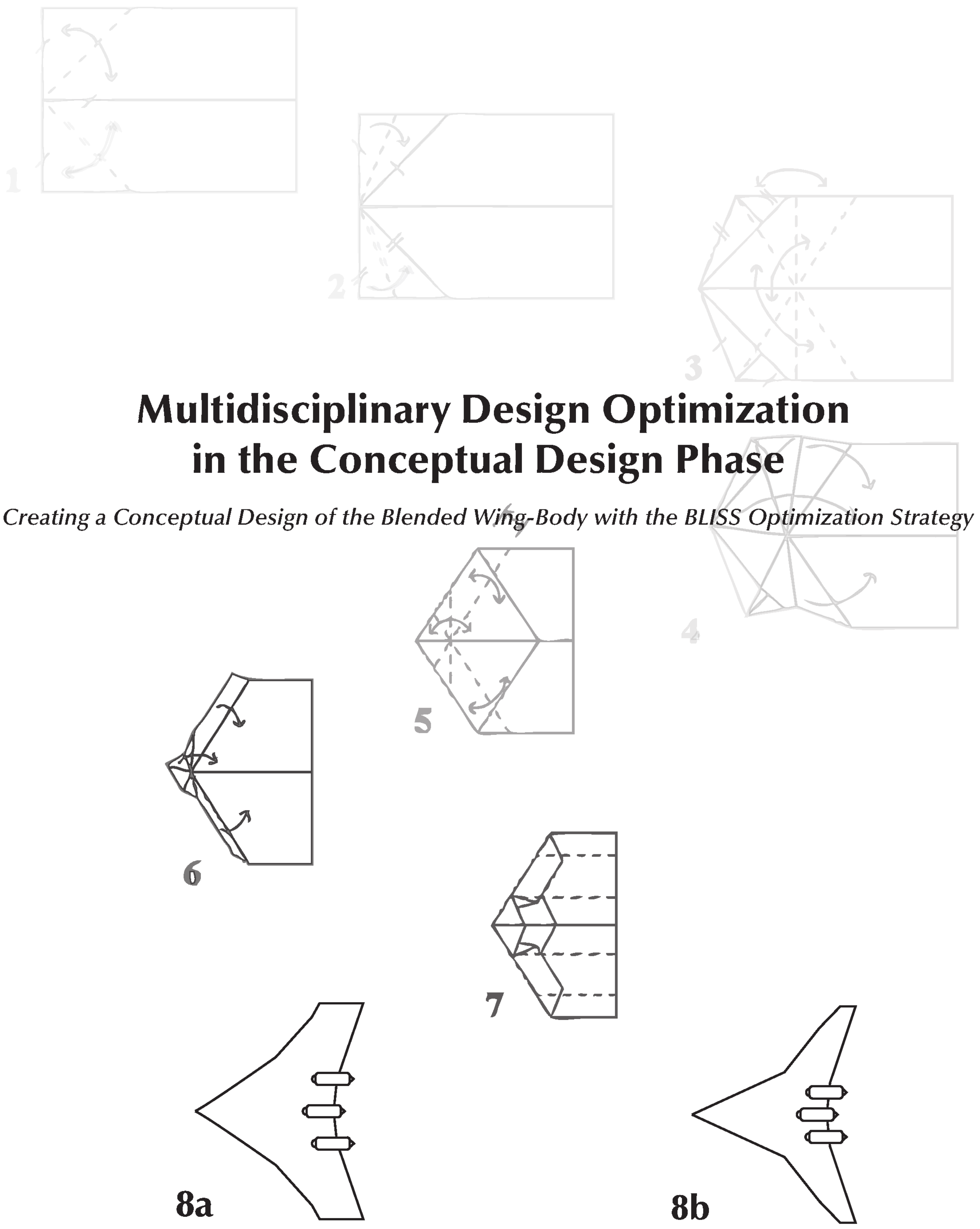
MASTER OF SCIENCE THESIS

Multidisciplinary Design Optimization in the Conceptual Design Phase

Creating a Conceptual Design of the Blended Wing-Body with the BLISS Optimization Strategy



T.J.M. Hendrich, BSc.

April 4, 2011

Delft University of Technology

Faculty of Aerospace Engineering

Challenge the future

Creating a Conceptual Design of the Blended Wing-Body with the BLISS Optimization Strategy

Multidisciplinary Design Optimization in the Conceptual Design Phase

Creating a Conceptual Design of the Blended Wing-Body with the BLISS Optimization Strategy

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

T.J.M. Hendrich, B.Sc.

April 4, 2011

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Systems Engineering and Aircraft Design

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled ``**Multidisciplinary Design Optimization in the Conceptual Design Phase**'' by **T.J.M. Hendrich, B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: April 4, 2011

Readers:

Prof. Dr. ir. drs. H. Bijl

Dr. ir. H.G. Visser

Dr. ir. G. La Rocca

ir. M. J. T. Schroijen

Abstract

Traditionally, the aircraft design process is divided into three phases: conceptual, preliminary and detailed design. In each subsequent phase, the fidelity of the analysis tools increases and more and more details of the design geometry are frozen. In each phase a number of design variants is generated, fully analyzing them with the tools available, and then doing trade studies between important design variables to finally choose the best variant. In the past, this approach has shown good results for 'Kansas city' type aircraft, which could be decomposed into different airframe parts with distinct functions, such as wings, tail, engines and fuselage. Each part needs to fullfill its own set of requirements and could be designed and optimized relatively independently from the others.

For the new generation of large transport aircraft, such as the Blended Wing Body (BWB), the traditional design approach is less suited. The Blended Wing-Body - studied by Boeing and many others as a future long-haul transport aircraft concept - is characterized by an integrated airframe, in which the aforementioned parts can no longer be clearly distinguished. The Blended Wing-Body features many and strong interactions between the various design disciplines and airframe subparts. Using the traditional design doctrine, these interactions greatly increase the required time to design. Over the past years,Multidisciplinary Design Optimization (MDO) is being considered as an alternative.

Nowadays, in industry the MDO approach is mainly used in the detail design phase and for isolated, well-defined design cases. The goal of this project is to create an MDO framework which can aid the designer in optimizing entire aircraft designs in the conceptual phase. This framework is shaped to the Bi-level Integrated System Synthesis (BLISS) strategy. This strategy splits the optimization into two levels: a disciplinary level, and a system one. Before optimization, BLISS performs a sensitivity analysis to obtain linearized global sensitivities of the design objective and constraints to each of the design variables. Validation is done using three cases: two sample problems from literature with known solutions, and the optimization of a simplified Boeing 747 wing for maximum aerodynamic efficiency using an aerodynamic and structural model. All three cases were optimized succesfully.

Finally, as a proof-of-concept for MDO, the framework is required to find an conceptual design of the Blended Wing-Body with minimum structural weight and minimum drag across a given mission. Meanwhile, structural, aerodynamic and performance constraints had to be satisfied. The problem features 5 disciplines, 93 constraints, 110 states and in total 92 design variables. Again, BLISS could converge to a solution, requiring 4 hours per cycle. By tuning the design variables, BLISS managed to converge to a final design in 22 cycles. The final design satisfies all constraints, except for the large local Mach number on the outboard wing. Similar problems were identified in several other Blended Wing-Body studies. The results support BLISS as a viable candidate method for introducing MDO in the conceptual design practice.

Preface

When I returned from my internship in Florence, the time had come for me to choose a graduation project. As a true aerospace fanatic, I had come to the faculty hoping to learn how to design an aircraft. Already when playing with LEGO, airplanes were all I would ever build. From that perspective, diving into aircraft design for my graduation seemed like a natural choice. Due to circumstances, my graduation period has also been a personal journey, and a long one at that. Still, I never ceased to be motivated for this project, because creating something that might actually fly is one of the most fun things to do.

Over the years I found out that besides aircraft design, studying at the aerospace faculty taught me so much more. I learned how to solve problems of many different natures, keeping a clear perspective on them and approaching them from different angles. I travelled to Russia and China, and learned italian in Florence. None of these things I would have learned however, without the people in aerospace engineering.

First of all I would like to thank my tutors, Marcel and Gianfranco. They were always available for support, tips and advice when I needed it. In particular, I would like to express the appreciation I have for the sessions with Marcel, where we discussed the mathematics behind it all quite extensively. The same holds for Gianfranco, who helped me find good weight estimation methods and took time to read my report even when his own PhD defense was imminent.

Many thanks to all of my roommates of 2.40, with whom I could always relax if Matlab was being stubborn, but also at times when it wasn't. Thanks Tom, Michael, Mark (all three), Bjorn, Jan, Roel, Harm, Karl, Fred, Ricardo and all the others!

My deepest gratitude to my parents and sister, who have supported me throughout my whole graduation, studies in Delft, and my life. Finally, thanks to a special person: my girlfriend Sanneke van der Ouw.

Tijl Hendrich
Delft, april 2011

Contents

Abstract iv

Preface v

1 Background 1

1-1 Traditional Design Approach . . . 1

1-2 Emerging Design Methods . . . 3

2 Thesis Objective & Approach 5

2-1 Thesis Objective . . . 5

2-2 Research plan . . . 6

3 The Blended Wing Body 8

3-1 Aircraft Concept . . . 8

3-2 Design Studies . . . 10

3-2-1 Boeing MDO study . . . 10

3-2-2 European MOB Project . . . 11

3-3 Design Issues . . . 12

3-3-1 Aerodynamics . . . 13

3-3-2 Structures . . . 13

3-3-3 Control . . . 14

3-3-4 Passenger comfort . . . 15

4 MDO: a short introduction 16

4-1 Basic Concepts . . . 16

4-2 BLISS Classification . . . 19

4-3 Methods of Design Analysis . . . 20

4-4 Single- and Multilevel MDO Strategies . . . 20

4-5 Gradient- and Non-gradient Based Optimization . . . 21

5 BLISS Framework **23**
5-1 BLISS Theory 23
5-1-1 BlackBox Sensitivity Analysis 24
5-1-2 System Sensitivity Analysis 25
5-1-3 Move Limits in X and Z 26
5-1-4 BlackBox Optimization 27
5-1-5 System Optimization 28
5-1-6 Update Design Vector 29
5-2 Framework Capabilities 29
5-2-1 Genericity and Problem Definition 32
5-2-2 Design Problem Modification 32
5-2-3 Results Postprocessing 32
5-3 Framework Implementation 32
5-4 Initialization 35
5-5 System Analysis 37
5-6 Sensitivity analysis 37
5-7 Optimizers 38

6 Validation Cases **41**
6-1 Validation Approach 41
6-2 Analytical Sample Problem 43
6-2-1 Problem Structures 44
6-2-2 Results 46
6-3 The NASA Business Jet 50
6-3-1 Problem Structure 50
6-3-2 Results 51
6-4 B747 Wing Model 53
6-4-1 Problem Structure 53
6-4-2 Results 55

7 Design Problem Formulation and Integration **61**
7-1 Formulation Process 61
7-2 Requirements Identification 63
7-3 Functions analysis 64
7-3-1 Aircraft Functions 64
7-3-2 Discipline Model Roles 66
7-4 Discipline Model Selection 68

7-4-1 Aerodynamics . . . 68
7-4-2 Structures . . . 69
7-4-3 Propulsion . . . 70
7-4-4 Aircraft Performance . . . 71
7-4-5 Weight Estimation . . . 71
7-5 Constraints Definition . . . 72
7-6 Aircraft Parametrization . . . 72
7-6-1 Aerodynamics . . . 75
7-6-2 Structures . . . 76
7-6-3 Propulsion . . . 78
7-7 BLISS Integration . . . 79
7-7-1 BlackBox Decomposition . . . 79
7-7-2 Design Variables and States . . . 81
7-7-3 Constraint Allocation . . . 84
7-8 Validation . . . 85

8 Blended Wing-Body Optimization Results 87
8-1 Preliminary BLISS Run . . . 87
8-2 Final BLISS Run . . . 92
8-3 Framework Utilization . . . 97

9 Conclusions 100

10 Recommendations 102

Bibliography 104

A SQP Algorithm 107

B Discipline Model Theory 109
B-1 Aerodynamics . . . 109
B-1-1 Tornado . . . 109
B-1-2 Xfoil . . . 111
B-1-3 Trim & Balance . . . 112
B-2 Structures . . . 113
B-2-1 Structural Strength . . . 113
B-2-2 Structural Weight . . . 115
B-2-3 Loads & Center of Gravity . . . 116
B-3 Propulsion . . . 117
B-3-1 Gasturbine Simulation Tool . . . 117
B-3-2 Airfield performance . . . 118
B-4 Weight . . . 119

C Discipline Model Flowcharts 122

C-1 Aerodynamics 122

C-2 Structures 123

C-3 Propulsion 124

C-4 Weight 125

D BLISS FrameWork Manual 126

D-1 BLISS Main Menu 126

D-2 Starting BLISS 127

D-3 Defining a New Problem 127

D-4 Generic Problem Initialization File 128

D-5 Running BLISS 131

D-6 Problem Structure Modification 132

D-7 Manually Increasing Problem Feasibility 132

D-8 Viewing Results 133

E BLISS Problems Checklist 134

F BWB Design and State Variables 135

G BWB Constraints 137

H 747 Design and State Variables 138

I 747 Constraints 140

Nomenclature

Roman Symbols

c	Chord	[m]
h	Altitude	[m]
R	Range	[m]
c	Chord	[m]
C_D	Drag Coefficient	[-]
C_L	Lift Coefficient	[-]
C_t	Thrust Coefficient	[-]
D	Drag	[N]
G	Vector of Constraints	[-]
h	Altitude	[m]
L	Lift	[N]
R	Range	[m]
S	Surface Area	$[m^2]$
T	Thrust	[N]
t	Thickness	[m]
V	Airspeed	[m/s]
W	Weight	[N]
X	Vector of Local Design Variables	[-]
Y	Vector of State Variables	[-]
Z	Vector of Global Design Variables	[-]

Greek Symbols

γ	Flight Path Angle	[deg]
α	Angle-of-attack	[deg]
δ	Control Surface Deflection	[deg]
Γ	Dihedral	[deg]
λ	Taper Ratio, Lagrange Multiplier	[-]
Λ	Sweep angle	[deg]
Φ	System Objective	n.a.
ρ	Air Density	[kg/m^3]

Sub/Superscripts

$_s$	Structures

tr	Transition
eng	Engines
cr	Cruise
$*$	Optimum
0	Initial
r	Belonging to BlackBox *r*
f	Fuel
p	Payload
$content$	Payload/Fuel
$0.25c$	Quarter chord
r,s	Quantity passed from BlackBox *s* to BlackBox *r*

Abbreviations

BB	BlackBox
$BBOpt$	BlackBox Optimizer
$BBSA$	BlackBox Sensitivity Analysis
$BLISS$	Bi-Level Integrated System Synthesis
BWB	Blended Wing Body
CDE	Computational Design Engine
CFD	Comptational Fliud Dynamics
DEE	Design Engineering Engine
DSM	Design Structure Matrix
FEM	Finite Element Method
FMM	Flight Mechanics Model
GST	Gasturbine Simulation Tool
HLP	High-Level Primitive
KBE	Knowledge-Based Engineering
LE	Leading Edge
LEF	Leading Edge Fraction
MDA	Multidisciplinary Design Analysis
MDO	MultiDisciplinary Design Optimization
MLW	Maximum Landing Weight
MMG	Multi-Model Generator
MOB	Multidisciplinary Optimization of the Blended Wing Body
$MTOGW$	Maximum Take-Off Gross Weight
$MZFW$	Maximum Zero-FuelWeight
$NAND$	Nested Analysis and Design
NLR	Dutch Aerospace Laboratory
OEI	One Engine Inoperative
OEW	Operational Empty Weight
SA	System Analysis
$SAND$	Simultaneous Analysis and Design
SE	Systems Engineering
SFC	Specific Fuel Consumption
SQP	Sequential Quadratic Programming
SSA	System Sensitivity Analysis
$SysOpt$	System Optimizer
TE	Trailing Edge
TEF	Trailing Edge Fraction
VLM	Vortex-Lattice Method

Chapter 1
Background

Designing an aircraft is a complex and challenging task, which can take up to 10 years for a long haul airliner design. It involves thousands of design specialists organised in teams, each team with a different task and field of expertise. The demands put on aviation by society in terms of safety and environmental impact and by airliners in terms of performance are extremely high. These demands will be ever increasing for the future to come (*ICAO: Act Global. Uniting aviation on climate change* (2011)). Without doubt, these increasing demands will have their impact on aircraft design and design methods. Section 1-1 starts of by outlining the traditional aircraft design approach, section 1-2 continues by looking to alternatives.

1-1 Traditional Design Approach

Traditionally, the aircraft design process can be divided into three phases (Torenbeek (1982)): conceptual design, preliminary design and detail design. The conceptual phase is started with the identification of a complete set of requirements that the design should fulfill. The goal of this phase is to decide on the most suitable configuration for the design and the initial sizing of its basic geometry. According to Hamann & Tooren (2004), the traditional design process is given by the basic design cycle (see Figure 1-1).

The basic design cycle is applicable to all three aforementioned design phases, where the detail level of the concepts and the accuracy of analysis tools increases with each phase. As the traditional design process progresses, larger portions of the design are frozen, they may not be changed anymore from a certain point onwards.The conceptual design phase uses low fidelity tools to size the geometry of the concepts and analyze their (physical) behaviour. If design experience is present for the concept under consideration, it can be sized using empirical sizing rules and handbook methods based on previous designs.

After concluding the conceptual design phase, the design configuration is frozen in the preliminary design phase, which means that there will be no major changes in the design layout from that point. This phase seeks to increase the level of detail in the design and to model its physical behaviour and cost estimation as accurately as possible. At the end of this phase, the company evaluates design performance and feasibility, and decides whether or not to continue the design program and create a detail design.

In the detail design phase, the aircraft design is extended to such a level, that it can be manufactured and sold. This involves creating a CAD model of the whole aircraft and of each part. Only small changes in geometry are allowed in this phase, since weight, cost and performance targets set in

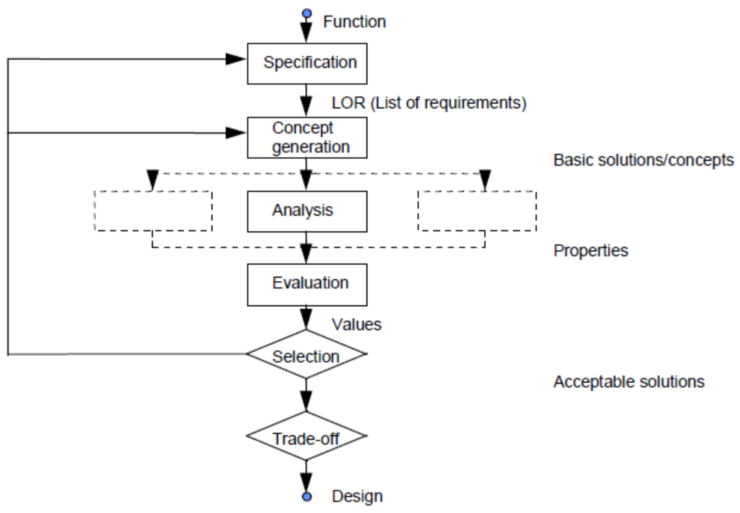


Figure 1-1: The basic design cycle

the previous phases should still be met. The detailed geometry on the one hand and the required modeling accuracy on the other demand the use of high-fidelity analysis tools, which can have runtimes in the order of days or even weeks.

In the traditional design process, trade studies are conducted between pairs of the most general design variables (e.g. thrust-to-weight ratio vs wing loading), meanwhile accounting for constraints. The so-called carpet plot (see figure 1-2) is one of the methods to do this. If a trade study changes a design variable, this will influence the properties of the design, hence requiring change of other design variables as well. Thus, trade studies have to be performed between many pairs of design variables, which means that their character is highly iterative. In subsequent phases, trade studies are needed as well, but then they require more time due to longer computation times. This limits the number of design variants that can be explored within a reasonable timeframe.

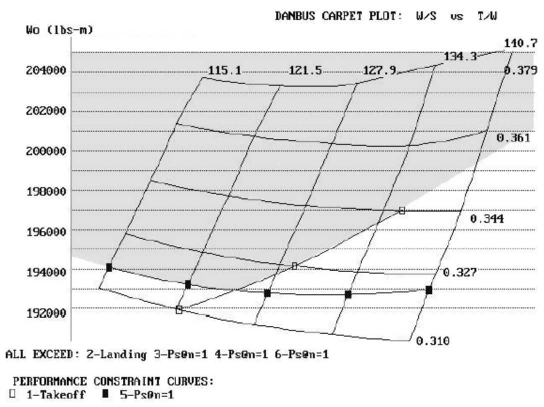


Figure 1-2: W/S versus T/W carpet plot

The procedure for designing aircraft has not changed fundamentally since the introduction of electronic computing. Computers made it possible to automate the analyses needed for aircraft design, such as Computational Fluid Dynamics and Finite Element Modeling. Also, they allowed the use of more elaborate calculation schemes, because of the increased amount of computing power. This gave a better insight into what the consequences of a particular design choice or configuration would be. At the beginning of the computer age, the design world predicted a significant decrease of both the time to come to a final design and the design costs. However, it turned out that a major part of the time reduction gained by speeding up calculation processes, was lost by the need to manage all the different analysis tools (see M. V. Tooren et al. (2005)). In addition, complexity of aircraft designs is ever increasing, requiring more and more complex and time-consuming analysis models to be able to still achieve performance gains. The increased information generated during the design process also led to an equal increase in information requirements by the airworthiness authorities.

1-2 Emerging Design Methods

A different approach to complex design problems is MDO: Multidisciplinary Design Optimization. This method for optimizing a design aims to decompose the ever growing design problem into smaller, more manageable parts, the design disciplines. In turn, decomposition requires the identification of the couplings between disciplines, which also exist in real life. By only conducting trade studies between 2 or 3 variables, these interdisciplinary couplings are not accounted for, and possible design benefits from them cannot be indentified (Weck et al. (2007)).

MDO calls for techniques that can automate the repetitive and routine tasks in the design process. These include the creation of input files for the analysis tools for a given geometry, implementation of desired geometry changes and passing data from one analysis model to the other. One such technique is Knowledge Based Engineering (KBE). KBE is aimed at reducing the overall time to design, increasing the time that engineers can spend on the creative side of design work, and increasing the number of design iterations that can be done in a given time (Rocca & Tooren (2009)).

Model and output generation are automated, and instead of losing time on routine tasks, engineers can focus on what will happen is a certain change in the design is implemented. Within the Delft University of Technology (DUT) aerospace faculty, focus is on developing a so-called Design Engineering Engine (DEE), which is a framework that supports designing using KBE. A DEE for fixed wing design called 'DARwing' was developed recently by Koning (2010). A DEE consists of three parts (see Figure 1-3 , adapted from Schut & Tooren (2008)):

1. Initiator
2. Multi-Model Generator (MMG)
3. Optimizer

The design process is started by a first guess found in the initiator. This first guess is used as an initial point for the MMG. The MMG is capable of creating a product model for a wide variety of designs by using the concept of High-Level Primitives (HLP's). These HLP's are the 'building blocks' of the design, which go beyond primitives used in modern day CAD programs(lines, curves, surfaces). Each HLP captures a specific set of design knowledge. Examples of HLP's are wing trunks, ailerons and wing trunk connection elements.

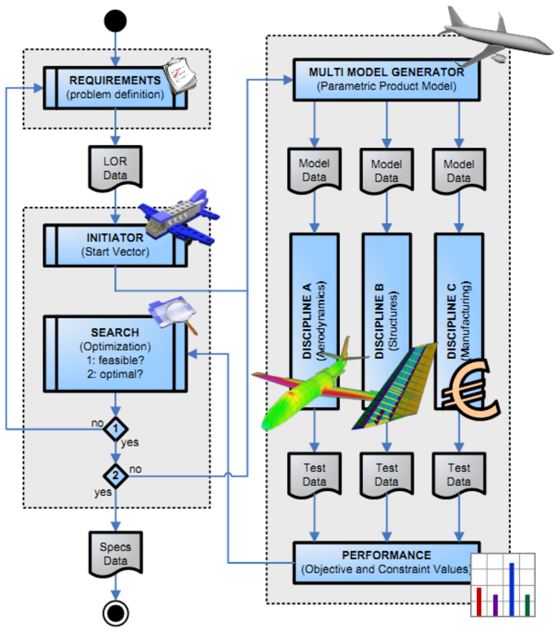


Figure 1-3: Design Engineering Engine schematic

After creating the product model, the MMG features a set of 'capability modules'. Capability modules generate input files for a range of analysis models, allowing these to be run. The MMG then gathers results from all analysis models and computes design objective and constraint values. The optimizer checks requirements compliance and convergence of the design. If the design has converged to an optimum one but does not comply to all requirements, an entirely new design is initialized by the initiator and the whole process is repeated.

From Weck et al. (2007) it is clear that the concept of MDO is becoming increasingly popular in industry. At the same time, MDO is exclusively used in detail design. There, it can solve isolated problems that consist of only a small part of the airframe, or even on individual parts. In this type of problem, the problem size and properties are well known and controllable. In the above discussion, MDO is pinpointed as a strategy which has great potential in conceptual design as well. In that phase, MDO could be used to:

- Identify and exploit performance benefits in conventional aircraft design, which would not have been found with traditional trade-offs
- Be able to design and evaluate novel, highly integrated aircraft concepts
- Be able to design aircraft incorporating a larger number of design variables and disciplines already in the conceptual phase

When these notions are related to the DEE, an MDO conceptual design tool would fulfill the role of the initiator.

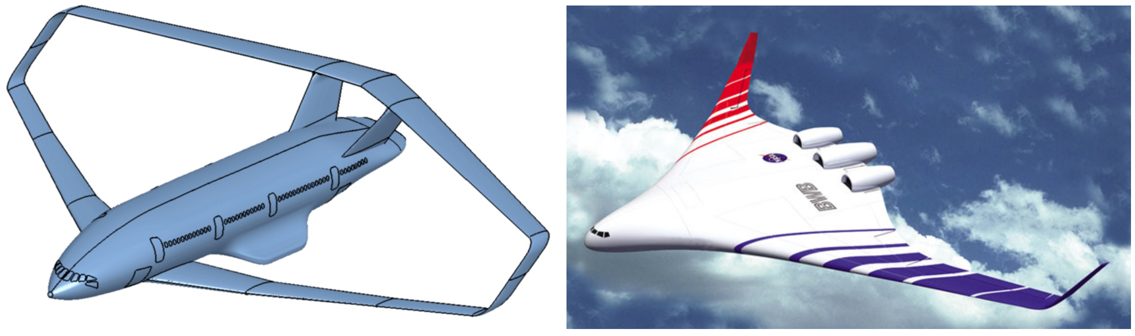
Chapter 2

Thesis Objective & Approach

Chapter 1 showed that the increasing focus towards designing novel aircraft configurations also asks for investigation of new aircraft design methodologies, MDO in particular. In the coming chapter, section 2-1 will define the thesis goal and sub-goals. Then, section 2-2 outlines the approach that was envisioned to achieve them.

2-1 Thesis Objective

The use of MDO for designing an aircraft instead of the traditional process becomes more and more beneficial as the aircraft concept itself has a more integrated nature. In this perspective, integrated means that many couplings exist between the shapes and/or functions of the different aircraft components, so that there is potential for synergy between them. Currently, the faculty of aerospace engineering is doing research on two revolutionary aircraft concepts: the Prantlplane and the Blended Wing-Body (BWB). These aircraft are shown in figures 2-1a and 2-1b respectively



(a) PrandtlPlane **(b)** Blended Wing-Body

Figure 2-1: Examples of novel aircraft configurations

The Prandtlplane still features a conventional fuselage (Frediani et al. (2003)), but the conventional wings have been replaced by box wings according to Ludwig Prantdl's Best-Wing system for minimum induced drag. The BWB is a flying wing with an adapted center section as its fuselage. Since a large amount of model and design knowledge exists on these aircraft within the faculty they are the two prime candidates for conducting MDO. Because of the apparent higher level of component integration in the Blended Wing-Body , this aircraft configuration is selected as the design case for current research. Contrary to the present design practice of limiting MDO to detail design, this research aims to use MDO in the conceptual phase, which means that the models describing the BWB will be of low-fidelity. Over the last thirty years, various MDO methods have been defined and researched. After a short, literature based trade-off, 'Bi-Level Integrated System Synthesis' or 'BLISS' was selected as the MDO method of choice.

Taking all of these aspects into account, the following research goal is formulated:

> *"The creation of an accurate, flexible and adaptable Multidisciplinary Design Optimization tool using the Bi-Level Integrated System Synthesis strategy. The tool is able to optimize a conceptual design of the Blended Wing Body"*

The main goal is accompanied by the following sub-goals:

- Prove that BLISS is able to solve a conceptual design problem of a larger size than common sample problems given in literature in a few hours of computation time with a standard PC
- Show that MDO is capable of producing plausible Blended Wing-Body conceptual design results, which reproduce known Blended Wing-Body design trends and issues

Up to now, a literature survey indicated that the BLISS method was tested on optimization problems of 10-15 independent variables. The size of theBlended Wing-Body design problem in this thesis will be considerably larger. In the context of these research goal, the following meaning is given to the words *accurate*, *flexible* and *adaptable*:

- **Accurate**: the tool is able to give a reasonably accurate prediction of aircraft performance, in particular of the design objectives, resulting in realistic values for the principal design variables
- **Flexible**: the tool is able to handle a wide variety of designs and situations (in terms of constraints, mission, aircraft geometry, objective functions) suchthat a major portion of the conceptual design space can be explored
- **Adaptable**: it should be easy to make adjustments and modifications to the tool (replacing discipline models by lower or higher fidelity versions, adding or removing constraints)

The Matlab language is chosen to program the tool, because the author is already experienced in its use and it already has an extensive optimization toolbox. This project will focus on the MDO methodology and structure in aircraft designing. This means that less focus is put on the identification of requirements and the design option analysis, traditional first steps in the systems engineering process. It is assumed that customer requirements (e.g. mission requirements) are already given, and that requirements derived from them (e.g. noise, stall and c.g. limits) are similar to those of conventional long-haul transports. In a DEE perspective, the tool could work as an initiator, which provides a reasonable 'first guess' of the BWB, which can then be used as a starting point for more detailed analysis by the DEE (see Figure 1-3). Alternatively, stand-alone usage of the design tool should also be possible.

