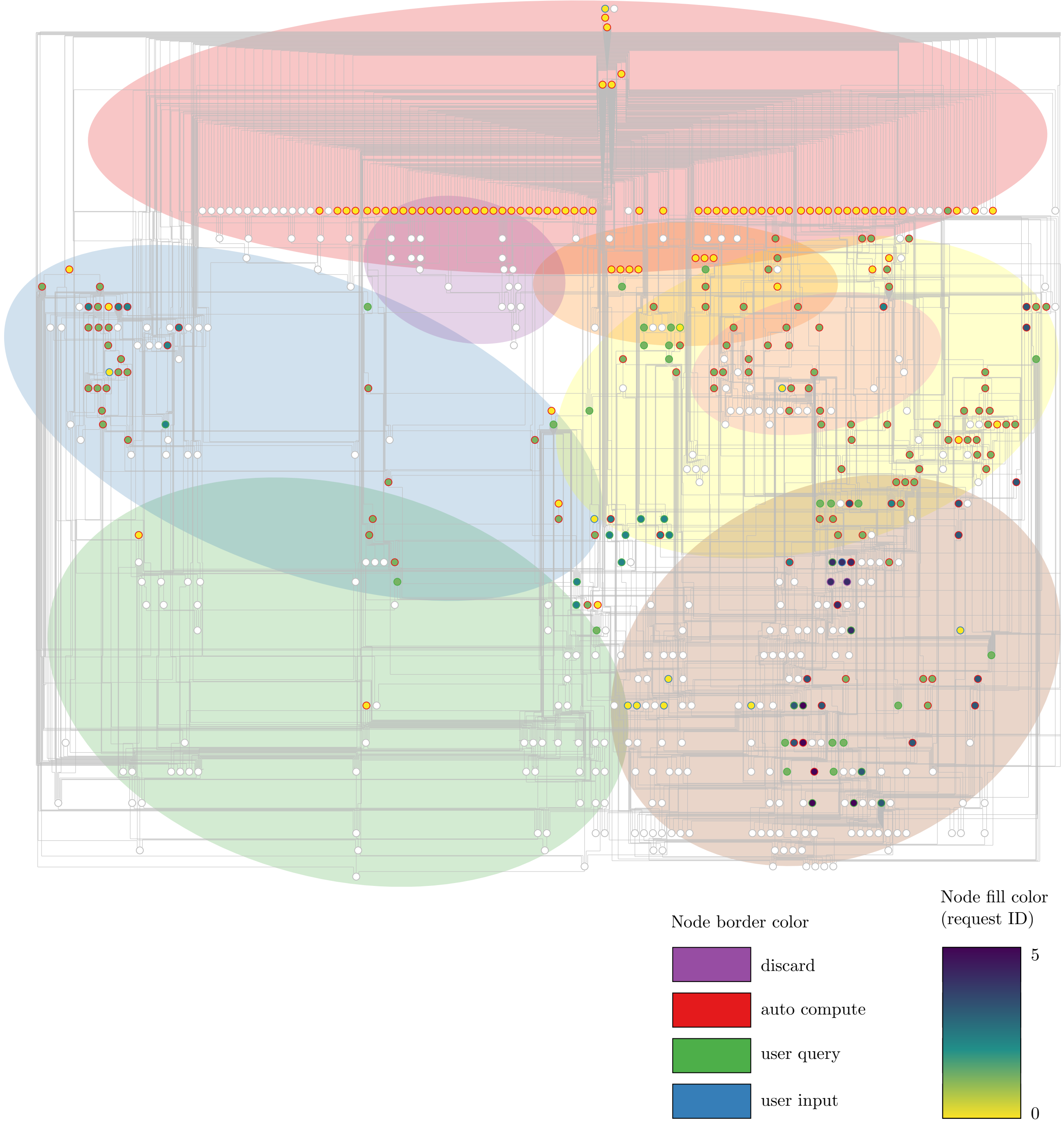


Figure 7.3.13.: Knowledge graph execution chart for GFEM generation

more, the cross-section or thickness data as well as materials must be determined to provide element properties. As a result, the base grid building step requires the largest number of rule evaluations, excluding the initialization. That said, the number of rules evaluated is still substantially lower than for the design synthesis in table 7.1.2. This shows that only those design rules were evaluated, which are essential to the task at hand.

Nevertheless, a performance benefit in the subsequent steps can be identified, since a lot of the necessary input data is already available in the data repository. For instance, the floor grid can be assembled in the next step based on a ready-made stringer- and frame layout. Since the exit layout is also known, the determination of the seat positions, which are inputs for the longitudinal beam definitions, also requires only very few additional rule evaluations.

Even fewer rules are necessary to compute the center fuselage cutouts. In this step it is determined, which elements to delete from the base grid to subsequently include the detailed center fuselage model. Despite the lower number of rules involved, this operation is in fact more computationally expensive than the floor assembly because Boolean geometry operations are again required to determine the nodes, which lie inside the space of the center fuselage area.

In the two final steps the component geometry models are built based on the updated base shapes. As such, only very few rules, linked directly to the component model generation, are triggered. However, since Boolean operations are necessary once again, the operations are comparatively computationally expensive.

Nevertheless, overall, the determination of the GFEM geometry remains on a similar scale as the design synthesis in terms of computational cost. Even though fewer rules are evaluated in total, the overall run time is nonetheless slightly increased, due to the higher share of rules related to geometry operations, which are more expensive to compute. A further caveat is that neither the mesh generation for the detailed components nor the FEM solution are included in these times, as they are not controlled by FUGA rules, but by an overarching control code. Consequently, the mesh generation run times must still be added to the model generation times given in table 7.3.1. Nevertheless, the available numbers show that the necessary geometric data and shapes, which can be used as inputs for optimized mesh generation tools developed by disciplinary experts, can be provided quickly using the KBE methodology.

7.3.2. Immersive visualization for human factors analysis

In section 7.3.1, it was shown, how the KBE-driven modeling approach can be applied to provide GFEMs for structural sizing. However, it is furthermore stated in **working hypothesis 2** that the approach should be capable of supporting a variety of disciplinary analyses related to the fuselage.

Human factors analysis using VR is introduced in section 4.2.3.2. The field is of high interest to many stakeholders, e.g. to rapidly assess customization options, improve customer experience or as a sales tool. However, in early design phases the exploitation of the potential is constricted due to high requirements of geometric detail, which can not typically be met at this point of the product development cycle. Once again, the KBE driven design and modeling approach implemented in FUGA can be leveraged to make more detailed geometry models available sooner in the development process.

The human factors model is a suitable choice as the second use case to illustrate the disciplinary model generation capabilities of the knowledge-based approach because it complements the previously discussed GFEM in many ways. For one, it is a use case that is primarily driven by the cabin, whereas the GFEM is primarily driven by the structure. Furthermore, the challenge during the generation of the GFEM model is the effective abstraction of the geometry, i.e. to provide component models at low levels of fidelity to reduce computational cost. The human factors model, on the other hand, needs to provide high amounts of detail in order to be sufficiently immersive to get useful responses from human test subjects or convince potential customers. In this way, the use cases cover two ends of the spectrum of requirements for multi-fidelity geometry.

The model by Fuchs et al. [FB+21] shown in figure 4.2.10 is used as a reference for an automatically generated cabin model for VR applications. In this model, textures and animations are integrated via hard-coded features of the external component models. Consequently, they are not considered part of the design process and thus not taken into account in the following. Instead, the goal is to provide models of the fuselage geometry and cabin layout in such a way that they can be used by experts in the field to quickly set up their respective analyses. This implies that:

- Models shall be provided as 3D visualization meshes either in a 3D graphics environment or a video game engine,
- Meshes of the CAD geometry shall be provided, which can be rendered efficiently to not deteriorate performance, but are also sufficiently detailed to ensure an immersive user experience,
- Meshes of sub-components shall be traceable to their original CPACS definition e.g. via their uID, to enable targeted additional manual processing or integration of metadata for interactive data display as shown by Fuchs et al. [FB+21].

The outcome is a design and visualization pipeline, which can read CPACS inputs and generate meshes in VTK via FUGA and provide the model to Blender via the Blender Python scripting interface. Blender, in turn, offers options for manual model processing and enables the export of the model to the Unity video game engine using the glTF or FBX formats.

7.3.2.1. Visualization mesh generation

As described in section 6.1, a hybrid approach is adopted to describe the fuselage and cabin geometry, which combines CAD models with triangulated meshes. The overall layout can be assembled from the triangulated components models which can be retrieved, for cabin components such as seats, by simply transforming copies of the mesh, as described in section 6.1.3. For CAD-based models, the Delaunay mesh generation algorithm from OCCT is leveraged.

As discussed in section 4.2.1.2, the requirements for visualization meshes are profoundly different to FEM analysis meshes. Most obviously, the GFEM mesh is dominated by quadrilateral elements, whereas the visualization mesh consists exclusively of triangles to allow for efficient rasterization. Furthermore, the goal of the FEM mesh is to model the structural behavior, whereas the visualization mesh is meant to accurately represent the underlying mathematical surface. Consequently, visualization meshes are typically coarse where the geometry is flat and refined in curved regions.

The meshing parameters provided by OCCT allow the user to control how closely the mesh should follow the underlying geometry, where higher accuracy requirements will result in a finer mesh with more triangles. The parameters include the angular deflection α_{mesh} and linear deflection d_{mesh} of the mesh as illustrated by figure 7.3.14. They can be specified both on the boundary and in the interior of a face. Furthermore a minimum triangle edge length is specified to prevent infinitely small triangles.

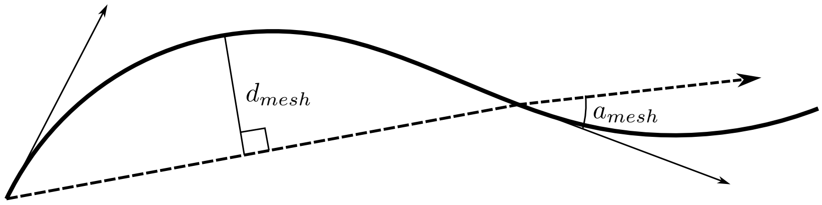


Figure 7.3.14.: Linear and angular deflection (on the boundary) for OCCT meshing algorithm (after [OCC23b])

The effect of the meshing parameters on the number of faces of the resulting mesh for the example of the fuselage skin with cutouts is shown in figure 7.3.15. The substantial impact of the parameters on the number of faces is visible. In particular, very small angular deflections result in very high numbers of faces and thus in meshes, which are expensive to compute and process. A significant influence of the linear deflection can only be detected for angular deflections $\alpha_{mesh} > 0.5$.

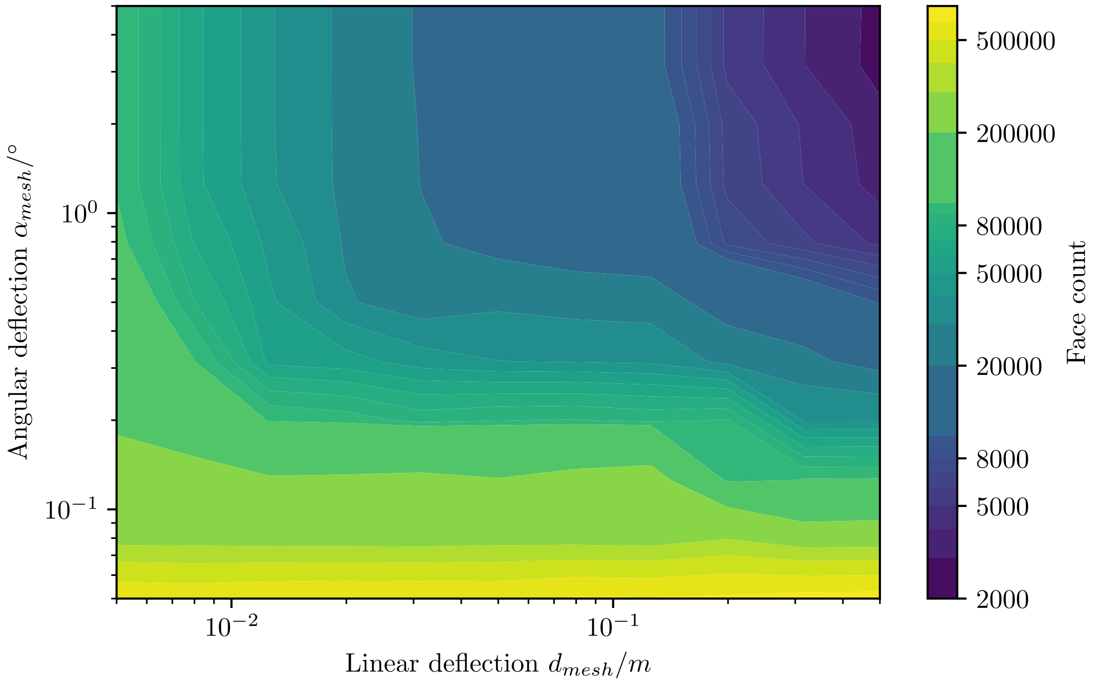
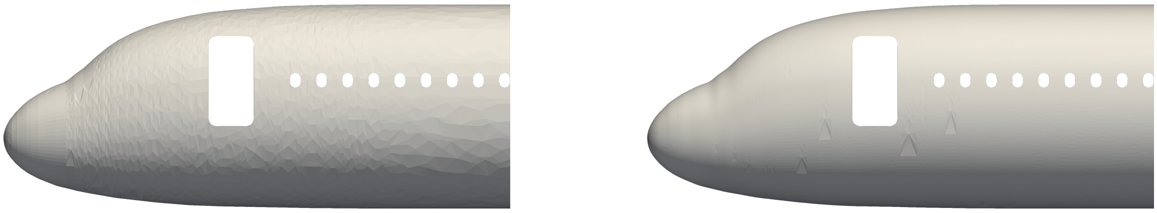


Figure 7.3.15.: Number of faces due to linear and angular deflection

A comparison of two resulting meshes with flat shading applied is given in figure 7.3.16. The first mesh has a face count of approximately 50.000, which is increased by a factor of 10 for the second mesh. The finer mesh clearly appears smoother as individual triangles are more difficult to discern, aside from some artifacts in the lower part of the fuselage. Meanwhile, in the coarser mesh, the triangular faces can be seen very clearly, resulting in a less credible appearance. The overall shape of the fuselage is, however, represented quite well, as evidenced by the very similar outline compared to the finer mesh.



(a) 53854 faces

(b) 573812 faces

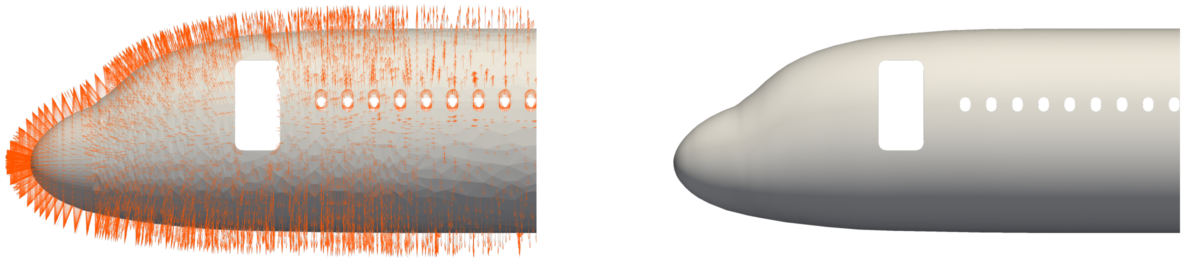
Figure 7.3.16.: Comparison of shaded meshes with different face counts

Even though the number of 500.000 faces may not appear very high compared to the state of the art in modern gaming[1], it is worth noting that this is only for a single component mesh, i.e. the skin. The final model will, in addition, contain separate meshes for each structure and cabin component.

[1] Vehicle models in 2022's Gran Turismo 7 consist of 500.000 faces compared to 300 faces in the original Gran Turismo from 1997 [Smi22].

Consequently, it is advisable to keep poly counts low for all component meshes in order to optimize performance.

In order to improve the appearance of coarser meshes, different shading techniques have been discussed in section 4.2.1.2. Common to all of these techniques is that the normal vectors of the surface must be provided in addition to the node positions and cell connectivity. A common technique to determine the normals at the mesh points, which is implemented e.g. in the VTK library, is to compute the average of the adjacent cell normals weighted by the area. This procedure has the drawback that the accuracy of the normals depends on the mesh resolution, as finer meshes result in more accurate normals. Since the underlying CAD geometry is known, it is therefore preferable to use the known uv-coordinates of the mesh nodes on the CAD faces and determine the normal vectors directly from the geometry. In this way, accurate normals can be obtained independently from the selected meshing parameters. The resulting normal distribution for the coarse mesh is given in figure 7.3.17a. Applying Gouraud shading yields the result shown in figure 7.3.17b.



(a) Mesh point normal vectors

(b) Gouraud shading applied

Figure 7.3.17.: Normal vectors and shading result (53854 faces)

Visibly, the quality of the visualization surpasses the quality of either of the previous visualizations using flat shading, including the fine mesh. It can thus be concluded that providing meshes, which are augmented with normal data, is typically preferable to providing very fine meshes for effective visualization of CAD data. This is also advantageous, given the high number of separate component models provided in FUGA, e.g. due to high frame and stringer counts, which can quickly lead to a high accumulated face count. Consequently, the coarse mesh generation settings have been adopted as defaults for FUGA, where $d_{mesh} = 0.01m$ and $\alpha_{mesh} = 0.5°$. Nevertheless, the possibility to request higher resolution meshes is retained via optional inputs.

7.3.2.2. Overall visualization mesh in the Blender 3D graphics environment

The outputs of the mesher, including the point normals, can be transferred to a polygonal mesh data type in VTK. Hierarchical CAD tree structures using e.g. compounds in OCCT, can furthermore be mapped to a multi-block data structure in VTK. In this way, the hierarchy of the CAD model and thus the accessibility of individual component sub-models can be retained.

The pool of component types, which should be contained in the mesh can be determined based on the meshes shown by Fuchs et al. [FB+21] and includes the floor-based components of the cabin, e.g. seats and monuments, and the secondary structure including sidewalls, OHSCs, and ceiling panels. Furthermore, elements of the primary structure are contained, including frames and the skin. The floor is represented by a floor plane. The cabin system details shown by Fuchs et al. are omitted here, since they exceed the scope of the modeling capabilities of CPACS as of version 3.4. In return, a model of the full fuselage is assembled, expanding upon the barrel-based models by Fuchs et al. A realistic depiction of the exterior not only promises an enhanced user experience, but also caters to applications involving the boarding process as shown by Gopani [Gop21].

As mentioned before, the need for manual modification of the models often arises during the setup of human factors analyses. A good example is given by De Crescenzio et al. [DB+19], who manually apply different sets of textures to a given geometry. Thus, the goal is formulated to provide the visualization mesh in a 3D graphics environment in such a way that facilitates the modification of models. Fuchs et al. [FB+21] apply the open source software Blender, which has been introduced in section 4.2.1.2.

Since FUGA natively provides the mesh in VTK format, an interface between the format and Blender is required. Two major requirements are the correct transfer of the mesh point normals and the mesh hierarchy. Out of all formats listed in section 4.2.1.2, the OBJ and glTF formats have been taken into consideration, based on these requirements. Both provide support for point normals and interfaces are included out of the box in both VTK and Blender. That said, support for glTF in the VTK library is currently still under development, whereas OBJ is an established format, which is why the latter has been selected.

That said, a challenge with either of these formats is the transfer of the model hierarchy. Typically, the individual component meshes are merged into a single mesh, which can be exchanged as a single file. However, if the separation of the component models is to be retained, a separate file for each component mesh must be written. To reflect the hierarchy, the mesh files can be stored in a custom-built folder structure. This approach requires a customized reader that can explore the folder structure and reconstruct the hierarchy tree, which may be implemented in Blender using the available Python scripting interface. Aside from the higher risk of errors due to customized output and input interfaces, this approach furthermore suffers from poor performance of the export and import operations for the plain text OBJ files.

A second option is therefore to access the VTK library directly from the Python scripting interface in Blender. Here, the established VTK multi-block format can be used for data exchange. The format is capable of storing data in binary form, which accelerates the data export and import operations. Using the VTK library, the mesh data arrays including any point or cell metadata, such as normals, can be made available to Blender directly via numpy arrays. Furthermore, the model hierarchy is contained in the data structure. As such, the only additional step is to traverse the multi-block data structure and recreate the hierarchy using Blender container types. Consequently, only one additional interface must be implemented, which significantly reduces the risk of errors.

The overall model as rendered by Blender is shown in figure 7.3.18. It can be seen that the smooth surface meshes of the CAD-based objects have been retained during the transfer. Furthermore, all types from the pool of components listed at the beginning in this section can be identified.

Figure 7.3.19 shows the component tree corresponding to the model in figure 7.3.18 in Blender. The individual components are subdivided into groups and can be identified via their uID, which is contained in the name. In this way it can be ensured that the components are easily accessible to Blender users. The model and the hierarchy can be stored and exchanged using e.g. a native Blender (.blend) file or an FBX file.

7.3.2.3. Interactive model exploration in the Unity video game engine

Having the visualization mesh available in Blender not only makes it possible to manipulate it manually, but also gives access to several additional exchange formats, which are not supported by the VTK library. Most prominently, the proprietary FBX format is supported, which is a popular exchange format for 3D assets in the Unity engine. However, whereas the model can be imported easily into Unity, some additional steps must be taken in order to prepare the model and its environment for interactive exploration.

Add user character model to enable interactive movement A user character model must be added to provide the physics rules for the character movement. This enables the user to move through the model and explore it from a first-person perspective. Off-the-shelf solutions are available for Unity, which can simply be imported into the scene (e.g. [Gra22]).

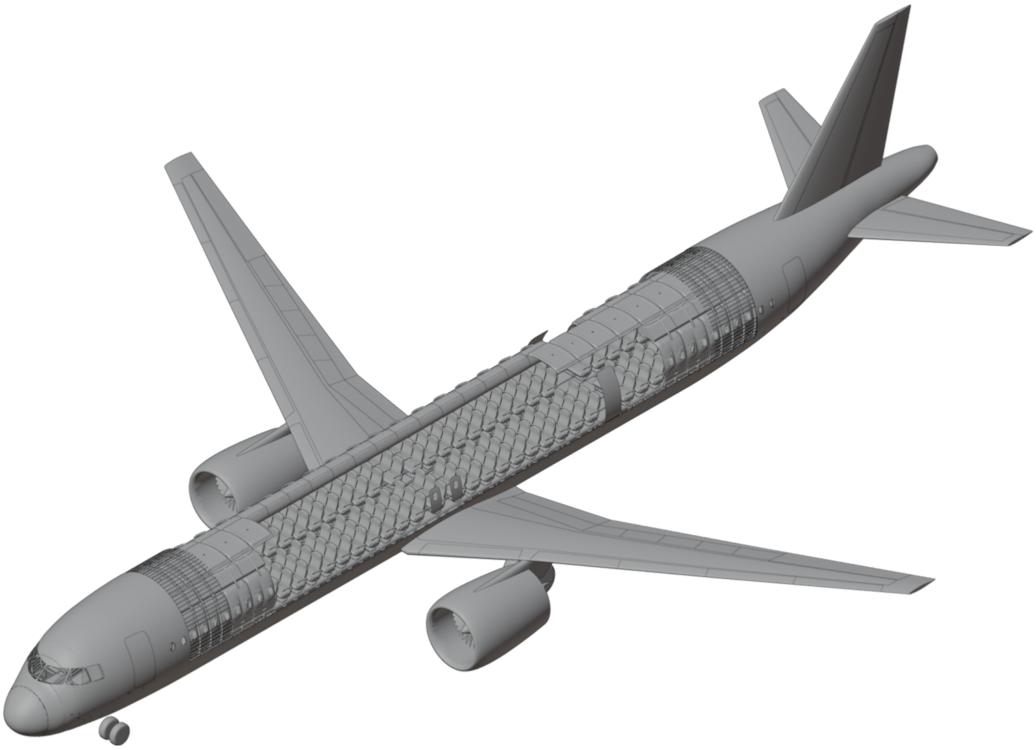


Figure 7.3.18.: Cut view of the overall configuration model in Blender

Add floor surfaces and colliders The user character model is subjected to gravity, which means it is necessary to define floor surfaces to walk on and keep the character from falling into infinity. This typically includes the cabin floor panels, but could also include the ground. To this end, an additional ground plane can be added to the environment. The vertical position of this plane can be determined via a FUGA rule based on the model bounds. Instead of a simple plane, an external model, e.g. of a hangar, could also be imported to represent the environment. The floor panels are available as individual meshes from the FBX import. Thus, their properties can also be manipulated to let them support the user model.

By default, the user character will not collide with another object in Unity, unless a mesh collider component is activated. This step must be completed both for the ground plane and the floor panels. To enable collisions, e.g. with floor components, the collider can furthermore be activated for the respective component meshes.