2-2 Research plan

Several components can be identified that are needed to realize the goals and subgoals in section 2-1. Clearly, the two main components are:

- The Blended Wing-Body design problem, which consists of requirements, design parametrization, disciplinary analysis models and the couplings between them.

- The BLISS framework able to solve generic MDO problems

It is decided to build these two components separately, so that both of them can be validated with an alternative Blended Wing-Body design and a known BLISS sample problem, respectively. After building and testing aforementioned components, the Blended Wing-Body design problem can be inserted into the BLISS framework for optimization. Probably, doing this will require a great deal of finetuning, on optimization parameters (methods, tolerances), formulation of the constraints and the problem structure itself.

To be able to handle these possible variety in formulation of the optimization problem, and allow for future use in other MDO problems, the BLISS framework is made as generic as possible. The Blended Wing-Body design problem is formulated according to systems engineering principles, assuming that a design configuration is already chosen and that requirements are given and fixed. Figure 2-2 presents an overview of the steps that were taken to try to realize the research goals.

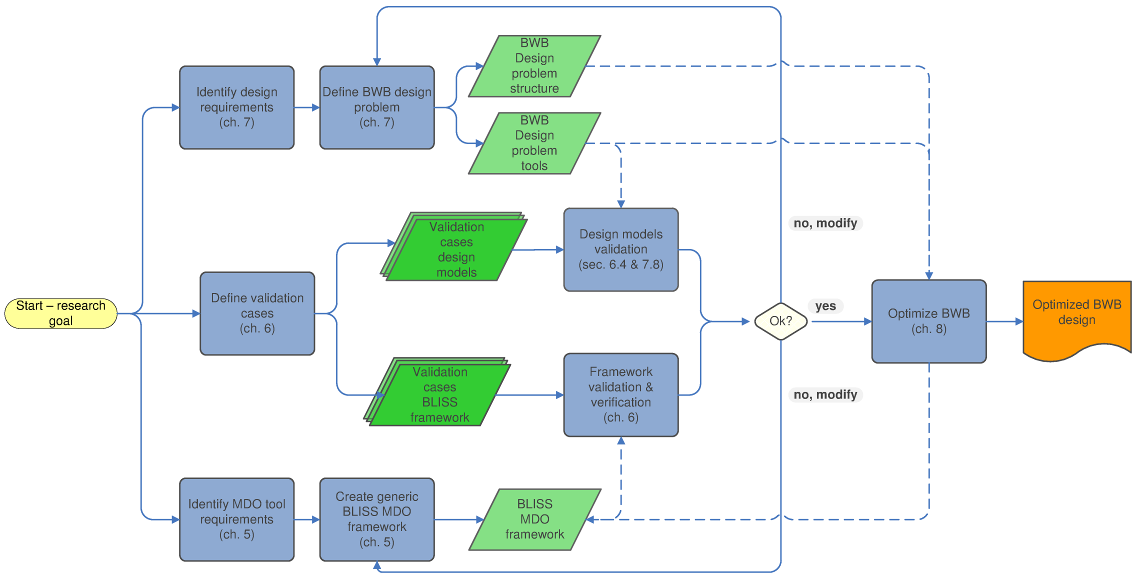

Start – research goal
Identify design requirements (ch. 7)
Define BWB design problem (ch. 7)
BWB Design problem structure
BWB Design problem tools
Define validation cases (ch. 6)
Validation cases design models
Design models validation (sec. 6.4 & 7.8)
Validation cases BLISS framework
Framework validation & verification (ch. 6)
Identify MDO tool requirements (ch. 5)
Create generic BLISS MDO framework (ch. 5)
BLISS MDO framework
no, modify
Ok?
yes
no, modify
Optimize BWB (ch. 8)
Optimized BWB design

Figure 2-2: Thesis research plan

The Blended Wing-Body design problem is discussed in chapter 3 with its advantages and shortcomings. Chapter 4 describes some concepts in MDO, that are fundamental to solving an optimization problem using MDO strategies, while chapter 5 elaborates the creation of the BLISS framework. The framework is tested in chapter 6 using 3 validation cases. Then, chapter 7 aims to devise a method to formulate a MDO problem for the BWB in a consistent way, dealing with structuring and parametrizing the BWB design and identifying design constraints. Subsequently, chapter 8 shows the road to the optimized Blended Wing-Body design that was obtained using BLISS. Based on the results, this chapter aims to provide the main guidelines for using BLISS and correctly setting up design problems for it. Finally, conclusions and recommendations are presented in chapter 10.

Chapter 3

The Blended Wing Body

This chapter will discuss the aircraft of interest for this study, the Blended Wing-Body into more detail. First of all, section 3-1 discusses motivation for conceiving the BWB concept, and the reasons why it might be a feasible replacement for the conventional long-haul transport. Then, section 3-2 treats four design studies conducted on the BWB. Finally, section 3-3 briefly touches upon the design issues and problems found in these studies.

3-1 Aircraft Concept

At the end of the 80's, engineers at the NASA Langley Flight Research Center realized that in the first 43 years of manned, powered flight, there had been great innovations both in terms of aircraft configuration (e.g. using swept wings) and technology (Liebeck (2004)).These efforts culminated in the B-47 in 1947, an large bomber from which all long-haul transports in service today are essentially derived (see E. Obert & R. Slingerland (2007)).

The direct incentive for starting the Boeing and NASA research was an overview of the aerodynamic efficiency of long-haul transports from the 60's up to the 80's. This overview, given in Figure 3-1), shows that the aerodynamic efficiency[1] of long-haul transport aircraft has not been increasing in the second half of the 20^{th} century.

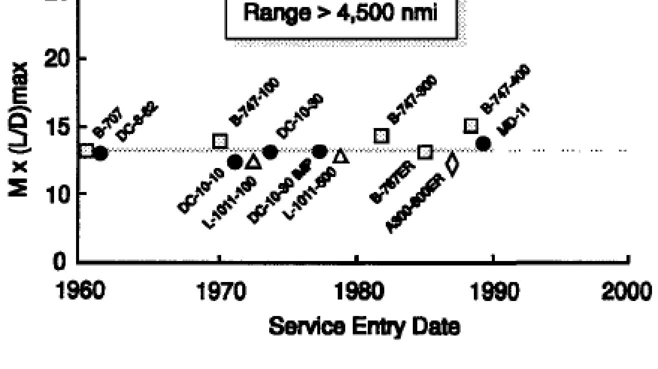


Figure 3-1: Trend in large transport aerodynamic efficiency through 40 years

[1]Aerodynamic efficiency is defined as the product of cruise Mach number and lift-to-drag ratio L/D.

Since the 1940s, technological innovations such as the use of composites, turbofan instead of turbojet engines and complicated airfoil shapes, have enabled a continuous improvement in aircraft performance. The configuration of long-haul transports however has not changed ever since. Even the newest aircraft entering service, such as the Boeing 787 Dreamliner feature the same basic configuration ('tube-with-wings') as the B-47. This basic configuration is also nicknamed the 'Kansas city aircraft'. In the 'Kansas city aircraft', the functions of each component are coupled quite loosely (Wakayama & Kroo (1998)). Simply put: the wing provides lift and keeps the aircraft flying, the tail surfaces provide stability and control, the engines propel it, while the fuselage carries the payload and serves as a main body to which all other elements connect.

The fundamental idea of the Blended Wing-Body concept is to integrate all of the components mentioned previously into one design. Instead of a cylindrical fuselage, the Blended Wing-Body has a thick wing with a large chord as its central body. This centerbody contains the payload, but it also generates lift, is partially responsible for the pitch control of the aircraft and acts as an inlet to the engines ((Liebeck (2004)). The outboard wing, which is considerably thinner, has a smooth and continuous (blended) connection with the centerbody. It provides lift, pitch and roll control. The shape with minimum surface area for a given volume is a sphere, which is however not suitable for using as aircraft fuselage. (Liebeck (2004)) argues that shaping this sphere into a streamlined disk instead of the conventional tube, there exists a overlap between wing and fuselage. This blended layout can yield a wetted area decrease up to 33% w.r.t. the conventional configuration.

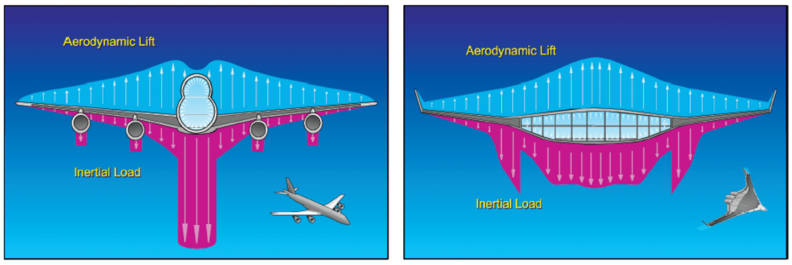


Figure 3-2: Spanwise division of loads, conventional and BWB

Besides the lower wetted area, the Blended Wing-Body concept may posses numerous other advantages with respect to Kansas City aircraft:

- Decreased **wave drag** due to more favourable area-ruling (Roman et al. (2003)), the Blended Wing-Body cross-sectional area distribution is closer to the Sears-Haack distribution for minimum wave drag (see Figure 3-3).
- **Lower wing loading**: Ruijgrok (1996) gives some relations, which show that a low wing loading beneficially affects stall speed, design lift coefficient, minimum turn radius and airfield performance due to a reduced wing loading. A low wing loading also eliminates the need for TE flaps[2].
- Reduced structural **bending moments** and shear loads due to more even spreading and a more continuous distribution of aerodynamic and inertial loads. In contrast, in conventional aircraft a peak bending moment occurs at the wing root (see Figure 3-2)
- Engines are located above the centerbody, which protects them from ingestion of runway debris. In turn, the centerbody reflects fan noise upwards,yielding lower fly-over **noise levels**. More advanced engine integration (so-called Boundary-Layer Ingestion) may induce additional performance gains, but this technology still has many unknowns.
- Eliminating the need for wing-fuselage and fuselage-tail connections may result in a reduction of up to 30% in the total **number of parts** used according to Liebeck (2004).

[2]Wing loading is widely considered a principal design variable, in fact in (Raymer (2002)) it's called one of the 'basic five' variables in aircraft design.

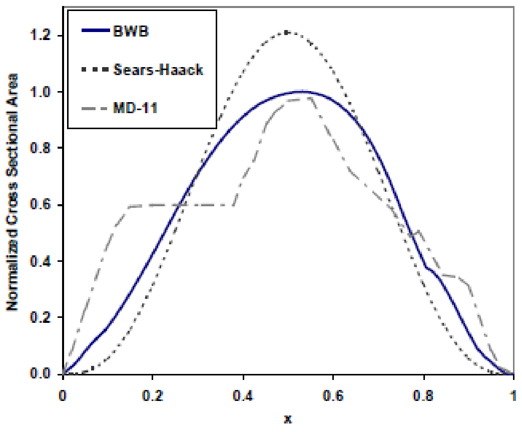


Figure 3-3: Blended Wing-Body and MD-11 area distributions compared to Sears-Haack

The NASA and Boeing first generation Blended Wing-Body design yielded the following significant performance improvements w.r.t. a conventional baseline aircraft, designed for the same mission and with the same technology level (Liebeck et al. (1998) and Liebeck (2004)):

- 27% lower fuel burn
- 15% lower take-off weight
- 12% lower Operational Empty Weight (OEW)
- 27% lower total thrust
- 20% higher lift-to-drag ratio

Summarizing, different studies show that the Blended Wing-Body could be a viable alternative to the convential long haul airliner. However, the difficulty of designing an aircraft with strong interdisciplinary couplings is hard to overcome with the traditional design approach.

3-2 Design Studies

After invention of the concept, several (optimization) studies were conducted to find the most suitable Blended Wing-Body configuration and shape and predict its performance. To see how these studies were conducted, what models were used in the analysis and for identifying possible pitfalls in Blended Wing-Body design, two of these studies will be briefly discussed here:

1. An MDO study by Boeing (see Wakayama & Kroo (1998))
2. An MDO study on designing a Blended Wing-Body within a large European design framework by various parties, the MOB project (see Laban et al. (2002) and Morris et al. (2004))

3-2-1 Boeing MDO study

During the BWB study described in (Liebeck et al. (1998)), the engineers discovered that the traditional aircraft design approach was not able to handle the design of the BWB very well, due to its integrated and highly coupled nature. This makes the BWB a highly suitable candidate to be subjected to MDO. The MDO study was aimed at finding an optimized wing planform shape, along with the shape of the pressure cabin, meanwhile accounting for payload and fuel volume, static

Discipline	Model
Aerodynamics	Vortex-Lattice method
Structures	Aircraft structure is modeled as a singe, hollow beam (monocoque beam analysis)
Weight & c.o.g.	Based on fuel, payload, structural and systems masses and relative positions
Stability & Control	20 flight conditions identified throughout mission, addressing low-speed aerodynamic behaviour and stability

Table 3-1: WingMOD models

margin, maximum stresses, weights and c.g. limits. The study used a Boeing in-house developed code for MDA on conventional aircraft wings called WingMOD, using the models and disciplines given in table 3-1.

The optimization resulted in a 7% take-off weight reduction and a 7.5% L/D increase with respect to the initial values. Wakayama & Kroo (1998) identify a number of problems and deficiencies with the WingMOD framework:

- No mission analyzer was incorporated
- A lack of intermediate fidelity tools, while low fidelity (lofi) tools cannot reveal essential benefits (such as 3D relief effects) as well as limits (such as flow separation) of the concept. High fidelity (hifi) tools (such as CFD) are computationally too expensive to include

In total, the set-up explained above yielded a design problem with 44 unconstrained design variables, which according to Wakayama & Kroo (1998) is impossible to solve using the conventional design approach and requires MDO.

3-2-2 European MOB Project

From Morris et al. (2004), the goal of the MOB (Multidisciplinary Optimisation of a Blended Wing Body) project was:

> "The development of tools and working methods to facilitate the multidisciplinary design of large scale and complex aeronautical products by distributed teams, employing different design approaches and a variety of discipline-based programs, employing either commercial off the shelf (COTS) products or proprietary codes."

To facilitate MDO within the MOB , a so-called Computational Design Engine (CDE) was built by the MOB consortium. The CDE covered four disciplines:

- Aerodynamics and trim
- Structures
- Weight and balance
- Flight mechanics

After the multidisciplinary optimization, the changes in the overall design vector are captured and fed back to the product model (Rocca et al. (2002)), which creates a new geometric model and its representations for each respective discipline based on the new design vector. The Breguet range equation (see 3-1) was used as a system objective, with the MTOGW assumed constant.

$$R = V_{cr} C_t \frac{L}{D}_{cr} ln \frac{W_{MTOW}}{W_{MTOW} - W_{fuel}} \tag{3-1}$$

Each discipline optimizes its respective contribution to equation 3-1. To do this, a 'driving scenario' was calculated by the weight & balance model for each of the other modules. Table 3-2 summarizes their objectives and driving scenarios.

Table 3-2: CDE modules and scenarios

CDE module	Module objective	Scenario
Aerodynamics & trim	Maximum lift-to-drag ratio	transonic cruise flight condition at Mach 0.85 at 35000 feet altitude in standard atmosphere conditions, maximum payload, half the trip fuel available in the wing tanks, empty trim tanks
Structures	Minimum structural weight	+2.5G pull-up manoeuvre at sea-level altitude and Mach 0.50, maximum payload, and fuel, empty trim tanks
Weight & balance	Finding critical loading scenarios for the other modules by assessing weight and cg envelopes	(none)
Flight mechanics	Satisfactory stability, control and handling qualities	low-speed approach flight scenario, cruise trimmability

Lofi modeling is done by modeling the structure as a hollow bending beam, with all loads being transferred through the skin. The CDE uses FEM to do the hifi structural computations. Aeroelastic effects were also considered in this phase. A extra scaling factor of 1.5 was used in the hifi results, for not including manufacturing constraints and structural details (Laban et al. (2002)). The weight and balance module calculates aircraft weight, along with c.g. position and moments of inertia for the aerodynamics and flight mechanics modules.

In conclusion, the MOB project features a complete and detailed design and geometry generation capability within a distributed framework. However, the broadness of the actual BWB design case is limited: few scenarios are taken into account in each discipline, and each discipline module optimizes a component of the Breguet equation. This means that the way in which each discipline module contributes to minimizing the global objective is fixed beforehand and benefits from interdisciplinary trade-offs cannot be identified.

3-3 Design Issues

Although a BWB configuration has many advantages above a conventional aircraft, a number of design challenges specific to the Blended Wing-Body was identified from literature. These challenges are divided in four groups:

- Aerodynamics
- Structures
- Control
- Passenger comfort

Each of these challenges is discussed in the following subsections.

3-3-1 Aerodynamics

To carry passengers and payload, the BWB centerbody airfoil must be thicker than any conventional wing designed for high subsonic speeds. Typical values indicate a maximum t/c of around 17-20% (Liebeck et al. (1998)), compared to a value of 13% for a Boeing 747. The large centerbody t/c will cause an increased pressure and wave drag, and high supervelocities on a thick airfoil can result in a low critical Mach number (E. Obert & R. Slingerland (2007)). Increasing the centerbody chord is not a solution, as this would yield an increase in wetted area and an associated penalty on friction drag (Roman et al. (2000)).

The BWB outboard wing t/c on the other hand is similar to that of a conventional wing. The flow across the wing will therefore be strongly three-dimensional. Air is not only able to flow straight across the centerbody, but also outwards, decreasing the maximum supervelocities of the flow on the centerbody. The opposite effect is felt on the outboard wing, where effective thickness increases. The effect can be so strong, that the effective t/c of the outboard wing becomes equally large or even larger than that of the centerbody (see Figure 3-4.

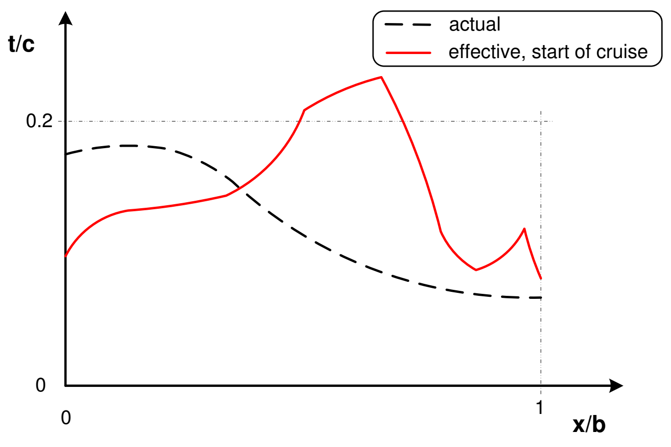


Figure 3-4: 3D effects between BWB centerbody and wing, adapted from Wakayama & Kroo (1998)

Another issue is the large chord length of the centerbody compared to those of the outboard wing. This wing therefore requires high sectional lift coefficients w.r.t. the centerbody sections. As a consequence, the outboard wing stall becomes critical. The traditional problem of preventing that the wing tips stall first (see Raymer (2006)) is aggravated by this. Wing tip stall on the Blended Wing-Body would not only cause ineffectiveness of the outboard TE control surfaces, but also affect the winglets, decreasing yaw control authority.

3-3-2 Structures

The major structural challenge in Blended Wing-Body design, according to Liebeck (2004) and Geuskens et al. (2008), lies in coming up with a suitable pressure cabin within the centerbody, that is non-cylindrical. Pressure loads result in axial and circumferential stresses in the cabin structure. The challenge now is to find another pressure cabin concept which does not give a large weight penalty w.r.t. cylindrical cabins. For this, two concepts are identified:

1. Using a thick sandwich structure as one composite shell, combining outer skin and pressure vessel, carrying both wing bending and pressurization loads

2. Using a multi-bubble composite structure as pressure vessel, separated from the aerodynamic skin

Another structural challenge is the difference in (absolute) thickness between centerbody and outboard wing, along with the fact that their connection should be smooth and continuous, which requires complicated 3D shapes. This will increase complexity of the manufacturing process (see Roman et al. (2000)). Once more data and research is available on the cabin concepts, correction factors for structural weight can be derived from them, which are subsequently applied to a conceptual Blended Wing-Body design to have a general idea of the effect of such a cabin on the aircraft weight and performance.

3-3-3 Control

A conventional aircraft uses of flaps to increase lift in the take-off and landing phases, so that its approach and take-off speeds can be sufficiently low. Flaps also generate significant nose-down pitching moments (Anderson (2001)), which need to be balanced by the horizontal tail surfaces. The BWB cannot use flaps as high lift devices, since a horizontal tail is absent. This has two consequences. First of all, the landing approach attitude will be high, because maximum lift occurs at high angles-of-attack. Secondly, not having flaps constrains the BWB wing loading to a maximum value for an approach speed that is sufficiently low. To prevent separation and stall at high angles-of-attack during the landing phase, the outboard wings of the NASA Blended Wing-Body are equipped with LE slats.

A critical problem is the power needed for controlling the aircraft according to Roman et al. (2000)). Without horizontal tail, the BWB will be longitudinally unstable requiring a fly-by-wire system. Moment arms of the TE control surfaces w.r.t. the aircraft center-of-gravity will be smaller than those of conventional aircraft (see Figure 3-5).

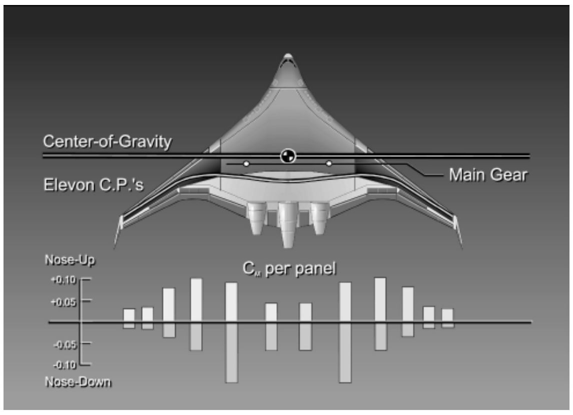


Figure 3-5: Control authority of BWB TE control surfaces

If moment arms become smaller, the forces that need to be generated by the control surfaces need to grow proportionally. This requires higher surface deflections and consequently larger hinge moments. According to Roman et al. (2000), the control power needed to generate these moments becomes prohibitive, even when using full span TE surfaces.

Finally, trim drag is significantly more critical in the Blended Wing-Body compared to conventional transports. The Blended Wing-Body will have one c.g. condition, where the aircraft is exactly in equilibrium and no control surface deflections are necessary. This condition occurs once during the mission, when a certain amount of fuel is used, or for a period of time when the aircraft is equipped with trim tanks used to maintain this c.g. position. As the c.g. moves further away from this ideal position, larger deflections of the TE control surfaces are necessary to maintain equilibrium (not considering the continuous variations in deflection needed for artificial longitudinal stability). The above is also true for conventional aircraft but with this difference: a Blended Wing-Body requires larger control surface deflections to keep accounting for the moving c.g. due its small moment arm.

3-3-4 Passenger comfort

The passenger cabin is positioned inside the centerbody airfoil, which must be at sufficient angle-of-attack to generate enough lift. However, requirements exist regarding the maximum angle the passenger cabin floor is allowed to have w.r.t. the horizontal (Liebeck et al. (1998)). This requirement is limiting the BWB cruise angle-of-attack. Furthermore, the width of the passenger compartment must not be too large, to limit passenger offset from the aircraft centerline, for reasons of passenger comfort. Both limits can be easily implemented by constraining floor angle and fuselage width in the upcoming Blended Wing-Body design problem.

Chapter 4

MDO: a short introduction

In the research goal defininition, BLISS was selected as the MDO approach of interest. MDO has become a diverse field of science in whcih many new concepts and strategies have emerged over the past decades. This chapter aims to cover some of the basics of MDO, and discuss the position and classification of BLISS within the range of available MDO methods. Section 4-1 discusses some fundamental concepts used in MDO. Section 4-2 gives a classification of the BLISS strategy. This components of this classification are explained in sections 4-3 to 4-5.

4-1 Basic Concepts

Multidisciplinary Design Optimization saw its first application in the 60's in structural optimization. Structural optimization problems were solved using Non-Linear Programming approach, using the most basic layout of a single design discipline with an optimizer, as in Figure 4-1.

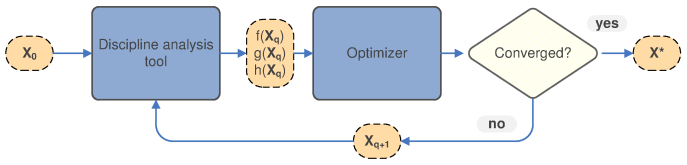


Figure 4-1: Basic optimization layout

X is the so-called design vector, a list of design variables representing a given design concept that possesses specific geometry and properties. The analysis code computes the objective function f and the constraints **g** and **h**. If all constraints are satisfied, the design is feasible. All three functions are (unknown) functions of the **X** and some constant parameters **P**. The elements of **X** themselves can be bounded by upper and lower limits $\mathbf{X}_{min}$ and $\mathbf{X}_{max}$ The design space is defined as the set of all possible designs, and a specific design vector is a 'point' in this space. Following Figure 4-1, the mathematical formulation of a basic optimization problem is as follows:

Problem 4-1.1. *Minimize* $f = f(\mathbf{X}, \mathbf{P})$

subject to:
$g = g(\mathbf{X}, \mathbf{P}) \leq 0 \; h = h(\mathbf{X}, \mathbf{P}) = 0$

and:

$\mathbf{X}_{min} \leq \mathbf{X} \leq \mathbf{X}_{max}$

There exist two types of design spaces: concave and convex ones. In a convex design space, each design point on a line between two design vectors is also inside the design space. For a concave design space, this is not necessarily true. Figure 4-2 shows an example of both types, together with definitions of the design point and design cycle. For the time being, it is assumed that the design spaces of the MDO problems treated in this research are convex, i.e. that a design can always be formulated when the design variables respect their bounds.

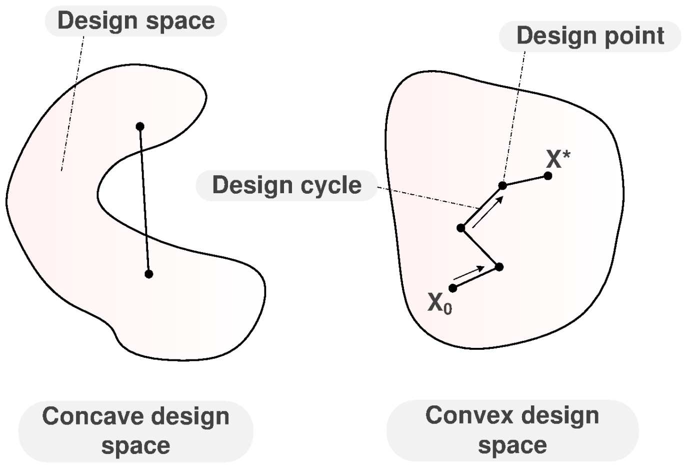


Figure 4-2: Design space, cycle and point

The optimizer tries to move from the initial design point $\mathbf{X}_0$ to the (supposed) optimum design $\mathbf{X}^*$. Going through the loop of Figure 4-1 once (so one update of the design vector) is termed a design cycle. As MDO problems became more and more complex and involved more disciplines, the computational load and problem complexity had to be brought down by decomposing the problem. A problem can be decomposed in several ways. According to Tosserams et al. (2010) and M. van Tooren et al. (2009) there exist two principal decomposition strategies for a system:

1. **Aspect based**: decomposition according to physical aspects (e.g. aerodynamics, structures, propulsion, etc.)
2. **Object based**: decomposition according to system components (e.g. fuselage, tail, wing, engines etc.)

Since the Blended Wing-Body , by its nature is an integrated design, focus will be on aspect based decomposition. Decomposition does not end at design disciplines: they can be decomposed further into sub-disciplines, and for each sub-discipline, a set of models is available to analyze the behaviour of a design (see Figure 4-3).

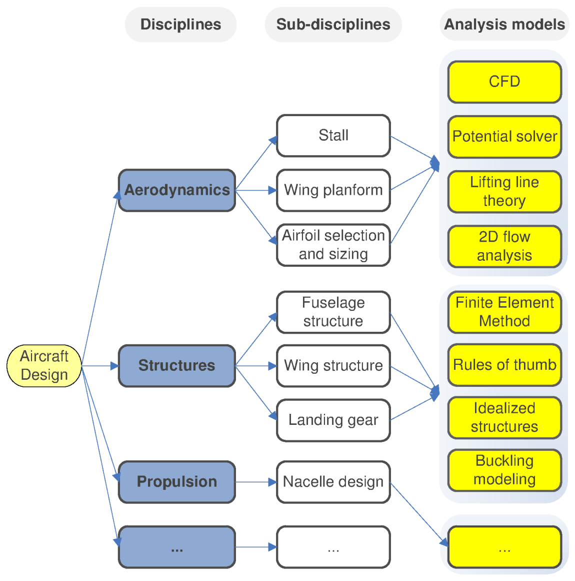


Figure 4-3: Aspect based decomposition of an aircraft design problem

In practice, the issue of decomposition also considers the implementation of an MDO problem. Analysis models can have large variations in their durations, the amount of data they generate et cetera. If an MDO method requires running the 'blocks' in which the problem was decomposed in parallel at some point, the computational effort of these blocks must be balanced. In the actual problem implementation, the problem is therefore decomposed in so-called BlackBoxes (BB's), across which the design disciplines can be divided in various ways to spread computational loads. Figure 4-4) illustrates the relation between BB's, (sub-)disciplines and models in the decomposition process.

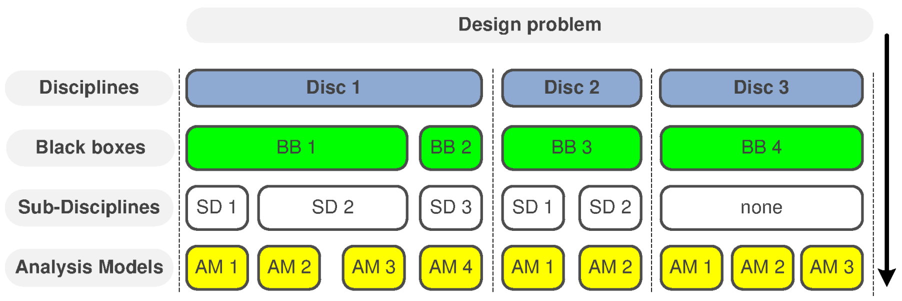


Figure 4-4: Decomposition hierarchy possibilities for an MDO problem implementation

When multiple disciplines are used, the issue of coupling them arises. One discipline may require inputs that are provided as output by another. Thus, by definition, coupling variables are state variables. According to Weck et al. (2007), there exist two types of couplings: *feedforward* and *feedback*. Another possibility is that the disciplines are not coupled, which essentially makes MDO unnecessary for this type of problem. Table 4-1 shows the properties of each coupling type.

Figure 4-5 is a visual illustration of the coupling directions and naming convention, shown for a simple example with two disciplines.

Table 4-1: Coupling directions

Coupling direction	Description
Feedforward	discipline i needs input coming from discipline j, **sequential** analysis, **decoupled** system
Feedback	discipline i needs input coming from discipline j and vice versa, **iterative** analysis, **coupled** system
None	discipline i and discipline j don't exchange information, **separate** analysis, **uncoupled** system

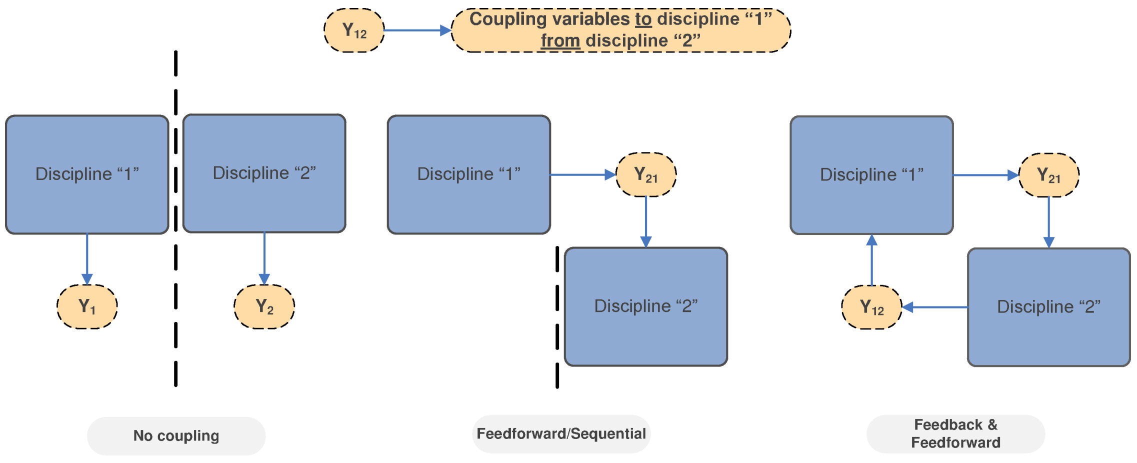


Figure 4-5: Types of interdisciplinary couplings

The charts of Figure 4-5 are handy tools to have an overview of the complete structure of the design problem, expecially if many disciplines are involved. In the MDO community, such charts are called N2-charts or Design Structure Matrices (see Perez et al. (2001)).

4-2 BLISS Classification

Over the decades MDO was investigated further, involving more and more complex algorithms, problem structures and discipline models. There exist a number of directions, which can be used to classify an MDO algorithm. These directions are discussed in sections 4-3, 4-4 and 4-5, respectively (M. van Tooren et al. (2009)):

- Method of design analysis: **Simultaneous** or **Nested** Analysis And Design
- **Single-** and **multilevel** optimization
- Use of **gradient-** or **non-gradient** based optimizer

The above list is useful for getting an overview of the different MDO strategies currently available. An extensive overview of these methods is not given here since BLISS was already selected in Hendrich (2009). The BLISS or Bi-Level Integrated System Synthesis method is placed in the MDO method range as follows:

Definition 4-2.1. *BLISS is a two-level method that employs Nested Analysis and Design in combination with gradient-based optimization algorithms that provide Lagrangians*

4-3 Methods of Design Analysis

Analysis is defined by M. van Tooren et al. (2009). as the activity of finding the values of the state variables Y, given the design vector X, using some suitable computational model in each discipline. An important concept in the analysis of a system of coupled disciplines is consistency. The output of a discipline '1' depends on input coming from another discipline '2', so if this input changes, the output of '1' changes, in turn influencing the output of '2' and so on. A design is consistent if the system input state $\mathbf{Y} = [Y_{12}\ Y_{21}]$ is equal to the system input state $\mathbf{Y}' = [\mathbf{Y}'_{12}\ \mathbf{Y}'_{21}]$.

Definition 4-3.1. *A system of disciplines is consistent if the change in state vector **Y** after a sequential run of all disciplines is sufficiently small.*

Consistency of a design tells nothing about feasibility: a consistent design may still be unfeasible because not all constraints are satisfied. At any rate, the designer wishes to obtain a consistent design at the end of the design optimization. If the resulting design is inconsistent, it literally makes no sense.

There exist two ways to perform design analysis (see Balling & Sobieszczanski-Sobieski (1996)):

1. Nested Analysis and Design (NAND)
2. Simultaneous Analysis and Design (SAND)

In NAND, all coupled disciplines are evaluated iteratively given the design vector until the system state variables converge (and the design becomes consistent). This convergence is reached if the difference in each state variable (the so-called residual for that variable) between two iterations is sufficiently small. Using NAND yields a consistent design, however the computational cost can become large because disciplinary models must be run iteratively. BLISS is a method that uses NAND, requiring a full Multidisciplinary Analysis (MDA) each cycle.

While NAND iterations require evaluating the discipline models one after another, using SAND they can be evaluated in parallel. To do this, the couplings between the disciplines must be torn, so that they become independent of each other. In this case, consistency is no longer maintained, and an inconsistent design is allowed. MDO strategies that use SAND make sure that the optimal design is consistent by introducing consistency constraints[1]. These constraints require that the state of the entire system is equal to the collection of states from the individual dsciplines. Thus, SAND avoids expensive system analyses but its drawback is that it needs to converge to a point where all consistency constraints are satisfied. Before this point is reached, the design is inconsistent and the designer cannot avaluate the properties of the design at intermediate stages.

4-4 Single- and Multilevel MDO Strategies

After dealing with design *analysis* methods in section 4-3, this section will focus on design *optimization*. When a single optimizer has to handle more design variables it will need more time to find an optimum. When the system scale is truly large[2], the optimizer may not be able to converge to a solution at all. In MDO, convergence doesn't only depend on the convergence of the optimization algorithm(s) (e.g. SQP), but also on the changes that occur from design cycle to design cycle.

So, as the design vector contains more and more elements, it becomes desirable to use optimizers at multiple levels. They only have to optimize part of the problem, meaning that they are likely to obtain results faster (Balling & Sobieszczanski-Sobieski (1996)). This leads to the division of the design vector in two parts, namely global and local design variables.

[1] An example of such a method is Collaborative Optimization, which is extensively described in I. Sobieski & Kroo (1996)

[2] In practice, the limit lies at around 15 design variables, again depending on the problem

Definition 4-4.1. *Local design variables (**X**-variables) serve as input to only one BlackBox. State variables (**Y**-variables) are defined as variables that are output of one BlackBox and used as input by one or more other BlackBoxes. Global design variables (**Z**-variables) are defined as design variables that serve as input to two or more BlackBoxes.*

Figure 4-6 gives a schematic showing a generic two-level problem structure. **Z**-variables are modified by the system optimizer, while an **X**-variable can only be changed by the optimizer of its respective discipline. In a local optimization, Z-variables are treated as constants, and **X**-variables as independent variables, while it is the other way around during a system optimization. An MDO method that uses separate optimizers for global and local design variables is called a multilevel method.

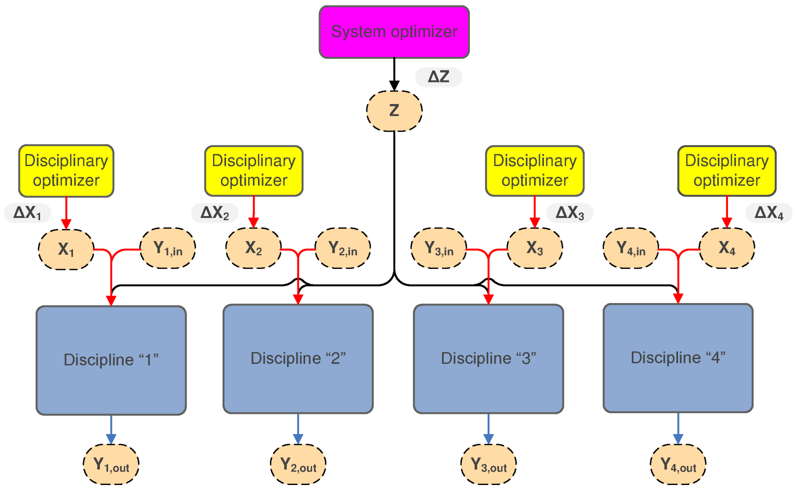


Figure 4-6: Two-level BLISS problem structure

According to literature (Weck et al. (2007), Perez et al. (2004)), multilevel methods should be able to tackle more complex problems, although they are inherently more complex than single level MDO formulations. Multilevel methods offer the possibility of performing the optimization of the local design variables in parallel and at different geographical locations.

4-5 Gradient- and Non-gradient Based Optimization

Finally, some attention is paid to the optimizer algorithm itself. These algorithms branch into two main directions: Gradient based and Non-gradient based algorithms.

Gradient-based Optimization Methods

Gradient-based methods use information on both the objective value itself and the sensitivity of the objective function with respect to the design variables to determine the required change perturbation $\delta\mathbf{X}_q$. This sensitivity comes in the form of first or second order gradients of the objective function with respect to the design variables. In case of a gradient based method[3], $\delta\mathbf{X}_q$ is found by determining a search direction $\mathbf{S}_q$ and the optimal step size α^* in that direction to minimize the objective (M. van Tooren et al. (2009)).

$$\mathbf{X}_q = \mathbf{X}_{q-1} + \alpha^*\mathbf{S}_q \tag{4-1}$$

[3]It is assumed that the optimization problem at hand is constrained and continuous and that the objective function is smooth in the region of interest

Examples of gradient-based methods are Sequential Quadratic Programming (SQP), the method of steepest descent and Newton's method. Since SQP plays a main role in the Matlab optimization function `fmincon`, it will be explained in more detail in appendix A. Because Matlab will be used as programming language in current research, `fmincon` is an important candidate optimization algorithm. Different definitions for $\mathbf{S}_q$ constitute the different gradient-based methods. Gradient based methods are robust and quick in finding an optimum, but they have a number of important disadvantages. First of all, they walk from a starting point $\mathbf{X}_0$ to an optimum $\mathbf{X}^*$, getting closer to it each iteration. However if the design space has more than one optimum, a gradient-based algorithm may converge to a local optimum that is not the 'true' global optimum of the design space.

Non-gradient-based Optimization Methods

Non-gradient based optimization methods are suited for both finding a global optimum and dealing with discrete variables. These methods do not use gradients to converge to the optimum design, they only use evaluations of the objective function (M. van Tooren et al. (2009)). In this version of the algorithm, first a population of designs is generated using some probabilistic technique. Then, two designs $\mathbf{x}_1$ and $\mathbf{x}_2$ are drawn from this population and their objective values compared. The design with the best objective value 'survives', the other one is discarded. This is done until the whole population is covered. The design then having the largest objective value is considered the optimal design. To prevent that the genetic optimizer selects a final design too soon, it can 'mutate' designs in the population to see if a better design is possible.

Since non-gradient based methods don't rely on numerical procedures, they are easy to implement. The major disadvantage of non-gradient based methods is, that it is impossible to judge, if the end result is close to the real global optimum, because gradient information is not present and the shape of the objective function across the design space is unknown. Furthermore, genetic and other non-gradient based algorithms are computationally one or two orders of magnitude more expensive than gradient based algorithms (Soremekunt et al. (1996))[4].

[4]So, if a gradient based algorithm needs n iterations to converge, a genetic one may need n^2 or even n^3

Chapter 5

BLISS Framework

This chapter discusses the creation of the BLISS framework. The theory behind BLISS is illustrated in section 5-1, while section 5-2 argues which capabilities the BLISS framework should have. The structure of the actual implementation of the framework is discussed in sections 5-3 to 5-7.

5-1 BLISS Theory

BLISS is a multi-level method that does incorporate a MDA module. It utilizes the so-called Global Sensitivity Equations (GSE) to find the changes in the design vector required to minimize the system objective. As will become clear later on, this means that instead of using the actual - and generally non-linear - sensitivities of the design state to the design vector and the constraints to the state, BLISS optimizes the problem using linearized ones.

In BLISS, a problem is decomposed into two or more BlackBoxes (BB's), each containing one or more discipline models (see 7-7). A BB_r has as inputs the local design vector $\mathbf{X}_r$, the global design vector $\mathbf{Z}$, and coupling states $\mathbf{Y}_{r,s}$ coming from other BB (see Figure 5-1). It is assumed that there are n BB's. After analysis of BB_r, it outputs its state vector $\mathbf{Y}_r$ and constraint vector $\mathbf{G}_r$.

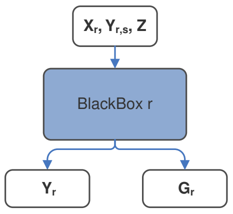


Figure 5-1: BlackBox in/outputs

The system state vector $\mathbf{Y}$ is the collection of all BlackBox output state vectors $\mathbf{Y}_r$. The system objective function Φ is given as an output of a particular BB_m, so as an element of its output state vector $\mathbf{Y}_{m,i}$. BLISS uses the gradients $\frac{d\Phi}{d\mathbf{X}}$ and $\frac{d\Phi}{d\mathbf{Z}}$ of the system objective Φ with respect to the design variables $\mathbf{X}$ and $\mathbf{Z}$. The optimization algorithms used in the system and disciplinary optimizations must therefore be gradient-based and provide Lagrangians. The design variables themselves must be continuous everywhere in the design space. Other than that, the designer is free to choose a suitable optimization algorithm. Variables for which it is not possible to calculate a gradient (e.g. integer variables) cannot be used in BLISS. The subsequent sections 5-1-1 to 5-1-6 cover the individual components of BLISS. A complete BLISS optimization cycle consists of a series of procedures:

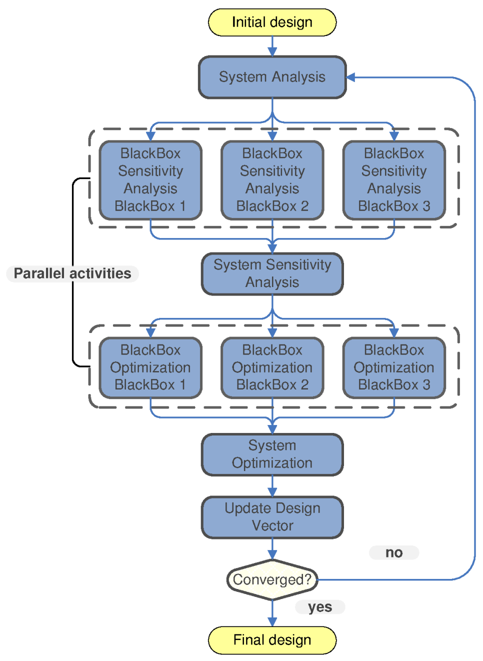

Initial design
System Analysis
BlackBox Sensitivity Analysis BlackBox 1
BlackBox Sensitivity Analysis BlackBox 2
BlackBox Sensitivity Analysis BlackBox 3
Parallel activities
System Sensitivity Analysis
BlackBox Optimization BlackBox 1
BlackBox Optimization BlackBox 2
BlackBox Optimization BlackBox 3
System Optimization
Update Design Vector
no
Converged?
yes
Final design

Figure 5-2: BLISS method components

As Figure 5-2 shows, the 'BlackBox Sensitivity Analysis' and the 'BlackBox Optimization' are performed for each BlackBox separately. During these activities, no information is exchanged between BlackBoxes, hence they can be executed in parallel. BLISS is started by initializing the design vector **X** and **Z**. Subsequently, an initial System Analysis (SA) is performed to find the complete system state **Y** (see section 4-3), the constraint values **G** and the system objective Φ.

5-1-1 BlackBox Sensitivity Analysis

BLISS continues by performing a BlackBox Sensitivity Analysis (BBSA), in which the inputs for each BlackBox ($\mathbf{X}_r$, $\mathbf{Y}_{r,s}$ and **Z**) are changed with a small amount to compute the gradient of the outputs ($\mathbf{Y}_r$ and $\mathbf{G}_r$) due to this change in input.

Since BLISS has both a discipline (local) level and a system (global) one, a clear distinction must be made between partial derivatives $\frac{\partial f}{\partial x}$ and total derivatives $\frac{df}{dx}$:

$$\frac{df(y(x))}{dx} = \frac{\partial f(y(x))}{\partial x} + \frac{\partial f(y(x))}{\partial y}\frac{dy(x)}{dx} \tag{5-1}$$

The BBSA calculates the local gradients $\frac{\partial Y_r}{\partial Y_{r,s}}$, $\frac{\partial Y_r}{\partial X_r}$, $\frac{\partial Y_r}{\partial Z}$, $\frac{\partial G_r}{\partial X_r}$, $\frac{\partial G_r}{\partial Y_{r,s}}$ and $\frac{\partial G_r}{\partial Z}$ (see Figure 5-3).

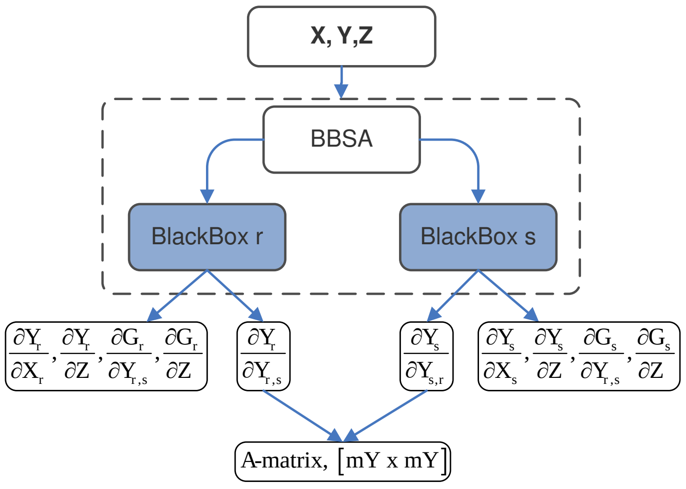

X, Y,Z
BBSA
BlackBox r
BlackBox s
$\frac{\partial Y_r}{\partial X_r}, \frac{\partial Y_r}{\partial Z}, \frac{\partial G_r}{\partial Y_{r,s}}, \frac{\partial G_r}{\partial Z}$
$\frac{\partial Y_r}{\partial Y_{r,s}}$
$\frac{\partial Y_s}{\partial Y_{s,r}}$
$\frac{\partial Y_s}{\partial X_s}, \frac{\partial Y_s}{\partial Z}, \frac{\partial G_s}{\partial Y_{r,s}}, \frac{\partial G_s}{\partial Z}$
A-matrix, [mY x mY]

Figure 5-3: Schematic of the BlackBox Sensitivity Analysis

This gradient calculation is executed element-wise. These gradients may be calculated by a method of choice, in this thesis, finite differencing is used. During sensitivity analysis, again no information from other BB is required. The gradients $\frac{\partial Y_r}{\partial Y_{r,s}}$ are collected in a matrix A:

$$A = [\text{m x m}] = \begin{bmatrix} I & A_{1,2} & \cdots & A_{1,n} \\ A_{2,1} & I & \cdots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \cdots & I \end{bmatrix} \tag{5-2}$$

where *m* is the number of elements of **Y**. A submatrix $A_{r,s}$ of A contains the sensitivities of the output state vector $\mathbf{Y}_r$ of BlackBox *r* to that of BlackBox s, $\mathbf{Y}_s$. $A_{r,s}$ can be written as:

$$A_{r,s} = [\text{m x n}] = -\begin{bmatrix} \frac{\partial Y_{r,1}}{\partial Y_{s,1}} & \frac{\partial Y_{r,1}}{\partial Y_{s,2}} & \cdots & \frac{\partial Y_{r,1}}{\partial Y_{s,n}} \\ \frac{\partial Y_{r,2}}{\partial Y_{s,1}} & \frac{\partial Y_{r,2}}{\partial Y_{s,2}} & \cdots & \frac{\partial Y_{r,2}}{\partial Y_{s,n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Y_{r,m}}{\partial Y_{s,1}} & \frac{\partial Y_{r,m}}{\partial Y_{s,2}} & \cdots & \frac{\partial Y_{r,m}}{\partial Y_{s,n}} \end{bmatrix} \tag{5-3}$$

where *m* and *n* are the number of elements of $\mathbf{Y}_r$ and $\mathbf{Y}_s$, respectively.

5-1-2 System Sensitivity Analysis

Following the BBSA, which obtained local **Y**-derivatives w.r.t. **X** and **Z**, the so-called System Sensitivity Analysis (SSA) aims to find their global equivalents. With the information from the BBSA, the GSE are solved in the SSA for a particular $X_{r,i}$ to obtain $\frac{dY}{dX_{r,i}}$ from $\frac{\partial Y}{\partial X_{r,i}}$ and $\frac{\partial Y}{\partial Y}$:

$$A\left(\frac{d\mathbf{Y}}{dX_{r,i}}\right) = \frac{\partial \mathbf{Y}}{\partial X_{r,i}} \tag{5-4}$$

In total, $\frac{d\mathbf{Y}}{d\mathbf{X}}$ is an [nY x mZ] matrix. Analogous to equation 5-4, the SSA finds $\frac{d\mathbf{Y}}{d\mathbf{Z}}$, which is a matrix of size [nY x mZ]. Now, the global **X**- and **Z**-sensitivities of all state variables are known. Since the objective function Φ is also a **Y**-variable, its sensitivity to the design variables is found by selecting the right column from $\frac{d\mathbf{Y}}{d\mathbf{X}}$and $\frac{d\mathbf{Y}}{d\mathbf{Z}}$ (see Figure 5-4).

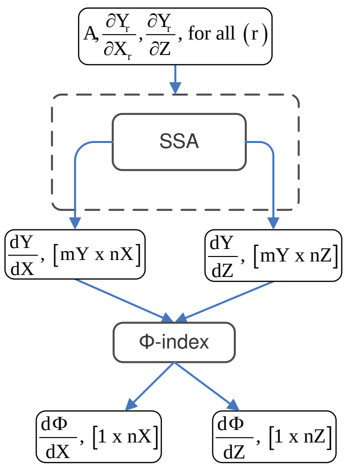

$A, \frac{\partial Y_r}{\partial X_r}, \frac{\partial Y_r}{\partial Z}$, for all (r)
SSA
$\frac{dY}{dX}$, [mY x nX]
$\frac{dY}{dZ}$, [mY x nZ]
Φ-index
$\frac{d\Phi}{dX}$, [1 x nX]
$\frac{d\Phi}{dZ}$, [1 x nZ]

Figure 5-4: Schematic of the System Sensitivity Analysis

5-1-3 Move Limits in X and Z

By linearizing the optimization problem, BLISS creates an approximation error e_{approx} in the objective and constraints. This error grows when the changes in design vector per cycle become larger. Figure 5-5 is a graphical description of this principle, where an optimizer increases parameter q with Δq. Based on the linearized sensitivity of q to the constraint (or element thereof) $\mathbf{G_i}$, the step Δq still lets the design satisfy $\mathbf{G_i}$, while according to the actual constraint value, the constraint is not satisfied anymore.

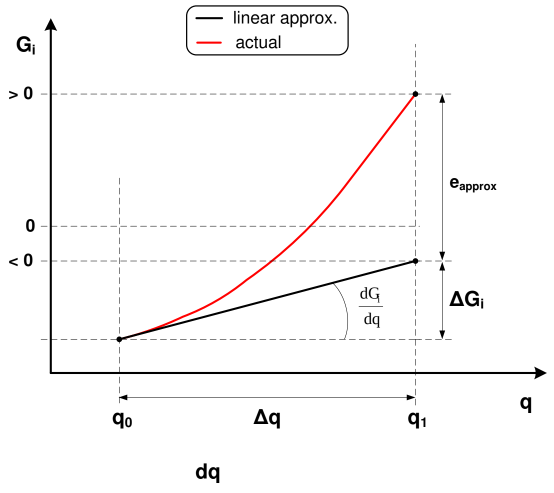
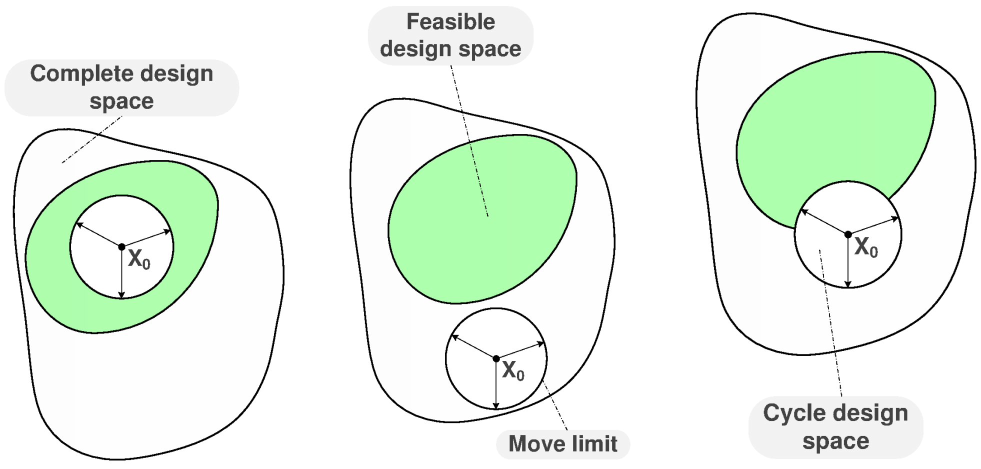


Figure 5-5: Approximation error

To limit the error, the maximum absolute change in **X** and **Z** is constrained by so-called move limits (eq. 5-5). A clear distinction should now be made between move limits and side constraints. Both are bounds directly imposed on design variables, but side constraints bound the overall design space, while move limits create a bounded region around a given design point to limit the changes the optimizers can make in going from that design point to the next. Since during one BLISS cycle, the optimizers have to find the next design point within this bounded region, it is termed 'cycle design space'.

$$\begin{aligned} \Delta\mathbf{X}^{-} &\leq \Delta\mathbf{X}_{r} \leq \Delta\mathbf{X}^{+} \\ \Delta\mathbf{Z}^{-} &\leq \Delta\mathbf{Z}_{r} \leq \Delta\mathbf{Z}^{+} \end{aligned} \tag{5-5}$$

Depending on the design point at hand, figures 5-6b to 5-6a show the three possible scenarios:



(a) Inside feasible region

(b) Partially inside feasible region

(c) Outside feasible region

Figure 5-6: Cycle design space locations

5-1-4 BlackBox Optimization

With the SSA completed, the BlackBox optimization (BBOpt) can now be performed for each BB_r using $\frac{\partial \Phi}{\partial X_r}$. The aim of the BBOpt is to alter $\mathbf{X}_r$ bya $\Delta\mathbf{X_r}$ that minimizes the objective function. Meanwhile, **Z**-variables are treated as constants. During optimization, the optimizer must satisfy side constraints, move limits on **X** and the constraints present in BB_r. If linearized constraints are used, a $\Delta\mathbf{X}$ in BB_r will influence constraint values of other BlackBoxes as well through $\frac{\partial \mathbf{G}}{\partial \mathbf{Y}}$ and $\frac{d\mathbf{Y}}{d\mathbf{X}}$:

$$\begin{aligned} \mathbf{G}^{T} &= \mathbf{G}^{T} + \left(\frac{\partial \mathbf{G_r}}{\partial \mathbf{X}_r} + \frac{\partial \mathbf{G}}{\partial \mathbf{Y}}\frac{d\mathbf{Y}}{d\mathbf{X}_r}\right)^{T} \Delta\mathbf{X_r} \\ &= \mathbf{G}^{T} + \frac{\mathbf{dG}}{\mathbf{dX}_r}\Delta\mathbf{X_r} \end{aligned} \tag{5-6}$$

Thus the constraint changes are approximated by using the total constraint sensitivity w.r.t. X instead of its local counterpart.

With the information from BBSA and SSA, BBOpt uses a linear approximation to the change in objective value $\Delta\Phi_{\mathbf{r}}$ due to $\Delta\mathbf{X_r}$:

$$\begin{aligned}\Phi &= \Phi_0 + \sum_{r=1}^{nBB} \Delta\Phi_r \\ &= \Phi_0 + \sum_{r=1}^{nBB} \frac{d\Phi}{d\mathbf{X}_r}^T \Delta\mathbf{X}_r \\ &= \Phi_0 + \sum_{r=1}^{nBB} \frac{d\Phi}{dX_r(1)} \Delta X_r(1) + \cdots + \frac{d\Phi}{dX_r(n)} \Delta X_r(n)\end{aligned} \tag{5-7}$$

The $\Delta\Phi_r$ terms can be interpreted as the contributions of each individual BlackBox to the objective improvement realized in the BlackBox optimizer. Mathematically, the BBOpt module has to solve the following optimization problem for a BB_r:

Problem 5-1.1. *Given:* $\mathbf{X}_r$, $\mathbf{Z}$,$\mathbf{Y}_{r,s}$, $\frac{d\Phi}{d\mathbf{X}_r}$
Find: $\Delta\mathbf{X}_r$
Minimizing: $\Delta\Phi_r = \frac{d\Phi}{d\mathbf{X}_r}\Delta\mathbf{X}_r$
Satisfy:
$\mathbf{G_r} \leq \mathbf{0}$
$\mathbf{H_r} = \mathbf{0}$
$\mathbf{X}_l \leq \mathbf{X}_r \leq \mathbf{X}_u$
$\Delta\mathbf{X}_{min} \leq \Delta\mathbf{X}_r \leq \Delta\mathbf{X}_{max}$

5-1-5 System Optimization

A similar procedure is used to optimize the global design variables. Following J. Sobieski et al. (1998), $\frac{d\Phi}{d\mathbf{Z}}$ is augmented with contributions from the constraint sensitivities w.r.t $\mathbf{Z}$ and w.r.t. $\mathbf{Y}$. The latter accounts for the possibility that $\mathbf{Y}$ = $\mathbf{Y}$($\mathbf{Z}$).

$$\frac{d\Phi}{d\mathbf{Z}}^T_{aug} = \frac{d\Phi}{d\mathbf{Z}}^T + \sum_{r=1}^{n} \left(\lambda_r^T \frac{\partial\mathbf{G}_r}{\partial\mathbf{Z}}\right) + \left[\sum_{r=1}^{n} \left(\lambda_r^T \frac{\partial\mathbf{G}_r}{\partial\mathbf{Y}}\right)\right] \frac{d\mathbf{Y}}{d\mathbf{Z}} \tag{5-8}$$

where λ_r is the vector of Lagrange multipliers of BB_r, which are obtained at the end of the BB optimization.