Add lighting Unity does not provide global lighting by default, but instead requires lights to be placed explicitly in the model. By default, a single directional light source is placed in the scene, which acts as a “sun” to provide illumination. However, this approach results in closed spaces, such as the cabin, remaining in the dark. Therefore, additional light sources must be placed in the model in order to ensure proper illumination of the interior. Once again, FUGA rules can be leveraged to determine the positions of the light sources, e.g. based on the ceiling panel positions. Blender provides the possibility to generate light sources automatically via the Python scripting backend and the results can be shared with Unity via the FBX format.

Figure 7.3.20 provides a collection of impressions from the resulting interactive scene in Unity. In figure 7.3.20a, a view of the aircraft exterior is provided. Once more, very smooth surfaces can be achieved by enhancing the relatively coarse mesh from the CAD data with the surface normals at the mesh points. The available aircraft details, which include the cabin and cockpit windows as well as the door cutouts and the engine help enhance the realism of the visualization, compared e.g. to the example given by Gopani [Gop21]. To give the impression of accessibility, the main door components

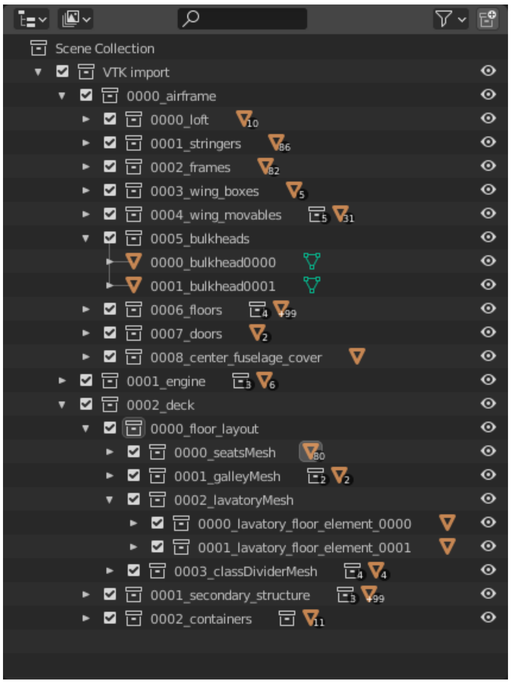


Figure 7.3.19.: Model hierarchy in Blender

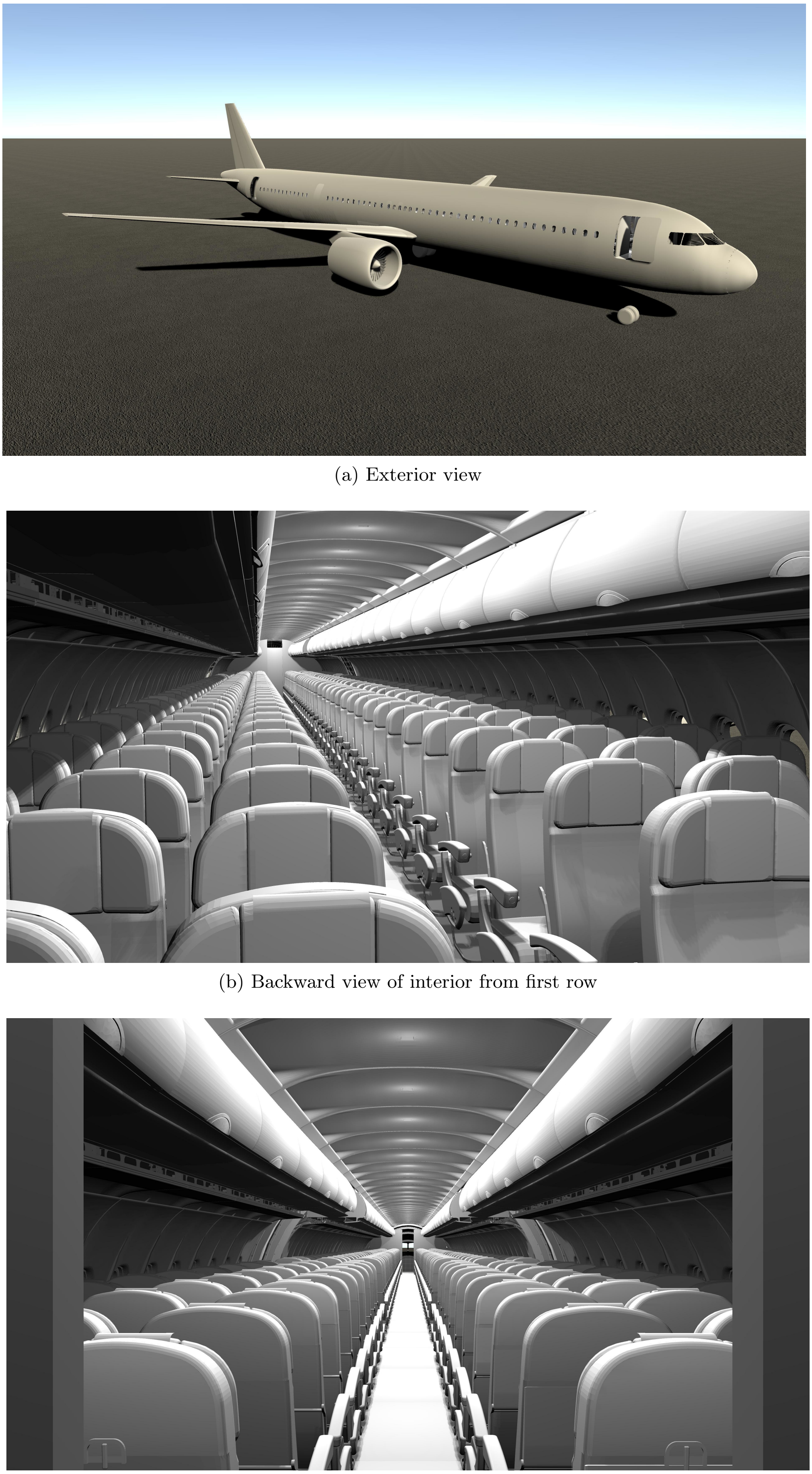
have been moved to appear open. This is easily accomplished as the individual component meshes are accessible in Unity as well.

The interior of the fuselage is shown in figures 7.3.20b and 7.3.20c. Unlike the fuselage, the majority of the cabin components is provided using external meshes in STL format. Consequently, the shading quality of the surfaces is not optimal as no normal vectors have been provided. Nonetheless, a good level of detail can be achieved, which closely matches the reference by Fuchs et al. [FB+21]. Aside from the aforementioned lack of cabin subsystem details, which have not been included in FUGA or CPACS yet, the main omission is the lack of different material properties and coloring, based on the component type. This is, however, not a major issue, since the individual component models can be accessed and manipulated in groups, as shown previously for the floor panels.

Apart from that, all cabin components found in the reference are present, including seats, sidewalls, OHSCs and ceiling panels. The floor-based monuments are also modeled, albeit as boxes. This means that, while they can convey a sense of the available space, they do not contribute significantly to an enhancement of the immersion via details. This simplification is, however, deemed acceptable, as no monuments are shown at all in the reference.

Nevertheless, this again brings up the issue of availability of detailed cabin component models, which can be identified as a key enabler for the generation of effective and immersive human factors models. Closer examination of figure 7.3.20c reveals the lack of suitable panel models around the exit behind the wing as well as a missing cockpit door. The result in both cases is that the underlying primary structure is exposed, which is clearly detrimental to the immersion. Thus, more detailed component models, which cover the entire cabin, must be made available, in order to enable truly immersive human factors analyses. In this context, including textures in the models should also be taken into consideration, as their usefulness has already been demonstrated e.g. by De Crescenzio et al. [DB+19] and Engelmann, Drust, and Hornung [EDH20].

To this end, 3D scanning has recently become an interesting alternative to creating the detailed component models manually, as exemplified e.g. by Rauscher et al. [RB+21] and Fuchs et al. [FB+22]. Additionally, it is possible to implement a new rule set in FUGA to model different types of components parametrically as shown for the seats in section 6.2.3.4. While the resulting models in either case will most likely not match the quality and detail of hand-made meshes, they could be of use to cover the distance between highly detailed models and bounding box representations.



(a) Exterior view

(b) Backward view of interior from first row

(c) Forward interior view from rear exit area

Figure 7.3.20.: Views from interactive scene in Unity

Irrespective of these issues, the above results show that the knowledge-based geometry modeling methodology can already be applied successfully to orchestrate an available set of component models in order to assemble a geometric representation of an aircraft fuselage and cabin, which provides a suitable starting point for the setup of a human factors analysis. They thus further support **working hypothesis 2**, confirming that the approach can support the generation of interactive visualization models in addition to the GFEM shown in section 7.3.1. Whereas further work on the model by the disciplinary expert might be required to improve the effectiveness of the analysis, the application of FUGA to provide the foundational model leads to a significantly reduced ambiguity w.r.t. the overall configuration of the fuselage and cabin geometry and thus substantially reduces the risk of inconsistencies in a collaborative setting.

7.3.2.4. Knowledge graph exploration analysis

Analogously to the GFEM generation, the coverage of the knowledge graph for the visualization mesh generation is given in figure 7.3.21. Unlike the GFEM, the process for the visualization mesh is fairly standardized in FUGA, which means the user only needs to make very few requests for models of specific subcomponents, as listed in table 7.3.2. Nevertheless, the coverage of the graph is significantly higher than in all previous examples.

Once again, the profiling data for the initialization remains the same and rules in the first topological layers are predominantly evaluated. For the first user query, the cabin model is assembled. The rule count in this step is higher than for any other, since much of the design details, which are not available, must be determined, triggering rules from the design rule sets. This is visible in figure 7.3.21, where, in the first step, not only most of the cabin design and modeling rules on the left are evaluated, but also a large number of structural design and geometry generation rules on the right. The drivers for the computational cost of the first query are similar to the design problem in section 7.1.1.4, i.e. the determination of the frame curves and the retrieval of the external models from the hard drive.

Table 7.3.2.: System requests and performance data for visualization mesh problem

Request ID	**Description**	**Rule count**	**Timing**	**Cumulative timing**
i			t_i/s	$\sum_0^i t_i/s$
0	Initialize	120	0.46	0.46
1	Build cabin model	161	14.10	14.56
2	Build engine model	16	0.60	15.16
3	Build airframe model	128	432.74	447.90

In contrast, the generation of the engine model is very fast, as it only requires the generation of a small number of NURBS surfaces, but no Boolean operations. In the outside-in baseline system used, it is furthermore based exclusively on CPACS inputs without interconnection to the fuselage or cabin design, which means no design rules must be triggered.

The generation of the airframe geometry model, which includes the computation of the cutouts for the doors and windows as well as for the center fuselage area, is significantly more demanding in comparison. The corresponding nodes are found in the bottom right corner of the chart. The coverage in this area is visibly higher than in the previous examples. Nevertheless a number of nodes are found, which have not been visited. They represent the multi-fidelity options for stiffener representation and cutouts. The benefit of these options is highlighted by the associated run times. In particular, the evaluation of the cutouts on the extruded stringers requires around one third of the overall run time, which is now in the order of minutes instead of tens of seconds. Furthermore, the determination of the parametric curves of the stringers on the fuselage surface as well as the extrusion of the cross section and the application of the center fuselage cutouts each roughly take half a minute. This shows that the stringers form a critical path, even in a parallelized implementation. Aside from the stringers, the computation of the cutouts for doors and windows on the fuselage also makes a significant contribution

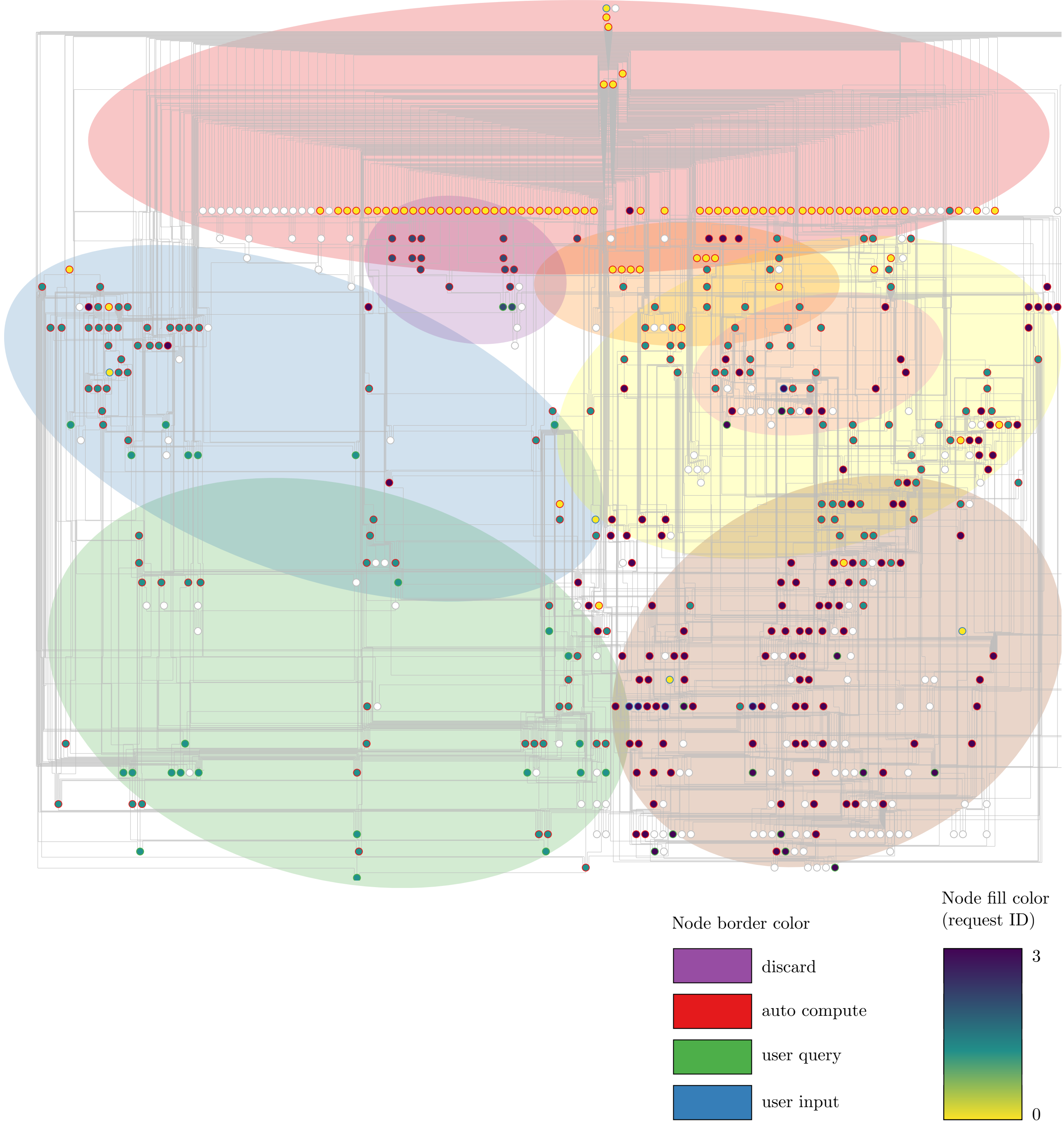


Figure 7.3.21.: Knowledge graph execution chart for visualization mesh generation

to the run time.

The reason for the substantial difference in performance between the frames and the stringers, despite being modeled using very similar procedures is the different approach to determining the trajectory curve, i.e. the intersection curve between the definition plane and the fuselage surface, which provides the basis for the extrusion. For the frames, the result of the intersection is typically an isoparametric curve, which means that the order and knot vector of the trajectory are the same as for the circumferential direction of the fuselage surface. In contrast, the stringer curves are not commonly isoparametric, which means they must be approximated, resulting in more complex curves with a higher polynomial order and more control points.

The above considerations once again highlight the value of the multi-fidelity geometry modeling approach, which makes it possible to bypass expensive operations if they are not necessary, as shown e.g. for the GFEM generation. Given that the stringers may not be essential e.g. for a cabin visualization model, there is also some potential for accelerating the visualization model generation process.

7.4. Discussion

In the application studies in this chapter the possibilities of the KBE-driven approach are further investigated. Broadly speaking, two dimensions can be identified, where the D240 baseline design from section 7.1.1.1 constitutes the origin of both axes.

The first axis, represents the versatility of the design capability. This axis is represented by the D380 use-case in section 7.1.2 as well as the tank integration and the BWB cabin integration examples in section 7.2. The D380 layout can be generated based on the same set of rules as the baseline, simply by manipulating the input parameters. It illustrates the big range of possible configurations, which can be supported by the baseline rule set, but also highlights that profound knowledge of the system is necessary to assure a consistent final design, even with two decks present. The example also exposes weaknesses in the approach of using “dead” external geometry models to represent cabin component models, which are not sufficiently flexible to adapt to significant changes of the surroundings. The adaptation of the sidewall panels to the cross-section of the fuselage is an example of this. Nevertheless, the two examples show that the approach can fulfill **working hypothesis 3**, since the missing structural and cabin data is filled in based on the given CPACS file and design parameters.

The tank integration example represents the next evolutionary step along the first axis. The vast majority of the design rules of the baseline system remains valid since the fundamental tube-and-wing configuration is retained, but the system is expanded to support the novel architecture and take into account the additional requirement of providing storage for a certain fuel volume in the fuselage. The similarities to the baseline configuration mean that a number of components, including cabin and primary structure can be designed to a very high level of detail.

This is no longer the case as one travels further along the first axis towards the BWB example from the AGILE project. The configuration is so substantially different from the D240 baseline that the structural design rule set is no longer valid. To still be able to provide the necessary inputs to the cabin design, the relevant interface nodes must therefore be determined using a mocking approach instead. The implementation of the mocking, which effectively bypasses large segments of the structural design rule set, results in a loss of detail in the design. Nevertheless, it is possible to provide a floor layout for the cabin. The two examples show the capability to support changes in product architecture with minimal changes to the knowledge-based system deployed through additional rule sets. This aligns with the requirements formulated in **working hypothesis 4**, showing that the proposed KBE methodology is also well-suited to support novel architectures.

The first axis is concerned with the design, which is why, geometry dependencies aside, the focus lies on the design rule sets and the effects of changed system composition and inputs. In contrast, neither the design inputs nor the system are changed going along the second axis, which is the model generation axis. Once more departing from the D240 baseline in section 7.1.1.1 geometry models are derived for a GFEM suitable for structural sizing applications and for a interactive visualization model for human factors studies, which are described in 7.3.1 and 7.3.2 respectively. The progression along this axis is illustrated by a graph exploration analysis in each case, which shows an increasing coverage of geometric modeling rules as the required level of detail is raised. The growth in coverage is accompanied by an increase in run time, in particular due to computationally expensive geometry operations. Conversely, whereas all design rules are executed in the baseline example to fill all possible CPACS nodes consistently, only the necessary subset of those rules is applied in the subsequent studies. This is a showcase of the capability of the KBE approach to deploy rules as required based on the incoming user query.

Based on the geometry assembled from the knowledge-based system, the analysis models are built automatically and deployed in a basic analysis to showcase the usefulness of the geometry to a disciplinary expert. The results show that the knowledge-based approach can indeed provide workable analysis models, thus confirming **working hypothesis 2**.

In addition to the above, some general observations can be made on how the user is expected to interact with the system in its current state. As shown by the exploration analyses, the KBE reasoner is capable of traversing the graph based on a user query if the node lies downstream from a set of given nodes. If this is not the case, the user can resort to numerical methods to iteratively solve for a target variable based on a number of input variables. The applicability of this approach is, however subject to some restrictions. For instance, both variables should ideally be floating point numbers or arrays, to support traditional root finding and optimization algorithms.

The solution of these kinds of problems, similarly to the resolution of loops in systems, where the MCG is not a DAG, require the user to carefully advance through the system to avoid loops in the FPG as well as reset node values based on findings in the iterative process. This introduces a complexity, which can presumably only be handled by very experienced users. Thus it would be desirable to introduce certain types of coordinator blocks, similar to the MDA or optimization blocks from MDAO, to the knowledge-based system, in order to control e.g. the resolution of a loop as shown in figure 7.1.6.

A second, related topic is the automated combination of rule sets. In FUGA, the user is currently responsible for finding and combining all necessary rule sets, supported by some helper classes, which offer pre-configured systems for the most common use-cases. This is acceptable for the number of rule sets available at this point. However, as more disciplines are added, e.g. to add parametric cabin component models or on-board systems, an automated solution for system composition based on a rule set dependency layer could be useful, which selects not only rules, but also eligible rule sets based on user inputs and queries. This would once again facilitate the application of the system to novice users.

Finally, the reliance on “dead” external geometry, is clearly identified as a major weakness of the current implementation. This is particularly drastic for the secondary structure. Whereas in the baseline configuration, the “damage” is limited to some specialized elements missing in the door neighborhood areas, configurations, which depart from the standard single-aisle cross-section are essentially not supported. Thus, a larger library of models, or even better a parametric component description is required.

It is nevertheless shown that the KBE methodology is, in principle, suitable to address all of the challenges posed by the above applications, confirming the **research hypothesis**.

8. Conclusion and perspectives for future research and development

8.1. Conclusion

The main goal of this thesis is to leverage knowledge-based engineering (KBE) techniques for generating design details and geometry, in order to provide consistent models for computational analysis of aircraft fuselages. The work is motivated by a gap identified between the level of design details typically provided by overall preliminary design tools and the level of design details necessary to build expressive numerical analysis models, which are required to enable end-to-end assessment of a new design using e.g. multidisciplinary design analysis and optimization (MDAO) techniques.

In the **research hypothesis**, the need for an integrated consideration of both the design and the model generation is formulated to enable a generation of design details, which is tailored to the geometry requirements of the use case, but remains consistent across disciplinary boundaries. Four working hypotheses are derived, which highlight the different aspects of the overall problem:

Working hypothesis 1: Description of aircraft fuselages and cabins as knowledge graph consisting of parameters and rules,

Working hypothesis 2: Capability to derive consistent geometry for analysis model generation at different levels of fidelity,

Working hypothesis 3: Provision of design capability to augment missing details,

Working hypothesis 4: Support for novel product architectures.

The KBE methodology applied here is implemented in the tool FUGA (Fuselage Geometry Assembler) using a graph-based approach, where design parameters, given by the Common Parametric Aircraft Configuration Schema (CPACS), are connected by design and model generation rules to form a directed knowledge graph. In reference to MDAO nomenclature, this graph is referred to as the maximal connectivity graph (MCG). Based on the MCG representation of the design and model generation system, techniques from graph-theory can be deployed to implement an inference engine, which can quickly determine solution paths based on data available in the system and requests for additional data made by the user. “Lazy evaluation” of the system, i.e. only evaluating rules if they are necessary to resolve a user request, results in significant performance benefits. Introducing the concept of rule sets, the MCG can furthermore be adapted depending on the type of design problem or aircraft architecture required. Thus, the implementation in FUGA fulfills the requirements stated in **working hypothesis 1**.

Applying the KBE methodology, rules to generate geometry models are formulated using state of the art computer-aided design (CAD) methods. By representing individual steps of the model generation process using separate rules, intermediate results of the model generation process can be exposed in the KBE system. This facilitates the provision of part geometry models at multiple levels of fidelity, such as curve or volume-based representations of stiffeners. The multi-fidelity capabilities, combined with the lazy system evaluation, not only allow the user to trade geometric detail for improved computational efficiency of the model generation, but also enable the creation of tailored geometry models for different disciplinary analysis use cases.

The approach is applied to generate a global finite-element model (GFEM) used for stress analysis and structure sizing and an immersive visualization mesh to support human factors analysis. The two cases are selected due to their substantially different focuses and requirements in terms of geometric fidelity. Whereas the GFEM is focused on the structural aspects and requires abstraction of the geometry, the cabin is the focus of the visualization mesh, which furthermore requires very detailed geometric representations. Profiling of both model generation processes reveals a substantial advantage in run time for the simplified geometry model as a result of the KBE approach. The analysis models are furthermore compared to the state of the art, and the fitness to perform relevant analyses in the field is assessed. Whereas the results of the analyses shown are sufficient to prove the applicability of the models generated and thus corroborate **working hypothesis 2**, the validity of the actual analysis results is compromised by a lack of necessary inputs e.g. loads for the stress analysis or certain component models for the immersive visualization, which are beyond the scope of this thesis.

That said, the capability to dynamically generate design details is identified as a key enabler to deploy the knowledge-based model generation process, especially in a collaborative MDAO environment, where the type and completeness of the incoming data may vary between projects. By leveraging the KBE methodology and a set of design rules, the available information can be analyzed to identify missing inputs, which are then added to the system. Differently from the geometry modeling rules, which use CPACS data as an input to generate geometric shapes, the design rules accept CPACS data and additional simplified design parameters to generate additional CPACS data. To this end, rule sets to generate structural and cabin design for conventional configurations are presented in depth. It quickly becomes apparent that the rule sets are inherently interconnected, e.g. via the positions of the exits, which are determined by the cabin layout and influence the frame positions, or by the cabin boundary, which is determined by the surrounding frames. Using the KBE approach, these interconnections can be resolved and a consistent design can be generated.

The versatility of the design rule sets is demonstrated using a short-to-medium range single-aisle configuration and a twin-aisle multi-deck configuration. It is shown that structure and cabin layouts can be generated for both configurations. That said, while the single-aisle design can be computed by evaluating the KBE system as-is, the more complex multi-deck configuration requires significantly more effort from the user. This is due to the fact that the rules are primarily designed to support single-deck use cases.