There may exist constraints $\mathbf{G}_{yz}$ that depend more on $\mathbf{Y}$ and $\mathbf{Z}$ than on $\mathbf{X}$, or that don't depend on $\mathbf{X}$ at all. Since a BB optimizer only manipulates $\mathbf{X}$, it will experience difficulties satisfying these constraints. This problem is circumvented by including the constraint also in the system optimizer. The $\mathbf{G}_{yz}$ is extrapolated to account for changes in the system design vector using its gradient w.r.t. $\Delta\mathbf{Z}$ analogously to equation 5-23:

$$\mathbf{G}_{yz}^T = \mathbf{G}_{yz,0}^T + \left(\frac{\partial\mathbf{G}_{yz}}{\partial\mathbf{Z}} + \frac{\partial\mathbf{G}_{yz}}{\partial\mathbf{Y}}\frac{d\mathbf{Y}}{d\mathbf{Z}}\right)^T \Delta\mathbf{Z} \tag{5-9}$$

With $\frac{d\Phi}{d\mathbf{Z}}$ found, the objective function of the System Optimization (SysOpt) becomes similar to BBOpt:

$$\begin{aligned}\Phi &= \Phi_0 + \Delta\Phi_{sys} \\ &= \Phi_0 + \frac{d\Phi}{dZ(1)}_{aug} \Delta Z(1) + \cdots + \frac{d\Phi}{dZ(n)}_{aug} \Delta Z(n)\end{aligned} \tag{5-10}$$

Again, $\Delta\Phi_{sys}$ can be interpreted as the objective improvement realized by the system optimizer. The system optimizer has to satisfy the constraints $\mathbf{G}_{yz}$, move limits and side constraints on **Z** itself. In SysOpt, **X**-variables are treated as constants. Hence, the SysOpt optimization problem is formulated as:

Problem 5-1.2. *Given:* **X**, **Z**,$\mathbf{Y}_{r,s}$, $\frac{d\Phi}{d\mathbf{Z}}_{aug}$
Find: $\Delta\mathbf{Z}$
Minimizing: $\Delta\Phi = \frac{d\Phi}{d\mathbf{Z}}_{aug}\Delta\mathbf{Z}$
Satisfy:
$\mathbf{G}_{yz} \leq \mathbf{0}$
$\mathbf{Z}_l \leq \mathbf{Z} \leq \mathbf{Z}_u$
$\Delta\mathbf{Z} \leq \Delta\mathbf{Z}_{max}$

5-1-6 Update Design Vector

The last step is to update the design vector **X** and **Z** with the changes suggested by the BlackBox and System optimizers:

$$\mathbf{X} = \mathbf{X}_0 + \Delta\mathbf{X} \tag{5-11}$$

$$\mathbf{Z} = \mathbf{Z}_0 + \Delta\mathbf{X} \tag{5-12}$$

With the new design vector obtained this way, the accompanying design state **Y** is found by doing a system analysis. If desired, the designer can halt the BLISS algorithm at this point to make changes in the problem formulation, move limits, optimization settings and design vector elements. These in-between changes allow the designer control over the design process while it is still ongoing.

A stopping criterion must be defined to determine if a problem is converged. For this thesis, BLISS is terminated when both BBOpt and SysOpt cannot not realize further objective improvement, the changes in design vector go to zero and the problem is feasible. Mathematically, this stopping criterion becomes:

$$max\left(\Delta\mathbf{X}\right) \leq \epsilon \tag{5-13}$$

$$max\left(\Delta\mathbf{Z}\right) \leq \epsilon \tag{5-14}$$

$$\Delta\Phi_{sys} \leq \epsilon \tag{5-15}$$

$$\sum_{r=1}^{nBB} \Delta\Phi_r \leq \epsilon \tag{5-16}$$

$$\mathbf{G} \leq \mathbf{0} \tag{5-17}$$

where $\epsilon << 1$.

5-2 Framework Capabilities

In its history Multidisciplinary Design Optimization has often been a subject of debate (see Belie (2002)). By some, it is viewed as an attempt to take the human designer more and more out of the loop and viewing the MDO process as a large black box with unverifiable contents. Others see it is an approach that is too complex, which prevents the designer to start the actual design work work right away because of the long time that is needed to set up the MDO problem structure.

Before the implementation of BLISS began, it was evaluated what capabilities the to-be-developed tool should have, in order to become a useful addition in the conceptual design process. To identify these capabilities, a short analysis is conducted on possible user scenarios if the framework is used as a stand-alone application. This was done with the possibilities offered by the BLISS strategy in mind. Figure 5-7 gives the result of this analysis, incorporating the following considerations regarding the framework functionality:

- Ability to quickly implement and structure a variety of design problems in the framework (section 5-2-1)
- Ability to quickly modify the problem formulation between cycles (section 5-2-2)
- Ability to view results per cycle, and distill conclusions (section 5-2-3)

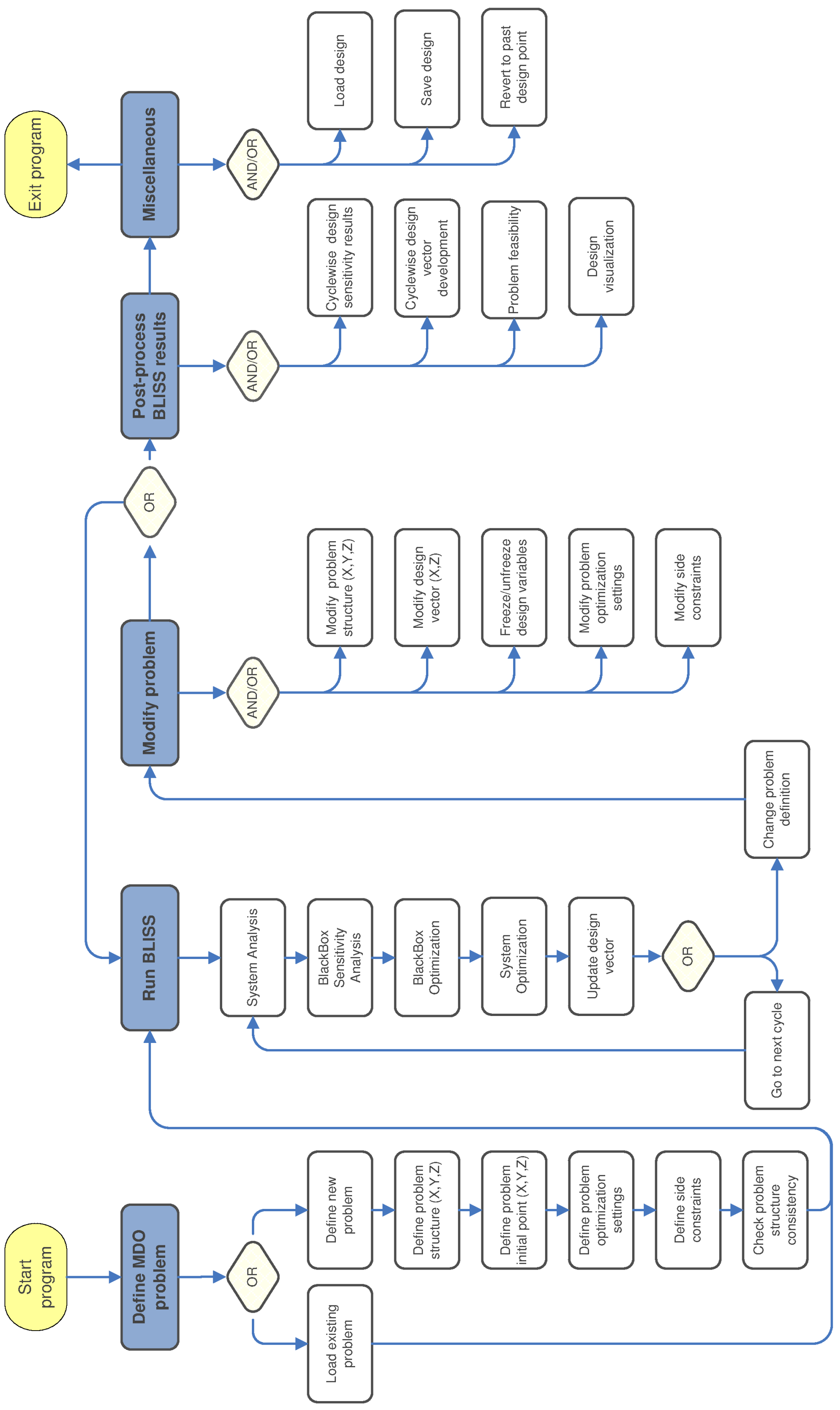

Start program
Define MDO problem
OR
Define new problem
Define problem structure (X,Y,Z)
Define problem initial point (X,Y,Z)
Define problem optimization settings
Define side constraints
Check problem structure consistency
Load existing problem
Run BLISS
System Analysis
BlackBox Sensitivity Analysis
BlackBox Optimization
System Optimization
Update design vector
OR
Go to next cycle
Change problem definition
Modify problem
AND/OR
Modify problem structure (X,Y,Z)
Modify design vector (X,Z)
Freeze/unfreeze design variables
Modify problem optimization settings
Modify side constraints
OR
Post-process BLISS results
AND/OR
Cyclewise design sensitivity results
Cyclewise design vector development
Problem feasibility
Design visualization
Miscellaneous
AND/OR
Load design
Save design
Revert to past design point
Exit program

Figure 5-7: BLISS framework functional analysis

5-2-1 Genericity and Problem Definition

First of all, the framework should give the user a large degree of freedom in defining the problem structure. Although a specific Blended Wing-Body design case is being defined for this research, the framework should also be able to tackle alternative cases as well. Examples are choosing another decomposition with different disciplines and choosing different discipline analysis models. To go even further: in the conceptual design stage, a designer will analyze a number of design options, which may be entirely different from each other. For doing this, he/she needs to define, set up and execute a separate MDO problem for each option. Hence, usefullness of the BLISS framework is greatly increased if it is generic and can be used for any MDO problem once it has been decomposed into two optimization levels.

5-2-2 Design Problem Modification

Secondly, the user cannot always be certain beforehand if a problem structure provides the desired results. For instance, the user may want to try out different objective functions, replace analysis models by others, or insert extra ones. In turn, this can have consequences for the BlackBox decomposition (see 7-7-1). Also, the user may want to examine different aircraft configurations. In this case, modifications in parametrization, analysis models, design vector and couplings can be necessary. In short, the BLISS framework should facilitate design problem modification, so that the user can quickly investigate all problem variants he/she wants. Furthermore, the designer may want the optimizer to use stricter settings (move limits, tolerances) as the design process advances. Fortunately, a BLISS cycle always results in a consistent[1] design, although the design is not necessarily feasible[2]. The designer can follow the changes in throughout the design process, and make changes at intermediate stages if necessary. Therefore, all BLISS framework settings that are defined on initialization should also be modifyable at the end of each BLISS cycle.

5-2-3 Results Postprocessing

Thirdly, each design cycle, BLISS provides variable sensitivities and values of coupling states and changes in the design vector. The sensitivities are needed by the optimization algorithm, but they also provide insight in the design characteristics themselves: to which variables is the design objective most (or least) sensitive? And are these sensitivities, and the in line with the designers experience? If not, a flaw might be present in the problem definition or structure, which the designer can track down using above information. The same discussion holds also for the coupling state values, which are provided after each system analysis. By inspection, the designer can judge if these states have a plausible order of magnitude. In case they don't, the coupling states allow the designer to easily trace the error back to its source (analysis model flaws, sign convention error, etc.) and repair it. Summarizing, the BLISS framework should give the user the opportunity to see design variable changes, coupling state values, sensitivities and their evolutions throughout the framework cycles.

5-3 Framework Implementation

After the qualitative description of the BLISS framework desired capabilities and structure, this has to be translated in to a Matlab implementation.

In the BLISS framework, variable parameters have been structured into three levels (see Figure 5-8):

[1]See section 4-3 for a definition of design consistency

[2]This is one of the main advantages from a conceptual design perspective of BLISS above multi-level MDO methods that don't use a system analysis each cycle, like Collaborative Optimization

- Variable types
  - Dependent variables: **Y**
  - Independent variables: **X** and **Z**
  - Constraints: **G**
- Discipline names
- Variable names

The amount of variable names in the Blended Wing-Body design problem is expected to be large. Therefore, it seems convenient to maintain arrays of variable names in the framework structure, instead of relying on one long list of **X**, **Y** and **Z** . This will prevent the user from mixing up variables and is convenient for defining couplings and displaying results to the user.

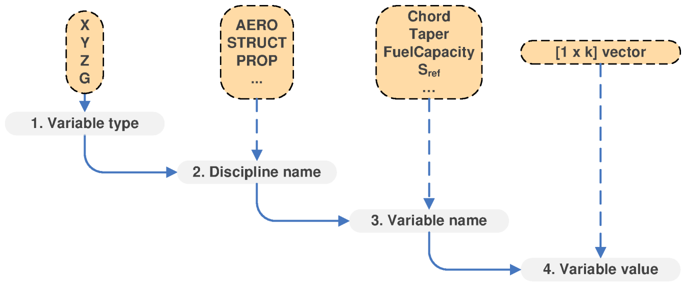


Figure 5-8: BLISS variable structure

Figure 5-8 shows how the aforementioned variable arrays are built up. For instance, the variable:

$$X.AERO.Twist = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7\ \theta_8] \tag{5-18}$$

tells the user that the parameter 'Twist' is a design variable that is only used in the 'AERO' BlackBox since it is an **X**-variable, that this twist is defined at 8 stations and currently has values θ_1 to θ_8.

Constants are are all gathered in one array, called CONSTANTS. Flight conditions are collected separately in the FCOND array for clarity. To run BLISS, all analysis models are wrapped in a BlackBox. BLISS thus only 'sees' the inputs and outputs to and from this BlackBox, and not the variables that are used in the model. Figure 5-9 shows a basic problem structure layout of two BlackBoxes, labeled '1' and '2'. This example layout will be used in the remainder of this section.

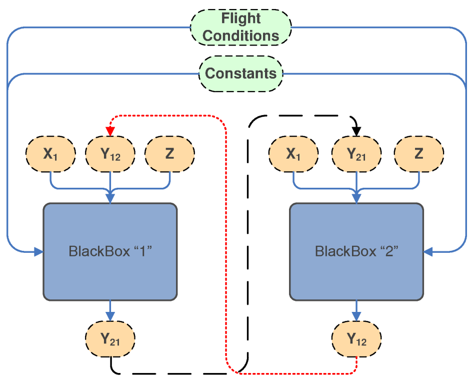


Figure 5-9: Basic problem structure layout

Since the naming formats in- and outside a BlackBox are generally different, a variable translation will have to take place. Before running the model in a BlackBox, the variables it needs are translated from their BLISS names, to their names inside the model itself. The BlackBox gathers its required variables from the total set of variables formed by **X**, **Y** and **Z** together. To rout the correct set of variables to the BlackBox, three arrays are defined and stored in the 'STRUCTURE' array: $\mathbf{X}_{\mathbf{in}}$.EXAMPLE, $\mathbf{Y}_{\mathbf{in}}$.EXAMPLE and $\mathbf{Z}_{\mathbf{in}}$.EXAMPLE.

They contain the names of the variables required by the BlackBox 'EXAMPLE'. Figure 5-10 shows the steps taken by the framework to go from framework variables **X**, **Y** and **Z** to variables that can be fed to the model inside the BlackBox. Analogously, an array $\mathbf{Y}_{\mathbf{out}}$.EXAMPLE defines the required output variables, which are extracted from the BlackBox model and written to **Y**.

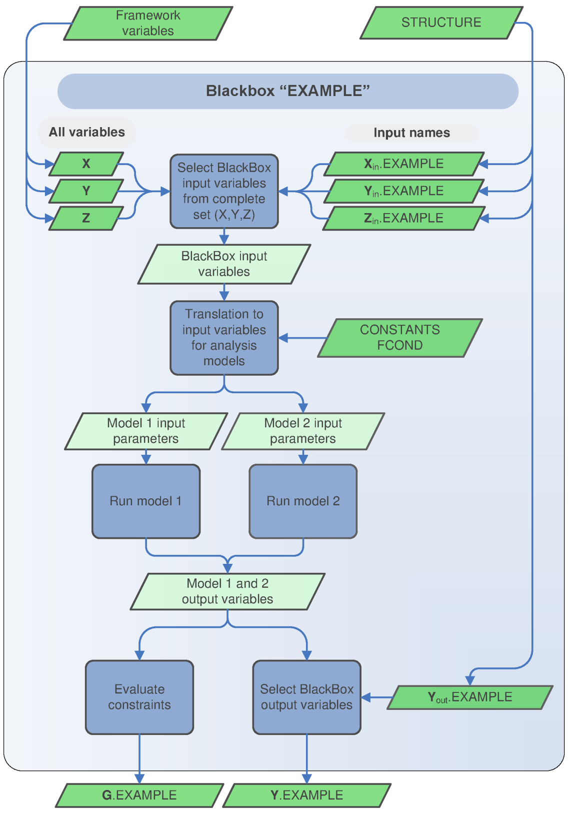


Figure 5-10: Overview of the variable translation process for a single BlackBox 'EXAMPLE'

Regardless of the disciplinary analysis model(s) inside, every BlackBox is structured the same way. Finally, to make the program more user-friendly, a user interface was implemented so that the user can:

- Start a new MDO problem from scratch
- Choose which problem he/she wants to run
- Change the type and value of design variables
- Set design variables to a fixed value ('freeze')
- Modify optimization and sensitivity analysis settings

A manual of the BLISS framework and supporting functions can be found in appendix D.. The next sections each will explain the major parts in the framework: initialization, system analysis, sensitivity analysis and optimization.

5-4 Initialization

Before a BLISS cycle can be run, a problem must first be initialized. A problem can be initialized in two ways. It can either be done by loading a previously created problem definition, or by running a file in which all initialization parameters are hardcoded.

BLISS initialization will now be explained by looking into the first case, where all problem data are hardcoded to an m-file by the user. It is assumed, that the user has the necessary analysis models available and wrapped in BlackBoxes, and that he/she has identified their input and output parameters as **X**, **Y** or **Z** variables[3].

First of all, the user is prompted to insert constant parameters, stored in 'CONSTANTS', BLISS problem settings stored in 'SETTINGS', and the discipline names. Then, for each BlackBox he/she has to insert the desired names of the **X** and **Z** variables - BlackBox inputs - and BlackBox output **Y** variables into the arrays $\mathbf{X_{in}}$, $\mathbf{Y_{out}}$ and $\mathbf{Z_{in}}$, respectively.

The initialization program subsequently checks if each X-name is used only once and each Z-name at least twice. Then, the user gets the opportunity to create couplings between the BlackBoxes. Therein he/she can only choose from the variables in $\mathbf{Y_{out}}$, so that it will immediately appear if a BlackBox is not providing the outputs it should. The program automatically creates a Design Structure Matrix from $\mathbf{X_{in}}$, $\mathbf{Y_{in}}$ and $\mathbf{Z_{in}}$ and displays it to the user. Figure 5-11 shows in a diagram how the problem data is entered.

If the user is satisfied with the problem structure, he/she can assign values to **X**, **Y** and **Z**, an SA is conducted. If the SA converges, the problem variables are saved and problem initialization is concluded.

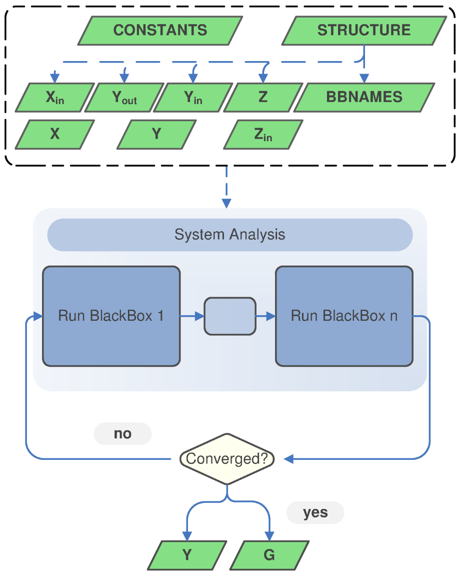


Figure 5-12: BLISS framework system analysis

The initialization routine checks if the problem structure is well-defined: all coupling variables are provided by the discipline models, all incoming design variables (**X**,**Z**) are named such that all models are able to run without error, and each design variable can hit its respective upper or lower bound again without model error. Both as a final check, and to find the initial state vector **Y**, an SA is conducted. If it runs and converges properly, the program is well defined. If it doesn't, control is handed to the user, who has to fix the source of the error before the problem can advance to the BLISS

[3]How this is done, is described in detail in chapter 7 for the Blended Wing-Body

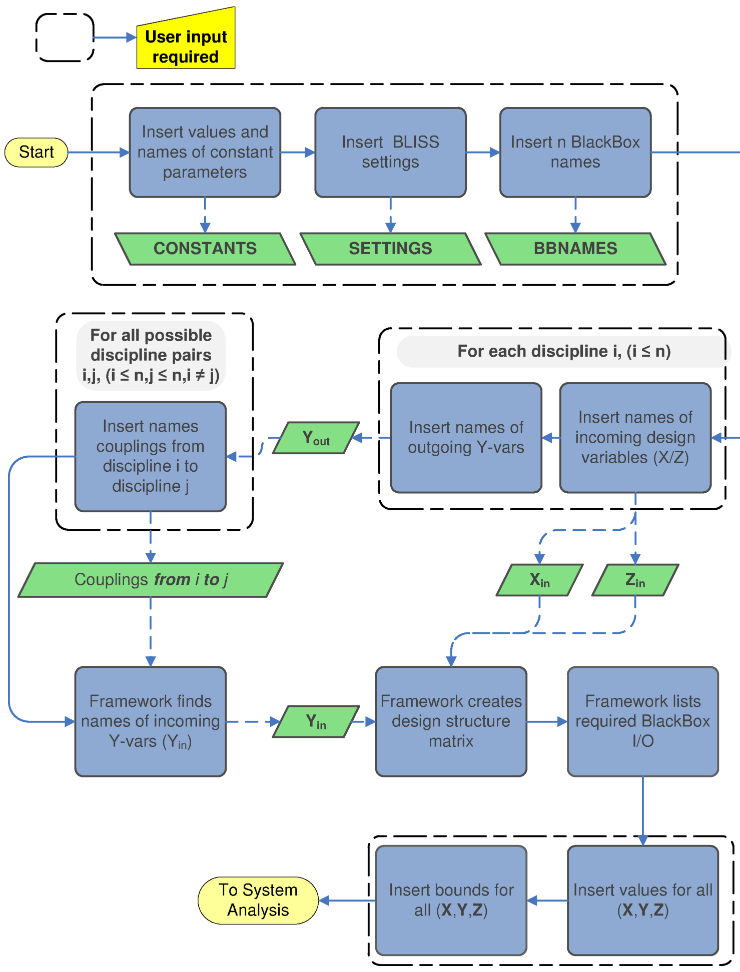

User input required
Start
Insert values and names of constant parameters
Insert BLISS settings
Insert n BlackBox names
CONSTANTS
SETTINGS
BBNAMES
For all possible discipline pairs i,j, (i ≤ n,j ≤ n,i ≠ j)
Insert names couplings from discipline i to discipline j
For each discipline i, (i ≤ n)
Insert names of outgoing Y-vars
Insert names of incoming design variables (X/Z)
Yout
Couplings from i to j
Xin
Zin
Framework finds names of incoming Y-vars (Yin)
Yin
Framework creates design structure matrix
Framework lists required BlackBox I/O
To System Analysis
Insert bounds for all (X,Y,Z)
Insert values for all (X,Y,Z)

Figure 5-11: Initialization of a BLISS problem from scratch

routine itself. This approach forces the user to thoroughly review the problem formulation. Also, the automatically generated DSM will help him/her to overview the design problem and quickly modify or redefine the problem structure if desired.

Summarizing the above, the input for a BLISS problem definition must consist of:

- BlackBox names
- Corresponding discipline models
- Variable names (STRUCTURE, containing $\mathbf{X}_{\mathbf{in}}$, $\mathbf{Y}_{\mathbf{out}}$, $\mathbf{Z}_{\mathbf{in}}$)
- Variable values (**X**,**Y**,**Z**)
- Couplings between disciplines
- Constants, names and values (CONSTANTS)
- BLISS framework settings (SETTINGS)
- Side constraints ($\mathbf{X}_{\mathbf{bound}}$,$\mathbf{Z}_{\mathbf{bound}}$)

From the input, the following items are generated automatically:

- Y_{in}, the array of input variable names for each BB
- Design Structure Matrix (DSM)

5-5 System Analysis

The system analysis is conducted as in 5-1. The user can specify the order in which the disciplines should be iterated. Using the problem DSM, the user can identify for which discipline ordering the amount of feedback coupling is minimal, decreasing the time-to-run for the System Analyzer. Furthermore, the System Analyzer loops until the convergence criterion is met with a user-defined 'TolY'. The SA has converged after n loops if:

$$\max\ (Y_n - Y_{n-1}) \leq (\text{TolY}) \max\ (Y_{n-1}) \tag{5-19}$$

5-6 Sensitivity analysis

The BBSA is the central component of the BLISS framework. According to BLISS theory, the output-to-input sensitivity of each BlackBox is analyzed separately, and this analysis can be done in parallel. The sensitivity is essentially a gradient, and the simplest way to determine it is using the 1_{st} order forward differencing scheme. If a generic BlackBox is written as a function BB of **X**, **Y** and **Z**, the forward differencing scheme for the gradient of **Y** to $\mathbf{X}_i$ with stepsize h becomes equation 5-21, with $\mathbf{X}_i$ being an element of **X**.

$$\begin{aligned}
\Delta X_i &= hO\left(\mathbf{X}\right) \\
\mathbf{X}_{i,1} &= X_{i,0} + \Delta X_i \\
\mathbf{Y}_0 &= BB\left(\mathbf{X_0}, \mathbf{Y_0}, \mathbf{Z}\right) \\
\mathbf{Y}_1 &= BB\left(\mathbf{X_1}, \mathbf{Y_0}, \mathbf{Z}\right) \\
\frac{d\mathbf{Y}}{d\mathbf{X}_i} &= \frac{\mathbf{Y}_1 - \mathbf{Y}_0}{\Delta X_i} + O\left(h\right)
\end{aligned} \tag{5-20}$$

To account for the different magnitudes the design variables may have, the generic step size 'h' is multiplied with the order of magnitude of the mean value of the variable $O\left(\mathbf{X}\right)$.

The optimization is based on data from the BBSA, so if the BBSA linearized sensitivities come closer to their actual values, the optimizers give better results and the number of design cycles required decreases. How well the BBSA performs depends on the numeric scheme used for linearization and the shape of the function describing the actual influence changing a variable has on another one. Figure 5-13 shows an example determining the gradient of the objective 'obj' w.r.t. an arbitrary design variable 'q'. In this example, forward differencing (figure **??**) may lead to problems because the approximation error in the objective increases if the optimizer chooses to move against the direction in which the gradient was taken (so a negative step Δq_{optim}). The approximation error may be less dependent on the sign of Δq_{optim} when a central differencing scheme is employed. In chapter 6 it is tested whether this behaviour actually occurs.

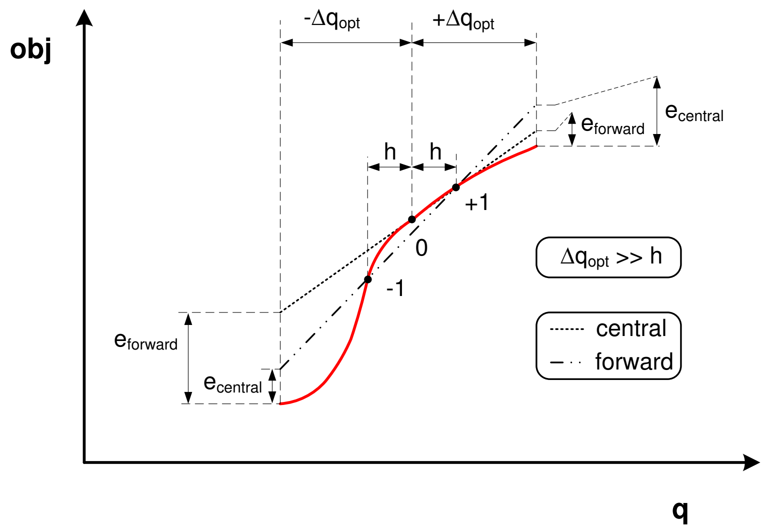


Figure 5-13: Sensitivity of objective to q with forward differencing

That is why in this research, the 2^{nd} order central differencing method is implemented. In the BLISS BBSA, this method is formulated analogously to 5-21 as:

$$\begin{aligned}
\Delta X_i &= hO(\mathbf{X}) \\
X_{i,1} &= X_{i,0} + \Delta X_i \\
X_{i,-1} &= X_{i,0} - \Delta X_i \\
\mathbf{Y_0} &= BB(\mathbf{X_0}, \mathbf{Y_0}, \mathbf{Z}) \\
\mathbf{Y_{-1}} &= BB(\mathbf{X_{-1}}, \mathbf{Y_0}, \mathbf{Z}) \\
\mathbf{Y_1} &= BB(\mathbf{X_1}, \mathbf{Y_0}, \mathbf{Z}) \\
\frac{d\mathbf{Y}}{dX_i} &= \frac{\mathbf{Y_{-1}} + \mathbf{Y_1}}{2\Delta X_i} + O\left(h^2\right)
\end{aligned} \tag{5-21}$$

For the gradient of **Y** to X_i. Other gradients are calculated in an analogous way. Higher order methods are not considered useful, because anyway the optimization problem itself is linearized to 1^{st} order. The time needed to run the complete BBSA depends on the number of coupling variables and the runtime of each BlackBox.

5-7 Optimizers

Both the system and the BlackBox optimizers use the Matlab fmincon function, which is extensively described in the Matlab help function. Using Sequential Quadratic Programming (SQP), fmincon is able to solve a continuous, nonlinear optimization problem that is constrained by inequality, equality and side constraints. To determine when the optimization should conclude, the user has a myriad of settings at his disposal, of which these are the principal ones:

- Termination tolerance on maximum constraint violation ('TolCon')
- Idem, change in design vector ('TolX')
- Idem, change in objective function value ('TolFun')

- Choice of SQP algorithm

Depending on these settings and the optimization problem at hand, the optimizer can finish the optimization in four possible ways:

1. A feasible, but not necessarily optimum point was found, optimization was terminated because 'TolX' was met
2. The optimizer converged to a local minimum
3. No feasible point was found
4. No convergence, but cyclic behaviour up to iteration limit

The most desirable termination scenario is of course the local minimum. If no feasible point is found, at least the optimizer will have attempted to move to the feasible region. The first scenario can however cause problems, because in some cases the optimizer meets 'TolX' directly at the first iteration and decides not to implement any change in design vector. If this happens in multiple consecutive BLISS cycles, BLISS will converge to a point that is not the optimum. This problem is discussed further in section 6-4. If the optimizer halts because it reaches the maximum number of iterations, it has not converged by meeting one of the termination criteria. Therefore, this limit is not specified in BLISS.

The move limits are implemented as percentages of the variable range, where X and Z may have a different move limit Δ_{max}. This is done to account for the different orders of magnitudes different variables or elements of tham can have. The allowed range of a variable is defined as the difference between its upper and lower bound $\mathbf{X}_{min}$ and $\mathbf{X}_{max}$, respectively. Now, the move limits can be formulated as:

$$\Delta\mathbf{X}^{-} \leq \Delta\mathbf{X} \leq \Delta\mathbf{X}^{+} \tag{5-22}$$

where

$$\begin{aligned} \Delta\mathbf{X}^{+} &= \mathbf{X} + \Delta_{max} abs\left(\mathbf{X}_{max} - \mathbf{X}_{min}\right) \\ \Delta\mathbf{X}^{-} &= \mathbf{X} - \Delta_{max} abs\left(\mathbf{X}_{max} - \mathbf{X}_{min}\right) \end{aligned} \tag{5-23}$$

Besides the settings in table **??** there is a large number of settings that is inherited from fmincon itself. Their definitions are found in the Matlab manual.

To prevent the optimizer from focusing on constraints involving large numbers (e.g. on range), all constraints are normalized. Equation 5-24 shows a generic normalized constraint G_i on a parameter l with maximum allowed value $l_{i,max}$.

$$G_i = \frac{(l_i - l_{i,max})}{l_{i,max}} \leq 0 \tag{5-24}$$

An optimizer like fmincon requires constraint values at each iteration to determine feasibility and decide which design vector change to apply to go to either a more feasible or a more optimum design point. An implementation issue arises here: calculating constraint values at during optimizer iteration requires running the appropriate model once. For the BlackBox optimizer, this is the discipline model at hand. For the System optimizer, an entire system analysis is required. Expecially the latter will consume an excessive amount of time, since it needs all BlackBoxes to run multiple times to converge. As the BlackBox models increase in complexity, providing constraint values is becoming infeasible in terms of computation time.

If the constraint are linearized like in sections 5-1-4 and 5-1-5 however, the optimizers don't require BB evaluations anymore. Then, running the optimization algorithm only takes a fraction of the time required when using actual constraint values. On the other hand, if the changes in the design vector proposed by the optimizer become larger, also the deviations of the linearized constraints with respect to the actual ones will increase. It may appear at the end of a cycle that the design has unsatisfied actual constraints, while the optimizers satisfied all linearized constraints. Because constraint values are normalized, they can be used to define a scalar measure of global constraint violation. Equation **??** shows the definition of this constraint violation measure G_{mean} with m constraints.

$$\begin{aligned} G_{pos} &= G\,(G > 0) \\ G_{mean} &= \frac{\sum G_{pos}}{n} \end{aligned} \tag{5-25}$$

where *n* is the number of unsatisfied constraints. In J. Sobieski et al. (1998), it is claimed that BLISS will first attempt to decrease constraint violation when starting from an infeasible point. With **??**, this claim could be verified for the validation cases of chapter 6.

Chapter 6

Validation Cases

The BLISS formulation of the Blended Wing-Body design problem is a large problem in terms of optimization. Therefore, the BLISS framework is tested on smaller problems first to see if it can find an (optimal) solution. Also, before Blended Wing-Body optimization is attempted, BLISS is set to work in a simplified aircraft design case to test the structural and aerodynamic models that will be used in the Blended Wing-Body formulation (see section 7-4).

To validate the BLISS framework that was constructed for the Blended Wing-Body , it was used on three increasingly complex problems:

- A sample problem with a known analytical solution presented in Perez et al. (2004) (section 6-2)
- The NASA Supersonic Bussiness Jet range optimization problem with known numerical solution (section 6-3)
- The L/D optimization of a simplified Boeing 747 wing, which is comapred to the actual 747 wing (section 6-4)

Besides *validation*, the sample problem of 6-2 is also used for framework *verification*, checking whether the BLISS framework implementation is mathematically equal to solving the problem with analytically obtained sensitivities and a single fmincon optimizer To be able to signal trends and possible 'best practices' across the testcases, each of the above problems is subjected to the same validation process and tests, which are discussed in section 6-1.

6-1 Validation Approach

In any optimization framework, there exist a large number of parameters that can be tuned to the user's needs. Often, these parameters have a strong effect on the final design and are crucial for finding it. Besides the obvious question if the BLISS framework is able to find a solution, the framework validation involves the following questions:

The questions above are answered for each validation case. The changes mentioned in the last item of table 6-1 include:

- Different move limits
- Different fmincon settings

Table 6-1: Validation tests

Question	Test
Optimum found?	If optimum is known, is it reached?⊠⊠"0309⊠
Substantial objective improvement?	Improvement order of magnitude - 1% to 10%
Freedom in initial point?	Let initial X and Z approach their side constraints, attempt to solve problem starting from arbitrary points
Best move limits?	Test different move limits for BLISS convergence, problem feasibility and objective improvement
Best BBSA differencing method?	Use 1st order forward or 2nd order central differencing with best move limits and other settings standard
Linearized constraints?	Compare results of linearized vs actual constraints using best BBSA method and movelimits
User intervention between cycles needed?	Manually make and record changes settings between BLISS cycles if the optimum, or at least a feasible pointis not found

- Manual modification of design vector to improve problem feasibility (i.e. drive unsatisfied constraints to zero)

To structure the questions in the above tests, Figure 6-1 shows a flowchart of the entire validation process. The ultimate way of deciding if a validation case confirms that BLISS is a useful addition to the conceptual design process, is the question whether it is able to improve the design, or at least make it feasible, with a total effort required from the user that is reasonable. This effort includes the time spent to create a problem as well as using the framework to solve it. What effort is reasonable is subjective and highly problem-dependent.

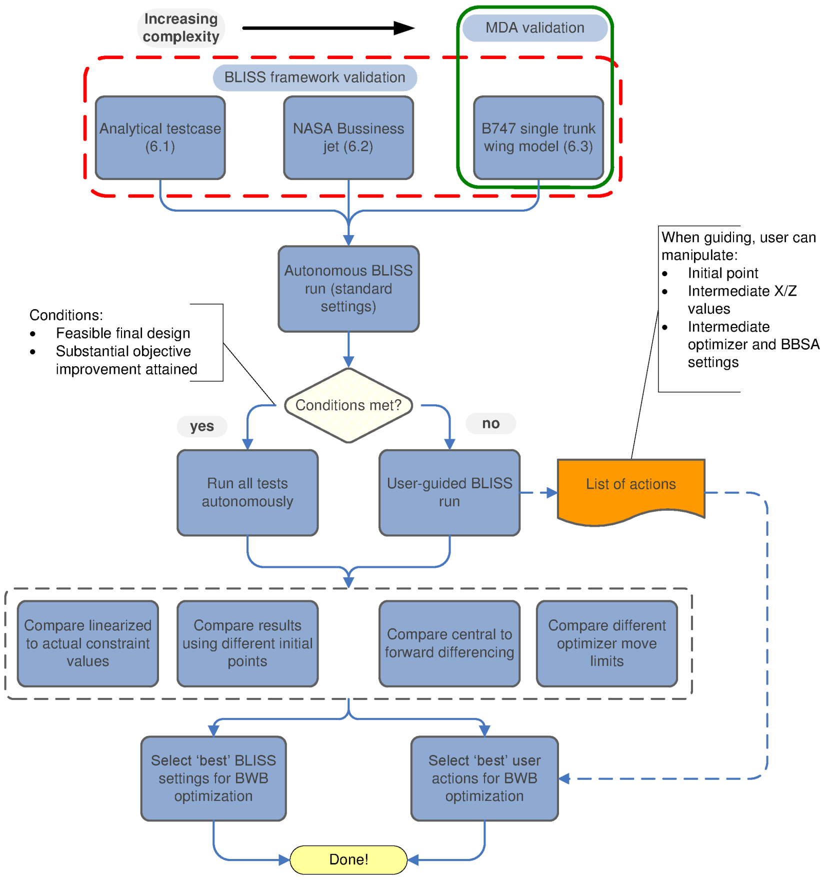

Increasing complexity
MDA validation
BLISS framework validation
Analytical testcase (6.1)
NASA Bussiness jet (6.2)
B747 single trunk wing model (6.3)
Autonomous BLISS run (standard settings)
When guiding, user can manipulate:
Initial point
Intermediate X/Z values
Intermediate optimizer and BBSA settings
Conditions:
Feasible final design
Substantial objective improvement attained
Conditions met?
yes
no
Run all tests autonomously
User-guided BLISS run
List of actions
Compare linearized to actual constraint values
Compare results using different initial points
Compare central to forward differencing
Compare different optimizer move limits
Select 'best' BLISS settings for BWB optimization
Select 'best' user actions for BWB optimization
Done!

Figure 6-1: BLISS framework validation process

6-2 Analytical Sample Problem

As a first test for the framework, a sample problem with uncomplicated structure, analytical functions and a known solution was taken from Perez et al. (2004). The problem has three design variables x_1, x_2 and x_3, two states y_1 and y_2 and an objective Φ. It is stated as follows:

Problem 6-2.1. *Minimize* $\Phi = x_2^2 + x_3 + y_1 + e^{-y_2}$
where:
$y_1 = x_1^2 + x_2 + x_3 - 0.2y_2$
$y_2 = \sqrt{y_1} + x_1 + x_3$

subject to:

$$g_1 = \frac{y_1}{3.16} - 1 \geq 0$$

$$g_2 = 1 - \frac{y_2}{24} \geq 0$$

and:
$-10 \leq x_1 \leq 10$
$0 \leq x_2 \leq 10$
$0 \leq x_3 \leq 10$

The optimum design vector is known to be $\mathbf{x} = [1.9776\ 0\ 0]$, with $\Phi = 3.1834$. In Perez et al. (2004), the problem is started from $\mathbf{x_0} = [1\ 5\ 2]$ and $\mathbf{y_0} = [10\ 4]$, which will also be used here as an initial point for optimization.

6-2-1 Problem Structures

To be run in BLISS, the variables of this problem need to be divided in **X**-, **Y**- and **Z**-variables and the objective. The two BlackBoxes containing Y1 and Y2 are called 'TEST1' and 'TEST2', respectively. The BlackBox 'TEST1' also calculates Y3, which is the objective value. X_2 has become an X-variable, since it is used solely in 'TEST1' for calculating Y_1 and Y_3 This value is then passed onto a separate BlackBox 'OBJ', which contains no calculations whatsoever. Later on, the 'OBJ' BlackBox is used in other problems to be able to easily change the objective functions and review its sensitivities w.r.t its different components.

This results in the following design structure, labeled 'Reference':

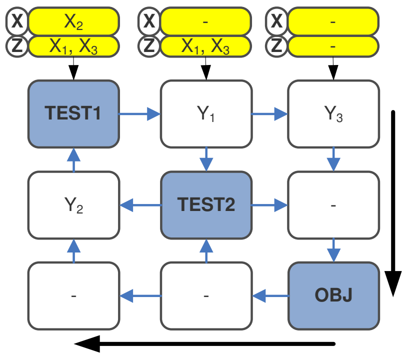


Figure 6-2: Analytical sample problem 'Integrated' DSM

Of course, this is not the only way to structure the problem. To see what is the effect of the problem structure in the BLISS framework, to other structures are evaluated as well. The first possibility is to calculate the objective value in 'OBJ'. This calculation requires X_2, which needs to be passed on through 'TEST1' to 'OBJ' as the variable X_{2s}. This alternative structure, labeled 'Constraint' is equivalent to Figure 6-3.

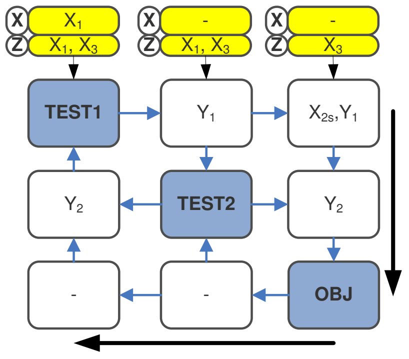


Figure 6-3: Analytical sample problem 'Constraint' DSM

Finally, the two structures can be combined by passing through X_2, then calculating the objective value in a new BlackBox 'TEST3' and passing it to a BlackBox 'OBJ' (see Figure 6-4). This structure is refferred to as 'Decoupled' from now on.

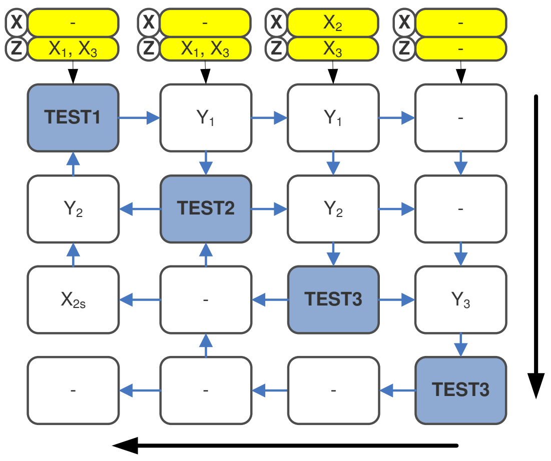
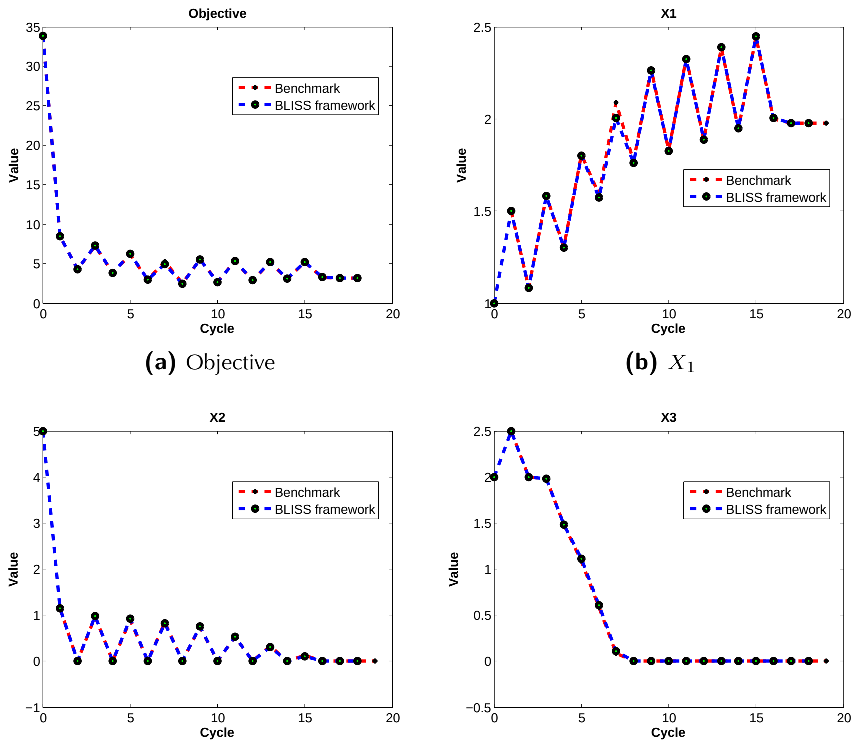


Figure 6-4: Analytical sample problem 'Decoupled' DSM

6-2-2 Results

The Perez et al. (2004) problem was implemented in the BLISS MDO framework, and the BlackBoxes and accompanying structure were tested at the known optimum. An SA showed that at the optimum, the objective and state values corresponded with literature, showing that the BlackBoxes were correctly implemented. To check whether the calculation of the objective and constraint sensitivities was correct, the framework results for the 'Reference' problem were compared to a benchmark solution that employed analytical sensitivities using the same move limits, problem structure and initial point. If the above is true, these two routines are mathematically identical and should produce the same path towards the optimum. Figure 6-5 shows that this is indeed the case. The absolute move limit for all **X** and **Z** was 0.5.



(a) Objective **(b)** X_1

(c) X_2 **(d)** X_3

Figure 6-5: Analytical sample problem: benchmark validation

Varying the move limits showed that the BLISS framework could solve the problem satisfactorily for a range of move limits (see Figure 6-6). It quickly appeared that ΔX could be left unconstrained without convergence issues, so the move limits discussed in this section refer to ΔZ only.

Figure **??** shows that the problem converges at different speeds and to different optima depending on move limit. For a small move limit of 1%[1], the problem converges to an optimum that is slightly higher than the actual one. For large movelimits of 40% and 80%, the objective overshoots its initial value at first, but still converges to the actual optimum. This behaviour is likely to be caused by the approximation error made in linearizing the optimization problem, which grows when larger move limits are allowed. A move limit of 10% gives the most coherent result (see Figure **??**), with a smooth and monotonous objective decrease. This move limit is used for testing BLISS performance for linearized constraints against actual constraint values. Since there are two optimization levels, this yields four options:

[1]Percentage of the difference between maximum and minimum allowed design variable values

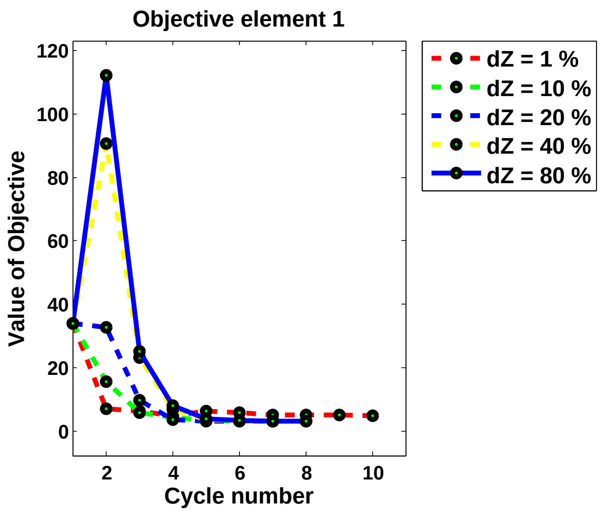


Figure 6-6: Objective vs move limit, 'Reference'

1. All constraints linearized
2. Only System constraints linearized
3. Only BlackBox constraints linearized
4. No constraints linearized

These are all depicted in Figure 6-7, numbered according to the overview above.

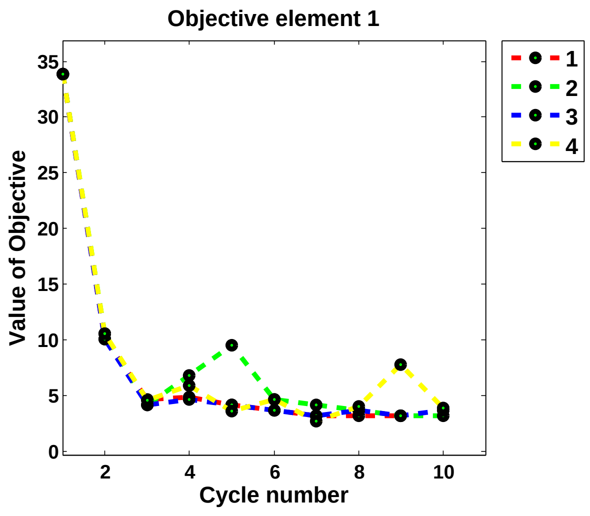
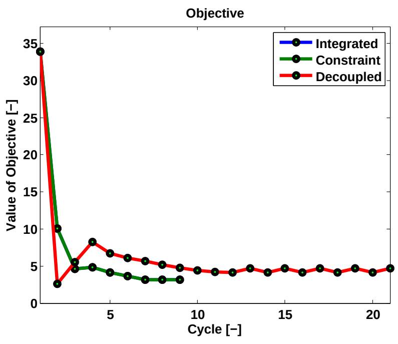


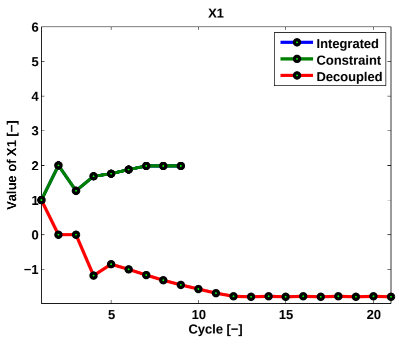
Figure 6-7: Analytical sample problem: constraint modeling methods

From Figure 6-7, all four options lead to the actual optimum in 9-10 cycles. Peaks in the convergence history occur for options '2' and '4', but not in '3', so they are likely to be caused by the use of actual constraint values in the BlackBox optimization. By using only actual values of the constraints within a BlackBox r, information on the sensitivity of the other constraints to ΔX_r is lost, while it isn't for linearized constraints (see equation 5-23). For a real design problem like the Blended Wing-Body design case, the duration of a System Analysis prohibits the use of option '4'.

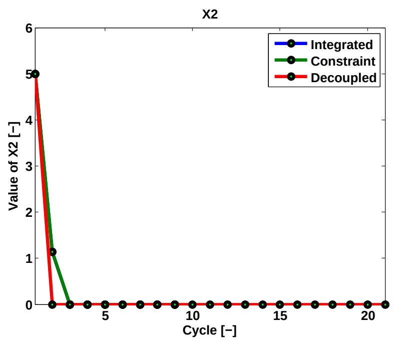
Subsequently, the 10% movelimit is applied to the three different problem structures defined previously. The best (and identical) results are obtained for the 'Integrated' and 'Constraint' formulations, while the decoupled formulation cycles around an optimum higher than the actual one (see Figure 6-8). It is clearly visible from this Figure that the cyclic behaviour is caused by X_3 and the different optimum by X_1, which converges to almost the opposite value for the 'Decoupled' problem.



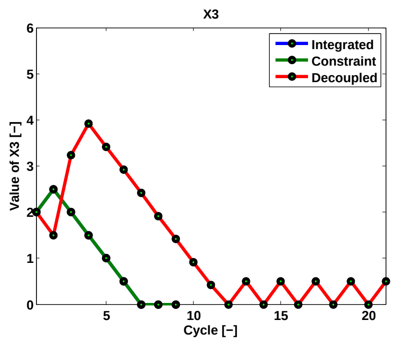
(a) Objective



(b) X_1



(c) X_2



(d) X_3

Figure 6-8: Problem structures comparison

The differences in Figure 6-8 can be explained by figures 6-2 and 6-4.

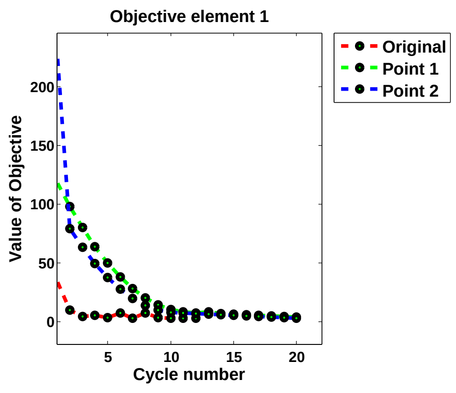
In the 'Decoupled' structure, the BlackBox 'TEST1' requires the **X**-variable X_2, but receives it as the **Y**-variable X_{2s}, which is controlled in 'TEST3'. Meanwhile 'TEST3' is unconstrained, which means that the BlackBox optimizer of 'TEST3' is free to choose a value

Then, with standard settings, the problem was run for one BLISS cycle from the initial point using both 2^{nd} order central and 1^{st} order forward differencing for the BBSA. This did not lead to differences in optimization results, which can be explained from the fact that the problem only consists of smooth functions (see also section 5-6).

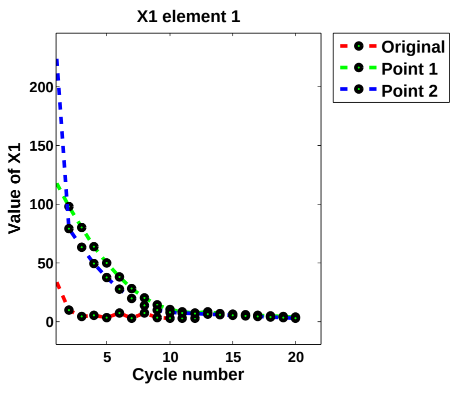
Since forward differencing is faster, the Perez et al. (2004) problem suggests that this technique should be preferred. The last step is to see if BLISS can still arrive at the optimum when varying the initial point. This was tested by varying initial values of the design variables at. The following three points were considered:

1. The orginal initial point, $\mathbf{x} = [1\ 5\ 2]$
2. 'Point 1', (x = $[-10\ 0\ 10]$
3. 'Point 2', (x = $[10\ 10\ 10]$

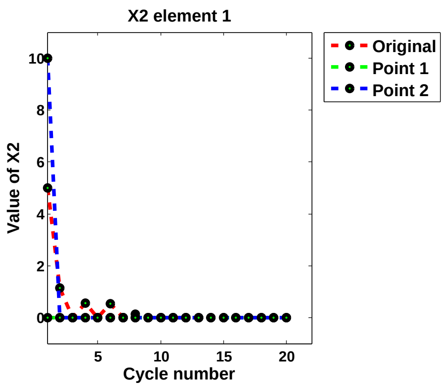
The known optimum was attained in all cases As can be seen from Figure 6-9. For 'point 1' and 'point 1', the convergence history is still smooth, but almost twice as many cycles are required to converge. Convergence speed is limited by the presence of a move limit in Z, so if the initial value of a Z-variable is far away from the optimum one, the move limit will determine the minimum required number of cycles for the optimizer to 'walk' to the optimum.



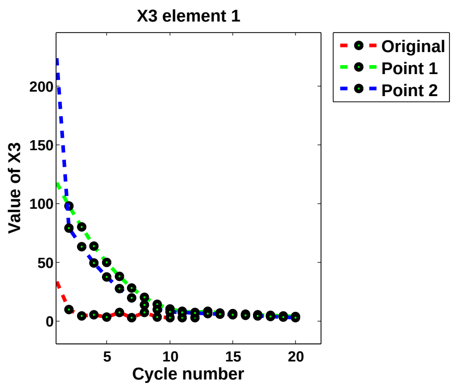
(a) Objective



(b) X_1



(c) X_2



(d) X_3

Figure 6-9: Analytical sample problem: different initial points

After testing the framework using the Perez et al. (2004) problem, table 6-2 summarizes the conclusions that could be drawn.

Question	Result	Remarks
Optimum can be reached?	yes	Move limits > 1%
Substantial objective decrease?	yes	Demonstrated for all tests
Best BBSA gradient estimation method?	Forward 1st order	Equal performance,but faster than central 2nd order
Best move limits (X/Z)?	$1\% \leq (\Delta Z) \leq 100^{+}\%$, ΔX free	Movelimits > 100% possible
User interventions required?	no	
Best optimizer algorithm?	'Active-set'	FMINCON small scale optimization algorithm
Linearized constraints?	Possible, not required	Linearized constraints gives faster runtimes
Robustness in choice initial point?	yes	Convergence shown for three points spread across the design space
Convergence to same optimum for different structures	partial	Unconstrained BB should not have X-variables that are passed onto other BB

Table 6-2: Validation tests: analytical sample problem

6-3 The NASA Business Jet

The supersonic bussiness jet testcase was used by J. Sobieski et al. (1998) to validate the BLISS implementation that is conceived in this source. This implementation was not a generic MDO optimizer, but a code specifically tailored to the business jet. The problem was divided into three disciplines: structures, aerodynamics and propulsion. A fourth discipline, range, was used to provide the objective value. The objective was to maximize the cruise range according to the Breguet equation. The system optimizer utilized linearized constraints according to equation 5-9,while the BlackBox one used actual constraint values. In this thesis, the cyclewise business jet results of J. Sobieski et al. (1998) are compared to those obtained with the BLISS framework, to see if it is possible to completely linearize this problem and still reach the same optimum.

6-3-1 Problem Structure

The NASA Business Jet problem was based on known results from J. Sobieski et al. (1998) and validated using an All-in-One approach, where a single optimizer solved the problem without the two level structure and without making a distinction between local and global design variables. Because J. Sobieski et al. (1998) focuses on the BLISS method itself and not the tools involved, many variables are found by evaluating a polynomial function of other variables S and coefficients A_{ij}, of the form:

$$pf(S) = A_0 + A_i S^T + \frac{1}{2} S A_{ij} S^T \tag{6-1}$$

The coefficients of these polynomials are given in J. Sobieski et al. (1998) and are not elaborated here. In the problem, initially the bussiness jet is cruising at Mach 1.4 and 60000 ft altitude. This result was within 1% of the benchmark value, and the Black Boxes nor the system level of BLISS could realize any further improvement.