More user intervention may also be necessary to approach certain design problems for the single-aisle configuration. For instance, the problem of solving for an upstream parameter based on downstream input parameters must be assessed using iterative numerical techniques, due to the directed nature of the graph-based system formulation. The problem of eliminating the gap between the galley and the rear pressure bulkhead in the single-aisle use case by changing the length of the fuselage is one example for this issue. Whereas the numerical solution must currently be implemented at user level, it is supported by the KBE system, which facilitates the formulation of the function to be solved and assures that only those nodes are reevaluated, which are actually affected by a changed input variable. In many cases this results in very efficient problem formulations, which can be evaluated very quickly, lowering the computational impact of the iterative solution.

The design capabilities demonstrated fulfill the requirements of **working hypothesis 3** and can be deployed in different ways to either provide a complete consistent design of the cabin and fuselage or reduce the scope of the design to those aspects required for the analysis model generation at hand. Either way, designs for different models will remain consistent as long as they share the same rule sets and problem formulation.

The applicability of the aforementioned rule sets is restricted to conventional tube-and-wing configurations. The need to be able to reflect architectures of unconventional configurations is, however, described in **working hypothesis 4**. This capability is essential to address the emerging challenges in aviation. The graph-based KBE methodology allows for modification of the system behavior by assembling additional rule sets, which can be integrated in the system to replace or extend existing rules. To illustrate this capability, two examples are investigated. On the one hand, the conventional hybrid design system is expanded with design and integration rules for a cryogenic tank to store liquid hydrogen (LH_2) in the fuselage. On the other hand, a rule set is presented, which bypasses large parts of the conventional structure design rules in order to enable the design of a cabin for a blended wing-body (BWB) configuration. In both cases, designs can be generated and evaluated. That said, the quality of the design strongly depends on the available knowledge about a given configuration. For instance, the lack of available knowledge on BWB structure layouts means that it is not represented in the final design. In return, very detailed layouts can be generated for the LH_2 configuration, since the available knowledge base can be applied almost in its entirety.

In summary, the findings on the four working hypotheses corroborate the **research hypothesis** of this thesis. Using the aircraft fuselage as an example, it is shown that KBE techniques can support third-generation MDAO activities by providing consistent augmentation of design details and tailored geometry models in order to facilitate disciplinary analysis model generation. The modularity of the graph-based implementation in FUGA furthermore makes the methodology sufficiently flexible to adapt to new requirements and architectures.

8.2. Perspectives for future research and development

Whereas this thesis successfully provides a proof of concept for the application for KBE-driven fuselage design and multi-model generation for MDAO applications, the road towards fully integrated end-to-end assessment and design of the aircraft is still very long. Further developments are required in several fields, which are considered in the following sections. First, possible development trajectories for the implementation of the KBE system presented in this thesis are discussed in section 8.2.1. Then, missing disciplines necessary to enable true end-to-end assessment of the fuselage are considered in section 8.2.2. In section 8.2.3, aspects of improving the knowledge base for unconventional configurations are listed. Finally, section 8.2.4 provides some insight on the necessary steps towards deployment of the methodology for large-scale MDAO applications.

8.2.1. KBE system implementation

The need for user intervention to realize certain types of problem formulations has been addressed on several occasions in this thesis. Cases include the solution of problems using numerical methods as well as setting up the system for various design tasks. In either case, the user is exposed to the complexity of the underlying system, which is typically not desirable. Therefore, further automation to support the aforementioned tasks should be considered.

On the one hand, improved automation of the set up of the solution workflow would be highly useful. As the close relationship to MDAO has been highlighted several times in this thesis, synergy effects with recent developments in that field could be exploited. To this end, a more advanced way of determining the problem solution graph, which takes into account coordinator blocks, e.g. convergers or optimizers, also known from MDAO [PGG13; Gen19a], could be implemented. The placement of these blocks could either be performed interactively using a tool such as MDAx [PB+20], which enables interactive manipulation of MDAO workflows, or automatically. However, the automatic placement of coordinators in an ad-hoc system, which is instantiated depending on the user query and does not comply with an established MDAO architecture pattern, is not well-understood at this point.

The second aspect, where user support can be improved is the composition of the overall system. The most complex design systems discussed in this thesis consist of approximately 10 rule sets. This order of magnitude can be handled manually quite well. However, as more disciplines and their corresponding rule sets are included, the dependencies between them become more difficult to trace. To this end, a rule set dependency meta-graph could be assembled to allow for reasoning on rule set dependencies. To illustrate, this graph could tell the system that the structural modeling rule set requires the CPACS rule set to be available. In turn, the structural design rule set requires the structural modeling rule set. Consequently, all three rule sets would be instantiated if the user requests the structural design rule set.

In addition to the user experience, the aspect of knowledge integration will grow in importance as more disciplinary design and modeling capabilities are made available. The choice to implement rules using Python functions and the rule protocol is an attempt to make it relatively easy for new developers to make contributions to the system. However, the effective development of the knowledge base requires highly qualified personnel, who unite engineering knowledge and coding skills.

Furthermore, over time, the overall understanding of the system and the management of new capability will become an increasingly complex and critical task. One of the most important points is to avoid duplication of rules as much as possible, as evaluation of multiple rules, which do very similar things, will thwart the computational efficiency of the system. Here, active maintenance and quality control as well as accessible, complete and up-to-date documentation are essential.

8.2.2. Missing disciplinary aspects for end-to-end fuselage assessment

In this thesis, it is attempted to capture the multidisciplinary nature of fuselage design by taking into account the outer mold line, structural design and cabin design. The combination of these disciplines is sufficient to illustrate the complexities of fuselage design. However, real fuselage designs are driven by a multitude of additional factors, which have not been taken into account. To include this knowledge into the design, additional rule sets must be implemented.

An issue, which has arisen repeatedly in the preceding chapters, is the lack of flexible cabin component design. Instead of utilizing dead component geometry models, a proper component design rule set, as suggested for the seat, could unlock additional design drivers such as the seat width, which could have an influence on an overall aircraft design level.

Other disciplines have been omitted altogether, such as the design of the electrical and mechanical systems. An example for integrated geometric systems design is shown e.g. by Fuchs et al. [FH+21] for the air conditioning system. Another interesting project is the GeneSys framework for parametric systems design described by Bielsky, Jünemann, and Thielecke [BJT20], which also uses CPACS as a starting point. Aside from geometric modeling capabilities, functional architecting capabilities are also in development [KB+22]. Making this knowledge available in the KBE system would add significantly to the value of the multidisciplinary design process. That said, the integration of system aspects is held back by limitations of CPACS, which, as of version 3.4, which is used in this thesis, does not provide sufficient support for geometric description of system components. However, development of an onboard systems description to be included in a subsequent version is ongoing [DLR21b].

Similarly to onboard systems, industrialization aspects will become increasingly important as growing demand for aircraft drives production rates at manufacturers. To assess manufacturing aspects, information such as assembly sequences and utilities are necessary, which are beyond the scope of CPACS as a product description format. An approach as shown by Markusheska et al. [MS+22] to use a complementary format to CPACS, which contains the manufacturing process data, could be a practical solution, which can be integrated in the knowledge-based process with manageable effort. Since the list of disciplines with possible relevance can easily be continued to include aspects e.g. of operations, maintenance or life cycle, the idea of complementary formats is promising for providing the necessary information while managing the complexity of CPACS.

Of course, aside from the design capability, the relevant analysis types must be supported as well, from the model generation side. Here the range of analyses shown provide a good foundation, where only a limited amount of detail must be added. A good example of this is provided by Hesse et al. [HW+23], who successfully adapt FUGA, the KBE system developed in this thesis, to provide high-resolution FEM models for vibro-acoustics analysis. Based on the examples shown in this thesis, it can be presumed that boarding and egress simulations will, too, only require very little additional geometry modeling capability to be implemented. For systems aspects, the effort is likely higher.

Moreover, the automation of the analyses is a factor to be taken into account. To enable distributed MDAO, the analyses must be set up to run without the need for user interaction. This is possible for human model simulations, but difficult for human-in-the-loop analyses, as shown by the corresponding use case in this thesis. Here, identifying key parameters and building a surrogate model of the test subject response could be an initial solution.

8.2.3. Improvement of the knowledge base for unconventional configurations

Aside from filling the gaps in disciplinary knowledge for conventional designs, building a reliable knowledge base for novel configurations is another task that needs further work. For the LH_2 configuration, the geometric integration of the tank in the rear fuselage has been demonstrated in this thesis. However, this is only the proverbial tip of the iceberg when it comes to necessary adaptations due to the change in propulsion architecture. To begin with, additional rules to provide a preliminary sizing and mass estimation and check against established requirements for pressure vessels should be implemented, as shown e.g. by Höhne [Höh22]. The systems to transport the fuel to the consumer, i.e. the fuel cell stack in an electric configuration or the engine for a direct burn architecture, must be designed to fundamentally different requirements than conventional configurations using kerosene. The need to provide a possibility to vent every section of pipe in order to release pressure from evaporating hydrogen is an example for this. A particularly critical component in terms of space allocation is the so-called capsule, a block of systems installed next to the tank, which contains pumps and heaters to distribute the fuel. The space and position requirements for the capsule may actually make it a driver for the fuselage length with an impact at a similar order of magnitude to the tank volume. Other aspects to be considered are accessibility of the tank for maintenance, which creates a need for a structural opening, and crash safety [Wee21].

For the BWB configuration, a fundamental investigation of feasible structural concepts is necessary, to gain a more complete understanding of the boundary conditions to the cabin layout generation demonstrated in this thesis. Furthermore, concepts must be evaluated w.r.t. egress scenarios and new exit concepts such as ventral emergency exit doors must be investigated, ideally in collaboration with certification authorities.

These points illustrate that the development of such novel configurations is an ongoing learning process for all involved. The benefit of the KBE methodology is that new knowledge can be formalized as a set of rules to go along with new discoveries. In this way, the knowledge can be stored, exchanged, and archived.

8.2.4. Deployment for collaborative MDAO

Ultimately, the motivation for the development of the multi-fidelity geometry modeling capabilities has been to facilitate the deployment of fuselage analysis and design capability to collaborative MDAO processes. The way to accomplish this is to facilitate the analysis model generation by providing geometry models tailored to the specific needs of the discipline, while design relationships ensure the consistency of the model.

To deploy this capability for collaborative MDAO processes, a change in paradigm for setting up these processes will be necessary. In current workflows, executed e.g. using the Remote Component Environment (RCE), the disciplinary tools are designed to communicate via CPACS files [CN16; WCN22]. This means that a tool will receive a CPACS file, build the geometry, possibly using the TiGL Geometry Library (TiGL), perform the analysis, and then write the results back to a CPACS file, which is fed back to the process. If the knowledge-based system is made available to the process instead, disciplinary tool developers could instead make a direct request for the geometry they require for the model generation, which would not only facilitate their work for model generation, but also reduce redundant computations of the same geometry components. In turn, the results, e.g. new skin thicknesses could be fed back directly to the system, triggering the invalidation capabilities. Logging capability ensures that no information gets lost completely and introduces a traceability and accountability w.r.t. who is the originator of certain changes. In this way, the tool could take on the role of a central hub, where the contributions from all collaborators are stored and validated.

An open technical aspect for such a scenario on the side of the KBE tool is how to make it available to the partners, who will likely interact with it via a network. The most obvious solution for this is to implement a REST API (representational state transfer application programming interface), which allows users to retrieve and update design data via a web interface. Python, like most modern programming languages, provides a multitude of packages to accomplish this. Instead of passing around a continuously updated CPACS file, the data would be communicated to and retrieved from a central web server, implying an evolution from a passive central data model to an active central data management service. A possible problem with this approach is the ensuing need to adapt all disciplinary design tools, which often only recently adopted CPACS as a standard. Here, a server-side data comparison tool, which can identify changes in a CPACS data set and map it to knowledge repository updates, could be helpful to ease the transition.

The capability of the KBE methodology to support different product architectures also makes it eligible to support architecture optimization scenarios [BS+21], where a multitude of multidisciplinary design optimization runs is performed for different architectural solutions to find the optimal solution to a top level design problem, e.g. to fulfill a given transportation task. In such a scenario, different design systems could be composed and deployed for each architectural solution. Such a scenario would also benefit greatly from the aforementioned capability for automated system composition.

Bibliography

[ATM12] M. V. Abritta, J. Thorbeck, and B. S. de Mattos. “Study of a lower-deck galley for airliners.” In: *Journal of Aerospace Technology and Management* 4.1 (2012), pp. 81–94. DOI: 10.5028/jatm.2012.04015311 (cit. on p. 12).

[ASS11] J. Abulawi, K. Seeckt, and D. Scholz. “Automatic Generation of 3D-CAD Models to Bridge the Gap between Aircraft Preliminary Sizing and Geometric Design.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2011 (cit. on pp. 8, 9, 44).

[AG+18] B. Aigner, I. van Gent, G. L. Rocca, E. Stumpf, and L. L. M. Veldhuis. “Graph-based algorithms and data-driven documents for formulation and visualization of large MDO systems.” In: *CEAS Aeronautical Journal* 9.4 (June 2018), pp. 695–709. DOI: 10.1007/s13272-018-0312-5 (cit. on p. 24).

[ASS21] B. Aigner, P. Strathoff, and E. Stumpf. “MICADO: Recent Developments of Models for Design and Evaluation of Electric Aircraft Propulsion Systems.” en. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. RWTH Aachen University, Sept. 2021. DOI: 10.18154/RWTH-2021-01260 (cit. on p. 8).

[AC+10] J. Ainsworth, C. Collier, P. Yarrington, R. Lucking, and J. Locke. “Airframe Wingbox Preliminary Design and Weight Prediction.” In: *SAWE 69th Annual Conference, Virginia Beach, Virginia*. 2010 (cit. on pp. 47, 48).

[ATA21] Air Transport Action Group. *Waypoint 2050 - Balancing growth in connectivity with a comprehensive global air transport response to the climate emergency: A vision of net-zero aviation by mid-century*. Tech. rep. Sept. 2021. URL: https://aviationbenefits.org/media/167417/w2050_v2021_27sept_full.pdf (cit. on pp. 1, 6, 12).

[Air23] Airbus. *Digital Design, Manufacturing & Services | Airbus*. 2023. URL: https://www.airbus.com/en/innovation/disruptive-concepts/digital-design-manufacturing-services (visited on 02/14/2023) (cit. on p. 1).

[Alc17] F. Alcock. *Airbus A321neo Airbus Cabin Flex Conf 9 - Data Basis for Design*. Tech. rep. (internal). Airbus Operations, Oct. 2017 (cit. on p. 168).

[AM+20] M. Alder, E. Moerland, J. Jepsen, and B. Nagel. “Recent Advances in Establishing a Common Language For Aircraft Design With CPACS.” In: *Aerospace Europe Conference – AEC2020, Bordeaux, France, 2020*. Jan. 2020 (cit. on p. 43).

[AL+04] J. Alonso, P. LeGresley, E. V. der Weide, J. R. R. A. Martins, and J. Reuther. “pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization.” In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2004. DOI: 10.2514/6.2004-4480 (cit. on p. 20).

[AD+16] E. Alyanak et al. “Multi-fidelity Geometry-centric Multi-disciplinary Analysis for Design.” In: *AIAA Modeling and Simulation Technologies Conference*. American Institute of Aeronautics and Astronautics, June 2016. DOI: 10.2514/6.2016-4007 (cit. on pp. 8, 9, 75).

[AJK08] K. Amadori, C. Jouannet, and P. Krus. “Aircraft Conceptual Design Optimization.” In: *26th Congress of the International Council of the Aeronautical Sciences (ICAS), Anchorage, Alaska, USA*. Vol. 4. Sept. 2008 (cit. on p. 56).

[ANA22] ANA Cargo. *ANA Cargo Dimension Guide*. 2022. URL: `https://www.anacargo.jp/ja/download/label/pdf/ANA_DG2022_07.pdf` (visited on 09/24/2023) (cit. on pp. 179, 259, 266).

[ANS23a] ANSYS, Inc. *Ansys Mechanical | Structural FEA Analysis Software*. 2023. URL: `https://www.ansys.com/products/structures/ansys-mechanical` (visited on 02/10/2023) (cit. on p. 48).

[ANS23b] ANSYS, Inc. *Ansys ModelCenter | MBSE Software*. 2023. URL: `https://www.ansys.com/de-de/products/connect/ansys-modelcenter/` (visited on 02/14/2023) (cit. on p. 23).

[AK05] N. E. Antoine and I. M. Kroo. "Framework for Aircraft Conceptual Design and Environmental Performance Studies." In: *AIAA Journal* 43.10 (Oct. 2005), pp. 2100–2109. DOI: `10.2514/1.13017` (cit. on p. 20).

[App14] H. S. Appel. "Analyse der Verzögerungen beim Boarding von Flugzeugen und Untersuchung möglicher Optimierungsansätze." PhD thesis. RWTH Aachen University, 2014 (cit. on p. 65).

[Aut23] Autodesk, Inc. *FBX | Adaptable File Formats for 3D Animation Software | Autodesk*. 2023. URL: `https://www.autodesk.com/products/fbx/overview` (visited on 02/10/2023) (cit. on p. 53).

[Aya15] U. Ayachit. *The ParaView guide: A parallel visualization application; updated for ParaView version 4.3*. Clifton Park, NY: Kitware, 2015. ISBN: 9781930934290 (cit. on p. 52).

[Ayy16] V. Ayyalasomayajula. "Evaluation of free and open-source finite element solvers for use in a process chain for aircraft pre-design." MA thesis. University of Stuttgart, 2016 (cit. on p. 48).

[BLK12] E. H. Baalbergen, W. F. Lammen, and J. Kos. "Mastering restricted network access in aeronautic collaborative engineering across organizational boundaries." In: *PDT Europe 2012, The Hague, The Netherlands*. Sept. 2012 (cit. on p. 23).

[Baa15] Y. M. Baan. "A hybrid method for the interior and exterior design of blended-wing-body cabins." MA thesis. Delft University of Technology, 2015 (cit. on pp. 13, 134, 164, 189, 255).

[BF+16] T. Bach, T. Führer, C. Willberg, and S. Dähne. "Automated sizing of a composite wing for the usage within a multidisciplinary design process." In: *Aircraft Engineering and Aerospace Technology* 88.2 (Mar. 2016), pp. 303–310. DOI: `10.1108/aeat-02-2015-0057` (cit. on p. 58).

[BL+18] S. Bagassi, F. Lucchi, F. D. Crescenzio, and S. Piastra. "Design for comfort: Aircraft Interiors Design Assessment Through A Human Centered Response Model Approach." In: *31st Congress of the International Council of the Aeronautical Sciences*. 2018 (cit. on pp. 61, 62).

[Ban19] M. Banović. "Efficient algorithmic differentiation of CAD frameworks." PhD thesis. Paderborn University, 2019 (cit. on p. 21).

[BL+17] N. Bartoli et al. "An adaptive optimization strategy based on mixture of experts for wing aerodynamic design optimization." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2017. DOI: `10.2514/6.2017-4433` (cit. on p. 19).

[BHJ09] M. Bastian, S. Heymann, and M. Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks." In: *International AAAI Conference on Web and Social Media*. Vol. 3. 1. Association for the Advancement of Artificial Intelligence (AAAI), Mar. 2009, pp. 361–362. DOI: `10.1609/icwsm.v3i1.13937` (cit. on p. 88).

[Bat08] K.-J. Bathe. *Finite Element Method.* John Wiley & Sons, Inc., June 2008. DOI: 10.1002/9780470050118.ecse159 (cit. on pp. 47, 261).

[BW01] A. Beckert and H. Wendland. "Multivariate interpolation for fluid-structure-interaction problems using radial basis functions." In: *Aerospace Science and Technology* 5.2 (2001), pp. 125–134. DOI: 10.1016/s1270-9638(00)01087-7 (cit. on p. 198).

[Bel02] G. Belie. "Non-Technical Barriers to Multidisciplinary Optimization in the Aerospace Industry." In: *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* American Institute of Aeronautics and Astronautics, Sept. 2002. DOI: 10.2514/6.2002-5439 (cit. on p. 23).

[BCM04] G. Bellinger, D. Castro, and A. Mills. *Data, Information, Knowledge, and Wisdom.* 2004. URL: https://www.systems-thinking.org/dikw/dikw.htm (visited on 04/12/2022) (cit. on p. 67).

[BRI08] A. Bérard, A. Rizzi, and A. Isikveren. "CADac: A New Geometry Construction Tool for Aerospace Vehicle Pre-Design and Conceptual Design." In: *26th AIAA Applied Aerodynamics Conference.* American Institute of Aeronautics and Astronautics, June 2008. DOI: 10.2514/6.2008-6219 (cit. on p. 8).

[BV16] L. P. Berg and J. M. Vance. "Industry use of virtual reality in product design and manufacturing: a survey." In: *Virtual Reality* 21.1 (Sept. 2016), pp. 1–17. DOI: 10.1007/s10055-016-0293-9 (cit. on p. 61).

[BA14] N. D. Bhagat and E. J. Alyanak. "Computational Geometry for Multi-fidelity and Multidisciplinary Analysis and Optimization." In: *52nd Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2014. DOI: 10.2514/6.2014-0188 (cit. on p. 75).

[BJT20] T. Bielsky, M. Jünemann, and F. Thielecke. "Parametric Modeling of the Aircraft Electrical Supply System for overall Conceptual Systems Design." en. In: *Deutscher Luft- und Raumfahrtkongress.* Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2020. DOI: 10.25967/530143 (cit. on p. 220).

[Bie22] J. Biermann. "Optimization of filament wound cryogenic composite tanks with unconventional geometries for future aircraft configurations." MA thesis. Institut Supérieur de l'Aéronautique et de l'Espace, Oct. 2022 (cit. on p. 162).

[Ble23] Blender Foundation. *blender.org - Home of the Blender project - Free and Open 3D Creation Software.* 2023. URL: https://www.blender.org/ (visited on 02/10/2023) (cit. on p. 53).

[BF+17] K. Bobrowski, E. Ferrer, E. Valero, and H. Barnewitz. "Aerodynamic Shape Optimization Using Geometry Surrogates and Adjoint Method." In: *AIAA Journal* 55.10 (Oct. 2017), pp. 3304–3317. DOI: 10.2514/1.j055766 (cit. on p. 21).

[Boe12] Boeing. *Pallets and containers.* 2012. URL: https://www.boeing.com/resources/boeingdotcom/company/about_bca/pdf/CargoPalletsContainers.pdf (visited on 09/24/2023) (cit. on pp. 259, 266).

[Böh15] D. Böhnke. "A Multi-Fidelity Workflow to Derive Physics-Based Conceptual Design Methods." PhD thesis. Technische Universität Hamburg-Harburg, 2015. URL: https://elib.dlr.de/97905/ (cit. on pp. 8, 22, 43, 67).

[BNG11] D. Böhnke, B. Nagel, and V. Gollnick. "An approach to multi-fidelity in conceptual aircraft design in distributed design environments." In: *2011 Aerospace Conference.* IEEE, Mar. 2011. DOI: 10.1109/aero.2011.5747542 (cit. on p. 27).

[Boo01] C. de Boor. *A Practical Guide to Splines.* Springer New York, Nov. 2001. 372 pp. ISBN: 0387953663 (cit. on p. 28).

[BW+16] E. M. Botero et al. “SUAVE: An Open-Source Environment for Conceptual Vehicle Design and Optimization.” In: *54th AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: 10.2514/6.2016-1275 (cit. on p. 8).

[BH+19] M. A. Bouhlel et al. “A Python surrogate modeling framework with derivatives.” In: *Advances in Engineering Software* 135 (Sept. 2019), p. 102662. DOI: 10.1016/j.advengsoft.2019.03.005 (cit. on p. 19).

[BC+18] F. Bouissiere et al. “Co-engineering in aeronautics? The A320 forward section case study.” In: *9th Congress on Embedded Real Time Software and Systems (ERTS^2'18).* 2018 (cit. on p. 12).

[Bre91] G. D. Brewer. *Hydrogen Aircraft Technology.* CRC Press, 1991. ISBN: 0-8493-5838-8 (cit. on pp. 162, 184).

[Bro06] P. Brooker. “Civil aircraft design priorities: air quality? climate change? noise?” In: *The Aeronautical Journal* 110.1110 (Aug. 2006), pp. 517–532. DOI: 10.1017/s0001924000001408 (cit. on p. 14).

[Bro11] Y. H. A. Brouwers. “Development of KBE applications to support the conceptual design of passenger aircraft fuselages.” MA thesis. TU Delft, 2011 (cit. on pp. 11, 74).

[BV18] M. Brown and R. Vos. “Conceptual Design and Evaluation of Blended-Wing Body Aircraft.” In: *2018 AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-0522 (cit. on p. 12).

[Bru73] E. F. Bruhn. *Analysis and Design of Flight Vehicle Structures.* Jacobs Pub, 1973. ISBN: 0961523409 (cit. on pp. 17, 47, 141, 202).

[BC+21] T. Burschyk, Y. Cabac, D. Silberhorn, B. Boden, and B. Nagel. “Liquid hydrogen storage design trades for a short-range aircraft concept.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* 2021 (cit. on pp. 13, 162, 184).

[BS+21] J. H. Bussemaker, T. D. Smedt, G. L. Rocca, P. D. Ciampa, and B. Nagel. “System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem.” In: *AIAA AVIATION 2021 FORUM.* American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-3078 (cit. on p. 222).

[Car23] P. Carbonnelle. *PYPL PopularitY of Programming Language.* Sept. 2023. URL: https://pypl.github.io/PYPL.html (visited on 09/02/2023) (cit. on p. 79).

[CRT11] L. Cavagna, S. Ricci, and L. Travaglini. “NeoCASS: An integrated tool for structural sizing, aeroelastic analysis and MDO at conceptual design level.” In: *Progress in Aerospace Sciences* 47.8 (Nov. 2011), pp. 621–635. DOI: 10.1016/j.paerosci.2011.08.006 (cit. on p. 58).

[CL+18] J. Cavalcanti, P. London, R. Wallach, and P. Ciloni. “A Case of Success: MDO Applied on the Development of Embraer 175 Enhanced Wingtip.” In: *31st Congress of the International Council of the Aeronautical Sciences (ICAS), Belo Horizonte, Brazil.* Sept. 2018 (cit. on p. 20).

[Cav06] M. Cavcar. “Bréguet Range Equation?” In: *Journal of Aircraft* 43.5 (Sept. 2006), pp. 1542–1544. DOI: 10.2514/1.17696 (cit. on p. 6).

[CP99] C. B. Chapman and M. Pinfold. “Design Engineering – A need to rethink the solution using KBE.” In: *Applications and Innovations in Expert Systems VI.* Springer London, 1999, pp. 112–131. DOI: 10.1007/978-1-4471-0575-6_9 (cit. on pp. 68, 69, 74).

[CP01] C. B. Chapman and M. Pinfold. “The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure.” In: *Advances in Engineering Software* 32.12 (Dec. 2001), pp. 903–912. DOI: 10.1016/s0965-9978(01)00041-2 (cit. on pp. 74, 75).

[CB14] P. Chiusano and R. Bjarnason. *Functional Programming in Scala.* Manning Publications, Oct. 2014. 320 pp. ISBN: 1617290653 (cit. on p. 93).

[CN16] P. D. Ciampa and B. Nagel. "Towards the 3rd Generation MDO Collaborative Environment." In: *30th Congress of the International Council of the Aeronautical Sciences, DCC, Daejeon, Korea.* 2016 (cit. on p. 221).

[CN17] P. D. Ciampa and B. Nagel. "The AGILE Paradigm: the next generation of collaborative MDO." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.* American Institute of Aeronautics and Astronautics, June 2017. DOI: 10.2514/6.2017-4137 (cit. on p. 6).

[CN20] P. D. Ciampa and B. Nagel. "AGILE Paradigm: The next generation collaborative MDO for the development of aeronautical systems." In: *Progress in Aerospace Sciences* 119 (Nov. 2020), p. 100643. DOI: 10.1016/j.paerosci.2020.100643 (cit. on pp. 5, 22).

[CN21] P. D. Ciampa and B. Nagel. "Accelerating the Development of Complex Systems in Aeronautics via MBSE and MDAO: a Roadmap to Agility." In: *AIAA AVIATION 2021 FORUM.* American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-3056 (cit. on pp. 5, 14, 15, 255).

[CZN10] P. D. Ciampa, T. Zill, and B. Nagel. "CST parametrization for unconventional aircraft design optimization." In: *27th Congress of the International Council of the Aeronautical Sciences (ICAS), Nice, France.* 2010 (cit. on pp. 13, 57, 59).

[CP+19] P. D. Ciampa et al. "Streamlining Cross-Organizational Aircraft Development: Results from the AGILE Project." In: *AIAA Aviation 2019 Forum.* American Institute of Aeronautics and Astronautics, June 2019. DOI: 10.2514/6.2019-3454 (cit. on pp. 23, 24).

[CF20] Clean Sky 2 JU and Fuel Cells and Hydrogen 2 JU. *Hydrogen-powered aviation: a fact based study of hydrogen technology, economics, and climate impact by 2050.* Publications Office of the European Union, 2020. DOI: 10.2843/471510 (cit. on pp. 1, 12).

[CFS17] J. Q. Coburn, I. Freeman, and J. L. Salmon. "A Review of the Capabilities of Current Low-Cost Virtual Reality Technology and Its Potential to Enhance the Design Process." In: *Journal of Computing and Information Science in Engineering* 17.3 (July 2017). DOI: 10.1115/1.4036921 (cit. on p. 61).

[Col46] A. R. Collar. "The Expanding Domain of Aeroelasticity." In: *Journal of the Royal Aeronautical Society* 50 (1946), pp. 613–636 (cit. on p. 15).

[EC19] Commission to the European Parliament, European Council, Council, European Economic and Social Committee, and Committee of the Regions. *The European Green Deal.* Dec. 2019. URL: https://eur-lex.europa.eu/resource.html?uri=cellar:b828d165-1c22-11ea-8c1f-01aa75ed71a1.0002.02/DOC_1&format=PDF (visited on 02/14/2023) (cit. on p. 1).

[CM60] S. Coons and R. Mann. *Computer-aided Design Related to the Engineering Design Process.* Technical memorandum 8436. M.I.T. Electronic Systems Laboratory, 1960 (cit. on p. 28).

[CL07] D. Cooper and G. La Rocca. "Knowledge-based Techniques for Developing Engineering Applications in the 21st Century." In: *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences,17th LTA Systems Tech Conf; followed by 2nd TEOS Forum.* American Institute of Aeronautics and Astronautics, Sept. 2007. DOI: 10.2514/6.2007-7711 (cit. on p. 70).

[CJ+94] E. J. Cramer, J. J. E. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin. "Problem Formulation for Multidisciplinary Optimization." In: *SIAM Journal on Optimization* 4.4 (Nov. 1994), pp. 754–776. DOI: 10.1137/0804044 (cit. on p. 17).

[CCS15] E. Crawley, B. Cameron, and D. Selva. *System Architecture, Global Edition.* Pearson Education Limited, Dec. 2015. 480 pp. ISBN: 1292110848 (cit. on p. 14).

[Dan13] J. Dannenhoffer. "OpenCSM: An Open-Source Constructive Solid Modeler for MDAO." In: *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* American Institute of Aeronautics and Astronautics, Jan. 2013. DOI: 10.2514/6.2013-701 (cit. on pp. 38, 39).

[DH15] J. Dannenhoffer and R. Haimes. "Design Sensitivity Calculations Directly on CAD-based Geometry." In: *53rd AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2015. DOI: 10.2514/6.2015-1370 (cit. on pp. 21, 38, 39).

[DH16] J. Dannenhoffer and R. Haimes. "Generation of Multi-fidelity, Multi-discipline Air Vehicle Models with the Engineering Sketch Pad." In: *54th AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: 10.2514/6.2016-1925 (cit. on pp. 22, 56, 75).

[DH17] J. Dannenhoffer and R. Haimes. "Using Design-Parameter Sensitivities in Adjoint-Based Design Environments." In: *55th AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2017. DOI: 10.2514/6.2017-0139 (cit. on p. 21).

[3DS23a] Dassault Systèmes. *Abaqus Unified FEA - Mechanical Simulation.* 2023. URL: https://www.3ds.com/products-services/simulia/products/abaqus/ (visited on 02/10/2023) (cit. on p. 48).

[Das23] Dassault Systèmes. *Isight & SIMULIA Execution Engine | Dassault Systèmes®.* 2023. URL: https://www.3ds.com/products-services/simulia/products/isight-simulia-execution-engine/ (visited on 02/14/2023) (cit. on p. 23).

[3DS23b] Dassault Systèmes. *Shape the world we live in | CATIA - Dassault Systèmes.* 2023. URL: https://www.3ds.com/products-services/catia/ (visited on 02/10/2023) (cit. on pp. 39, 70).

[3DS23c] Dassault Systèmes. *Transform Your Business with 3DEXPERIENCE Platform | Dassault Systèmes.* 2023. URL: https://www.3ds.com/3dexperience (visited on 09/24/2023) (cit. on p. 40).

[DF+21] C. Davey et al. *Systems Engineering Vision 2035.* Tech. rep. International Council on Systems Engineering (INCOSE), 2021. URL: https://www.incose.org/about-systems-engineering/se-vision-2035 (cit. on p. 14).

[DD+21] C. David et al. "From FAST to FAST-OAD: An open source framework for rapid Overall Aircraft Design." In: *IOP Conference Series: Materials Science and Engineering* 1024.1 (Jan. 2021), p. 012062. DOI: 10.1088/1757-899x/1024/1/012062 (cit. on p. 8).

[DB+19] F. De Crescenzio, S. Bagassi, S. Asfaux, and N. Lawson. "Human centred design and evaluation of cabin interiors for business jet aircraft in virtual reality." In: *International Journal on Interactive Design and Manufacturing (IJIDeM)* 13.2 (Apr. 2019), pp. 761–772. DOI: 10.1007/s12008-019-00565-8 (cit. on pp. 61, 62, 209, 211, 255).

[DBS21] F. De Crescenzio, S. Bagassi, and F. Starita. "Preliminary user centred evaluation of regional aircraft cabin interiors in virtual reality." In: *Scientific Reports* 11.1 (May 2021). DOI: 10.1038/s41598-021-89098-3 (cit. on pp. 61, 62).

[DeM18] A. De Marco. *jPAD, A Java Application Programming Interface for Aircraft Design.* en. 2018. DOI: 10.5281/ZENODO.2064963 (cit. on p. 45).

[Dei16] S. M. Deinert. "Shape and Sizing Optimization of Aircraft Structures with Aeroelastic and Induced Drag Requirements." PhD thesis. TU München, 2016 (cit. on p. 59).

[DLR21a] Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR). *Auf dem Weg zu einer emissionsfreien Luftfahrt: Luftfahrtstrategie des DLR zum European Green Deal.* 2021. URL: `https://www.dlr.de/content/de/downloads/publikationen/broschueren/2021/auf-dem-weg-zu-einer-emissionsfreien-luftfahrt.pdf` (cit. on pp. 1, 12).

[Die17] R. Diestel. *Graph Theory.* Springer Berlin Heidelberg, June 2017. 448 pp. ISBN: 3662536218 (cit. on p. 83).

[DB23] R. van Dijk and M. Baan. *ParaPy.* 2023. URL: `https://parapy.nl/` (visited on 02/10/2023) (cit. on pp. 9, 70).

[DV14] J. van Dommelen and R. Vos. "Conceptual design and analysis of blended-wing-body aircraft." In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 228.13 (Jan. 2014), pp. 2452–2474. DOI: `10.1177/0954410013518696` (cit. on p. 12).

[Dor14] F. Dorbath. "A flexible wing modeling and physical mass estimation system for early aircraft design stages." en. PhD thesis. TU Hamburg, 2014. DOI: `10.15480/882.1159` (cit. on pp. 47, 48, 53, 55, 57, 58, 67, 74–76, 131, 136).

[DNG11] F. Dorbath, B. Nagel, and V. Gollnick. "A Knowledge Based Approach for Automated Modelling of Extended Wing Structures in Preliminary Aircraft Design." In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* 2011 (cit. on pp. 42, 106).

[DNG13] F. Dorbath, B. Nagel, and V. Gollnick. "Extended physics-based wing mass estimation in early design stages applying automated model generation." In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 228.7 (Apr. 2013), pp. 1010–1019. DOI: `10.1177/0954410013482657` (cit. on pp. 106, 136).

[Dre10] M. Drela. *TASOPT 2.00: Transport Aircraft System OPTimization - Technical Description.* Tech. rep. MIT, 2010 (cit. on p. 8).

[DY24] M. Drela and H. Youngren. *MIT AVL User Primer - AVL 3.36.* 2024. URL: `https://web.mit.edu/drela/Public/web/avl/AVL_User_Primer.pdf` (visited on 06/12/2024) (cit. on p. 75).

[Dro18] M. Drougard. "Computer-Aided Design for Aircraft." MA thesis. École Polytechnique Fédérale de Lausanne (EPFL), 2018 (cit. on p. 45).

[DP+22] T. Y. Druot, N. Peteilh, P. Roches, and N. Monrolin. "Hydrogen Powered Airplanes, an exploration of possible architectures leveraging boundary layer ingestion and hybridization." In: *AIAA SCITECH 2022 Forum.* American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: `10.2514/6.2022-1025` (cit. on pp. 12, 13).

[DR19] R. J. Durscher and D. Reedy. "pyCAPS: A Python Interface to the Computational Aircraft Prototype Syntheses." In: *AIAA Scitech 2019 Forum.* American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: `10.2514/6.2019-2226` (cit. on p. 8).

[Ecm17] Ecma International. *The JSON data interchange syntax.* Ecma. 2017. URL: `https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf` (visited on 02/25/2023) (cit. on p. 160).

[EP+16] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. "SU2: An Open-Source Suite for Multiphysics Simulation and Design." In: *AIAA Journal* 54.3 (Mar. 2016), pp. 828–846. DOI: `10.2514/1.j053813` (cit. on p. 20).

[EWP18] M. Eder, K. Wicke, and A. A. Pohya. "A Set of Decision Rules for Aircraft Lifecycle Cost-Benefit Assessment." In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* Sept. 2018. URL: `https://elib.dlr.de/121769/` (cit. on p. 6).

[EDF23] EDF SA. *Code_Aster.* 2023. URL: `https://code-aster.org/` (visited on 02/10/2023) (cit. on p. 48).

[ES+04] S. Eelman, D. Schmitt, A. Becker, and W. Granzeier. "Future Requirements and Concepts for Cabins of Blended Wing Body Configurations - A Scenario Approach." In: *Journal of Air Transportation* 9.2 (2004) (cit. on p. 13).

[Eib59] A. M. Eiband. *Human Tolerance to Rapidly Applied Accelerations: A Summary of the Literature.* Tech. rep. NASA, Lewis Research Center, Cleveland, OH, 1959 (cit. on p. 60).

[EVL14] R. Elmendorp, R. Vos, and G. La Rocca. "A conceptual design and analysis method for conventional and unconventional airplanes." In: *29th Congress of the International Council of the Aeronautical Sciences (ICAS), St. Petersburg, Russia.* Sept. 2014 (cit. on p. 8).

[EDH20] M. Engelmann, D. Drust, and M. Hornung. "Automated 3D cabin generation with PAX-elerate and Blender using the CPACS file format." en. In: (2020). DOI: 10.25967/530014 (cit. on pp. 62, 126, 211).

[EH19] M. Engelmann and M. Hornung. "Boarding Process Assessment of the AVACON Research Baseline Aircraft." en. In: Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2019. DOI: 10.25967/490049 (cit. on p. 64).

[EKH20] M. Engelmann, T. Kleinheinz, and M. Hornung. "Advanced Passenger Movement Model Depending On the Aircraft Cabin Geometry." In: *Aerospace* 7.12 (Dec. 2020), p. 182. DOI: 10.3390/aerospace7120182 (cit. on p. 64).

[Epi23] Epic Games. *Unreal Engine.* Version 4.22.1. 2023. URL: https://www.unrealengine.com (visited on 02/10/2023) (cit. on p. 53).

[EST23] ESTECO SpA. *ESTECO modeFRONTIER | Simulation process automation and design optimization.* 2023. URL: https://engineering.esteco.com/modefrontier/ (visited on 02/14/2023) (cit. on p. 23).

[EAS14] European Aviation Safety Agency. *Proposed Equivalent Safety Finding on CS25.807(g) at Amdt 13 "Emergency Exits".* Nov. 2014. URL: https://www.easa.europa.eu/sites/default/files/dfu/ESF%20D-01_consultation.pdf (cit. on p. 168).

[EAS21] European Aviation Safety Agency. *Certification specifications and acceptable means of compliance for large aeroplanes CS-25 - Amendment 23.* 2021. URL: https://www.easa.europa.eu/document-library/certification-specifications/cs-25-amendment-23 (cit. on pp. XIII, 10, 11, 62, 63, 126, 264).

[FHK02] G. Farin, J. Hoschek, and M. S. Kim. *Handbook of Computer Aided Geometric Design.* Elsevier Science & Technology, Aug. 2002. 848 pp. ISBN: 0444511040 (cit. on p. 28).

[Far01] G. Farin. *Curves and Surfaces for CAGD: A Practical Guide.* MORGAN KAUFMANN PUBL INC, Oct. 2001. 520 pp. ISBN: 1558607374 (cit. on pp. 28, 31, 140, 141).

[FH85] R. Farouki and J. Hinds. "A Hierarchy of Geometric Forms." In: *IEEE Computer Graphics and Applications* 5.5 (1985), pp. 51–78. DOI: 10.1109/mcg.1985.276393 (cit. on p. 35).

[FP79] I. D. Faux and M. J. Pratt. *Computational geometry for design and manufacture.* Chichester, Eng. New York: Horwood Halsted Press, 1979. ISBN: 0853121141 (cit. on p. 28).

[FH+14] S. Freund, F. Heinecke, T. Führer, and C. Willberg. "Parametric Model Generation and Sizing of Lightweight Structures for a Multidisciplinary Design Process." In: *NAFEMS Konferenz: "Berechnung und Simulation - Anwendungen, Entwicklungen, Trends.* May 2014. URL: https://elib.dlr.de/89992/ (cit. on p. 57).

[FB+21] M. K. Fuchs, F. Beckert, J. Biedermann, and B. Nagel. "Experience of Conceptual Designs and System Interactions for the Aircraft Cabin in Virtual Reality." In: *AIAA AVIATION 2021 FORUM.* American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-2773 (cit. on pp. 62, 63, 205, 206, 208, 209, 211, 255).

[FFB19] M. K. Fuchs, J. C. Fuchte, and J. Biedermann. “Ein MBSE-Ansatz für die Auslegung von Flugzeugkabinen am Beispiel eines Passenger Supply Channel.” de. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2019. DOI: 10.25967/490018 (cit. on p. 61).

[FH+19] M. K. Fuchs, C. Hesse, J. Biedermann, and J. C. Fuchte. “A multi-disciplinary design optimization of the passenger supply channel in the aircraft cabin.” In: *MATEC Web of Conferences* 304 (2019). Ed. by S. Pantelakis and C. Charitidis. DOI: 10.1051/matecconf/201930404009 (cit. on p. 62).

[FH+21] M. K. Fuchs, C. Hesse, J. Biedermann, and B. Nagel. “Formalized Knowledge Management for the Aircraft Cabin Design Process.” In: *32nd Congress of the International Council of the Areronautical Sciences (ICAS), Shanghai, China*. 2021 (cit. on pp. 43, 44, 62, 220).

[FB+22] M. K. Fuchs et al. “Virtual Reconfiguration and Assessment of Aircraft Cabins using Model-Based Systems Engineering.” In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden*. Sept. 2022 (cit. on p. 211).

[Fuc14] J. C. Fuchte. *Enhancement of Aircraft Cabin Design Guidelines with Special Consideration of Aircraft Turnaround and Short Range Operations*. Tech. rep. Technische Universität Hamburg-Harburg, Apr. 2014. URL: https://elib.dlr.de/89599/ (cit. on pp. 9, 63, 64, 195).

[FDG11] J. C. Fuchte, N. Dzikus, and V. Gollnick. “Cabin Design for Minimum Boarding Time.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Sept. 2011. URL: https://elib.dlr.de/77237/ (cit. on pp. 63–65, 255).

[FGN13] J. C. Fuchte, V. Gollnick, and B. Nagel. “Integrated Tool for Cabin and Fuselage Modeling in Future Aircraft Research.” In: *Workshop on Aircraft System Technology (AST)*. Apr. 2013. URL: https://elib.dlr.de/82767/ (cit. on pp. 12, 42, 62, 122, 126, 136, 148).

[FP+14] J. C. Fuchte, T. Pfeiffer, P. D. Ciampa, B. Nagel, and V. Gollnick. “Optimization of Revenue Space of a Blended Wing Body.” In: *29th Congress of the International Council of the Aeronautical Sciences*. Sept. 2014. URL: https://elib.dlr.de/95128/ (cit. on p. 13).

[FRW11] J. C. Fuchte, S. Rajkowski, and A. Wick. “Rapid Creation of CFD-Capable CAD-Models for Cabin Air Ventilation Simulation.” In: *Workshop on Aviation System Technology (AST)*. 2011 (cit. on p. 63).

[FW+16] T. Führer, C. Willberg, S. Freund, and F. Heinecke. “Automated model generation and sizing of aircraft structures.” In: *Aircraft Engineering and Aerospace Technology* 88.2 (Mar. 2016), pp. 268–276. DOI: 10.1108/aeat-02-2015-0054 (cit. on p. 57).

[GV+18] F. Gallard et al. “GEMS: A Python Library for Automation of Multidisciplinary Design Optimization Process Generation.” In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-0657 (cit. on pp. 24, 76).

[GG+11] A. Gazaix et al. “Investigation of Multi-Disciplinary Optimisation for Aircraft Preliminary Design.” In: *SAE Technical Paper Series*. SAE International, Oct. 2011. DOI: 10.4271/2011-01-2761 (cit. on p. 20).

[Gen19a] I. van Gent. “Agile MDAO Systems: A Graph-based Methodology to Enhance Collaborative Multidisciplinary Design.” PhD thesis. Delft University of Technology, 2019. DOI: 10.4233/uuid:c42b30ba-2ba7-4fff-bf1c-f81f85e890af (cit. on pp. 24, 76, 83, 85, 172, 219).

[GLH18] I. van Gent, G. La Rocca, and M. F. M. Hoogreef. "CMDOWS: a proposed new standard to store and exchange MDO systems." In: *CEAS Aeronautical Journal* 9.4 (May 2018), pp. 607–627. DOI: `10.1007/s13272-018-0307-2` (cit. on p. 76).

[Gen19b] Genworks International. *Genworks GDL - assembly.* 2019. URL: `https://www.genworks.com/` (visited on 02/14/2023) (cit. on p. 70).

[DLR21b] German Aerospace Center, Institute for System Architectures in Aeronautics. *Combining system definitions · Issue #721 · DLR-SL/CPACS · GitHub.* July 2021. URL: `https://github.com/DLR-SL/CPACS/issues/721` (visited on 09/10/2023) (cit. on p. 220).

[DLR22] German Aerospace Center, Institute for System Architectures in Aeronautics. *Provide more versatile fuselage cutout profile definition · Issue #780 · DLR-SL/CPACS · GitHub.* June 2022. URL: `https://github.com/DLR-SL/CPACS/issues/780` (visited on 09/10/2023) (cit. on p. 122).

[DLR23] German Aerospace Center, Institute for System Architectures in Aeronautics. *CPACS Website.* 2023. URL: `http://cpacs.de` (visited on 02/11/2023) (cit. on p. 43).

[Ger15] F. H. Gern. "Update on HCDstruct - A Tool for Hybrid Wing Body Conceptual Design and Structural Optimization." In: *15th AIAA Aviation Technology, Integration, and Operations Conference.* American Institute of Aeronautics and Astronautics, June 2015. DOI: `10.2514/6.2015-2544` (cit. on p. 59).

[GR09] C. Geuzaine and J.-F. Remacle. "Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities." In: *International Journal for Numerical Methods in Engineering* 79 (Sept. 2009), pp. 1309–1331. DOI: `10.1002/nme.2579` (cit. on pp. 51, 193).

[GB98] J. Giesing and J.-F. Barthelemy. "A summary of industry MDO applications and needs." In: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* American Institute of Aeronautics and Astronautics, Aug. 1998. DOI: `10.2514/6.1998-4737` (cit. on p. 20).

[GS+22] S. T. Gil, J. Schminder, A. G. i Almirall, J. I. T. Prieto, and R. V. Linn. "Zonal Model Based Aircraft Passenger Thermal Comfort Comparative Study." In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden.* Sept. 2022 (cit. on p. 63).

[GDG96] J. Gloudemans, P. Davis, and P. Gelhausen. "A rapid geometry modeler for conceptual aircraft." In: *34th Aerospace Sciences Meeting and Exhibit.* American Institute of Aeronautics and Astronautics, Jan. 1996. DOI: `10.2514/6.1996-52` (cit. on p. 8).

[Gob15] A. Gobbin. "Numerische Modellierung des Auslegungsprozesses für Passagierkabinen von Verkehrsflugzeugen unter Berücksichtigung der wichtigstenAuslegungsforderungen und Implementierung in MatLab." MA thesis. TU Berlin, 2015 (cit. on pp. 8, 11, 61, 63, 168).

[GKB21] A. Gobbin, R. Khosravi, and A. Bardenhagen. "Emergency evacuation simulation of commercial aircraft." In: *SN Applied Sciences* 3.4 (Mar. 2021). DOI: `10.1007/s42452-021-04295-z` (cit. on pp. 63–65, 255).

[God08] P.-J. Goderis. "Conceptual Design of Fuselages, Cabins and Landing Gears - Methods, Statistics, Tool Setup." MA thesis. HAW Hamburg, 2008 (cit. on p. 9).