The Business Jet problem has Breguet range as its system objective. With its 10 design variables, it is much smaller than the Blended Wing-Body design problem, as known optimization cases in BLISS with a size comparable to that of the Blended Wing-Body were not available. Still, the NASA case can only confirm that the framework is consistently programmed and attains the correct (and known) optimum. Meanwhile, the validation cannot show that the Blended Wing-Body designs it produces are actually optimal, because it depends on problem size whether an MDO strategy can find the actual optimum or not.

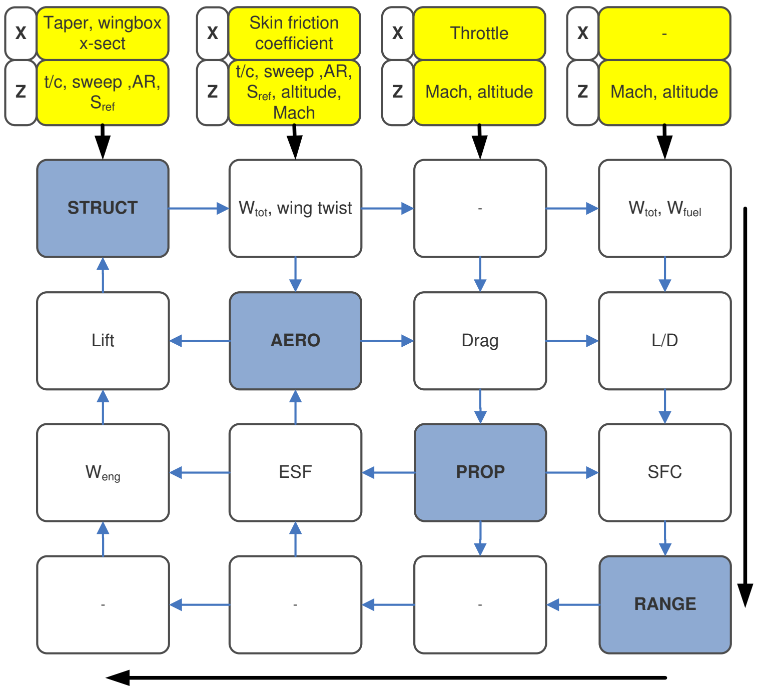


Figure 6-10: NASA Supersonic Bussiness Jet case DSM

The (schematic) structure of the Business Jet problem is given in Figure 6-10. The problem has four BlackBoxes: aerodynamics, structures, propulsion and range. The range BlackBox only computes the system objective range and does not pass this parameter to any other BlackBox.

6-3-2 Results

The BLISS framework intended for the Blended Wing-Body gave satisfactory results without user interference. Both the BLISS formulation of J. Sobieski et al. (1998) and the one used in this thesis needed 6 cycles to arrive at the optimum. The BLISS framework optimum is . Also, there are minor differences in the final design vector and the path towards it. Since these differences are small, the discrepancies are likely to be due to an error in the implementation of one of the polynomial functions. Running the NASA problem using different initial points was not tested, since the variation in initial point is restricted by the polynomial functions (6-1) that form surrogate models for the design state. However, as stated by J. Sobieski et al. (1998), this is an implementation issue which will not occur for problems that use actual analysis models instead of surrogate ones. Apart from initial point variation, all validation tests were repeated. First of all, Figure 6-11 shows the objective development for different move limits.

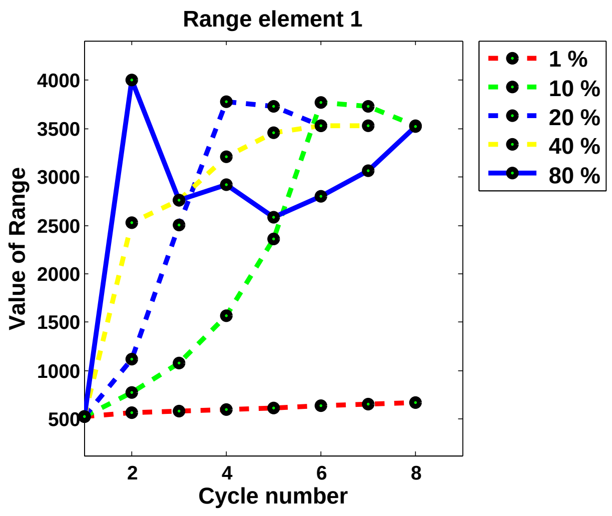


Figure 6-11: NASA results for different move limits

BLISS converges for all move limits, including the 1% run, which needs more than 50 cycles to arrive at the optimum. Values obtained from J. Sobieski et al. (1998) for a move limit of 20% are used as a benchmark in Figure 6-12, where these values are compared to the framework results. The BLISS framework follows a path that closely resembles the one from J. Sobieski et al. (1998), although it converges to a lower optimum range, 3500 instead of 3900 km.

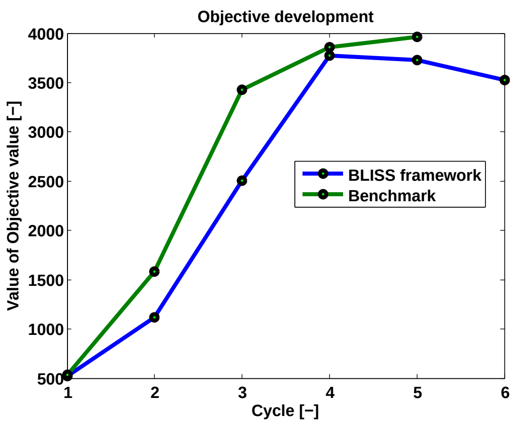


Figure 6-12: NASA Business jet benchmark

J. Sobieski et al. (1998) also claims that the problem showed satisfactory performance up to move limits of 60%, which is confirmed by the current framework even up to move limits of 80%. In J. Sobieski et al. (1998) the actual constraint values were used in the BlackBox optimizer. In the NASA case, a complete constraint linearization is possible without deterioration of the results, as in the problem of section 6-2, all four constraint modeling possibilities produced the same results. For large problems, this implies a significant decrease in computation times. Using central and forward differencing techniques did not produce a difference in results. Again, the NASA uses surrogate polynomials to represent the actual physics behind the problem, which are smooth functions. Table 6-3 summarizes the conclusions for the NASA problem.

Question	Result	Remarks
Optimum can be reached?	yes	Move limits > 1%
Substantial objective decrease?	yes	Demonstrated for all tests
Best BBSA gradient estimation method?	Forward 1st order	Equal performance,but faster than central 2nd order
Best move limits (X/Z)?	1% ≤ (ΔX,ΔZ) ≤ 100+%	
User interventions required?	no	
Best optimizer algorithm?	'Active-set'	FMINCON small scale optimization algorithm
Linearized constraints?	Possible, not required	Equal performance linearized constraints w.r.t. NASA reference solution
Robustness in choice initial point?	N.a.	Could not be tested due to implementation of polynomial functions inside BB's

Table 6-3: Validation tests: NASA Bussiness jet problem

6-4 B747 Wing Model

The NASA and analytical validation cases are problems with a known solution which are used to demonstrate that the BLISS framework can reproduce these results, proving that the routines for e.g. sensitivity analysis and passing data from component to component are correctly implemented. However these cases give no indication of the performance of the framework when it is dealing with an actual aircraft design problem. Especially the Bussiness Jet problem with all its polynomial functions is 'rigged' to give a monotonously converging optimal solution.

To test if the BLISS framework is suited for aircraft optimization, the 'Boeing 747 wing problem' was defined. This problem will function as a benchmark of the BlackBox models that are to be used for the Blended Wing-Body . At the same time, it can be used to quickly verify theories and hypotheses on the functioning of the framework without having to run the costly Blended Wing-Body problem each time. The structure of the B747 problem is of a much smaller complexity than the Blended Wing-Body and plenty of data on the existing design are available. Table 6-4 gives some general idea of the size of the 747 problem.

Vector	Number of elements
X	19
Y	41
Z	2
G	14

Table 6-4: 747 problem size

6-4-1 Problem Structure

The problem is based on Torenbeek (1992), where it was used as a sample problem for his wing weight calculation method. The problem deals with a B747-400 cruising at M = 0.85 at 35000 ft. Besides the information from Torenbeek (1992), actual B747-400 data were taken from *Boeing 747-400 Airplane Characteristics for Airport Planning* (2009) to formulate an initial point for the optimization. Since the cruise fuel consumption of the B747 was unknown, it was calibrated to let the range calculated by the BLISS aerodynamics model match the actual B747 range. The following simplifiying assumptions ensure fast calculation times:

- No **engine model** is included, disciplines are aerodynamics, structures and weight. Engine SFC is calibrated to match the actual 747-400 range with an equal amount of fuel
- The wing consists of **one trunk**, which is identical to the Blended Wing-Body trunk described in 7-6
- **Only the wing** is modeled in the framework, fuselage and tail dimensions, weight and drag coefficient are fixed.
- The **centersection**, that represents the structural connection of the left and right wing through the fuselage, is left out
- No **control surfaces** included, hence no trim drag
- **One flight condition** is analyzed: the beginning of cruise with full fuel and payload

The fuselage drag contribution is modeled using a profile drag estimation of Raymer (2006), approximating the wetted area of the fuselage as a cylinder of equal size. The complete set of B747 constraints and problem parameters are found in appendix **??**. The objective of the 747 BLISS optimization is maximum cruise L/D for a given range of 14000 km. Figure 6-13 gives the DSM of the B747 case.

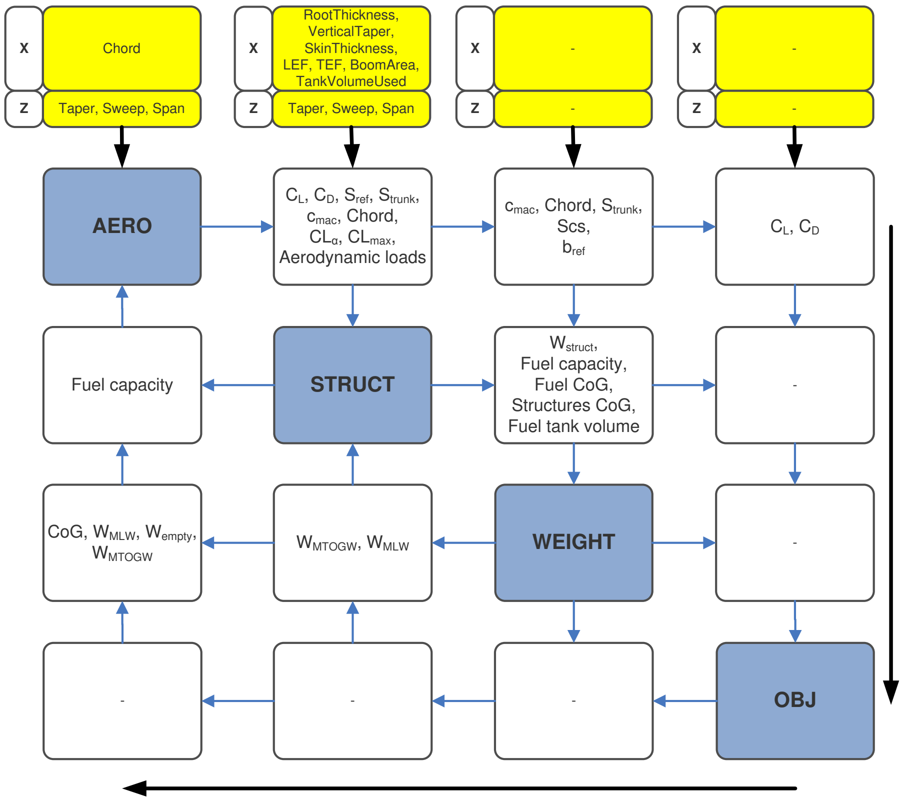
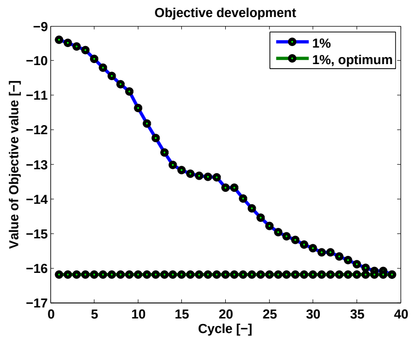

X
Chord
Z
Taper, Sweep, Span
X
RootThickness, VerticalTaper, SkinThickness, LEF, TEF, BoomArea, TankVolumeUsed
Z
Taper, Sweep, Span
X
-
Z
-
X
-
Z
-
AERO
C_L, C_D, S_{ref}, S_{trunk}, c_{mac}, Chord, CL_α, CL_{max}, Aerodynamic loads
c_{mac}, Chord, S_{trunk}, Scs, b_{ref}
C_L, C_D
Fuel capacity
STRUCT
W_{struct}, Fuel capacity, Fuel CoG, Structures CoG, Fuel tank volume
-
CoG, W_{MLW}, W_{empty}, W_{MTOGW}
W_{MTOGW}, W_{MLW}
WEIGHT
-
-
-
-
OBJ

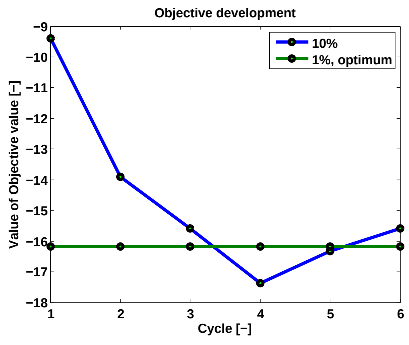
Figure 6-13: Boeing 747 wing DSM

6-4-2 Results

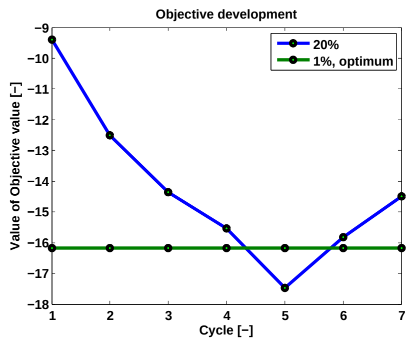
After defining the problem structure, the B747 was subjected to the tests described in section 6-1. It appeared that like the previous cases, the B747 does not require user intervention to get to the (presumed) optimum. Figure 6-14 shows that a larger problem like the B747 has a lower maximum move limit. For small move limits the framework performs satisfactorily, but at a move limit of 40%, the objective convergence history is erratic and does not converge. The larger move limits converge in less cycles compared to the 1% one, but their final objective values are slightly worse (less negative). A small move limit of 1% on the other hand has the smoothest cyclewise convergence. Based on this conclusion, the even larger Blended Wing-Body optimization will also be attempted using small move limits.



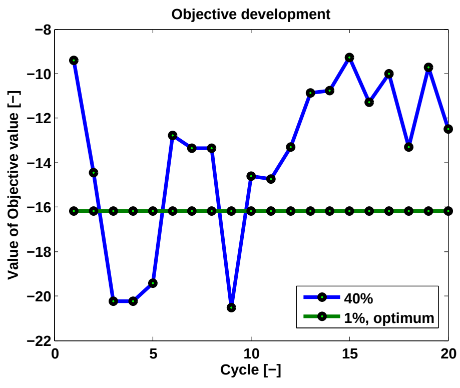
(a) 1%



(b) 10%



(c) 20%



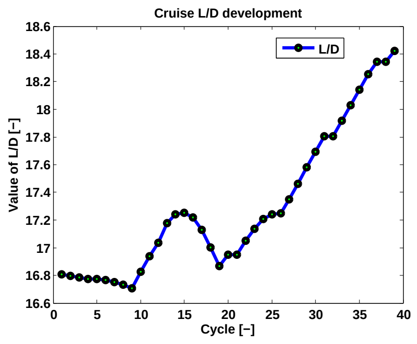
(d) 40% (not converged)

Figure 6-14: B747 results for different move limits

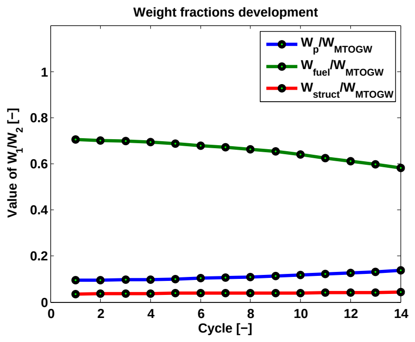
It appears from Figure 6-14 that for a move limit of 1% in **X** and **Z** the framework converged to the best objective value. The B747 problem converges to a feasible and optimal design decreasing the objective with 73% to -16.15. The results overview continues with the display of some common design parameters in Figure 6-15.

The different move limits also have a clear effect on the mean constraint violation (see Figure 6-16). As the move limit gets larger, approximation errors increase the framework has more opportunity to enter the infeasible region, a phenomenon identified already in section 5-1-3. The resulting unsatisfied constraints have to be repaired the next cycle, leading to the peak pattern for the 40% move limit in Figure 6-16.

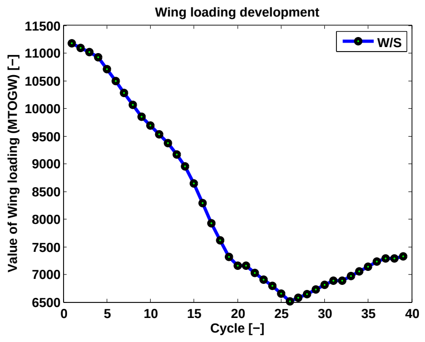
However, it was noted that the runs with 10 and 20% move limits came close to the optimum of the 1% run when they first entered the feasible region (a zero value in Figure 6-16). At that point they had not converged according to the criterion in equation 5-17. Summarizing, larger move limits can



(a) Mission lift-to-drag ratio



(b) Weight fractions



(c) Wing loading at MTOGW

Figure 6-15: Development of common design characteristics

be used to quickly move in the right direction towards the optimum. Close to the optimum itself, the designer should switch to smaller move limits for complete convergence. Otherwise, large move limits are a good setting for doing a 'quick check' on a design.

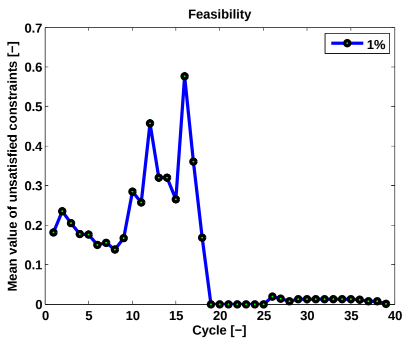
Changing the BBSA stepsize did not affect results for the previous two problems, and neither does it for the B747. BLISS was run for 5 cycles, with BBSA step sizes of 10^{-2}, 10^{-4}, 10^{-6} and 10^{-8}, all yielding the same results. Unlike the other validation cases, the B747 does involve actual analysis behaviour, which is not necessarily smooth.

Before Figure 6-14 could be produced, the problem constraints had to be adapted. Since the 747 problem does not model compressibility effects, wave drag and structural deformation, the BLISS routine was virtually unrestricted in its search for a design with minimum (induced) drag, which has the large span and high aspect ratio of a sailplane. This was repaired by imposing a stiffness constraint of the wing, by modeling it as a simple, cantilever beam of length L, with an end load F and flexural rigidity EI which should have a tip deflection δ_{tip} of less than 10%:

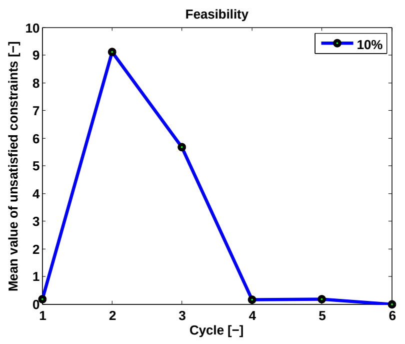
$$\delta = \frac{FL^3}{EI} \leq 0.1b \tag{6-2}$$

With L/D as an objective, the optimizer showed the tendency to increase fuel weight, to increase aircraft weight and thereby the required lift. This produces unrealistic cruise angles of attack, which are not constrained in the BLISS formulation. The large increase in fuel weight required could not have been caused by the 'range' constraint, as its value was below -1 during the complete optimization run. To counter this tendency, the objective was expanded with a penalty for fuel weight (divided by payload weigth to non-dimensionalize), and became:

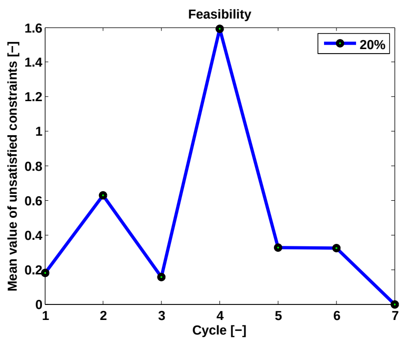
$$\frac{L}{D} + \frac{W_f}{W_p} \tag{6-3}$$



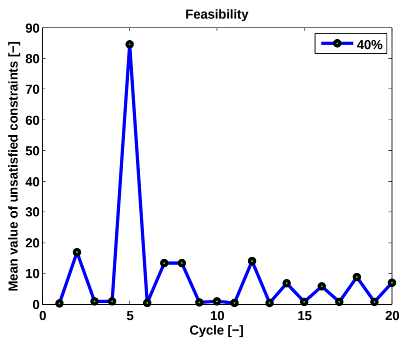
(a) 1%



(b) 10%



(c) 20%



(d) 40% (not converged)

Figure 6-16: Constraint violation histories for the 747

Through this modification, the aforementioned tendency was eliminated and the optimizer started minimizing the required fuel weight as it is supposed to according to design practice. A more 'complete' optimization problem like the Blended Wing-Body is likely to be less susceptible to this kind of phenomena. However, this shows that a designer should not thrust the optimizer results at first sight if it makes counterintuitive modifications to the design. Rather, he/she should check the problem to see what changes in design vector produced the results, and if these changes are related to a physical phenomenon, or to a program flaw, such as the absence of a constraint which is exploited by the optimizer. Figure 6-17 contains top views of the 747 wing before and after the optimization with a 1% move limit. It shows that for the 1% limit the optimization resulted in a slightly more slender wing, increasing L/D.

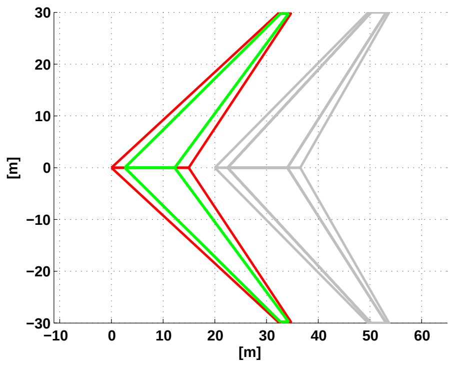
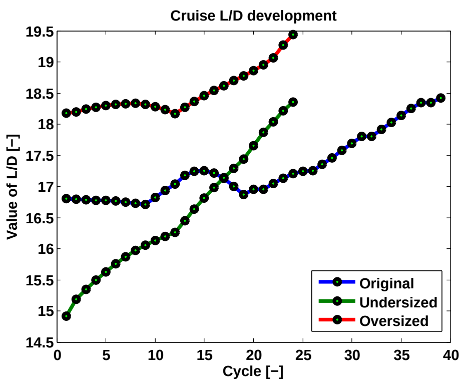


Figure 6-17: Final results, 1% move limit

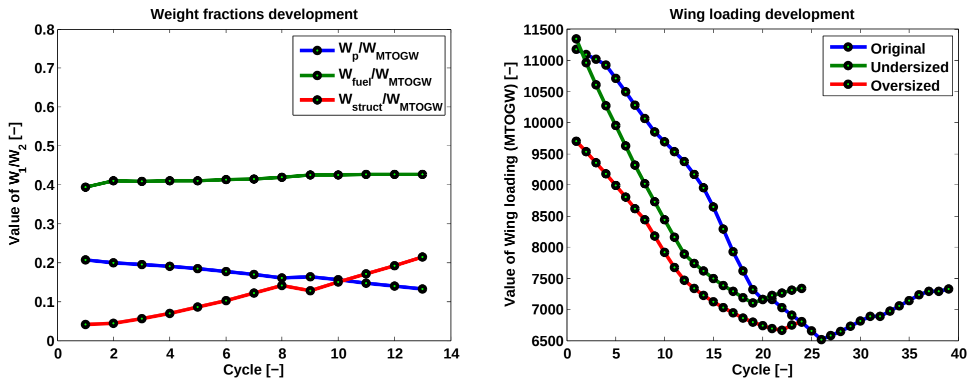
The initial point variation was tested with the following case. Suppose a designer was designing a wing for a 747-like aircraft and uses the current 747 formulation. Instead of using values that resemble the actual 747 quite closely, he/she assumes initial values of **X** and **Z** based on rules of thumb. The influence of the initial design point was tested by solving two optimization problems:

- The initial wing span, wingbox thickness and wing chord were **decreased with 15%** with respect to the original optimization run (the 'Undersized' problem)
- The initial wing span, wingbox thickness and wing chord were **increased with 15%** with respect to the original optimization run (the 'Oversized' problem)

The original optimization run is labeled 'Original' for clarity. BLISS settings were equal in all three cases. For robustness, **X** and **Z** are given movelimits of 1%. Figure 6-18 contains the results of the initial point study, showing that the BLISS framework is still able to converge. The 'Undersized' and 'Original' problems both converge to similar optima. The 'Oversized Problem on the otherhand outperforms the other two designs for the given objective, acknowledging the importance of a thorough initial point study before concluding the conceptual design process.



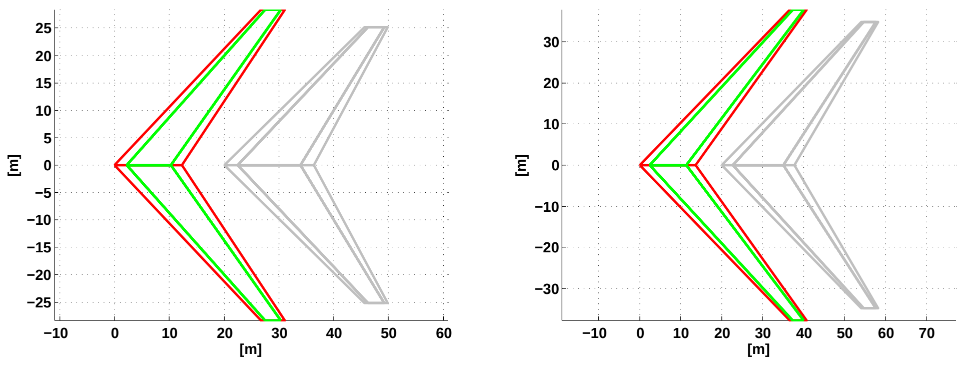
(a) Mission lift-to-drag ratio



(b) Weight fractions **(c)** Wing loading at MTOGW

Figure 6-18: Development of common design characteristics, initial point study

Again, planform plots show that BLISS arrives at more slender wings, which are 'in the ball park' for a real 747 wing (Figure 6-19). According to BLISS, both designs are feasible, but one can doubt if a wing as slender as the 'Oversized' one is structurally and aerodynamically feasible in practice for a long haul airliner. To see if it is, the BLISS design can be compared to results from higher fidelity methods using equivalent geometry. Such tests demonstrate how BLISS conceptual design results 'translate' to improved or deteriorated performance of the design in the preliminary phase. For this thesis however this is considered beyond its scope.



(a) 'Undersized' run (b) 'Oversized' run

Figure 6-19: Planform changes, 'Oversized' and 'Undersized' runs

A particularily interesting comparison is the one between the BLISS-optimized 747 and the real one (which is an optimized design as well, of course). Table 6-5 shows some design characteristics of the final '1%' B747 design. The initial point was chosen quite arbitrarily. Indeed, as the first column shows, the initial fuel weight was grossly overestimated. The optimized properties of the BLISS model however, and the wing shapes of Figure 6-17 are in reasonable accordance with the actual B747 dimensions and properties.

Table 6-5: Boeing 747 final results

Parameter	Initial	Final	Actual 747	% Difference
W_{MTOGW} (tonnes)	704	387.3	362.9	6.7
W_{OEW}/W_{MTOGW}	0.20	0.44	0.49	-10
L/D	16.8	18.4	16	15
W_{fuel}/W_{MTOGW}	0.70	0.38	0.45	-15
W_{wing}/W_{MTOGW}	0.035	0.14	0.12	17
b	60	59.6	59.6	0.06
c_{root}	16.56	14.94	16.56	-9.0
t_{root}	2.22	2.45	2.22	10

Table 6-5 shows that, with the stiffness constraint and objective modification presented in this section, the BLISS framework is able to produce results that follow actual design trends. The stiffness constraint, which in itself highly simplistic, confirms the statement made in Wakayama & Kroo (1998) on whether to allow large approximations into an MDO problem formulation:

"A chain with missing links is worse than one with weak links"

Finally, table presents the conclusions drawn from the 747 validation case. The 747 was developed with models and BLISS framework readily available (as they were constructed for the Blended Wing-Body at the same time). Consequently, the implementation of the B747 was only a fraction of the time needed for the Blended Wing-Body , because the work of getting the BLISS framework to run was completed for the larger part. Although in size the 747 problem is incomparably smaller than the Blended Wing-Body one, still, this feature shows the value of using a generic framework.

Question	Result	Remarks
(Local) optimum can be reached?	yes	Different move limits yield different optima
Substantial objective decrease?	yes	Demonstrated for all tests
Best BBSA gradient estimation method?	Forward 1st order	Equal performance,but faster than central 2nd order
Best move limits (X/Z)?	(ΔX,ΔZ)= 1%	Smoothest convergence history, highest optimum
User interventions required?	no	
Best optimizer algorithm?	‘interior-point’	FMINCON large scale optimization algorithm
Linearized constraints?	Don’t use nonlinear constraints	Longer computation times and no optimization benefit for nonlinear constraints
Robustness in choice initial point?	Partial	Other initial points can deliver similar designs, but local optima exist. Initial point sweep required.