[GS+11] V. Gollnick, E. Stumpf, J. Szodruch, and S. Lehner. "Virtual Integration Platforms (VIP) - A concept for Integrated and Interdisciplinary Air Transportation Research and Assessment." In: *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference.* American Institute of Aeronautics and Astronautics, June 2011. DOI: `10.2514/6.2011-6889` (cit. on p. 43).

[GS19] A. Gomez and H. Smith. “Liquid hydrogen fuel tanks for commercial aviation: Structural sizing and stress analysis.” In: *Aerospace Science and Technology* 95 (Dec. 2019), pp. 105–438. DOI: 10.1016/j.ast.2019.105438 (cit. on p. 13).

[Gop21] M. P. Gopani. “Evaluation of aircraft cabin design integrated with agent based algorithm in virtual reality.” MA thesis. Carl von Ossietzky Universität Oldenburg, 2021 (cit. on pp. 64, 65, 208, 210, 255).

[GK+18] S. Görtz, A. Krumbein, M. Ritter, and J. Hofmann. “DLR-Projekt VicToria - Virtual Aircraft Technology Integration Platform.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2018. URL: https://elib.dlr.de/121695/ (cit. on p. 23).

[Gou71] H. Gouraud. “Continuous Shading of Curved Surfaces.” In: *IEEE Transactions on Computers* C-20.6 (June 1971), pp. 623–629. DOI: 10.1109/t-c.1971.223313 (cit. on p. 51).

[GP21] B. E. Granger and F. Perez. “Jupyter: Thinking and Storytelling With Code and Data.” In: *Computing in Science & Engineering* 23.2 (Mar. 2021), pp. 7–14. DOI: 10.1109/mcse.2021.3059263 (cit. on p. 89).

[Gra19] GraphML Working Group. *The GraphML File Format*. Jan. 2019. URL: http://graphml.graphdrawing.org/ (visited on 02/11/2023) (cit. on p. 88).

[Gra98] J. M. Grasmeyer. “Multidisciplinary Design Optimization of a Strut-Braced Wing Aircraft.” MA thesis. Virginia Polytechnic Institute and State University, 1998 (cit. on p. 12).

[Gra22] A. Graves. *Character Controller SUPER | Game Toolkits | Unity Asset Store*. Oct. 2022. URL: https://assetstore.unity.com/packages/tools/game-toolkits/character-controller-super-135316 (visited on 08/28/2023) (cit. on p. 209).

[GMN10] J. Gray, K. Moore, and B. Naylor. “OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization.” In: *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2010. DOI: 10.2514/6.2010-9101 (cit. on p. 20).

[GH+19] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor. “OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization.” In: *Structural and Multidisciplinary Optimization* 59.4 (Mar. 2019), pp. 1075–1104. DOI: 10.1007/s00158-019-02211-z (cit. on p. 20).

[GH+14] J. S. Gray et al. “Automatic Evaluation of Multidisciplinary Derivatives Using a Graph-Based Problem Formulation in OpenMDAO.” In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2014. DOI: 10.2514/6.2014-2042 (cit. on p. 20).

[GP89] J. Griessmair and W. Purgathofer. “Deformation of Solids with Trivariate B-Splines.” In: *Eurographic Conference 1989*. Eurographics Association, 1989. DOI: 10.2312/EGTP.19891010 (cit. on p. 21).

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. CAMBRIDGE, Nov. 2008. 460 pp. ISBN: 0898716594 (cit. on p. 19).

[Gri04] T. Grimm. *User’s guide to rapid prototyping*. Dearborn, Mich: Society of Manufacturing Engineeers, 2004. ISBN: 0-87263-697-6 (cit. on p. 53).

[Gu17] X. Gu. “Application of Computational Aerodynamic Analysis and Optimization in a Multi-Fidelity Distributed Overall Aircraft Design System.” PhD thesis. RWTH Aachen University, 2017 (cit. on pp. 20, 104).

[HSS08] A. A. Hagberg, D. A. Schult, and P. J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX.” In: *Proceedings of the 7th Python in Science Conference*. Ed. by G. Varoquaux, T. Vaught, and J. Millman. Pasadena, CA USA, 2008, pp. 11–15 (cit. on p. 83).

[Hah10] A. Hahn. "Vehicle Sketch Pad: A Parametric Geometry Modeler for Conceptual Aircraft Design." In: *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition.* American Institute of Aeronautics and Astronautics, Jan. 2010. DOI: 10.2514/6.2010-657 (cit. on p. 8).

[HD13] R. Haimes and J. Dannenhoffer. "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry." In: *21st AIAA Computational Fluid Dynamics Conference.* American Institute of Aeronautics and Astronautics, June 2013. DOI: 10.2514/6.2013-3073 (cit. on p. 9).

[HD12] R. Haimes and M. Drela. "On The Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design." In: *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* American Institute of Aeronautics and Astronautics, Jan. 2012. DOI: 10.2514/6.2012-683 (cit. on p. 60).

[HY+23] H. M. Hajdik et al. "pyGeo: A geometry package for multidisciplinary design optimization." In: *Journal of Open Source Software* 8.87 (July 2023), p. 5319. DOI: 10.21105/joss.05319 (cit. on p. 21).

[HM+13] A. Hall, T. Mayer, I. Wuggetzer, and P. Childs. "Future aircraft cabins and design thinking: optimisation vs. win-win scenarios." In: *Propulsion and Power Research* 2.2 (June 2013), pp. 85–95. DOI: 10.1016/j.jppr.2013.04.001 (cit. on p. 61).

[Han09] L. U. Hansen. "Optimierung von Strukturbauweisen im Gesamtentwurf von Blended Wing Body Flugzeugen." PhD thesis. TU Braunschweig, 2009 (cit. on pp. 12, 13, 54, 59).

[HG+20] N. A. Harrison et al. "Development of an Efficient M=0.80 Transonic Truss-Braced Wing Aircraft." In: *AIAA Scitech 2020 Forum.* American Institute of Aeronautics and Astronautics, Jan. 2020. DOI: 10.2514/6.2020-0011 (cit. on p. 12).

[HB21] M. Häußler and A. Borrmann. "Knowledge-based engineering in the context of railway design by integrating BIM, BPMN, DMN and the methodology for knowledge-based engineering applications (MOKA)." In: *Journal of Information Technology in Construction* 26 (May 2021), pp. 193–226. DOI: 10.36680/j.itcon.2021.012 (cit. on p. 74).

[HG12] C. Heath and J. Gray. "OpenMDAO: Framework for Flexible Multidisciplinary Design, Analysis and Optimization Methods." In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference & 20th AIAA/ASME/AHS Adaptive Structures Conference & 14th AIAA.* American Institute of Aeronautics and Astronautics, Apr. 2012. DOI: 10.2514/6.2012-1673 (cit. on p. 20).

[Hei12] J. Heinemann. "Preliminary Sizing of FAR Part 23 and Part 25 Aircraft." MA thesis. Hamburg University of Applied Sciences, 2012 (cit. on p. 8).

[Hei18] R. Heinrich, ed. *AeroStruct: Enable and Learn How to Integrate Flexibility in Design.* Springer International Publishing, 2018. DOI: 10.1007/978-3-319-72020-3 (cit. on p. 23).

[Hei94] W. Heinze. "Ein Beitrag zur quantitativen Analyse der technischen und wirtschaftlichen Auslegungsgrenzen verschiedener Flugzeugkonzepte für den Transport großer Nutzlasten." PhD thesis. Braunschweig: TU Braunschweig, 1994. ISBN: 3928628143 (cit. on p. 8).

[HAR20] C. Hesse, P. Allebrodt, and A. Rege. "Multi-Physikalische Untersuchungen zum Transmissionsverhalten neuartiger Kabinenseitenwände." de. In: (2020). DOI: 10.25967/530130 (cit. on p. 63).

[HB19] C. Hesse and J. Biedermann. "Acoustic Evaluation and Optimization of Aircraft Cabin Concepts - A Systems Engineering Approach." In: *7th International Workshop on Aircraft System Technologies (AST 2019).* Feb. 2019. URL: https://elib.dlr.de/126614/ (cit. on p. 63).

[HW+23] C. Hesse et al. “Wissensbasierte Modellgenerierung für die Vorhersage von Kabinenlärm in Kontext des Flugzeugvorentwurfs.” In: *49. Jahrestagung für Akustik (DAGA)*. Mar. 2023 (cit. on pp. 112, 220).

[Hex23] Hexagon AB. *MSC Nastran | Hexagon*. 2023. URL: `https://hexagon.com/products/product-groups/computer-aided-engineering-software/msc-nastran` (visited on 02/10/2023) (cit. on p. 48).

[Hil90] F. S. Hill. *Computer Graphics*. Macmillan, 1990 (cit. on p. 48).

[HS+22] J. Hoelzen, D. Silberhorn, T. Zill, B. Bensmann, and R. Hanke-Rauschenbach. “Hydrogen-powered aviation and its reliance on green hydrogen infrastructure – Review and research gaps.” In: *International Journal of Hydrogen Energy* 47.5 (Jan. 2022), pp. 3108–3130. DOI: `10.1016/j.ijhydene.2021.10.239` (cit. on p. 12).

[Hof89] C. M. Hoffmann. *Geometric and solid modeling: An introduction*. San Mateo, Calif: Morgan Kaufmann, 1989. ISBN: 1558600671 (cit. on pp. 37, 38, 255).

[Höh22] P. Höhne. “Development of conceptual design and analysis procedures for a cryogenic hydrogen tank.” MA thesis. University of Stuttgart, Dec. 2022 (cit. on pp. 162, 221).

[HD+13] J. F. Hughes et al. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2013. ISBN: 9780321399526 (cit. on pp. 48–50, 52, 255).

[Hun07] J. D. Hunter. “Matplotlib: A 2D Graphics Environment.” In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: `10.1109/mcse.2007.55` (cit. on p. 131).

[Hür10] F. Hürlimann. “Mass Estimation of Transport Aircraft Wingbox Structures with a CAD/CAE-Based Multidisciplinary Process.” PhD thesis. ETH Zürich, 2010 (cit. on p. 56).

[HKM14] J. Hwang, G. Kenway, and J. R. R. A. Martins. “Geometry and Structural Modeling for High-Fidelity Aircraft Conceptual Design Optimization.” In: *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, June 2014. DOI: `10.2514/6.2014-2041` (cit. on pp. 21, 53, 55, 59, 60).

[HM12] J. Hwang and J. R. R. A. Martins. “GeoMACH: Geometry-Centric MDAO of Aircraft Configurations with High Fidelity.” In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, Sept. 2012. DOI: `10.2514/6.2012-5605` (cit. on pp. 21, 59).

[HM18] J. T. Hwang and J. R. Martins. “A fast-prediction surrogate model for large datasets.” In: *Aerospace Science and Technology* 75 (Apr. 2018), pp. 74–87. DOI: `10.1016/j.ast.2017.12.030` (cit. on p. 19).

[IJs11] S. T. IJsselmuiden. “Optimal Design of Variable Stiffness Composite Structures Using Lamination Parameters.” PhD thesis. Delft University of Technology, 2011 (cit. on p. 47).

[IM+20] C. Ilic et al. “Cybermatrix Protocol: A Novel Approach to Highly Collaborative and Computationally Intensive Multidisciplinary Aircraft Optimization.” In: (June 2020). DOI: `10.2514/6.2020-3169` (cit. on pp. 17, 24).

[ISO21] International Organization for Standardization. *Industrial automation systems and integration – Product data representation and exchange*. ISO. 2021 (cit. on p. 41).

[JG+19] J. Joe, V. Gandhi, J. Dannenhoffer, and H. Dalir. “Rapid Generation of Parametric Aircraft Structural Models.” In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: `10.2514/6.2019-2229` (cit. on p. 8).

[JS14] A. Johanning and D. Scholz. “Adapting life cycle impact assessment methods for application in aircraft design.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2014 (cit. on p. 6).

[Joh77] E. H. Johnson. *PASS – A Computer Program for Preliminary Aircraft Structural Synthesis.* Tech. rep. NASA Ames Research Center, 1977 (cit. on p. 8).

[Jon17] R. L. A. de Jonge. "Development of a Knowledge-Based Engineering Application to Support Conceptual Fuselage Sizing and Cabin Configuration." MA thesis. TU Delft, Jan. 2017 (cit. on pp. 9–11, 33, 74, 79, 105, 131, 132, 145, 255).

[Jun14] A. Jungo. "Development of the CEASIOM Aircraft Design Environment for Novel Aircraft Configurations." MA thesis. École Polytechnique Fédérale de Lausanne (EPFL), 2014 (cit. on p. 13).

[Jup23] Jupyter Team. *Project Jupyter | Home.* 2023. URL: `http://jupyter.org/` (visited on 02/11/2023) (cit. on p. 89).

[JS+14] C. V. Jutte, B. Stanford, C. D. Wieseman, and J. B. Moore. "Aeroelastic Tailoring of the NASA Common Research Model via Novel Material and Structural Configurations." In: *52nd Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2014. DOI: `10.2514/6.2014-0598` (cit. on p. 12).

[Kah62] A. B. Kahn. "Topological sorting of large networks." In: *Communications of the ACM* 5.11 (Nov. 1962), pp. 558–562. DOI: `10.1145/368996.369025` (cit. on pp. 84, 87).

[Kar13] W. Karush. "Minima of Functions of Several Variables with Inequalities as Side Conditions." In: *Traces and Emergence of Nonlinear Programming.* Springer Basel, July 2013, pp. 217–245. DOI: `10.1007/978-3-0348-0439-4_10` (cit. on p. 17).

[Kel99] C. T. Kelley. *Iterative Methods for Optimization.* Society for Industrial and Applied Mathematics, Jan. 1999. DOI: `10.1137/1.9781611970920` (cit. on p. 18).

[Kel03] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method.* Society for Industrial and Applied Mathematics, Jan. 2003. 104 pp. ISBN: 0898715466. DOI: `10.1137/1.9780898718898` (cit. on p. 87).

[KM14] G. J. Kennedy and J. R. Martins. "A parallel finite-element framework for large-scale gradient-based design optimization of high-performance structures." In: *Finite Elements in Analysis and Design* 87 (Sept. 2014), pp. 56–73. DOI: `10.1016/j.finel.2014.04.011` (cit. on pp. 12, 20).

[KKM10] G. Kenway, G. Kennedy, and J. R. R. A. Martins. "A CAD-Free Approach to High-Fidelity Aerostructural Optimization." In: *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference.* American Institute of Aeronautics and Astronautics, June 2010. DOI: `10.2514/6.2010-9231` (cit. on p. 21).

[Khr17] Khronos Group. *glTF 2.0 Specification.* June 2017. URL: `https://github.com/KhronosGroup/glTF/tree/master/specification/2.0` (visited on 02/10/2023) (cit. on p. 53).

[KJ+23] J. C. Kirchhof, N. Jansen, B. Rumpe, and A. Wortmann. "Navigating the Low-Code Landscape: A Comparison of Development Platforms." In: *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).* IEEE, Oct. 2023. DOI: `10.1109/models-c59198.2023.00135` (cit. on p. 41).

[KR+23] J. Kleinert, A. Reiswich, M. Siggel, and M. Banović. "A Generic Parametric Modeling Engine Targeted Towards Multidisciplinary Design: Goals and Concepts." In: *CAD'23, Mexico City.* May 2023. DOI: `10.14733/cadconfp.2023.101-105` (cit. on p. 21).

[Kli16] T. Klimmek. "Statische aeroelastische Anforderungen beim multidisziplinären Strukturentwurf von Verkehrsflugzeugflügeln." PhD thesis. TU Braunschweig, 2016 (cit. on pp. 47, 48, 53, 58).

[KO+16] T. Klimmek, P. Ohme, P. D. Ciampa, and V. Handojo. "Aircraft Loads - An Important Task from Pre-Design to Loads Flight Testing." In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* Sept. 2016 (cit. on pp. 197, 198).

[KS+19] T. Klimmek, M. Schulze, M. Abu-Zurayk, C. Ilic, and A. Merle. “cpacs-MONA – An independent and in high fidelity based MDO tasks integrated process for the structural and aeroelastic design for aircraft configurations.” In: *IFASD 2019 - International Forum on Aeroelasticity and Structural Dynamics*. June 2019. URL: `https://elib.dlr.de/128099/` (cit. on pp. 20, 45, 58, 167).

[KM+02] A. Ko, W. H. Mason, B. Grossman, and J. Schetz. *A-7 Strut Braced Wing ConceptTransonic Wing Design*. Tech. rep. Multidisciplinary Analysis et al., 2002 (cit. on p. 13).

[KL+11] A. Koch et al. “Climate impact assessment of varying cruise flight altitudes applying the CATS simulation approach.” In: *3rd CEAS Air & Space Conference, 21st AIDAA Congress*. 2011 (cit. on p. 43).

[KHH13] D. S. C. Kowollik, P. Horst, and M. C. Haupt. “Fluid-structure interaction analysis applied to thermal barrier coated cooled rocket thrust chambers with subsequent local investigation of delamination phenomena.” In: *Progress in Propulsion Physics*. EDP Sciences, 2013. DOI: `10.1051/eucass/201304617` (cit. on p. 15).

[Kra88] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Tech. rep. Institut für Dynamik der Flugsysteme, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1988 (cit. on p. 18).

[Kri05] T. Krishnamurthy. “Comparison of Response Surface Construction Methods for Derivative Estimation Using Moving Least Squares, Kriging and Radial Basis Functions.” In: *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, Apr. 2005. DOI: `10.2514/6.2005-1821` (cit. on p. 19).

[KA+15] N. Kroll et al. “DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods.” In: *CEAS Aeronautical Journal* 7.1 (Dec. 2015), pp. 3–27. DOI: `10.1007/s13272-015-0179-7` (cit. on p. 23).

[Kro04] I. Kroo. “Distributed multidisciplinary design and collaborative optimization.” In: *VKI lecture series on optimization methods & tools for multicriteria/multidisciplinary design* (Jan. 2004) (cit. on pp. 5, 22).

[Kro97] I. Kroo. “Multidisciplinary optimization applications in preliminary design - Status and directions.” In: *38th Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, Apr. 1997. DOI: `10.2514/6.1997-1408` (cit. on p. 22).

[KM00] I. Kroo and V. Manning. “Collaborative optimization - Status and directions.” In: *8th Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, Sept. 2000. DOI: `10.2514/6.2000-4721` (cit. on p. 24).

[KT51] H. W. Kuhn and A. W. Tucker. “Nonlinear Programming.” In: *Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 2. 1951, pp. 481–492 (cit. on p. 17).

[KD+11] O. Kuhn, T. Dusch, P. Ghodous, and P. Collet. “Knowledge-Based Engineering Template Instances Update Support.” In: *Enterprise Information Systems*. Springer Berlin Heidelberg, 2011, pp. 151–163. DOI: `10.1007/978-3-642-19802-1_11` (cit. on p. 88).

[Kul08] B. M. Kulfan. “Universal Parametric Geometry Representation Method.” In: *Journal of Aircraft* 45.1 (Jan. 2008), pp. 142–158. DOI: `10.2514/1.29958` (cit. on pp. 33, 34, 255).

[KL+17] A. R. Kulkarni, G. La Rocca, T. van den Berg, and R. van Dijk. “A knowledge based engineering tool to support front-loading and multidisciplinary design optimization of the fin-rudder interface.” In: *Aerospace Europe 6th CEAS Conference*. 2017 (cit. on p. 74).

[KB+22] N. Külper, J. Bröhan, T. Bielsky, and F. Thielecke. “Systems architecting assistant (SArA): enabling a seamless process chain from requirements to overall systems design.” In: *33rd Congress of the International Council of the Aeronautical Sciences (ICAS).* Sept. 2022. URL: https://www.icas.org/ICAS_ARCHIVE/ICAS2022/data/preview/ICAS2022_0665.htm (cit. on p. 220).

[LLB12] 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.” In: *28th Congress of the International Council of the Aeronautical Sciences (ICAS), Brisbane, Australia.* 2012 (cit. on pp. 8, 71).

[LaR11] G. La Rocca. “Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization.” PhD thesis. Delft University of Technology, 2011 (cit. on pp. 5, 6, 12, 27, 67, 69–71, 73, 75, 76, 79, 80, 83, 88, 89, 93, 256).

[LaR15] G. La Rocca. “Multidisciplinary Design Optimization Supported by Knowledge Based Engineering.” In: ed. by J. Sobieszczanski-Sobieski, A. Morris, and M. van Tooren. John Wiley & Sons, Ltd., Sept. 2015. Chap. Knowledge Based Engineering, pp. 208–257. ISBN: 1118492129 (cit. on pp. 67, 70).

[LKT02] G. La Rocca, L. Krakers, and M. van Tooren. “Development of an ICAD Generative Model for Blended Wing-Body Aircraft Design.” In: *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.* American Institute of Aeronautics and Astronautics, Sept. 2002. DOI: 10.2514/6.2002-5447 (cit. on p. 74).

[Laa08] A. H. van der Laan. “Knowledge based engineering support for aircraft component design.” PhD thesis. Delft University of Technology, 2008 (cit. on pp. 73, 74).

[LDL19] R. Lafage, S. Defoort, and T. Lefebvre. “WhatsOpt: a web application for multidisciplinary design analysis and optimization.” In: (June 2019). DOI: 10.2514/6.2019-2990 (cit. on p. 24).

[LK+09] X. B. Lam, Y. S. Kim, A. D. Hoang, and C. W. Park. “Coupled Aerostructural Design Optimization Using the Kriging Model and Integrated Multiobjective Optimization Algorithm.” In: *Journal of Optimization Theory and Applications* 142.3 (Mar. 2009), pp. 533–556. DOI: 10.1007/s10957-009-9520-9 (cit. on p. 19).

[LM12] A. B. Lambe and J. R. R. A. Martins. “Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes.” In: *Structural and Multidisciplinary Optimization* 46.2 (Jan. 2012), pp. 273–284. DOI: 10.1007/s00158-012-0763-y (cit. on pp. 15, 16, 172).

[LW94] H. Lamousin and N. Waggenspack. “NURBS-based free-form deformations.” In: *IEEE Computer Graphics and Applications* 14.6 (Nov. 1994), pp. 59–65. DOI: 10.1109/38.329096 (cit. on p. 21).

[LB19] S. C. Langer and C. Blech. “Cabin noise prediction using wave-resolving aircraft models.” In: *PAMM* 19.1 (Nov. 2019). DOI: 10.1002/pamm.201900388 (cit. on p. 62).

[LHW09] D. Lazzara, R. Haimes, and K. Willcox. “Multifidelity Geometry and Analysis in Aircraft Conceptual Design.” In: *19th AIAA Computational Fluid Dynamics.* American Institute of Aeronautics and Astronautics, June 2009. DOI: 10.2514/6.2009-3806 (cit. on p. 22).

[LEK06] C. Ledermann, P. Ermanni, and R. Kelm. “Dynamic CAD objects for structural optimization in preliminary aircraft design.” In: *Aerospace Science and Technology* 10.7 (Oct. 2006), pp. 601–610. DOI: 10.1016/j.ast.2006.07.001 (cit. on pp. 54–57, 74, 255).

[LH+04] S. LeDoux, W. Herling, G. Fatta, and R. Ratcliff. “MDOPT - A Multidisciplinary Design Optimization System Using Higher Order Analysis Codes.” In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.* American Institute of Aeronautics and Astronautics, Aug. 2004. DOI: 10.2514/6.2004-4567 (cit. on p. 20).

[LF+21] D. Lee et al. "The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018." In: *Atmospheric Environment* 244 (Jan. 2021), p. 117834. DOI: 10.1016/j.atmosenv.2020.117834 (cit. on p. 12).

[Lee89] E. Lee. "Choosing nodes in parametric curve interpolation." In: *Computer-Aided Design* 21.6 (July 1989), pp. 363–370. DOI: 10.1016/0010-4485(89)90003-1 (cit. on p. 32).

[Lee03] S. Lee. "Konzeptionelle Untersuchung einer Flying Wing Zweideckkonfiguration." MA thesis. Hochschule für angewandte Wissenschaften Hamburg, 2003 (cit. on pp. 13, 61, 164).

[LLL17] I. Levkivskyi, J. Lehtosalo, and Ł. Langa. *PEP 544 – Protocols: Structural subtyping (static duck typing).* Mar. 2017. URL: https://peps.python.org/pep-0544/ (visited on 02/11/2023) (cit. on p. 82).

[Lie04] R. H. Liebeck. "Design of the Blended Wing Body Subsonic Transport." In: *Journal of Aircraft* 41.1 (Jan. 2004), pp. 10–25. DOI: 10.2514/1.9084 (cit. on pp. 12, 189).

[LH11] C. M. Liersch and M. Hepperle. "A distributed toolbox for multidisciplinary preliminary aircraft design." In: *CEAS Aeronautical Journal* 2.1-4 (Aug. 2011), pp. 57–68. DOI: 10.1007/s13272-011-0024-6 (cit. on pp. 42, 100).

[LMc23] J. Linietsky, A. Manzur, and contributors. *Godot Engine - Free and open source 2D and 3D game engine.* 2023. URL: https://godotengine.org/ (visited on 02/10/2023) (cit. on p. 53).

[Lip18] M. Lipovaca. *Learn You a Haskell for Great Good!* Random House LCC US, May 2018. ISBN: 1593272839 (cit. on p. 93).

[Lom96] T. L. Lomax. *Structural Loads Analysis.* AIAA, Jan. 1996. 280 pp. ISBN: 1563471140. URL: l (cit. on p. 46).

[Luf14] Lufthansa Cargo AG. *Pallets, containers and fleet.* Apr. 2014. URL: https://lufthansa-cargo.com/documents/20184/26391/Brochure_Pallets_Containers_and_Fleet_2019.pdf/1d182b27-2271-1666-bbe5-699218f14636 (visited on 02/10/2023) (cit. on pp. 259, 266).

[LW+15] T. W. Lukaczyk et al. "SUAVE: An Open-Source Environment for Multi-Fidelity Conceptual Vehicle Design." In: *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.* American Institute of Aeronautics and Astronautics, June 2015. DOI: 10.2514/6.2015-3087 (cit. on p. 8).

[Lut10] M. Lutz. *Programming Python.* 4th ed. O'Reilly, 2010, p. 1584. ISBN: 9780596158101 (cit. on p. 93).

[MC+17] T. MacDonald, M. Clarke, E. M. Botero, J. M. Vegh, and J. J. Alonso. "SUAVE: An Open-Source Environment Enabling Multi-Fidelity Vehicle Optimization." In: *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference.* American Institute of Aeronautics and Astronautics, June 2017. DOI: 10.2514/6.2017-4437 (cit. on p. 8).

[MG+18] R. Maierl et al. "Aero-structural Optimization of a MALE Configuration in the AGILE MDO Framework." In: *31st Congress of the International Council of the Aeronautical Sciences (ICAS), Belo Horizonte, Brazil.* Sept. 2018. URL: https://www.icas.org/ICAS_ARCHIVE/ICAS2018/data/papers/ICAS2018_0675_paper.pdf (cit. on p. 53).

[MPD13] R. Maierl, Ö. Petersson, and F. Daoud. "Automated creation of aeroelastic optimization models from a parameterized geometry." In: *15th International Forum on Aeroelasticity and Structural Dynamics, Bristol.* 2013 (cit. on pp. 45, 55, 58, 59).

[MZM21] A. Mancini, J. Zamboni, and E. Moerland. “A Knowledge-based Methodology for the Initiation of Military Aircraft Configurations.” In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021. DOI: `10.2514/6.2021-2789` (cit. on p. 8).

[MM02] S. Maneewongvatana and D. Mount. “Analysis of approximate nearest neighbor searching with clustered point sets.” In: (Dec. 2002), pp. 105–123. DOI: `10.1090/dimacs/059/06` (cit. on p. 196).

[MS+22] N. Markusheska et al. “Implementing a system architecture model for automated aircraft cabin assembly processes.” In: *CEAS Aeronautical Journal* (May 2022). DOI: `10.1007/s13272-022-00582-6` (cit. on pp. 43, 44, 65, 220).

[Mar13a] S. Marlow. *Parallel and Concurrent Programming in Haskell*. O’Reilly Media, Inc, USA, Aug. 2013. 256 pp. ISBN: 1449335942 (cit. on p. 95).

[Mar13b] D. D. Marshall. “Creating Exact Bezier Representations of CST Shapes.” In: *21st AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2013. DOI: `10.2514/6.2013-3077` (cit. on p. 33).

[MH13] J. R. R. A. Martins and J. T. Hwang. “Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models.” In: *AIAA Journal* 51.11 (Nov. 2013), pp. 2582–2599. DOI: `10.2514/1.j052184` (cit. on p. 18).

[ML13] J. R. R. A. Martins and A. B. Lambe. “Multidisciplinary Design Optimization: A Survey of Architectures.” In: *AIAA Journal* 51.9 (Sept. 2013), pp. 2049–2075. DOI: `10.2514/1.J051895` (cit. on p. 17).

[MN21] J. R. R. A. Martins and A. Ning. *Engineering Design Optimization*. Cambridge University Press, Nov. 2021. 475 pp. ISBN: 1108833411. URL: `https://www.ebook.de/de/product/41096500/joaquim_r_r_a_martins_andrew_ning_engineering_design_optimization.html` (cit. on pp. 18, 19).

[MSA03] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. “The complex-step derivative approximation.” In: *ACM Transactions on Mathematical Software* 29.3 (Sept. 2003), pp. 245–262. DOI: `10.1145/838250.838251` (cit. on p. 19).

[MD00] D. N. Mavris and D. DeLaurentis. “Methodology for Examining the Simultaneous Impact of Requirements, Vehicle Characteristics, and Technologies on Military Aircraft Design.” In: *22nd International Congress of Aeronautical Sciences (ICAS), Harrogate, UK*. 2000 (cit. on pp. 6, 7, 255).

[MTA08] D. N. Mavris, C. de Tenorio, and M. Armstrong. “Methodology for Aircraft System Architecture Definition.” In: *46th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2008. DOI: `10.2514/6.2008-149` (cit. on pp. 5, 14).

[McC84] L. A. McCullers. “Aircraft configuration optimization including optimized flight profiles.” In: *Symposium on Recent Experiences in Multidisciplinary Analysis and Optimization*. 1984, pp. 395–412 (cit. on p. 8).

[MG22] R. A. McDonald and J. R. Gloudemans. “Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design.” In: *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: `10.2514/6.2022-0004` (cit. on pp. 8, 9).

[McK10] W. McKinney. “Data Structures for Statistical Computing in Python.” In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and J. Millman. 2010, pp. 56–61. DOI: `10.25080/Majora-92bf1922-00a` (cit. on p. 81).

[MSR17] A. Merle, A. Stueck, and A. Rempke. “An Adjoint-based Aerodynamic Shape Optimization Strategy for Trimmed Aircraft with Active Engines.” In: *35th AIAA Applied Aerodynamics Conference*. American Institute of Aeronautics and Astronautics, June 2017. DOI: `10.2514/6.2017-3754` (cit. on p. 21).

[Met08] R. Metzger. “Gesamtheitliche Optimierung von Rumpfquerschnitten im Flugzeugvorentwurf.” PhD thesis. München: TU München, 2008. ISBN: 9783899638141 (cit. on p. 9).

[MY10] S. Minami and S. Yoshimura. “Performance evaluation of nonlinear algorithms with linesearch for partitioned coupling techniques for fluid-structure interactions.” In: *International Journal for Numerical Methods in Fluids* 64.10-12 (Nov. 2010), pp. 1129–1147. DOI: `10.1002/fld.2274` (cit. on p. 16).

[MBN15] E. Moerland, R.-G. Becker, and B. Nagel. “Collaborative understanding of disciplinary correlations using a low-fidelity physics-based aerospace toolkit.” In: *CEAS Aeronautical Journal* 6.3 (Apr. 2015), pp. 441–454. DOI: `10.1007/s13272-015-0153-4` (cit. on p. 22).

[MD+16] E. Moerland, S. Deinert, F. Daoud, J. Dornwald, and B. Nagel. “Collaborative aircraft design using an integrated and distributed multidisciplinary product development process.” In: *30th Congress of the internatinal Council of the Aeronautical Sciences (ICAS), Daejon, Korea*. Sept. 2016. URL: `https://elib.dlr.de/111005/` (cit. on p. 5).

[MT97] T. Möller and B. Trumbore. “Fast, Minimum Storage Ray-Triangle Intersection.” In: *Journal of Graphics Tools* 2.1 (Jan. 1997), pp. 21–28. DOI: `10.1080/10867651.1997.10487468` (cit. on p. 50).

[Moo12] K. Moore. “Calculation of Sensitivity Derivatives in an MDAO Framework.” In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, Sept. 2012. DOI: `10.2514/6.2012-5481` (cit. on p. 19).

[Mot16] M. Motzer. “Integrierte Flugzeugrumpf- und Kabinenentwicklung mit graphenbasierten Entwurfssprachen.” PhD thesis. Unversität Stuttgart, 2016 (cit. on pp. 11, 43, 73, 75).

[Mun14] R. C. Munjulury. “Knowledge Based Integrated Multidisciplinary Aircraft Conceptual Design.” MA thesis. Linköping University, 2014 (cit. on p. 11).

[Mun17] R. C. Munjulury. “Knowledge-Based Integrated Aircraft Design: An Applied Approach from Design to Concept Demonstration.” PhD thesis. Linköping University, 2017 (cit. on pp. 8, 74, 75).

[MS+15] R. C. Munjulury, I. Staack, P. Berry, and P. Krus. “A knowledge-based integrated aircraft conceptual design framework.” In: *CEAS Aeronautical Journal* 7.1 (Nov. 2015), pp. 95–105. DOI: `10.1007/s13272-015-0174-z` (cit. on pp. 8, 11).

[Nag21] B. Nagel. “Ein Beitrag zur strukturellen Vorauslegung von Tragflächen im digitalen Zulieferverbund des Flugzeugbaus.” PhD thesis. Otto-von-Guericke-Universität Magdeburg, 2021 (cit. on pp. 48, 57).

[NKS08] B. Nagel, M. Kintscher, and T. Streit. “Active and passive structural measures for aeroelastic winglet design.” In: *26th Congress of the International Council of the Aeronautical Sciences (ICAS), Anchorage, Alaska, USA*. Sept. 2008 (cit. on p. 47).

[NB+12] B. Nagel et al. “Communication in Aircraft Design: Can We Establish a Common Language?” In: *28th Congress of the International Council of the Aeronautical Sciences (ICAS), Brisbane, Australia*. 2012 (cit. on pp. 27, 42).

[Nah86] M. N. Nahas. “Survey of Failure and Post-Failure Theories of Laminated Fiber-Renforced Composites.” In: *Journal of Composites Technology and Research* 8.4 (1986), p. 138. DOI: `10.1520/ctr10335j` (cit. on p. 17).

[NH22] R. K. Nangia and L. Hyde. “Arriving at certifiable novel airliner using liquid hydrogen & efficiancy metrics.” In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden*. Sept. 2022 (cit. on p. 185).

[Neg11] M. Negnevitsky. *Artificial Intelligence*. Pearson Education Limited, May 2011. 504 pp. ISBN: 1408225743 (cit. on pp. 70–72, 255).

[Nic12] C. Nickol. “Hybrid Wing Body Configuration Scaling Study.” In: *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Jan. 2012. DOI: 10.2514/6.2012-337 (cit. on p. 13).

[NM+16] F. Nicolosi, A. D. Marco, L. Attanasio, and P. D. Vecchia. “Development of a Java-Based Framework for Aircraft Preliminary Design and Optimization.” In: *Journal of Aerospace Information Systems* 13.6 (June 2016), pp. 234–242. DOI: 10.2514/1.i010404 (cit. on p. 45).

[NS10] M. Nita and D. Scholz. “From preliminary aircraft cabin design to cabin optimization.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2010 (cit. on p. 9).

[Nit13] M. F. Nita. “Contributions to Aircraft Preliminary Design and Optimization.” PhD thesis. Hamburg University of Applied Sciences, 2013 (cit. on p. 8).

[Niu88] M. C. Y. Niu. *Airframe Structural Design: Practical Design Information and Data on Aircraft Structures*. Hong Kong Los Angeles, Calif: Conmilit Press Technical Book Co. distributor, 1988. ISBN: 962712804X (cit. on pp. 9, 54, 106, 109, 115, 119, 138, 141).

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag GmbH, Sept. 2006. ISBN: 0387303030 (cit. on p. 18).

[Noe23] Noesis Solutions. *Optimus | Noesis Solutions | Noesis Solutions*. 2023. URL: https://www.noesissolutions.com/our-products/optimus/ (visited on 02/14/2023) (cit. on p. 23).

[Num23] NumFOCUS, Inc. *pandas - Python Data Anaylsis Library*. 2023. URL: https://pandas.pydata.org/ (visited on 02/11/2023) (cit. on p. 81).

[OMG17] Object Management Group (OMG). *OMG Unified Modeling Language (OMG UML) - Version 2.5.1*. Tech. rep. Dec. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF (visited on 02/10/2023) (cit. on p. 43).

[OMG19] Object Management Group (OMG). *OMG Systems Modeling Language (OMG SysML) - Version 1.6*. Tech. rep. Nov. 2019. URL: https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf (visited on 02/10/2023) (cit. on p. 43).

[OCC23a] Open Cascade. *Open Cascade - software development company*. 2023. URL: https://www.opencascade.com/ (visited on 02/10/2023) (cit. on pp. 21, 37, 41, 259).

[OCC23b] Open Cascade. *Open CASCADE Technology User Guide: Mesh*. Jan. 2023. URL: https://dev.opencascade.org/doc/occt-7.7.0/overview/html/occt_user_guides__mesh.html (visited on 01/30/2023) (cit. on pp. 206, 258).

[Öst03] C. M. Österheld. “Physikalisch begründete Analyseverfahren im integrierten Flugzeugvorentwurf.” PhD thesis. TU Braunschweig, 2003 (cit. on pp. 7, 13, 54–56, 107, 109, 115, 255).

[PB+20] A. Page Risueño, J. Bussemaker, P. D. Ciampa, and B. Nagel. “MDAx: Agile Generation of Collaborative MDAO Workflows for Complex Systems.” In: *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, June 2020. DOI: 10.2514/6.2020-3133 (cit. on pp. 24, 76, 219).

[PN19] J. Page Risueño and B. Nagel. "Development of a Knowledge-Based Engineering Framework for Modeling Aircraft Production." In: *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019. DOI: 10.2514/6.2019-2889 (cit. on pp. 43, 65).

[PT+18] A. Papageorgiou, M. Tarkian, K. Amadori, and J. Ölvander. "Multidisciplinary Design Optimization of Aerial Vehicles: A Review of Recent Advancements." In: *International Journal of Aerospace Engineering* 2018 (2018), pp. 1–21. DOI: 10.1155/2018/4258020 (cit. on p. 19).

[Pap17] V. Papantoni. "Towards Active Structural Psychoacoustic Control." PhD thesis. TU Braunschweig, 2017 (cit. on p. 63).

[PGG13] D. J. Pate, J. Gray, and B. J. German. "A graph theoretic approach to problem formulation for multidisciplinary design analysis and optimization." In: *Structural and Multidisciplinary Optimization* 49.5 (Oct. 2013), pp. 743–760. DOI: 10.1007/s00158-013-1006-6 (cit. on pp. 24, 83, 85, 172, 219).

[PGB95] S. N. Patnaik, J. D. Guptill, and L. Berke. "Merits and limitations of optimality criteria method for structural optimization." In: *International Journal for Numerical Methods in Engineering* 38.18 (Sept. 1995), pp. 3087–3120. DOI: 10.1002/nme.1620381806 (cit. on p. 47).

[PM10] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer Berlin Heidelberg, Feb. 2010. 424 pp. ISBN: 364204073X (cit. on p. 38).

[Pav20] T. Paviot. *pythonocc*. en. 2020. DOI: 10.5281/ZENODO.3605364 (cit. on pp. 41, 79).

[PG20] H. Percival and B. Gregory. *Architecture Patterns with Python*. O'Reilly UK Ltd., Apr. 2020. ISBN: 1492052205 (cit. on p. 70).

[Pet15] M. Petsch. "Erweiterung einer Auslegungsprozesskette für Transportflugzeuge auf Konfigurationen mit abgestrebten Tragflügeln." MA thesis. Universität Stuttgart, 2015 (cit. on pp. 57, 117).

[PKH18] M. Petsch, D. Kohlgrüber, and J. Heubischl. "PANDORA - A python based framework for modelling and structural sizing of transport aircraft." In: *MATEC Web of Conferences* 233 (2018). Ed. by K. Kontis and S. Pantelakis, p. 00013. DOI: 10.1051/matecconf/201823300013 (cit. on pp. 45, 57).

[PM+17] T. Pfeiffer et al. "Ergebnisse des Flugzeugvorentwurfprojekts "FrEACs" (Future Enhanced Aircraft Configurations)." In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2017 (cit. on p. 12).

[Pho75] B. T. Phong. "Illumination for computer generated pictures." In: *Communications of the ACM* 18.6 (June 1975), pp. 311–317. DOI: 10.1145/360825.360839 (cit. on p. 51).

[PBC17] M. Picchi Scardaoni, V. Binante, and V. Cipolla. "WAGNER: a new code for parametrical structural study of fuselages of civil transport aircraft." In: *Aerotecnica Missili & Spazio* 96.3 (2017). ISSN: 03657442. DOI: 10.19249/ams.v96i3.311 (cit. on p. 59).

[PM20] M. Picchi Scardaoni and M. Montemurro. "A general global-local modelling framework for the deterministic optimisation of composite structures." In: *Structural and Multidisciplinary Optimization* 62.4 (May 2020), pp. 1927–1949. DOI: 10.1007/s00158-020-02586-4 (cit. on p. 47).

[PT96] L. Piegl and W. Tiller. *The NURBS Book*. Springer-Verlag GmbH, Nov. 14, 1996. ISBN: 3540615458 (cit. on pp. 28, 34, 111, 262).

[PDH13] P. Piperni, A. DeBlois, and R. Henderson. "Development of a Multilevel Multidisciplinary-Optimization Capability for an Industrial Environment." In: *AIAA Journal* 51.10 (Oct. 2013), pp. 2335–2352. DOI: 10.2514/1.j052180 (cit. on p. 20).

[Plu21] R. van der Pluijm. “Cockpit Design and Integration into the Flying V.” MA thesis. TU Delft, 2021 (cit. on p. 79).

[Poe84] T. Poeschl. “Detecting surface irregularities using isophotes.” In: *Computer Aided Geometric Design* 1.2 (Nov. 1984), pp. 163–168. DOI: 10.1016/0167-8396(84)90028-1 (cit. on p. 132).

[Poh86] H. Pohl. “Ein Beitrag zur quantitativen Analyse des Einflusses von Auslegungsparametern auf den optimalen Entwurf von Verkehrsflugzeugen.” PhD thesis. TU Braunschweig, 1986 (cit. on p. 8).

[PR+17] T. Polacsek et al. “Towards Thinking Manufacturing and Design Together: An Aeronautical Case Study.” In: *Conceptual Modeling*. Springer International Publishing, 2017, pp. 340–353. DOI: 10.1007/978-3-319-69904-2_27 (cit. on p. 12).

[Por18] C. Pornet. “Conceptual Design Methods for Sizing and Performance of Hybrid-Electric Transport Aircraft.” PhD thesis. Technical University of Munich, June 2018. URL: http://mediatum.ub.tum.de?id=1399547 (cit. on p. 12).

[Pow94] M. J. D. Powell. “Advances in Optimization and Numerical Analysis.” In: ed. by S. Gomez and J.-P. Hennart. Springer Netherlands, 1994. Chap. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation, pp. 51–67. DOI: 10.1007/978-94-015-8330-5 (cit. on p. 19).

[Pow09] M. J. D. Powell. *The BOBYQA algorithm for bound constrained optimization without derivatives*. Tech. rep. Department of Applied Mathematics and Theoretical Physics, Cambridge University, 2009 (cit. on p. 19).

[PT+07] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes*. Cambridge University Pr., Sept. 2007. 1248 pp. ISBN: 0521880688 (cit. on p. 18).

[Puc96] A. Puck. *Festigkeitsanalyse von Faser-Matrix-Laminaten*. Carl Hanser Verlag, 1996. ISBN: 978-3-446-18194-6 (cit. on p. 47).

[Pyt23] Python Software Foundation. *Welcome to Python.org*. 2023. URL: https://www.python.org/ (visited on 02/11/2023) (cit. on p. 79).

[QA21a] J. Qian and J. J. Alonso. “Structural Optimization of Blended Wing Body Transport Aircraft With Buckling Constraints.” In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-3022 (cit. on pp. 13, 59).

[QA21b] J. Qian and J. J. Alonso. “Wing Structural Optimization through Highly-Parameterized Design.” In: *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2021. DOI: 10.2514/6.2021-1967 (cit. on p. 59).

[QMM05] G. Quaranta, P. Masarati, and P. Mantegazza. “A conservative mesh-free approach for fluid structure problems.” In: *Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering*. 2005 (cit. on p. 198).

[QG16] J. R. Quinlan and F. H. Gern. “Optimization of an Advanced Hybrid Wing Body Concept using HCDstruct Version 1.2.” In: *16th AIAA Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics, June 2016. DOI: 10.2514/6.2016-3460 (cit. on p. 53).

[QG17] J. R. Quinlan and F. H. Gern. “Extension of HCDstruct for Transonic Aeroservoelastic Analysis of Unconventional Aircraft Concepts.” In: *17th AIAA Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics, June 2017. DOI: 10.2514/6.2017-4379 (cit. on p. 59).

[RB+21] F. Rauscher, J. Biedermann, A. Gindorf, F. Meller, and B. Nagel. “Permanente geometrische Digitalisierung der Flugzeugkabine zur Änderungsnachverfolgung.” de. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2021. DOI: `10.25967/530008` (cit. on pp. 150, 211).

[Ray89] D. P. Raymer. *Aircraft design: A conceptual approach*. Washington, D.C: American Institute of Aeronautics and Astronautics, 1989. ISBN: 0930403517 (cit. on pp. 5, 10).

[Ray16] D. P. Raymer. “RDSwin: Seamlessly-Integrated Aircraft Conceptual Design for Students & Professionals.” In: *54th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: `10.2514/6.2016-1277` (cit. on p. 8).

[Reb93] S. Rebay. “Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm.” In: *Journal of Computational Physics* 106.1 (May 1993), pp. 125–138. DOI: `10.1006/jcph.1993.1097` (cit. on p. 51).

[RN+12] F. Rebelo, P. Noriega, E. Duarte, and M. Soares. “Using Virtual Reality to Assess User Experience.” In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 54.6 (Nov. 2012), pp. 964–982. DOI: `10.1177/0018720812465006` (cit. on p. 61).

[Rec22] D. Reckzeh. “Technology Integration for Future Aircraft Configurations.” In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden*. Sept. 2022 (cit. on p. 12).

[RR+20] F. Reimer et al. “Applikation des Design Thinking Ansatzes auf den Flugzeugkabinen Entwurfsprozess.” de. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2020. DOI: `10.25967/530203` (cit. on pp. 61, 178).

[Rei15] L. Reimer. “The FlowSimulator – A software framework for CFD-related multidisciplinary simulations.” In: *NAFEMS European Conference: Computational Fluid Dynamics (CFD) – Beyond the Solve, Munich 2015*. 2015 (cit. on pp. 20, 22).

[RJT98] D. Rentema, F. Jansen, and E. Torenbeek. “The application of AI and geometric modelling techniques in conceptual aircraft design.” In: *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, Aug. 1998. DOI: `10.2514/6.1998-4824` (cit. on p. 74).

[Ric07] T. Richter. “Simulationsmethodik zur Effizienz- und Komfortbewertung von Menschenflussprozessen in Verkehrsflugzeugen.” PhD thesis. TU München, 2007 (cit. on p. 65).

[Rie13] J. Rieke. “Bewertung von CFK-Strukturen in einem multidisziplinären Entwurfsansatz für Verkehrsflugzeuge.” PhD thesis. TU Braunschweig, Mar. 2013. 260 pp. ISBN: 3954043688 (cit. on pp. 54, 55).

[Ris16] K. Risse. “Preliminary overall aircraft design with hybrid laminar flow control.” PhD thesis. Aachen: RWTH Aachen University, 2016. ISBN: 9783844049503 (cit. on pp. 8, 44).

[RA+12] K. Risse, E. Anton, T. Lammering, K. Franz, and R. Hoernschemeyer. “An Integrated Environment for Preliminary Aircraft Design and Optimization.” In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference & 20th AIAA/ASME/AHS Adaptive Structures Conference & 14th AIAA*. American Institute of Aeronautics and Astronautics, Apr. 2012. DOI: `10.2514/6.2012-1675` (cit. on pp. 8, 42).

[RZ+12] A. Rizzi, M. Zhang, B. Nagel, D. Boehnke, and P. Saquet. “Towards a Unified Framework using CPACS for Geometry Management in Aircraft Design.” In: *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, Jan. 2012. DOI: `10.2514/6.2012-549` (cit. on p. 8).

[McN23] Robert McNeel & Associates. *Rhino - Rhinoceros 3D*. 2023. URL: `https://www.rhino3d.com/` (visited on 02/10/2023) (cit. on p. 40).

[Ron06] A. Ronzheimer. “Prospects of Geometry Parameterization based on Freeform Deformation in MDO.” In: *ERCOFTAC 2006 International Conference*. Ed. by G. Winter et al. DLR Projekt MEGAOPT. Nov. 2006, pp. 1–10. URL: https://elib.dlr.de/46472/ (cit. on p. 20).

[Ros97] J. Roskam. *Airplane Design - Part I: Preliminary Sizing of Airplanes*. Lawrence, Kan: DARcorporation, 1997. ISBN: 9781884885433 (cit. on pp. 5, 8).

[Ros02] J. Roskam. *Airplane Design - Part III: Layout Design of Cockpit, Fuselage, Wing and Empennage: Cutaways and Inboard Profiles*. Lawrence, Kan: DARcorporation, 2002. ISBN: 188488542X (cit. on pp. 9, 10).

[Ros18] J. Roskam. *Airplane Design - Part V: Component Weight Estimation*. DARcorporation, 2018 (cit. on p. 8).