Table 6-6: Validation tests: B747 wing design problem

Chapter 7

Design Problem Formulation and Integration

A key element in MDO is the formulation of the optimization problem. As is generally known, the results of an optimization run depend entirely on how the problem was formulated (see for instance Weck et al. (2007)). As the so called 'No-free-lunch theorem' states that (Ho & Pepyne (2002)):

> "...a general-purpose, universal optimization strategy is impossible. The only way one strategy can outperform another is if it is specialized to the structure of the specific problem under consideration."

Unfortunately, the consequences of the choices made in the definition phase cannot be exactly predicted before the optimization. Instead, the problem formulation has to be iteratively revised before satisfactory results are obtained. The process described in this chapter corresponds to one such iteration. It is assumed that the Blended Wing-Body configuration is similar to the one appearing most in literature: a highly swept centerbody, fairly conventional outboard wing sections, winglets for yaw control and elevons for roll and pitch control. The Blended Wing-Body design that is one of the goals of this research will be reffered to as 'Blended Wing-Body ' or 'BLISS Blended Wing-Body ' from now on. Since the BLISS Blended Wing-Body uses data from the MOB project, the MOB design is labeled 'MOB Blended Wing-Body '.

7-1 Formulation Process

In the traditional aircraft design process, (see 1-1), the designer can start the design process using established processes of systems engineering such as requirements identification, design options analysis, functional analysis (see Hamann & Tooren (2004)). For an MDO problem, the systems engineering approach is still followed, but additional considerations have to be made with regard to the structuring of the problem as an MDO case and its integration into the BLISS framework. Finally, when the problem structure is ready, it should be validated against a similar case with known results, if such a case is available. According to this reasoning, the BLISS design problem formulation can be split up into three phases, shown by table 7-1. This table also explains which steps in Figure 7-1 are related to which phase.

Table 7-1: Three phases in MDO problem set-up

Phase	Description	Steps in 7-1
Formulation	The design problem itself is formulated using SE techniques	1 to 3
Transformation	The problem is translated to a structure BLISS can analyze and optimize	4 to 6
Validation	the problem MDO structure is compared with a known benchmark case	7 to 9

Both phases consist of a number of sub-processes, which are treated in the remaining sections of this chapter. Figure 7-1 shows the complete problem formulation process and indicates which section deals with which sub-process.

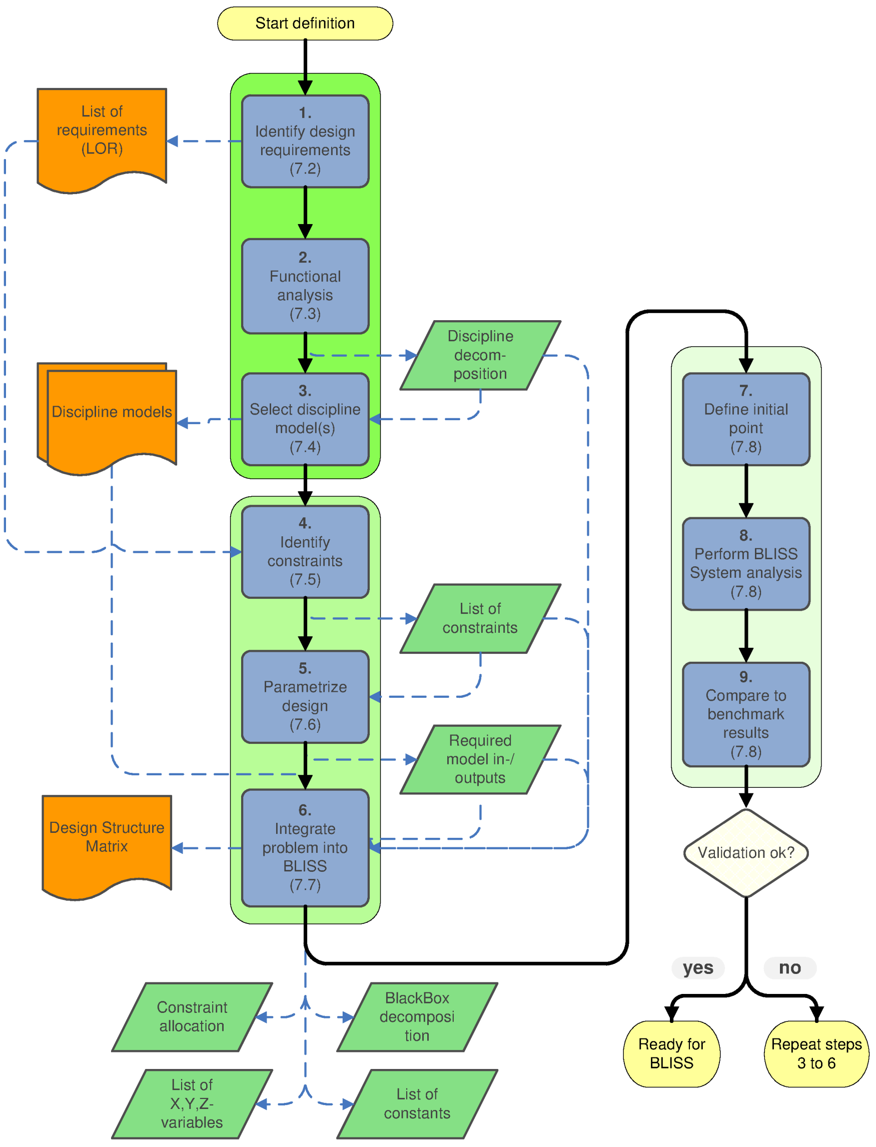

Start definition
1. Identify design requirements (7.2)
List of requirements (LOR)
2. Functional analysis (7.3)
Discipline decom-position
3. Select discipline model(s) (7.4)
Discipline models
4. Identify constraints (7.5)
List of constraints
5. Parametrize design (7.6)
Required model in-/outputs
6. Integrate problem into BLISS (7.7)
Design Structure Matrix
Constraint allocation
BlackBox decomposi tion
List of X,Y,Z-variables
List of constants
7. Define initial point (7.8)
8. Perform BLISS System analysis (7.8)
9. Compare to benchmark results (7.8)
Validation ok?
yes
no
Ready for BLISS
Repeat steps 3 to 6

Figure 7-1: MDO problem formulation process

In the 'formulation' phase, the designer has to find out what design problem he/she is dealing with. The first two steps (requirements identification, functions) correspond to steps in the Systems Engineering process (see Hamann & Tooren (2004)). Subsequently, it is decided into which disciplines the problem should be divided. Then, the design constraints are identified. To model and optimize the Blended Wing-Body in the BLISS framework, the actual aircraft has to be parametrized by its design vector[1]. During parametrization, the discipline model in- and outputs are identified. Already, a large part of the possibilities for parametrizing the Blended Wing-Body is fixed by choosing the analysis models that are to be wrapped inside the BB's. The fidelity of these models has a large impact on the number of parameters for parametrizing the aircraft, and the geometric features this parametrization can represent.

The final goal of the 'transformation' phase is to define a problem structure with BlackBoxes, coupling variables **Y**, global and local design variables (**X** and **Z**) and constants and integrate it in the BLISS framework. The last step in the design problem formulation process is to conduct a system analysis on the aforementioned structure, using an initial point that corresponds as much as possible with a benchmark aircraft of the same type of which the properties are known and verified. In this research, the MOB Blended Wing-Body (see Laban et al. (2002)) is taken as a benchamrk case. In the validation phase, the designer checks if the BLISS system analysis of the problem produce results that are 'in the ball park' for the MOB Blended Wing-Body .

7-2 Requirements Identification

Identifying requirements is a first step in the design process. Since this thesis treats the Blended Wing-Body as a testcase for BLISS, executing the complete requirements identification process is beyond the scope of this project. Instead, it the Blended Wing-Body requirements are chosen to match those found in literature on the BWB (see table 7-2). Besides requirements, design problems need a design objective. In some cases the customer (e.g. airliner) is imposing which objective should be used. If the designer can choose an objective himself, he needs to determine which objective will actually result in the 'best' aircraft for the customer. An aircraft optimized for minimum weight having a large noise contour is still potentially unsuccesful. This calls for a large flexibility in defining and changing the objective function or the weight factors of its components during the design process. As became clear from section 5-1, BLISS provides this flexibility.

Table 7-2: BWB Design Requirements

Requirement	**Value**	**Source**
Satisfy OEI, stall, stresses and T-O certification requirements	see **??**	*FAR* (2009)
Take-Off distance	$\leq$ 3350 m	Liebeck (2004)
Range	7000 n.m.	idem
Payload	450 pax. + luggage	idem
Payload weight	133 tonnes	Smith & Yarf-Abbasi (1999)
Payload volume	1460 m^3	idem
Reserve fuel	5% of block fuel	idem
Cruise Mach	M = 0.83	idem
Cruise altitude	h = 35000 ft	Smith & Yarf-Abbasi (1999)

The maximum T-O distance requirement of Liebeck et al. (1998) is in accordance with existing runways of large airports. The Kaagbaan runway at Schiphol for instance measures 3500 m (*CleanEra*

[1]Parametrization: defining the parameters used to represent a design in the design space

(2010)). With regards to the design mission, the Blended Wing-Body is intended as a long haul airliner and should therefore match performance of other long haul aircraft, such as the B747, A380, B777 and A350 in terms of range and cruise Mach number. Trim drag was identified as a more critical issue for the Blended Wing-Body compared to conventional airliners, so the aerodynamics discipline cannot limit itself to one design cruise condition. To capture L/D decrease due to trim drag, the mission is composed of 5 flight conditions: the aircraft climbs to cruising altitude, flies two cruise legs and descends again to sea level. A single flight condition consists of:

- Freestream Mach number M
- Altitude h
- Momentary fuel fraction (0-1) $\frac{W_{f,mom}}{W_{f,max}}$
- Payload fraction (0-1) W_p
- Flight path angle γ

It is assumed that the distance flown during climb and descent phases is neglegible. The details of the Blended Wing-Body mission flight conditions, stored in the array 'FCOND', are given in table 7-3. The third contidion is used as a mid-cruise reference condition in for instance the calculation of the engine design operating point. The implementation of the BLISS Blended Wing-Body allows for a design mission with any number of flight conditions and any combination of cruise, climb and descent legs.

Table 7-3: Blended Wing-Body Mission flight conditions

Flight condition	Name	h (m)	M	γ (deg)	Fuel fraction	Payload fraction
1	climb	0	0.32	5	1	1
2	cruise	10668	0.83	0	0.9	1
3	cruise (ref)	10668	0.83	0	0.5	1
4	cruise	10668	0.83	0	0.1	1
5	descent	0	0.32	-3	0.05	1

7-3 Functions analysis

Functional analysis determines what functions the to-be-designed product should fullfill. The functional analysis is split up into two parts. First, section 7-3-1 treats the aircraft functions that will be covered in Blended Wing-Body conceptual design. This results in a list of design disciplines that need to be covered in the design problem formulation. Then, section 7-3-2 identifies what role these disciplines should fullfill in the MDO formulation itself: which phenomena, design issues and peculiarities of the Blended Wing-Body need to be modeled by the disciplinary models to arrive at a realistic final result with a reasonable time investment?

7-3-1 Aircraft Functions

To keep the scope of the BWB design case within bounds (due to time constraints), only the flight phase is considered, so pre- and post-flight phases are ignored. In terms of functional analysis, Hamann & Tooren (2004) defines the flight phase as the 'Perform flight operations' function (see Figure 7-2). Of course, each of the functions in this Figure can again be split up into subfunctions.

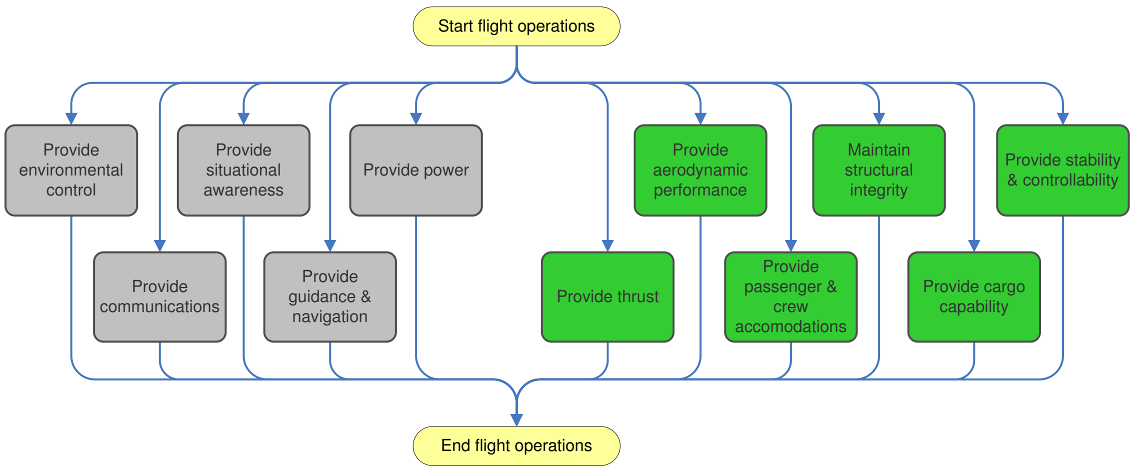

Start flight operations
Provide environmental control
Provide situational awareness
Provide power
Provide aerodynamic performance
Maintain structural integrity
Provide stability & controllability
Provide communications
Provide guidance & navigation
Provide thrust
Provide passenger & crew accomodations
Provide cargo capability
End flight operations

Figure 7-2: The 'Perform flight operations' function

Again, not all of the subfunctions of 'Perform flight operations' given in Figure 7-2 can be modeled and included in the conceptual design phase. The grey subfunctions in Figure 7-2 are considered to be too detailed and of small influence to the overall configuration in the conceptual phase, and will therefore be ignored.

The remaining functions determine the disciplines in which the problem will be decomposed. Each of these functions can be imagined as representing one or more disciplines. Table 7-3 shows how the problem is decomposed into four disciplines based on the functions previously defined relevant. Some disciplines have subparts which can be grouped under one header (such as weight).

Figure 7-3: Determination of relevant disciplines

Function	Discipline	Sub-discipline
Provide stability & control	**Aerodynamics**	Trim & balance
Provide aerodynamic performance		Aircraft planform analysis
Maintain structural integrity	**Structures**	Fuel weight calculation
		Structural weight estimation
		Structural strength & stiffness
Provide thrust	**Propulsion**	Engine performance
		Aircraft performance
Provide cargo capability	**Weight**	Payload/cargo weight
Provide accomodations for passengers & crew		Overall weight

The step of making a subdivision of the problem into disciplines based on aircraft functions is subjective and determined by the designer. Amongst others, this division can be determined by the depth and breadth of the problem, and the runtime and availability of the models. It was decided to to exclude some aspects of the 'Provide Stability Control' function:

- Lateral controllability

- Control system design (fly-by-wire)
- Handling qualities

Lateral controllability is an issue in Blended Wing-Body design that also depends on its the configuration: it can be improved by adding an extra vertical tail on the centerbody (), and by using split drag rudders. Since this thesis focuses on a single Blended Wing-Body configuration, this comparison is left out and the accent is on longitudinal flight. The Blended Wing-Body longitudinal control surfaces are sized for sufficient longitudinal control authority while the lateral ones (winglet rudders) are not (see section 7-6-1). Control system design (fly-by-wire) and handling qualities are design activities not commonly associated with conceptual design (Torenbeek (1982)) and are considered beyond the scope of current work. For reasons above, control is not appearing as a separate discipline in table 7-3.

7-3-2 Discipline Model Roles

Now that the problem is decomposed into disciplines, it is time to decide what design issues and challenges each discipline should tackle in the optimization framework, i.e. in what ways is a discipline model going to indicate that a design will be of good quality (or not) in the real world? If these issues are well-chosen, the Blended Wing-Body representation in the framework exhibits the same general behaviour as the actual aircraft with the same properties would have had. Designers of conventional aircraft use well-established and proven design practices to make sure that this is the case. These practices are largely absent for the Blended Wing-Body , and so the choice of what issues to treat in the conceptual phase is an educated guess, with lofi modeling capabilities and computational time as constraints.

A number of design issues specific to the Blended Wing-Body was identified in section 3-3. From Torenbeek (1982), Raymer (2006) and E. Obert & R. Slingerland (2007), Figure 7-4 was composed, which contains the design aspects deemed most important in conceptual design of aircraft in general and of the Blended Wing-Body in particular. Some design challenges, for example finding the structural c.g. can be grouped under both the 'weight' and 'structures' disciplines. Here, 'structures' was the most convenient choicebecause calculating the structural c.g. requires using the structural geometry, which is not available in the 'weight' discipline.

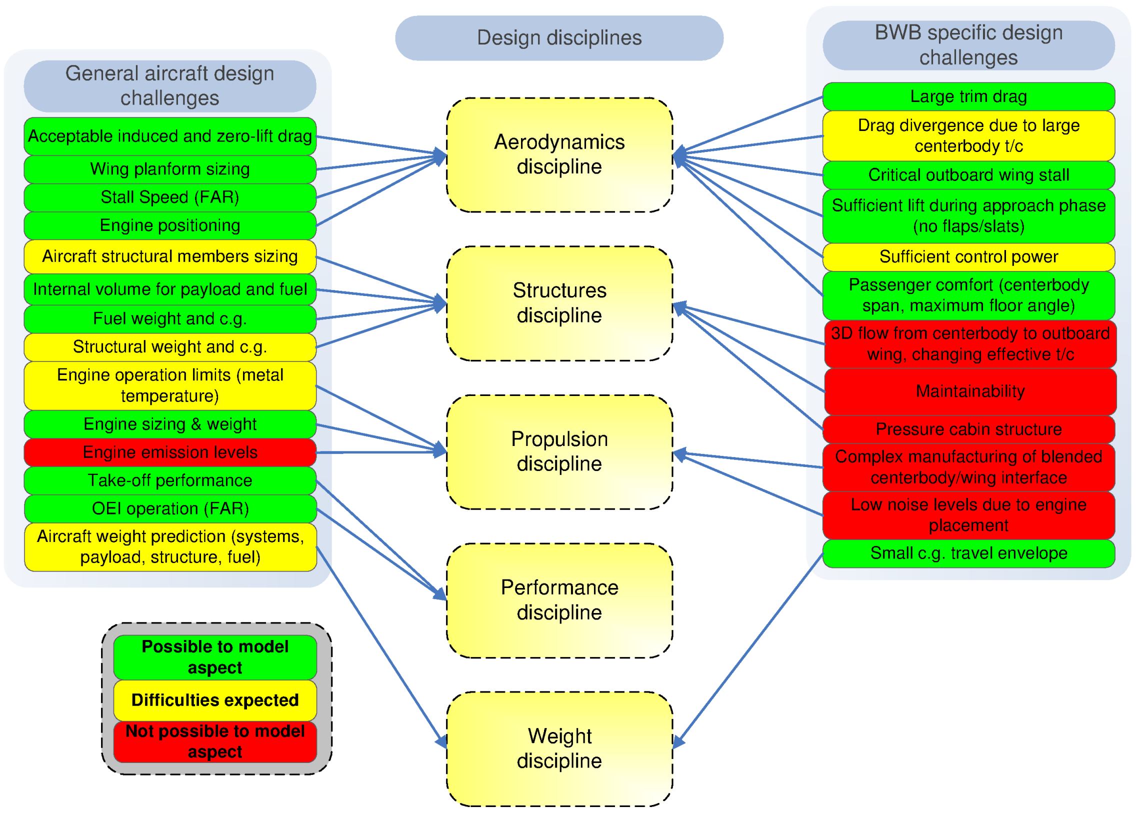

General aircraft design challenges
Acceptable induced and zero-lift drag
Wing planform sizing
Stall Speed (FAR)
Engine positioning
Aircraft structural members sizing
Internal volume for payload and fuel
Fuel weight and c.g.
Structural weight and c.g.
Engine operation limits (metal temperature)
Engine sizing & weight
Engine emission levels
Take-off performance
OEI operation (FAR)
Aircraft weight prediction (systems, payload, structure, fuel)
Design disciplines
Aerodynamics discipline
Structures discipline
Propulsion discipline
Performance discipline
Weight discipline
BWB specific design challenges
Large trim drag
Drag divergence due to large centerbody t/c
Critical outboard wing stall
Sufficient lift during approach phase (no flaps/slats)
Sufficient control power
Passenger comfort (centerbody span, maximum floor angle)
3D flow from centerbody to outboard wing, changing effective t/c
Maintainability
Pressure cabin structure
Complex manufacturing of blended centerbody/wing interface
Low noise levels due to engine placement
Small c.g. travel envelope
Possible to model aspect
Difficulties expected
Not possible to model aspect

Figure 7-4: Design challenges of Blended Wing-Body across design disciplines

By inspection, some of the above issues can be immediately discarded because modeling them in the conceptual phase is considered difficult or impossible. For the pressure cabin, assessing and dimensioning the different concepts (as in Geuskens et al. (2008)) requires high fidelity structures routines, able to handle composite materials. Also 3D flow effects can only be identified with time consuming CFD programs. Finding out wether the Blended Wing-Body could actually be more maintenance-friendly can only be done in the detail design phase, when engineers begin to know the number of parts and their maintainability. It is expected that the remaining design challenges (labeled yellow and green according to complexity) in Figure 7-4 can be modeled in the conceptual design phase.

7-4 Discipline Model Selection

Selection of disciplinary models is a crucial step in MDO and optimization in general, because of the large influence model assumptions have on the optimization process and its results. This section shows the process of selecting models for each discipline. For this, the following four criteria are used:

1. **Speed**: the order of magnitude of the computational speed required for one model run as a BlackBox, preferably on a normal PC[2]
2. **Availability**: the effort that is needed to adapt the model or program it from scratch in such a way that it gives all outputs that are required by other BlackBoxes.
3. **Integrability**: the amount of work needed to integrate the model into the BLISS framework. The model should be easily modifyable to communicate with Matlab if it is written in another language
4. **Parametrization**: the amount of work and variables needed to define the design geometry and characteristics in the model. Generally, parametrization becomes more difficult when a model is able to handle more complex geometries.
5. **Accuracy**: the extent to which the model is capable of handling complex phenomena occuring in reality. Generally, models that include more simplifying assumptions are also less accurate.
6. **Robustness**: in order not to disturb the progress of the BLISS algorithm, a disciplinary model should be able to come up with an answer for a wide range of inputs

For practical reasons, emphasis in the discipline analysis model trade-offs that are to follow is on model availability and speed. Since the focus of this project is on MDO and not on development of the analysis tools themselves, it was decided to make use of available tools as much as possible, while accepting their possible limitations.

7-4-1 Aerodynamics

A wide variety of tools is readily available for the aerodynamic analysis of aircraft. These tools can range from analytical models, such as the Prandtl lifting line theory, to CFD computations from which the complete behaviour of the flow can be found, accounting for viscosity and turbulence. The selected candidates are described below:

- **Vortex-lattice Method**: wing is modeled as a flat plate divided in panels, each containg horseshoe vortex. An example of a VLM implementation in Matlab is Tornado, constructed by Melin (2000)
- **CFD**: CFD features a complete solution of Navier-Stokes equations including viscosity and turbulence with complex 3D geometries
- **Linearized potential flow**: linearized potential flow (Neumann problem), with complex 3D geometries. An example is VSAero.
- **Modified Prandtl Lifting Line** Prandtl lifting line theory, modified to account for high sweep and dihedral wings. The wing is modeled as a single lifting line with a large number of trailing vortices

Table 7-4 lists the candidates for the aerodynamic model of the Blended Wing-Body with their scores on aformenetioned criteria.

[2]The PC used for running BLISS was a 3 GHz, dual core system with 4G RAM

Table 7-4: Aerodynamics model selection

Aerodynamic model selection	Speed	Integrability	Availability	Parametriza-tion	Accuracy	Robustness	Choice
Vortex Lattice Method	+	+	+	+	-	+	✓
Computational Fluid Dynamics	-	-	+	-	+	-	✗
Linearized potential flow	+/-	+/-	+	+/-	+/-	+/-	✗
Modern lifting line theory (Prandtl)	+	+	-	+	-	+	✗

Based on the data in table 7-4, it can be concluded that VLM, and Tornado (Melin (2000)) in particular is the most suitable option. Tornado is readily available, fast and programmed in Matlab . Geometry and flight conditions are easy to define and change. The geometry includes airfoil camber, but not thickness. The mathematical details of Tornado are found in appendix B-1-1 and Melin (2000). Zero-lift (viscous) drag and wetted area are computed using the Eckerts equation in the formulation of Raymer (2006), but the accuracy of this method for use in Tornado has not been tested. The modified Prandtl method of Philips & Snyder (2000) isn't readily available, and computation times of CFD are unsuitable for conceptual design. The main disadvantage of a linearized potential flow routine like VSaero is its integrability: several translation steps are needed to go from the design vector to the actual calculation and vice versa (Koning (2010)). VSAero uses potential flow, coupled with a boundary layer routine that yields a prediction of viscous drag. The choice for the Vortex Lattice Method is supported by Wakayama & Kroo (1998).

One addition to the aerodynamics model is not mentioned in table 7-4 on purpose: the use of Xfoil in the Blended Wing-Body problem formulation. Xfoil is a well-established routine for analyzing airfoils in subsonic flows and is described in appendix B-1-2. During the implementation of the models of the Blended Wing-Body in the BLISS framework, it appeared that Tornado was not well able to provide realistic values for the minimum pressure coefficient at spanwise stations. With this coefficient, it is possible to give a first indication of the minimum pressure coefficient exceeding a critical value (see appendix B-1-2). Hence, it was decided to include Xfoil in the problem formulation to get a minimum pressure coefficient prediction that incorporated airfoil thickness. The foregoing discussion illustrates the iterative character of the problem formulation and implementation process.

7-4-2 Structures

The following structural models are considered candidates:

- **Hollow beam method**: the wing is modeled as a hollow cantilever beam with varying cross-section, (equivalent) thickness is sized to withstand shear and normal stresses
- **Idealized structures method**: the wingbox is replaced by and equivalent, idealized structure of booms (normal stress) and plates (shear stress)
- **Finite Element Method**: the structure is divided into finite elements, model accounts for complex geometries, (plastic) deformations and failure

The hollow beam model was used in the research on the Blended Wing-Body of Carpentieri et al. (2004). In this research, it turned out that this method provided poor estimations of the Blended

Wing-Body structural strength, because standard eam theory assumes that the beam under consideration should be slender, whereas the Blended Wing-Body wing is not (especially at its center section).

FEM methods can be highly accurate when it comes to structural stresses and deformations, but their complicated mesh geometry and computational burden make them unsuitable for this research.

Table 7-5: Structures model selection

Structural model selection	Speed	Integrability	Availability	Parametrization	Accuracy	Robustness	Choice
Hollow beam method	+	+	-	+	-	+	✗
Idealized structures method	+	+	+/-	+	+/-	+	✓
Finite Element Method	-	-	+	-	+	-	✗

Next to a model for structural strength, the structural BlackBox will also require one for structural weight. To model this, literature suggests many empirical methods, that rely on large databases of existing aircraft to derive statistical formulae for the weights of structural components. However one of the key points of the Blended Wing-Body is that this statistical knowledge is absent, because no Blended Wing-Body designs have been realized as yet. Luckily also semi-empirical estimation methods exist, where the structural weight depends on both statistical knowledge and the actual dimensions of the design. One of these methods that has shown is accuracy for conventional wing structural weight was defined by Torenbeek (1982). It is decided to select this method for estimating Blended Wing-Body structural weight, using it separately for each trunk instead of the whole wing at once.

Appendix B-2 explains both the idealized structures and Torenbeek methods.

7-4-3 Propulsion

Also for the propulsion model, a number of viable candidates was identified. There is a vast number of tools able to simulate steady and unsteady behaviour of a generic gasturbine. Below, the candidates for the 'propulsion' BlackBox are summarized:

- **GasTurb Tool**: commercial engine performance prediction software, using 0-D thermodynamic component analysis and Newton Rhapson iteration (see Kurzk (1995)). The user can define which component arrangement he wants.
- **Rubber engine scaling method**: A reference engine with known properties is scaled up or down using statistically established nonlinear scaling factors (see Raymer (2006)).
- **Engine Genetic Optimization Tool**: Matlab tool developed by Kok (2008) based on GasTurb calculation scheme, made suitable for optimization with the genetic algorithm
- **GSP Tool**: an engine perforance prediction tol similar to GasTurb, developed by the NLR (see Visser (1995))

GasTurb and the GSP program developed by the NLR are commercial tools use so-called 0-D gas path analysis, where flow properties are averaged over the flow cross-sectional areas and only considered at a number of stations throughout the engine, at the intakes and exits of the engine components. In both programs, many different engine configurations can be constructed from these components by the user.

Table 7-6: Propulsion model selection

Propulsion model selection	Speed	Integrability	Availability	Parametriza-tion	Accuracy	Robustness	**Choice**
GasTurb tool	+	-	-	+/-	+	+	✗
Rubber engine sizing method	+	+	+	+	-	+	✗
Engine genetic optimization tool	+/-	+	+	+	+	+	✓
GSP tool	+	-	+/-	+/-	+	+	✗

Finally, the Engine genetic optimization tool of Kok (2008) was chosen because of availability, ease of integration and completeness. The tool is written in Matlab . Furthermore, it calculates not only engine performance, but also engine weight and nacelle drag. In Kok (2008) the model was validated satisfactorily with GasTurb, a program which is known to predict performance with reasonable accuracy.

7-4-4 Aircraft Performance

Besides engine performance, a number of other performance parameters play a role in the Blended Wing-Body design. These include Take-Off (T-O) distance, performance with One Engine Inoperative (OEI) and the minimum required climb gradient imposed by airworthiness authorities. Since these parameters are closely related to engine performance, they are also incorporated in the 'propulsion' BlackBox (see section 7-7). Appendix B-3 gives details on both the Kok (2008) engine model and miscellaneous performance calculations. For evaluating these performance constraints on the Blended Wing-Body , Ruijgrok (1996) presents a set of well-known analytical formulas.