[Ros98] G. van Rossum. “Glue It All Together With Python.” In: *OMG-DARPA-MCC Workshop on Compositional Software Architecture, Monterey, California*. Jan. 1998 (cit. on p. 96).

[Rot03] B. A. van de Rotten. “A limited memory Broyden method to solve high-dimensional systems of nonlinear equations.” PhD thesis. Universiteit Leiden, 2003 (cit. on p. 87).

[Row07] J. Rowley. “The wisdom hierarchy: representations of the DIKW hierarchy.” In: *Journal of Information Science* 33.2 (Feb. 2007), pp. 163–180. DOI: 10.1177/0165551506070706 (cit. on pp. 67, 68, 255).

[RR+21] F. Rudolph, F. Reimer, I. Moerland-Masic, and T.-M. Bock. “Modellierung von Personenbewegungen zur Unterstützung des Flugzeugkabinenentwurfsprozesses.” de. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2021. DOI: 10.25967/550102 (cit. on pp. 65, 177, 178, 180, 257).

[SW+89] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. “Design and Analysis of Computer Experiments.” In: *Statistical Science* 4.4 (1989), pp. 409–423 (cit. on p. 19).

[SAZ03] S. Sakata, F. Ashida, and M. Zako. “Structural optimization using Kriging approximation.” In: *Computer Methods in Applied Mechanics and Engineering* 192.7-8 (Feb. 2003), pp. 923–939. DOI: 10.1016/s0045-7825(02)00617-5 (cit. on p. 19).

[Sam04] J. Samareh. “Aerodynamic Shape Optimization Based on Free-Form Deformation.” In: *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics, Aug. 2004. DOI: 10.2514/6.2004-4630 (cit. on p. 20).

[Sam99] J. A. Samareh. “Status and Future of Geometry Modeling and Grid Generation for Design and Optimization.” In: *Journal of Aircraft* 36.1 (Jan. 1999), pp. 97–104. DOI: 10.2514/2.2417 (cit. on p. 20).

[Sán17] R. Sánchez Fernández. “A Coupled Adjoint Method for Optimal Design in Fluid-Structure Interaction problems with Large Displacements.” PhD thesis. Imperial College London, 2017 (cit. on pp. 20, 22).

[SM+22] S. Santhosh, M. C. Moruzzi, F. De Crescenzio, and S. Bagassi. “Auralization of Noise in a Virtual Reality Aircraft Cabin for Passenger Well Being Using Human Centred Approach.” In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden*. Sept. 2022 (cit. on p. 63).

[SPG02] M. J. Sasena, P. Papalambros, and P. Goovaerts. “Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization.” In: *Engineering Optimization* 34.3 (Jan. 2002), pp. 263–278. DOI: 10.1080/03052150211751 (cit. on p. 19).

[Sch17] L. A. van der Schaft. “Development, Model Generation and Analysis of a Flying V Structure Concept.” MA thesis. TU Delft, 2017 (cit. on p. 79).

[SW16] P. Schatrow and M. Waimer. "Crash concept for composite transport aircraft using mainly tensile and compressive absorption mechanisms." In: *CEAS Aeronautical Journal* 7.3 (July 2016), pp. 471–482. DOI: 10.1007/s13272-016-0203-6 (cit. on pp. 60, 115).

[Sch11] J. Scherer. "Koppelung der Modellgenerierung und der statischen Dimensionierung von Flugzeugrümpfen im Vorentwurf." MA thesis. Universität Stuttgart, 2011 (cit. on p. 141).

[SK16] J. Scherer and D. Kohlgrüber. "Fuselage structures within the CPACS data format." In: *Aircraft Engineering and Aerospace Technology* 88.2 (Mar. 2016), pp. 294–302. DOI: 10.1108/aeat-02-2015-0056 (cit. on pp. 42, 55, 57, 106, 107, 117, 136, 255).

[SK+13] J. Scherer, D. Kohlgrüber, F. Dorbath, and M. Sorour. "A Finite Element Based Tool Chain for Structural Sizing of Transport Aircraft in Preliminary Aircraft Design." In: *62. Deutscher Luft- und Raumfahrtkongress*. 2013. URL: https://elib.dlr.de/84917/ (cit. on pp. 57, 58, 104, 112, 140, 141, 167, 192, 193, 201, 202, 255).

[Sch86] K. Schittkowski. "NLPQL: A fortran subroutine solving constrained nonlinear programming problems." In: *Annals of Operations Research* 5.2 (June 1986), pp. 485–500. DOI: 10.1007/bf02022087 (cit. on p. 18).

[SS+20] D. Schmeling, A. Shishkin, T. Dehne, P. Lange, and I. Gores. "Evaluation of Thermal Comfort for Novel Aircraft Cabin Ventilation Concepts." In: *Aerospace Europe Conference – AEC2020, Bordeaux, France, 2020.* This project has received funding from the Clean Sky 2 Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 755596 (ADVENT project). 2020. URL: https://elib.dlr.de/129310/ (cit. on p. 63).

[SS+21] D. Schmeling, A. Shishkin, D. Schiepel, and C. Wagner. "Analysis of aerosol dispersion in the Do728 passenger compartment." In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* 2021 (cit. on p. 63).

[SK+15] D. Schmidt et al. "Development of a Door Surround Structure with Integrated Structural Health Monitoring System." In: *Smart Intelligent Aircraft Structures (SARISTU).* Springer International Publishing, Sept. 2015, pp. 935–945. DOI: 10.1007/978-3-319-22413-8_51 (cit. on p. 141).

[SR16] J. Schmidt and S. Rudolph. "Graph-Based Design Languages: A Lingua Franca for Product Design Including Abstract Geometry." In: *IEEE Computer Graphics and Applications* 36.5 (Sept. 2016), pp. 88–93. DOI: 10.1109/mcg.2016.89 (cit. on pp. 73, 74, 256).

[Sch12] J. G. W. Schmidt. *Eine Entwurfssprache für Geometrie.* Tech. rep. Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, 2012 (cit. on pp. 38, 43).

[SE+16] M. Schmidt, M. Engelmann, T. Brügge-Zobel, M. Hornung, and M. Glas. "PAXelerate - An Open Source Passenger Flow Simulation Framework for Advanced Aircraft Cabin Layouts." In: *54th AIAA Aerospace Sciences Meeting.* American Institute of Aeronautics and Astronautics, Jan. 2016. DOI: 10.2514/6.2016-1284 (cit. on pp. 63, 65).

[Sch18] P. Schmollgruber. "Enhancement of the conceptual aircraft design process through certification constraints management and full mission simulations." PhD thesis. Université de Toulouse, Institut Supérieur de l'Aéronautique et de l'Espace, Dec. 2018 (cit. on p. 8).

[SE07] A. Schneegans and F. Ehlermann. "Applying KBE Technologies to the Early Phases of Multidisciplinary Product Design." In: *The Future of Product Development.* Springer Berlin Heidelberg, 2007, pp. 587–595. DOI: 10.1007/978-3-540-69820-3_57 (cit. on pp. 8, 12).

[SH+96] A. Schneegans, C. Haberland, M. Kokorniak, and B. Dohmke. "An object oriented approach to conceptual aircraft design through component-wise modelling." In: *20th Congress of the International Council of the Aeronautical Sciences (ICAS), Sorrento, Italy.* Sept. 1996 (cit. on p. 8).

[SML06] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*. Clifton Park, N.Y: Kitware, 2006. ISBN: 978-1-930934-19-1 (cit. on p. 52).

[SD+12] G. Schuhmacher, F. Daoud, Ö. Petersson, and M. Wagner. "Multidisciplinary Airframe Design Optimization." In: *28th International Congress of the Aeronautical Sciences, Brisbane*. 2012 (cit. on pp. 15, 20, 47).

[SA+21] F. Schültke, B. Aigner, T. Effing, P. Strathoff, and E. Stumpf. "MICADO: Overview of Recent Developments within the Conceptual Aircraft Design and Optimization Environment." en. In: (2021). DOI: 10.25967/530093 (cit. on pp. 8, 42).

[SS21] F. Schültke and E. Stumpf. "UNICADO: Development and Establishment of a University Conceptual Aircraft Design Environment." de. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2021. DOI: 10.18154/RWTH-2021-09317 (cit. on p. 8).

[Sch16] M. Schürmann. "Supersonic Business Jets in Preliminary Aircraft Design." PhD thesis. TU Braunschweig, 2016 (cit. on p. 13).

[Sch84] H. R. Schwarz. *Methode der finiten Elemente*. Vieweg+Teubner Verlag, 1984. DOI: 10.1007/978-3-322-96758-9 (cit. on p. 47).

[SKH96] J. Schweiger, J. Krammer, and H. Hoernlein. "Development and application of the integrated structural design tool LAGRANGE." In: *6th Symposium on Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics, Aug. 1996. DOI: 10.2514/6.1996-4169 (cit. on pp. 20, 48).

[Sch15] D. B. Schwinn. "Parametrised fuselage modelling to evaluate aircraft crash behaviour in early design stages." In: *International Journal of Crashworthiness* 20.5 (Mar. 2015), pp. 413–430. DOI: 10.1080/13588265.2015.1022435 (cit. on pp. 60, 106).

[SS+13] D. B. Schwinn, J. Scherer, D. Kohlgrüber, and K. Harbig. "Development of a Multidisciplinary Process Chain for the Preliminary Design of Aircraft Structures." In: *NAFEMS World Congress 2013*. June 2013. URL: https://elib.dlr.de/86040/ (cit. on pp. 57, 121, 122).

[Sed12] T. W. Sederberg. *Computer Aided Geometric Design*. Course Notes, 2012 (cit. on p. 28).

[SP86] T. W. Sederberg and S. R. Parry. "Free-form deformation of solid geometric models." In: *ACM SIGGRAPH Computer Graphics* 20.4 (Aug. 1986), pp. 151–160. DOI: 10.1145/15886.15903 (cit. on p. 21).

[See10] K. Seeckt. "Conceptual Design and Investigation of Hydrogen-Fueled Regional Freighter Aircraft." PhD thesis. KTH School of Engineering, Stockholm, 2010 (cit. on p. 12).

[SS10] K. Seeckt and D. Scholz. "Application of the Aircraft Preliminary Sizing Tool PreSTo to Kerosene and Liquid Hydrogen Fueled Regional Freighter Aircraft." In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. 2010 (cit. on p. 8).

[SF+12] D. Seider, P. M. Fischer, M. Litz, A. Schreiber, and A. Gerndt. "Open Source Software Framework for Applications in Aeronautics and Space." In: *IEEE Aerospace Conference, Big Sky, Montana, USA*. Mar. 2012 (cit. on p. 23).

[Şen10] I. Şen. "Aircraft Fuselage Design Study: Parametric Modeling, Structural Analysis, Material Evaluation and Optimization for Aircraft Fuselage." MA thesis. TU Delft, 2010 (cit. on p. 9).

[SM+03] P. Shannon et al. "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks." In: *Genome Research* 13.11 (Nov. 2003), pp. 2498–2504. DOI: 10.1101/gr.1239303 (cit. on p. 88).

[SAM09] P. Shirley, M. Ashikhmin, and S. Marschner. *Fundamentals of Computer Graphics*. Taylor & Francis Ltd., July 2009. 804 pp. ISBN: 1439865523 (cit. on pp. 48, 51, 255).

[SD+18] P. Shiva Prakasha et al. “Model Based Collaborative Design & Optimization of Blended Wing Body Aircraft Configuration: AGILE EU Project.” In: *2018 Aviation Technology, Integration, and Operations Conference*. American Institute of Aeronautics and Astronautics, June 2018. DOI: `10.2514/6.2018-4006` (cit. on pp. 164, 189).

[SP+17] D. J. L. Siedlak, O. J. Pinon, P. R. Schlais, T. M. Schmidt, and D. N. Mavris. “A digital thread approach to support manufacturing-influenced conceptual aircraft design.” In: *Research in Engineering Design* 29.2 (Sept. 2017), pp. 285–308. DOI: `10.1007/s00163-017-0269-0` (cit. on p. 5).

[SS+17] M. H. Siemann, D. B. Schwinn, J. Scherer, and D. Kohlgrüber. “Advances in numerical ditching simulation of flexible aircraft models.” In: *International Journal of Crashworthiness* 23.2 (Aug. 2017), pp. 236–251. DOI: `10.1080/13588265.2017.1359462` (cit. on p. 60).

[SK+19] M. Siggel, J. Kleinert, T. Stollenwerk, and R. Maierl. “TiGL: An Open Source Computational Geometry Library for Parametric Aircraft Design.” In: *Mathematics in Computer Science* 13.3 (July 2019), pp. 367–389. DOI: `10.1007/s11786-019-00401-y` (cit. on pp. 35, 45, 58, 59, 105).

[SS+22] M. Siggel, T. Stollenwerk, J. Kleinert, B. M. Gruber, and R. Maierl. *TiGL - The TiGL Geometry Library*. 2022. DOI: `10.5281/ZENODO.858650` (cit. on p. 45).

[SA+19] D. Silberhorn, G. Atanasov, J.-N. Walther, and T. Zill. “Assessment of Hydrogen Fuel Tank Integration at Aircraft Level.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Sept. 2019. URL: `https://elib.dlr.de/129643/` (cit. on pp. 12, 13, 184).

[Sin14] D. Singer. “Entwicklung eines Prototyps für den Einsatz von Knowledge-based Engineering in frühen Phasen des Brückenentwurfs.” MA thesis. TU München, 2014 (cit. on p. 74).

[SW18] V. Singh and K. E. Willcox. “Engineering Design with Digital Thread.” In: *AIAA Journal* 56.11 (Nov. 2018), pp. 4515–4528. DOI: `10.2514/1.j057255` (cit. on p. 5).

[Sme17] T. S. L. M. Smeets. “Development of a Knowledge-Based Engineering Application to Support Conceptual Fuselage Design.” MA thesis. TU Delft, 2017 (cit. on p. 64).

[SAE14] D. Smith, R. S. Andre, and M. Eggen. *A Transition to Advanced Mathematics*. CENGAGE LEARNING, Aug. 2014. 448 pp. ISBN: 1285463269 (cit. on p. 85).

[Smi22] M. Smith. *GT7’s Polygon Count Tweet Shows How Racing Games Have Evolved in 20 Years*. Feb. 2022. URL: `https://www.racedepartment.com/news/gt7s-polygon-count-tweet-shows-how-racing-games-have-evolved-in-20-years.462/` (visited on 08/30/2023) (cit. on p. 207).

[SMR18] SMR SA. *B2000++ and B2000++ Pro Summary*. 2018. URL: `https://www.smr.ch/products/b2000/` (visited on 02/10/2023) (cit. on p. 48).

[SH97] J. Sobieszczanski-Sobieski and R. T. Haftka. “Multidisciplinary aerospace design optimization: survey of recent developments.” In: *Structural Optimization* 14.1 (Aug. 1997), pp. 1–23. DOI: `10.1007/bf01197554` (cit. on p. 19).

[Sob95] J. Sobieszczanski-Sobieski. “Multidisciplinary Design Optimization: An Emerging New Engineering Discipline.” In: *Advances in Structural Optimization*. Springer Netherlands, 1995, pp. 483–496. DOI: `10.1007/978-94-011-0453-1_14` (cit. on pp. 5, 17, 23).

[SM+15] J. Sobieszczanski-Sobieski, A. Morris, M. van Tooren, and M. V. Tooren. *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. John Wiley & Sons, Ltd., Sept. 2015. 392 pp. ISBN: 1118492129 (cit. on p. 17).

[SKH04] P. D. Soden, A. S. Kaddour, and M. J. Hinton. “Recommendations for designers and researchers resulting from the world-wide failure exercise.” In: *Failure Criteria in Fibre-Reinforced-Polymer Composites*. Elsevier, 2004, pp. 1223–1251. DOI: `10.1016/b978-008044475-8/50039-1` (cit. on p. 17).

[SM+21] V. Srinivasan et al. "Digital Shadow Model for Automated Cabin Assembly Process." en. In: *Deutscher Luft- und Raumfahrtkongress (DLRK)*. Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V., 2021. DOI: 10.25967/550163 (cit. on p. 65).

[SM+12] I. Staack et al. "Parametric aircraft conceptual design space." In: *28th International Congress of the Aeronautical Sciences*. Jan. 2012, pp. 311–320 (cit. on pp. 8, 9).

[Ste08] J. H. Steffen. "Optimal boarding method for airline passengers." In: *Journal of Air Transport Management* 14.3 (May 2008), pp. 146–150. DOI: 10.1016/j.jairtraman.2008.03.003 (cit. on p. 64).

[Sto01] M. Stokes. *Managing Engineering Knowledge: MOKA - Methodology for Knowledge Based Engineering Applications*. London: Wiley-Blackwell, 2001. ISBN: 1860582958 (cit. on pp. 68, 69, 255).

[Sut63] I. E. Sutherland. "Sketchpad: A man-machine graphical communication system." In: *Proceedings of the May 21-23, 1963, spring joint computer conference on - AFIPS '63 (Spring)*. ACM Press, 1963. DOI: 10.1145/1461551.1461591 (cit. on p. 28).

[Sza09] B. A. Szasz. "Cabin Refurbishing Supported by Knowledge Based Engineering Software." MA thesis. HAW Hamburg, 2009 (cit. on p. 12).

[Tec23] Tecplot, Inc. *Tecplot CFD Post-processing Data Visualization and Analysis Tools*. 2023. URL: https://www.tecplot.com/ (visited on 02/10/2023) (cit. on p. 52).

[TS19] C. Thomas and H. Scheel. "Kabinenakustik in der Luftfahrtforschung." de. In: (2019). DOI: 10.25967/480290 (cit. on p. 63).

[TK07] M. van Tooren and L. Krakers. "Multi-Disciplinary Design of Aircraft Fuselage Structures." In: *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2007. DOI: 10.2514/6.2007-767 (cit. on p. 76).

[Tor76] E. Torenbeek. *Synthesis of subsonic airplane design: an introduction to the preliminary design of subsonic general aviation and transport aircraft, with emphasis on layout, aerodynamic design, propulsion and performance*. Delft: Delft University Press, 1976. 624 pp. ISBN: 9029825057 (cit. on pp. 5, 6, 61, 135, 154, 168, 169).

[TB+18] F. Torrigiani et al. "Design of the strut braced wing aircraft in the AGILE collaborative MDO framework." In: *31st Congress of the Council of the Aeronautical Sciences, Belo Horizonte, Brazil*. 2018 (cit. on p. 24).

[TW+19] F. Torrigiani, J.-N. Walther, R. Bombardieri, R. Cavallaro, and P. D. Ciampa. "Flutter Sensitivity Analysis for Wing Planform Optimization." In: *International Forum on Aeroelasticity and Structural Dynamics - IFASD 2019*. June 2019. URL: https://elib.dlr.de/128448/ (cit. on p. 20).

[TD+21] F. Torrigiani et al. "An MBSE Certification-Driven Design of a UAV MALE Configuration in the AGILE 4.0 Design Environment." In: *AIAA AVIATION 2021 FORUM*. American Institute of Aeronautics and Astronautics, July 2021. DOI: 10.2514/6.2021-3080 (cit. on pp. 59, 106).

[Tra16] L. Travaglini. "PyPAD: A framework for multidisciplinary aircraft design." PhD thesis. Politechnico Milano, 2016 (cit. on pp. 15, 55, 58, 59).

[TE+20] F. M. Troeltsch et al. "Hydrogen Powered Long Haul Aircraft with Minimized Climate Impact." In: *AIAA AVIATION 2020 FORUM*. American Institute of Aeronautics and Astronautics, June 2020. DOI: 10.2514/6.2020-2660 (cit. on p. 13).

[TUB17] TU Berlin. *PADLab – Preliminary Aircraft Design Laboratory Aircraft Synthesis, Parametric Studies and Optimization – User Guide*. 2017 (cit. on pp. 8, 9, 44).

[UNF15] United Nations Framework Convention on Climate Change. *Paris Agreement.* 2015. URL: `https://unfccc.int/sites/default/files/english_paris_agreement.pdf` (visited on 02/14/2023) (cit. on p. 1).

[Uni23] Unity Technologies. *Unity Real-Time Development Platform | 3D, 2D VR & AR Engine.* 2023. URL: `https://unity.com/` (visited on 02/11/2023) (cit. on p. 53).

[VT10] A. Velicki and P. Thrash. "Blended wing body structural concept development." In: *The Aeronautical Journal* 114.1158 (Aug. 2010), pp. 513–519. DOI: `10.1017/s0001924000004000` (cit. on p. 13).

[VS+14] F. A. C. Viana, T. W. Simpson, V. Balabanov, and V. Toropov. "Special Section on Multidisciplinary Design Optimization: Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come?" In: *AIAA Journal* 52.4 (Apr. 2014), pp. 670–690. DOI: `10.2514/1.j052375` (cit. on p. 21).

[VG+20] P. Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." In: *Nature Methods* 17.3 (Feb. 2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2` (cit. on pp. 87, 172, 196).

[Wai13] M. Waimer. "Development of a Kinematics Model for the Assessment of Global Crash Scenarios of a Composite Transport Aircraft Fuselage." PhD thesis. Universität Stuttgart, 2013 (cit. on p. 60).

[WH76] D. V. Wallerstein and G. W. Haggenmacher. "Automated fully-stressed design with NASTRAN." In: *NASA Ames Res. Center NASTRAN: User's Experiences.* 1976 (cit. on p. 47).

[WCN22] J.-N. Walther, P. D. Ciampa, and B. Nagel. "Advances in Computational Methods and Technologies in Aeronautics and Industry." In: *Advances in Computational Methods and Technologies in Aeronautics and Industry.* Ed. by D. Knoerzer, J. Periaux, and T. Tuovinen. Springer Nature Switzerland AG, Dec. 2022. Chap. Disciplinary Implications of a System Architecting Approach to Collaborative Aircraft Design, pp. 159–173. DOI: `10.1007/978-3-031-12019-0_12` (cit. on pp. 22, 23, 221).

[WC18] J.-N. Walther and P. D. Ciampa. "Knowledge-based automatic Airframe Design using CPACS." In: *Transportation Research Procedia* 29 (Feb. 2018), pp. 427–439. DOI: `10.1016/j.trpro.2018.02.038` (cit. on pp. 109, 115, 136, 138, 141, 144, 146, 192, 193, 256, 258).

[WG+19] J.-N. Walther, A.-A. Gastaldi, R. Maierl, A. Jungo, and M. Zhang. "Integration aspects of the collaborative aero-structural design of an unmanned aerial vehicle." In: *CEAS Aeronautical Journal* 11.1 (Aug. 2019), pp. 217–227. DOI: `10.1007/s13272-019-00412-2` (cit. on pp. 15, 59).

[WH+22a] J.-N. Walther, C. Hesse, M. Alder, J. Y.-C. Biedermann, and B. Nagel. "Expansion of the cabin description within the CPACS air vehicle data schema to support detailed analyses." In: *CEAS Aeronautical Journal* (Aug. 2022). DOI: `10.1007/s13272-022-00610-5` (cit. on pp. 42, 110, 111, 122, 124, 128, 148, 159, 177, 180, 256).

[WH+22b] J.-N. Walther, C. Hesse, J. Y.-C. Biedermann, and B. Nagel. "Extensible aircraft fuselage model generation for a multidisciplinary, multi-fidelity context." In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden.* Sept. 2022 (cit. on pp. 135, 162–164, 184, 185, 257).

[WK+22] J.-N. Walther, B. Kocacan, C. Hesse, A. Gindorf, and B. Nagel. "Automatic cabin virtualization based on preliminary aircraft design data." In: *CEAS Aeronautical Journal* (Jan. 2022). DOI: `10.1007/s13272-021-00568-w` (cit. on pp. 121, 150, 156, 158, 256, 257).

[WPK17] J.-N. Walther, M. Petsch, and D. Kohlgrüber. “Modeling of CPACS-based fuselage structures using Python.” In: *Aircraft Engineering and Aerospace Technology* 89.5 (Sept. 2017), pp. 644–653. DOI: 10.1108/aeat-01-2017-0028 (cit. on pp. 45, 48, 57, 58, 81, 98, 101, 102, 193).

[WB+16] B. Wassermann, T. Bog, S. Kollmannsberger, and E. Rank. “A design-through-analysis approach using the Finite Cell Method.” In: *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016).* Institute of Structural Analysis and Antiseismic Research School of Civil Engineering National Technical University of Athens (NTUA) Greece, 2016. DOI: 10.7712/100016.1984.8920 (cit. on pp. 39, 255).

[Wat81] D. F. Watson. “Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes.” In: *The Computer Journal* 24.2 (Feb. 1981), pp. 167–172. DOI: 10.1093/comjnl/24.2.167 (cit. on p. 51).

[Wav92] Wavefront Technologies. *Advanced Visualizer manual, Appendix B1.* 1992 (cit. on p. 53).

[WA+07] O. de Weck et al. “State-of-the-Art and Future Trends in Multidisciplinary Design Optimization.” In: *48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference.* American Institute of Aeronautics and Astronautics, Apr. 2007. DOI: 10.2514/6.2007-1905 (cit. on p. 20).

[Wee21] R. Weerts. “The impact behavior of thick-walled composite-overwrapped pressure vessels.” English. Proefschrift. PhD thesis. Eindhoven University of Technology, Sept. 2021. ISBN: 978-90-386-5334-1 (cit. on p. 221).

[Wei09] D. Weiss. “Geometry-Based Structural Optimization on CAD Specification Trees.” PhD thesis. ETH Zürich, 2009 (cit. on p. 56).

[WSG11] J. Wenzel, M. Sinapius, and U. Gabbert. “Primary structure mass estimation in early phases of aircraft development using the finite element method.” In: *CEAS Aeronautical Journal* 3.1 (Nov. 2011), pp. 35–44. DOI: 10.1007/s13272-011-0040-6 (cit. on pp. 56–58, 74).

[WHH08] C. Werner-Westphal, W. Heinze, and P. Horst. “Structural sizing for an unconventional, environment-friendly aircraft configuration within integrated conceptual design.” In: *Aerospace Science and Technology* 12.2 (Mar. 2008), pp. 184–194. DOI: 10.1016/j.ast.2007.05.006 (cit. on p. 54).

[WB+10] M. Widhalm, J. Brezillon, C. Ilic, and T. Leicht. “Investigation on Adjoint Based Gradient Computations for Realistic 3d Aero-Optimization.” In: *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference.* American Institute of Aeronautics and Astronautics, June 2010. DOI: 10.2514/6.2010-9129 (cit. on p. 20).

[Wie96] J. Wiedemann. *Leichtbau.* Springer Berlin Heidelberg, 1996. DOI: 10.1007/978-3-642-61032-5 (cit. on p. 112).

[WK+18] C. Winnefeld, T. Kadyk, B. Bensmann, U. Krewer, and R. Hanke-Rauschenbach. “Modelling and Designing Cryogenic Hydrogen Tanks for Future Aircraft Applications.” In: *Energies* 11.1 (Jan. 2018), p. 105. DOI: 10.3390/en11010105 (cit. on p. 13).

[WA+20] S. Wöhler, G. Atanasov, D. Silberhorn, B. Fröhler, and T. Zill. “Preliminary Aircraft Design within a Multidisciplinary and Multifidelity Design Environment.” In: *Aerospace Europe Conference – AEC2020, Bordeaux, France, 2020.* Apr. 2020. URL: https://elib.dlr.de/135245/ (cit. on p. 8).

[WH+18] S. Wöhler, J. Hartmann, E. Prenzel, and H. Kwik. “Preliminary aircraft design for a midrange reference aircraft taking advanced technologies into account as part of the AVACON project for an entry into service in 2028.” In: *Deutscher Luft- und Raumfahrtkongress (DLRK).* Aug. 2018. URL: https://elib.dlr.de/126000/ (cit. on p. 104).

[WWG22] S. Wöhler, J.-N. Walther, and W. Grimme. “Design and eco-efficiency assessment of a people mover aircraft in comparison to state-of-the-art narrow body aircraft.” In: *33rd Congress of the International Congress of the Aeronautical Sciences (ICAS), Stockholm, Sweden.* Sept. 2022 (cit. on pp. 167, 183).

[W3C08] World Wide Web Consortium (W3C). *Extensible Markup Language (XML) 1.0 (Fifth Edition).* Nov. 2008. URL: `http://www.w3.org/TR/xml/` (visited on 02/10/2023) (cit. on p. 42).

[W3C12] World Wide Web Consortium (W3C). *OWL 2 Web Ontology LanguageDocument Overview (Second Edition).* Dec. 2012. URL: `https://www.w3.org/TR/owl2-overview/` (visited on 02/10/2023) (cit. on p. 43).

[W3C23] World Wide Web Consortium (W3C). *Semantic Web.* 2023. URL: `https://www.w3.org/standards/semanticweb/` (visited on 02/10/2023) (cit. on p. 43).

[WD+21] T. F. Wunderlich, S. Dähne, L. Reimer, and A. Schuster. “Global Aerostructural Design Optimization of More Flexible Wings for Commercial Aircraft.” In: *Journal of Aircraft* 58.6 (Nov. 2021), pp. 1254–1271. DOI: `10.2514/1.c036301` (cit. on pp. 15, 45).

[XJM13] S. Xu, W. Jahn, and J.-D. Müller. “CAD-based shape optimisation with CFD using a discrete adjoint.” In: *International Journal for Numerical Methods in Fluids* 74.3 (Sept. 2013), pp. 153–168. DOI: `10.1002/fld.3844` (cit. on p. 21).

[ZZM20] J. Zamboni, A. Zamfir, and E. Moerland. “Semantic Knowledge-Based-Engineering: The Codex Framework.” In: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management.* SCITEPRESS - Science and Technology Publications, 2020. DOI: `10.5220/0010143202420249` (cit. on p. 43).

[Zep07] O. Zeplin. “The Application of Virtual Reality in the Design of Aircraft Cabins.” In: *Vortragsreihe der DLGR Bezirksgruppe Hamburg.* 2007 (cit. on p. 61).

[ZT05] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method for Solid and Structural Mechanics.* Amsterdam Boston: Elsevier Butterworth-Heinemann, 2005. ISBN: 0750663219 (cit. on p. 47).

[Zil13] T. Zill. “Model hierarchy exploitation for efficient multidisciplinary design optimization in a distributed aircraft design enviroment.” PhD thesis. TU Hamburg-Harburg, 2013 (cit. on pp. 22, 43).

[ZCN12] T. Zill, P. D. Ciampa, and B. Nagel. “Multidisciplinary Design Optimization in a Collaborative Distributed Aircraft Design System.” In: *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* American Institute of Aeronautics and Astronautics, Jan. 2012. DOI: `10.2514/6.2012-553` (cit. on p. 167).

List of Figures

List of Tables

A. Appendix

A.1. FEM stiffness matrix derivation for beam elements

Section 4.2.1.1 provides an overview of the theoretical foundations of the finite element method. An elementary task in the finite element method is the assembly of the element stiffness matrix. In the chapter, equation 4.2.2 is provided as a generalized form of the stiffness matrix computation, which is valid for all element types. To reiterate,

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} dV, \tag{A.1.1}$$

where $\mathbf{B}$ is the strain-displacement matrix and $\mathbf{C}$ is the material matrix.

The derivation of the stiffness matrix for a beam element in 2D as described e.g. by Bathe [Bat08] is provided here as an example. The degrees of freedom of the element consist of the lateral displacements and the rotations at the end points $\mathbf{d} = [d_1, \theta_1, d_2, \theta_2]$. The strain-displacement vector is given by the second derivative of the deformation of the beam $\mathbf{B} = \frac{d^2\mathbf{d}(x)}{dx^2}$, whereas the material law is constant for isotropic materials $\mathbf{C} = EI$. These relationships are inserted into equation 4.2.2. Since the beam is a 1D element, the integration is performed only along the length of the element. As a simplification, the global coordinate x can be replaced by the normalized element coordinate $s \in [0, 1]$:

$$\mathbf{K} = EI \int_{x_1^{(e)}}^{x_2^{(e)}} \left(\frac{d^2\mathbf{d}(x)}{dx^2} \right)^2 dx = \frac{EI}{l^3} \int_0^1 \left(\frac{d^2\mathbf{d}(s)}{ds^2} \right)^2 ds. \tag{A.1.2}$$

The variable l denotes the element length. A cubic HERMITE polynomial ansatz is selected for the displacements $\mathbf{v}(s)$:

$$d_1(s) = 1 - 3s^2 + 2s^3, \tag{A.1.3}$$

$$d_2(s) = l \cdot \left(s - 2s^2 + s^3\right), \tag{A.1.4}$$

$$d_3(s) = 3s^2 - 2s^3, \tag{A.1.5}$$

$$d_4(s) = l \cdot \left(-s^2 + s^3\right). \tag{A.1.6}$$

Derivation of the polynomials yields the strain-displacement vector

$$\frac{d^2\mathbf{d}(s)}{ds^2} = \left[\begin{array}{cccc} -6 + 12s, & l \cdot (-4 + 6s), & 6 - 12s, & l \cdot (-2 + 6s) \end{array}\right]. \tag{A.1.7}$$

Computing the scalar product and evaluating the integral gives the well-known stiffness matrix for beam elements

$$\mathbf{K} = \frac{EI}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ 6l & 4l^2 & -6l & 2l^2 \\ -12 & -6l & 12 & -6l \\ 6l & 2l^2 & -6l & 4l^2 \end{bmatrix}. \tag{A.1.8}$$

A.2. Comparison of skinned surface and bivariate interpolation for two profile point clouds with equal number of sample points

In this section, the impact of the selected construction method on lofted surfaces is highlighted by comparing the surfaces generated by the skinned surface algorithm on the one hand and by bivariate

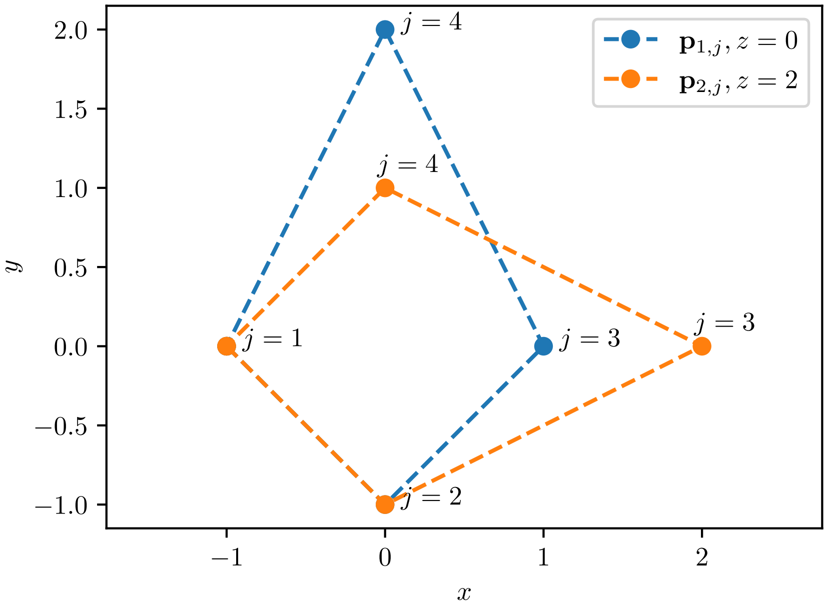


Figure A.2.1.: Point clouds for two profiles at $z = 0$ and z =1

interpolation on the other for two profile point clouds. The points along with the intended polygonal order are given in figure A.2.1.

For the skinned surface algorithm [PT96], both profiles are first interpolated independently to generate two profile curves. For this particular example, a linear interpolation with $k = 1$ and chord length parametrization is selected. Due to the choice of degree, the resulting curves C_1 and C_2 correspond to the dashed polygons in figure A.2.1 and the control points align with the original sample points indicated by the markers. Due to the chord length parametrization, both curves have different knot vectors, which reflect the normalized cumulative chord length at the given control points:

$$\mathbf{t}_1 = \left[\ 0., \quad 0., \quad 0.194, \quad 0.387, \quad 0.694, \quad 1., \quad 1.\ \right], \tag{A.2.1}$$

$$\mathbf{t}_2 = \left[\ 0., \quad 0., \quad 0.194, \quad 0.5, \quad 0.806, \quad 1., \quad 1.\ \right]. \tag{A.2.2}$$

Since the two curves do not share the same knot vector, they are *incompatible*. However, curves are required to be compatible in order to apply the skinned surface algorithm.

In order to make both curves compatible, first the union of the two knot vectors is formed:

$$\mathbf{t}_{1\cup 2} = \left[\ 0., \quad 0., \quad 0.194, \quad 0.387, \quad 0.5, \quad 0.694, \quad 0.806, \quad 1., \quad 1.\ \right]. \tag{A.2.3}$$

The resulting new knot vector contains all knot values of both curve knot vectors at the highest occurring multiplicity. Now, the missing entries from $\mathbf{t}_{1\cup 2}$ must be added using knot insertion [PT96] for each of the original curves. As a result, the number of control points is artificially increased, to match the increased number of knot positions. In the case of C_1 and C_2, two knots, and thus two control points, are added to each curve, resulting in the curves C_1^+ and C_2^+ shown in figure A.2.2.

The skinned surface algorithm can now be applied to the compatible curves to yield the surface shown in figure A.2.3. The isoparametric curves in skinning direction, which link the knots of both curves are clearly visible as discontinuities in the surface. This is because, as a result of the knot insertion, the isoparametric curve starting e.g. at the second corner point of first point cloud will end at the first additional knot on C_2^+, which lies on the straight polygon edge. Conversely, the second corner point of the second point cloud is mapped to an added node on the first curve C_1^+. The additional kinks thus result from the addition of knots to the respective knot vectors necessary to make the curves compatible.

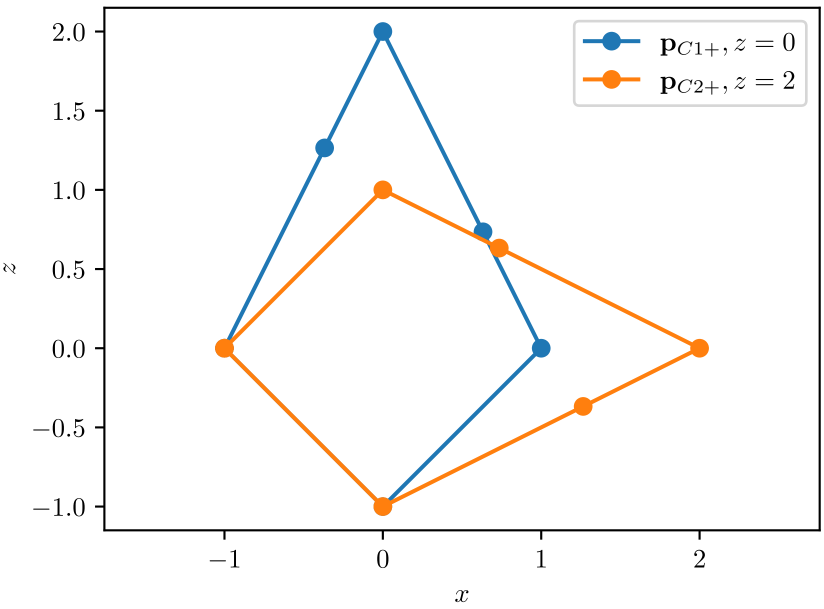


Figure A.2.2.: Compatible curves with additional control points

A different approach is taken in bivariate interpolation. Here, curve interpolation is not performed for each curve individually but for all curves at once. Again selecting linear interpolation, i.e. $k = 1$, and chord length parametrization, the global parameter vector is furthermore chosen to be the mean of the individual parameter vectors. For the example, the approach yields the knot vector

$$\mathbf{t}_{inter} = \begin{bmatrix} 0., & 0., & 0.194, & 0.443, & 0.75, & 1., & 1. \end{bmatrix}. \tag{A.2.4}$$

Since the length of the knot vector is coupled to the number of control points, selecting a single knot vector is only possible if the number of sample points is the same in all profile point clouds, allowing them to be assembled into a 2D point cloud grid.

Applying the bivariate interpolation then yields the surface shown in figure A.2.4. Compared to figure A.2.3, the surface appears more visually structured and less complex. Visibly, the isoparametric curves lead from one corner point of one profile exactly to the respective counterpart in the other. In this way, the number of kinks is minimized, which is also reflected by the shorter knot vector.

At first glance, comparing figures A.2.3 and A.2.4 may lead to the conclusion that bivariate interpolation, if applicable, yields the superior surfaces. However, in truth, which algorithm is best largely depends on the specific application. The example shown here clearly is a comparatively tidy edge case, where the number of sample points and the degree of the of the interpolation are very low. As a result, the chosen parametrization settings for the curve interpolation have no effect on the quality of the resulting profile curve. The result of the averaging approach for the bivariate interpolation is thus of the same quality as for the individual curve interpolation. This will likely not be the case for higher order interpolation, as illustrated by figure 4.1.4. For many applications, it could thus be expedient to interpolate each curve individually. In addition, the phenomenon of the undesirable discontinuities in the skinned surface is likely to be mitigated as well, as the order of the interpolation increases.

The majority of CPACS data sets provides a sufficiently high resolution of point clouds to make the effects of the circumferential interpolation secondary. Instead, it is desirable to minimize the complexity of the surface, i.e. keep the number of control points and knots low, in order to not impede the performance of subsequent CAD operations. Thus, as stated in section 6.1.1.3, if profiles are provided with an equal and sufficiently high number of sample points, bivariate interpolation is selected by default in FUGA when building surface representations of CPACS bodies.

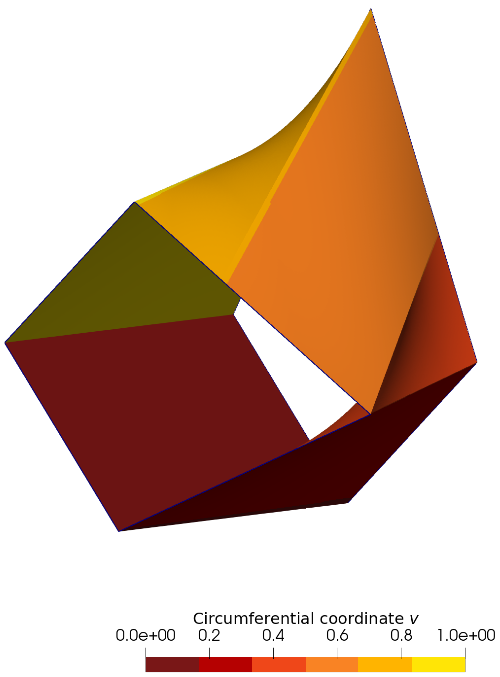


Figure A.2.3.: Skinned surfaces algorithm output

A.3. Design tables

Several of the design rules for the fuselage described in section 6.2 rely on data and requirements, which are determined by certification or operation aspects of the aircraft. The corresponding standards and limitations are stored in design tables, which can be referenced by the design rules. This section contains a collection of these design tables.

A.3.1. Certification data after CS-25 [EAS21]

In table A.3.1, the transport aircraft exit requirements given by CS 25.807 are listed. First, the door dimensions are given. The certification type determines the minimum width w_{door} and height h_{door} of the door. Furthermore, a maximum allowed corner radius r_{corner} is given. The passenger capacity $c_{PAX,exit}$ is listed for a pair of exits, where the smaller of the two is of the given type.

The certification specification also makes requirements w.r.t. spaces adjacent to the exits. To begin with, a passageway must lead to the exit from the nearest aisle. The minimum required width is given by $w_{passageway}$. For some types of exits assist spaces must furthermore be provisioned next to the passageways, to allow flight attendants to support passengers in an egress event. The required number of assist spaces is given by n_{assist}, whereas their width is determined by w_{assist}. If more than one aisle is present, the exit passageways must be connected via cross-aisles with a minimum width w_{xaisle}. For certain exits, it is required, for the cross-aisle to overlap completely with the adjacent passageway, whereas, for others, it is sufficient if the cross-aisle is placed in the vicinity of the passageway. Whether the latter is the case is indicated by a Boolean value in the “vicinity” column. For the rule implementation in FUGA, a more specific criterion is required than the ambiguous formulation of vicinity. Therefore, the maximum allowable longitudinal distance between the bounds an aisle and a passageway is set to $1m$.

Aside from the established exit types given in the certification specifications and acceptable means of compliance for large aeroplanes (CS-25), the exit types A+ and C+ are also included in table A.3.1. The plus indicates a proposed recertification of the corresponding CS-25-compliant exits for 10 additional passengers under the assumption that an additional flight attendant be present to assist during evacuation.

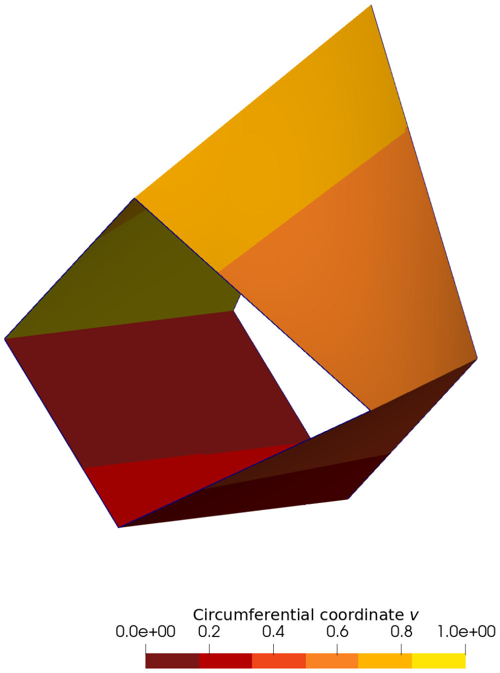


Figure A.2.4.: Bivariate interpolation output

Table A.3.1.: Exit requirements after CS 25.807

Type	$\frac{w_{door}}{in}$	$\frac{h_{door}}{in}$	$\frac{r_{corner}}{in}$	$c_{PAX,exit}$	$\frac{w_{passageway}}{in}$	n_{assist}	$\frac{w_{assist}}{in}$	$\frac{w_{xaisle}}{in}$	vicinity
I	24	48	8	45	20	1	12	20	True
II	20	44	7	40					
III		36		35	13	0			
IV	19	26	6.3	9					
A	42	72	6	110	36	2			False
A+				120					
B	32			75					
C	30	48	10	55	20	1			
C+				65	24				

Table A.3.2 lists the minimum aisle widths prescribed by CS 25.815. For a passenger number of up to 10, an even narrower aisle with a width no lower than $9in$ can be certified pending tests by the agency, as indicated by the asterisk in table A.3.2. However, since aircraft with less than 10PAX are not typically within the scope of FUGA, this exception has not yet been implemented.

Table A.3.2.: Aisle width requirements after CS 25.815

n_{PAX}	$\frac{w_{aisle,min}}{in}$	
	$z_{cab} < 25in$	$z_{cab} \geq 25in$
≤ 10	12*	15
11-19	12	20
> 20	15	20

A.3.2. Operational data

A list of cargo container types based on documents by airlines and OEMs [Boe12; Luf14; ANA22] is provided in table A.3.3. In general, the cargo containers are box-shaped. The outer dimensions are given by the height $h_{container}$, the depth $d_{container}$, and the overall width $w_{container}$. Furthermore, it is possible for a chamfer to be added in a 45 angle on the bottom at one (half) or two sides (full) in width direction. To determine the shape of the chamfer, the width of the container base w_{base} is provided as well as the width shape, which can be either of full or half. A visualization of the container types listed is given in figure A.3.1.

Table A.3.3.: List of cargo container types and dimensions (assembled from [Boe12; Luf14; ANA22])

Type	$\frac{V_{container}}{ft^3}$	$\frac{V_{usable}}{ft^3}$	$\frac{h_{container}}{in}$	$\frac{d_{container}}{in}$	$\frac{w_{base}}{in}$	$\frac{w_{container}}{in}$	Width shape	IATA
LD3-45	131	120	45	60.4	61.5	96	full	AKH
LD2	124		64		47	61.5	half	APE
LD3	159	120			61.5	79		AKE
LD1	175					92		AKC
LD4	195				96	96	full	AQP
LD8	245					125		AQF
LD11	256				125			ALP
PLA	250							PLA
LD6	316					160		ALF
LD26	470			88				AAF
LD7	495							P1P
P1P	379					125		P1P
LD9	381							AAP
LD29	510					186		AAU
LD39	560			96				AMU
P6P	407					125		P6P

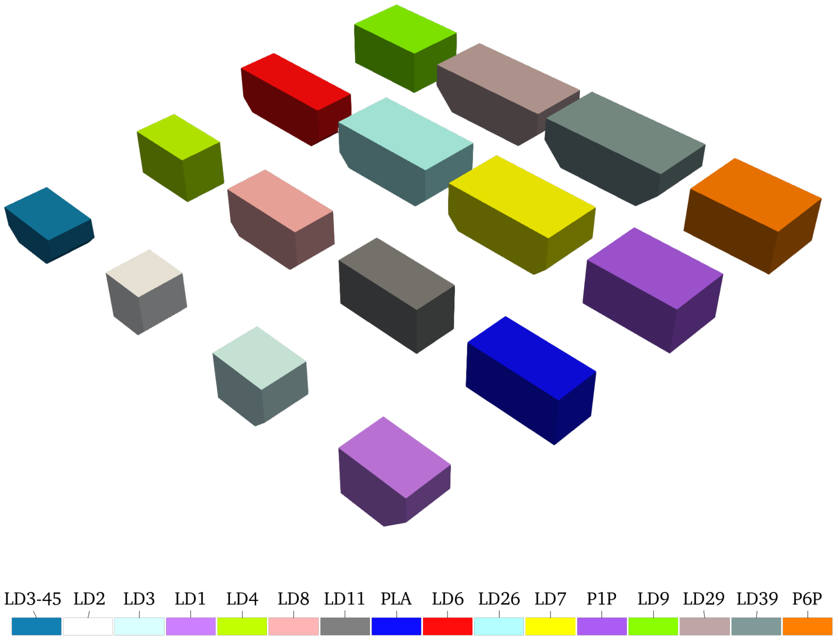


Figure A.3.1.: Visualization of cargo container types listed in table A.3.3