7-4-5 Weight Estimation

From the choices made in selecting the other discipline models, it becomes clear that structural and engine weight components are computed in their respective BlackBoxes while payload weight is given. This leaves just the calculation of the systems weight and c.o.g. Although the Blended Wing-Body is an unconventional aircraft, it will contain aircraft systems also found in conventional aircraft. Therefore, in contrast to the structural weight, empirical methods do apply here. Raymer (2006) suggests an easy-to-use method, giving simple formulae for components such as airconditioning, (installed) APU, avionics, hydraulics and more. The aircraft c.o.g. is calculated through a moment balance. The 'weight' Blackbox is described in detail in appendix B.

The aircraft weight is divided into structural weight W_s, systems weight W_{sys}, fuel weight W_f, engine weight W_{eng} and design payload weight $W_{p,des}$:

$$W_{ac} = W_s + W_f + W_{p,des} + W_{sys} + W_e \tag{7-1}$$

Of course, there exist a number of other aircraft weights that are commonly used:

$$\begin{aligned} W_{empty} &= W_s + W_{sys} + W_e \\ W_{MTOGW} &= W_{empty} + W_{f,max} + W_{p,des} \\ W_{MZFW} &= W_{empty} + W_{p,max} \\ W_{MLW} &= 0.7 W_{MTOGW} \end{aligned} \tag{7-2}$$

The MLW to MTOGW ratio of 0.7 was derived from that of the MOB (see Smith & Yarf-Abbasi (1999)).

7-5 Constraints Definition

As in every design optimization problem, the formulation of the Blended Wing-Body design case is accompanied by a set of constraints. It is assumed that these constraints are (in general) non-linear, and that there are only inequality constraints. Basically, constraints can be divided into three categories:

1. Constraints that come with the **design problem** itself
2. Additional constraints to account for **simplifications and assumptions** in the analysis models
3. **Side constraints** applied directly to design variables

Examples of the first category, are airworthiness regulations, requirements set by the customer and other operational constraints. An example is the maximum span which is allowed at airfields. ConstraintS for sufficient internal volume for storing fuel and payload also belongs to this group. The second category includes constraints that are needed to prevent the optimizer from exploiting assumptions and simplifications that are made in the discipline models. The optimization algorithm may seize these opportunities to attain a better objective value (higher or lower), while in reality this could render the design infeasible. Finally, so-called side constraints can be applied directly to the values of design variables, which must stay between upper and/or lower bounds. Examples are minimum and maximum values of the wing twist angle, which should prevent the optimizer from requesting a wing geometry that is impossible to manufacture.

Generally not all constraints of the second category can be identified beforehand, because a priori not all optimization loopholes are known. Hence, the process of finding an adequate set of constraints covering simplifications partially takes place during the optimization. Table **??** gives qualitative overview of the constraints found using the process of Figure **??**. Appendix G gives a complete list of Blended Wing-Body design constraints.

7-6 Aircraft Parametrization

In section 4-1 it is argued that aircraft optimization will only yield realistic results if the parametrization is consistent. This section describes the parametrization of the Blended Wing-Body representing it in the aerodynamic, structures and propulsion disciplines. In the majority of design optimization problems, the way in which the design is represented is largely dependent on in- and outputs of the the analysis models used, as it is generally difficult and time consuming to change them. In turn, the variables used for parametrizing the aircraft also form the set of design variables. Figure 7-5 shows the parametrization process.

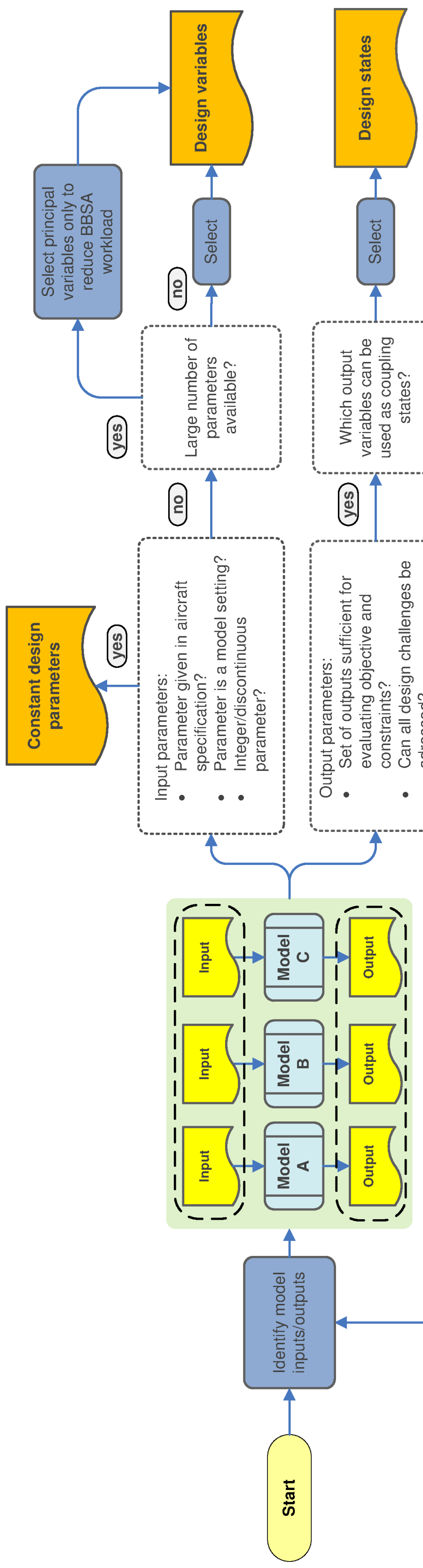

Start
Identify model inputs/outputs
Input
Input
Input
Model A
Model B
Model C
Output
Output
Output
Input parameters:
• Parameter given in aircraft specification?
• Parameter is a model setting?
• Integer/discontinuous parameter?
yes
Constant design parameters
no
Large number of parameters available?
yes
Select principal variables only to reduce BBSA workload
no
Select
Design variables
Output parameters:
• Set of outputs sufficient for evaluating objective and constraints?
• Can all design challenges be adressed?
yes
Which output variables can be used as coupling states?
Select
Design states
no
Can model be modified to provide missing output(s)?
yes
Modify
no
Add another model to MDO problem formulation
Model D

Figure 7-5: MDO parametrization process

First of all, off-the-shelf analysis models already adopt aircraft geometry in a certain way themselves. Xfoil requests an input file with (normalized) airfoil x- and y-coordinates from the user, while Tornado needs an array 'geo' containing number of wing partitions, partition dimensions, number of panels and so on. In the parametrization, the designer defines which quantities are variable and which are constant. During the parametrization process, the folllowing aspects are kept in mind:

1. The complete set of parametrization variables and constant should provide enough information to provide the required inputs to each analysis model
2. To allow gradient calculations, BLISS variables should be continuous. Integer and discontinuous[3] parameters can only be included as constants
3. The complete set of all model outputs should be sufficient for evaluating the system objective and constraints

Selection of the coupling variables from available model output parameters is straightforward when an output of one model is necessary for running another. For instance, to be able to calculate structural stresses, the aerodynamic loads are required from the aerodynamics discipline. On the other hand, in some cases the designer himself can decide which variable is a state and which is a design variable. If the engine is sized with a certain Engine Scaling Factor (see the section 6-3), running the engine model implies calculating the thrust. Another possibility would be to let the optimizer set a 'design thrust' value, and let the engine model size the engine to meet this required thrust. The latter option is considered less favourable, because it implies that the engine model has a design loop itself. In MDO, the task of a model is only to analyze an aspect of the complete design, but not perform design work on its own (M. van Tooren et al. (2009)), since this work interferes with the framework itself. Design work is solely the task of the optimizers.

An example of the last item in the above list is the wing chord of an aircraft. In a Blended Wing-Body design with *n* trunks, it could either be defined as:

1. A series of chords at spanwise stations (one design variable with n+1 elements)
2. A root chord and a series of taper ratios (one design variable with 1 and another one with n elements)

The latter formulation can produce misleading results, because the actual chord at all spanwise stations is determined by the root chord, and each chord depends on its neightbour inboard. In this case, if the optimizer makes a change in the root chord (local), this unintentionally changes the whole Blended Wing-Body planform (global). Generally, such a coupling between geometric parameters is highly undesirable.

Within the bounds set by the side constraints, the parametrization should not be able to yield designs that cannot be realized from a geometric perspective. A designer wishes to select a smart parametrization, where the amount of variables that has to be bounded explicitly is smallest. An example is the definition of wing planform using a LE and TE sweep angle instead of inner and outer chord and quarter chord sweep. The first option causes geometric oddities if both sweep angles are not properly bounded (see figure 7-6). This type of deficiencies in the parametrization is found by setting model in- and outputs to their extreme values and inspection a visualization of the result.

For the time being, the Blended Wing-Body aircraft will be modeled using 10 trunks across the complete span, so that the optimizer can produce a wide variety of wing planform shapes. This number of trunks has also been used in the theses of Dircken (2008). The parametrization is done according to these conventions:

- Symmetric aircraft, geometry can be mirrored in the aircraft center plane
- Global aircraft reference point located at the nose of the centerbody

[3]Example: a parameter that experiences a jump in value once a certain condition is met

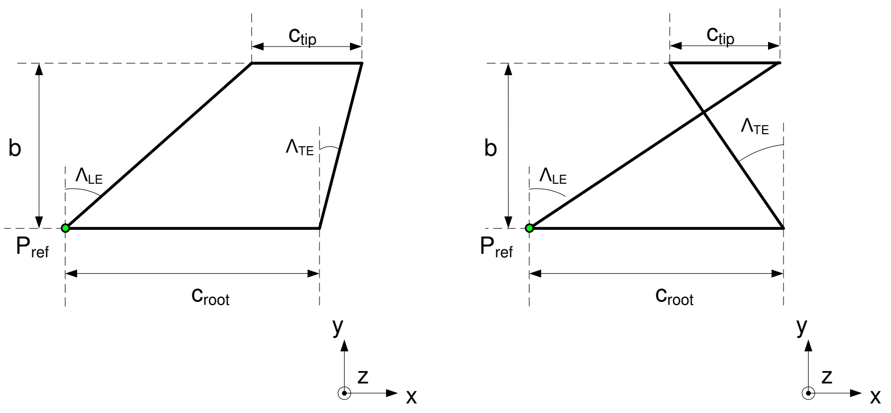


Figure 7-6: Wing planform definition requiring bounded sweep angles

7-6-1 Aerodynamics

The Vortex-Lattice Method selected to model the Blended Wing-Body aerodynamics models the aircraft as a series of flat panels. Tornado is able to handle wing twist, dihedral, sweep and taper and simple TE surfaces and offers the possibility of inserting root and tip airfoil sections for each trunk. It is convenient to use the Tornado parametrization, which is readily available, to get an overview of the variables that define the Blended Wing-Body geometry. Figure 7-7 shows the parametrization of a generic wing trunk in Tornado. Tornado defines control surfaces as fractions of the in- and outboard trunk chord. Only TE surfaces are possible, while the Blended Wing-Body of Liebeck et al. (1998) has LE slats to protect it from low-speed stall during the approach phase. To let all trunks have the same design variables, trunks that don't have a TE surface are defined with a zero control surfaces chord fraction.

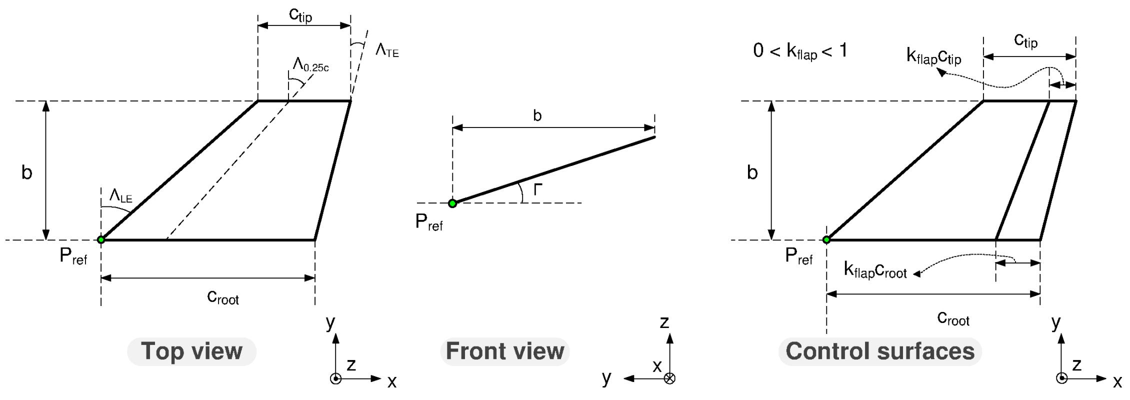


Figure 7-7: Trunk parameters in the aerodynamics discipline

A BLISS Blended Wing-Body can be created using an arbitrary number of trunks, where the root chord of a trunk is equal to the tip chord of its neighbour inboard. A 'winglet' is created by giving the most outboard trunk a large dihedral, e.g. close to 90°. The nose reference point of the Blended Wing-Body is the 'nose' of the VLM parametrization. The reference point of an individual trunk is defined as its root LE point. These reference points are indicated in Figure 7-8, which gives an example of a complete Blended Wing-Body consisting of 4 wing trunks and a winglet.

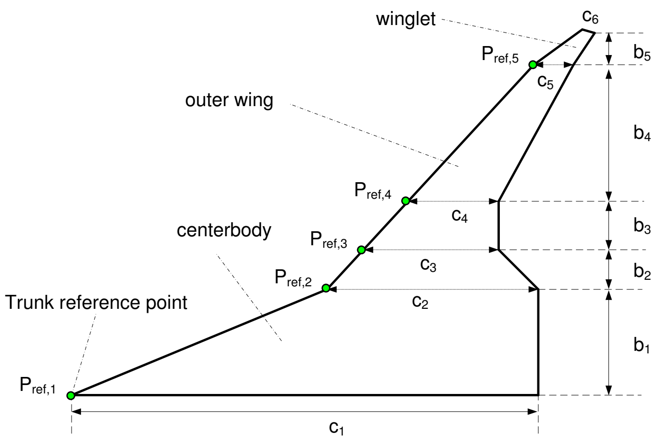
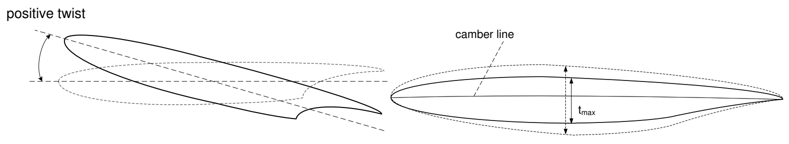


Figure 7-8: Five-trunk aerodynamic BWB parametrization

Appendix F shows the variables involved in parametrizing the wing planform for Tornado, and the number of elements they are required to have for a 10 trunk Blended Wing-Body . The airfoils of the MOB aircraft are used as a starting point for airfoil analysis since they already possess the properties Blended Wing-Body airfoils should have (large t/c, TE negative camber). To let BLISS tackle the problems of local supersonic flow and fitting the wingbox inside the aerodynamic contour, the thickness of these airfoils is scalable by changing their maximum thickness-to-chord ratios while maintaining the same camber line. The airfoil shapes themselves are derived from the airfoils of the MOB Blended Wing-Body . Figure 7-9 shows the definitions of twist and airfoil scaling.



(a) Wing twist **(b)** Airfoil Thickness scaling

Figure 7-9: Blended Wing-Body Airfoil representation

7-6-2 Structures

Also in the structural field, the Blended Wing-Body can be built up from trunks with different dimensions and properties but equal sets of design variables. Each trunk consists of booms and plates according to Megson (1999). The idealized structures method requires a definition of the chordwise cross-section in which the stresses are to be calculated, and the geometry of the wingbox in spanwise direction. To keep the number of structural design variables small, it was decided to use four booms and four plates in a rectangular cross-section. Figure 7-10 shows a typical trunk cross-section and 3D view.

A disadvantage of the rectangular approach is the consistency between aerodynamic and structural models. If the structural thickness t_{struct} becomes too large, the airfoil that has to be wrapped around it will be aerodynamically infeasible. The boom areas and skin thicknesses are considered constant throughout the trunk span but can vary from boom to boom and plate to plate. Material properties are assumed constant everywhere. Each wingbox trunk can contain an amount of 'content': either fuel or payload. It is assumed that the centerbody contains only payload and the outboard trunks contain only fuel. The main parameter in determining the content weight and c.g. for each trunk is defined as the so-called 'content span' $b_{content}$. The content span is related to the filled fraction of

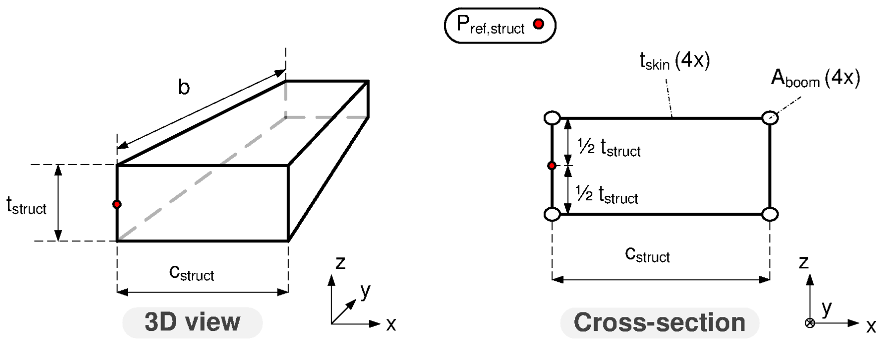


Figure 7-10: Wingbox cross-section

the trunk. This relation depends on the trunk shape and size (see appendix B-2). To maximize the bending moment relief effect of the fuel, the fuel tanks are emptied from wing root to tip.

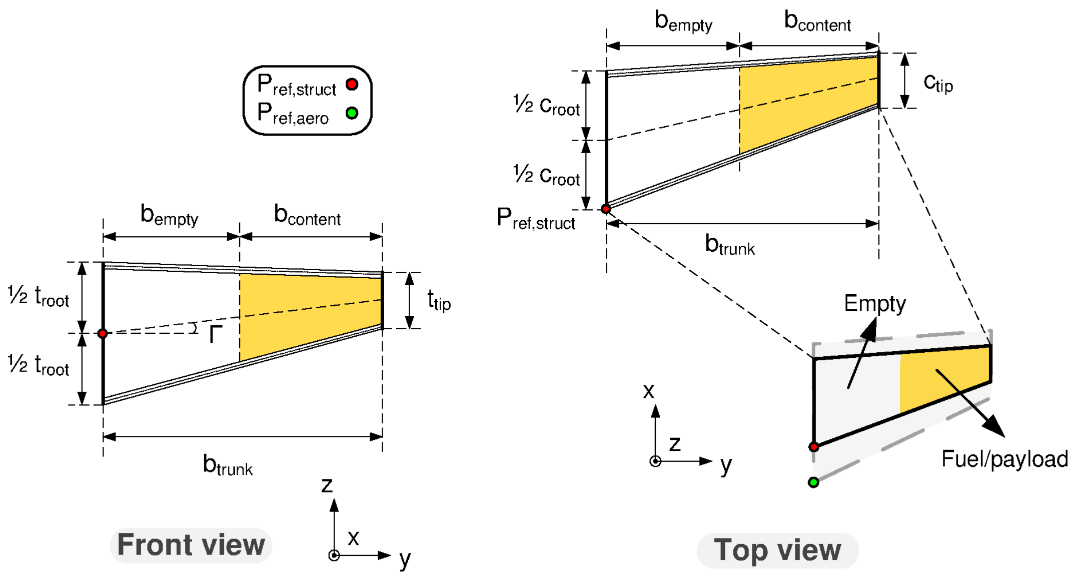


Figure 7-11: Structures, fuel and payload parametrization

It is assumed that the aerodynamic lifting surface divides the 'structures trunk' in two equal halves at any cross-section. This orientation produces an exact fit for a symmetric airfoil. For an unsymmetric or heavily cambered one, this will lead again to fitting problems. To accomodate the airfoil LE and TE, the wingbox chord is made smaller than the aerodynamic chord using a so-called LE- and TE fraction. Figure 7-12 shows how the aerodynamic and structural parametrizations are positioned w.r.t. each other. They both use the same reference point. To be able to follow the aerodynamic wing planform, the wingbox is tapered, swept, and may have dihedral.

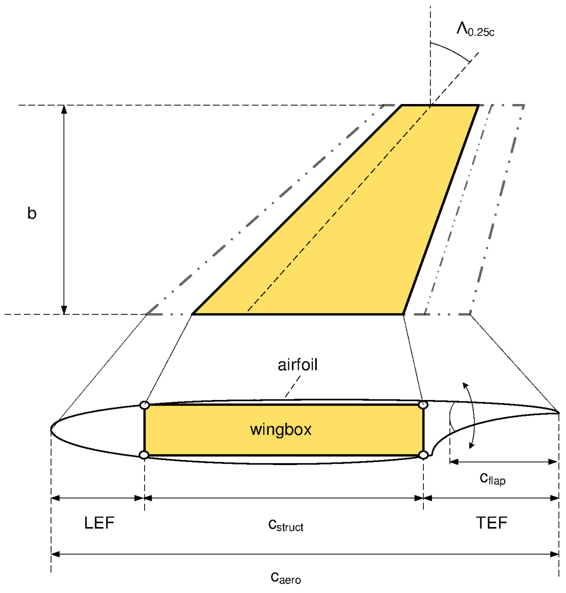


Figure 7-12: Orientation of structural parametrization w.r.t. aerodynamic one

7-6-3 Propulsion

Th engine geometry need not be defined, since the aerodynamic model nor the structural one is able to include it. Therefore it suffices to define the engines as point masses which can generate an certain amount of thrust and are located each at a position w.r.t. the aircraft reference point. The location and mass of the engines is relevant for the calculation of the aircraft center of gravity and for the pitching moments they generate w.r.t. the c.g.. The last aspect has an influence on trimming the aircraft. Two basic assumptions apply to the Blended Wing-Body propulsion system:

- Engines are placed somewhere above the aft part of the centerbody
- The Blended Wing-Body has three equal podded conventional engines

The engine thrust is net thrust since the engine model already includes engine drag taking into account. The advantage of engines above the wing is that the generate a downward pitching moment which is beneficial for aircraft stability. Figure 7-13 shows how the engines are positioned with respect to the aircraft.

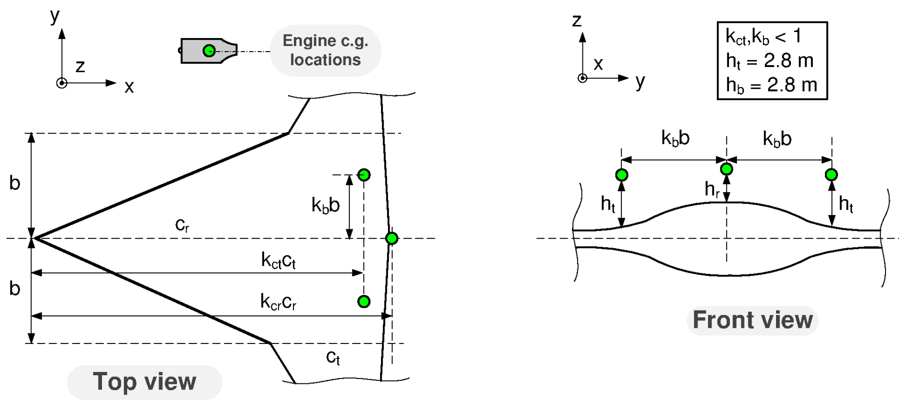


Figure 7-13: Engine positioning

Values of the parameters h_t and h_r were taken from Kok (2008) and kept constant throughout the design process.

7-7 BLISS Integration

The final step in the design problem formulation is to integrate it in the BLISS MDO framework at hand. From section 5-3, it can be deduced that the following three actions are needed to run a given problem in BLISS:

1. **Division** of the discipline models found in 7-4 across a number of BlackBoxes, where the number of BlackBoxes and discipline models is not necessarily equal.
2. **Definition** of the types of the variables found in the parametrization as local or global design variables, couplings or constant design parameters.
3. **Allocation** of constraints, where each constraint is placed inside a suitable BlackBox. Also, it is decided which constraints are system constraints.

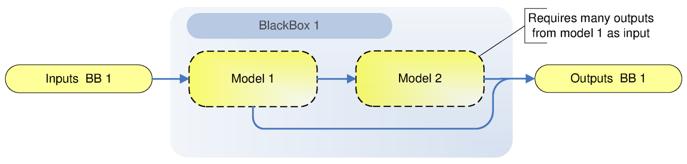
The above aspects will be discussed in sections 7-7-1 to 7-7-3, respectively.

7-7-1 BlackBox Decomposition

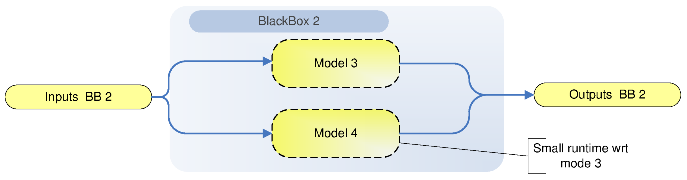
The BLISS methodology requires the discipline models to be wrapped inside BlackBoxes having fixed in- and output formats, namely **X**-,**Y**- and **Z**-variables and constant design parameters (in), and **Y**-variables and constraints (out). Only the BlackBox in- and outputs are 'seen' by the BLISS framework. The most straightforward method to do this is to create BlackBox around every discipline separately. When a certain discipline requires models that together are significantly more time-consuming compared to the other disciplines, it can be decided to 'cut' this discipline in half and locate the halves in separate BlackBoxes (see Figure 7-14c).

The aforementioned measures ensure that the time investment for running each BlackBox is in the same order of magnitude, which increases the benefits of parallel computing. Besides that, BLISS is designed to solve large problems decomposed into smaller parts (the BlackBoxes), which might not be solved with a single-level MDO method. If one of these parts involves much more design variables than the others, the advantage BLISS has over these methods is lost. Cutting up discipline models increases the number of couplings, something that can give more insight into the MDO problem. On the other hand, more couplings implies longer time is spent in the BBSA. The opposite is also true, the computational load can be balanced by merging two discipline models. This may be done for two reasons. First of all, if a discipline that requires many output variables from a one other discipline is wrapped separately, this would greatly increasing the amount of coupling variables, and hence BBSA duration. Merging them eliminates these coupling variables, since they become internal to the BlackBox (see Figure 7-14a). Secondly, a small discipline model can be merged with a larger one to balance computations across all BlackBoxes (see Figure 7-14b). Merging two disciplines decreases the duration of the BBSA but information on the interaction between the two is no longer available. When the above reasoning was adopted for the Blended Wing-Body , the following results emerged:

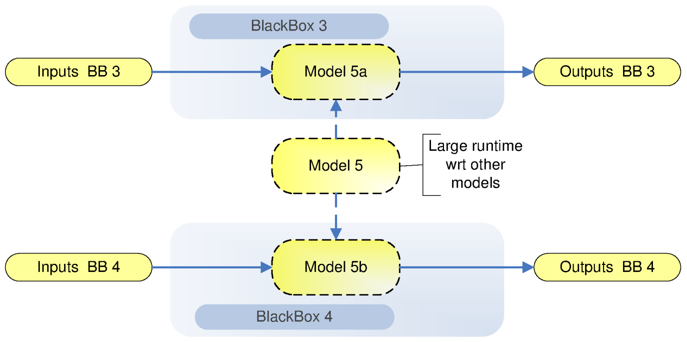
- The 'Performance' discipline requires data from the 'Propulsion' one and has a small runtime with respect to it, so both are merged into BB 'PROP'
- Idem for the 'Trim & balance' sub-discipline, it is merged with the 'Aircraft planform analysis (3D) ' one into BB 'AERO_3D'
- Although Xfoil is considered a fast program, running it for all Blended Wing-Body airfoil profiles for a range of AOA still required considerable time, and so it was implemented in a separate BlackBox 'AERO_2D'



(a) In series:output-to-input



(b) In parallel: small and large model together



(c) Large model split up

Figure 7-14: Possibilities for model division across BB's

- The 'Fuel weight', 'Structural strength & stiffness' and 'Structural weight estimation' disciplines all require the same structural geometry data as input, and so they are merged into BB 'STRUCT'
- The 'Overall weight estimation' discipline, although small, is kept in a separate BB 'WEIGHT' to see the contributions of other BlackBoxes to Blended Wing-Body MTOGW
- For convenience, the objective is placed in a separate BB 'OBJ', so that it can be easily changed

The resulting decomposition into BlackBoxes is displayed in table 7-7 together with their durations. Table shows that, the computation times of the two aerodynamic models 'AERO_3D' and 'AERO_2D' form the largest computational burden.

Table 7-7: BlackBox decomposition overview

Discipline	BlackBox	Sub-discipline	Tool	Duration, single run (s)
Aerodynamics	**AERO_3D**	Trim & balance	own	129
		Aircraft planform analysis (3D)	Tornado	
	AERO_2D	Airfoil analysis (2D)	Xfoil	75
Structures	**STRUCT**	Fuel weight calculation	own	9.5
		Structural weight estimation	Torenbeek method implementation (own)	
		Structural strength & stiffness	Ideal structures implementation (own)	
Propulsion	**PROP**	Engine performance	Gasturbine Simulation tool (Kok)	71
Performance		Aircraft performance	own	
Weight	**WEIGHT**	Systems weight estimation	Raymer implementation (own)	0.12
		Center of Gravity calculation	own	
		Overall weight estimation	own	
n.a.	**OBJ**	Objective value calculation	own	0.06

With these BlackBox durations, an SA with three loops takes about 15 minutes, and a complete BBSA around 3 hours when using forward differencing.

7-7-2 Design Variables and States

Some variables can be assigned a type and BlackBox by inspection, such as variables that belong specifically to a discipline model inside a BlackBox. An example of this is the 'PROP' BlackBox, which contains a model that requires a number of engine properties. Other BlackBoxes only need results of this BlackBox (thrust and engine weight). The issue of allocation arises especially when dealing with geometric parameters. Since both the 'AERO' and 'STRUCT' BB's are optimizing aircraft geometry, most design variables could be allocated:

1. To either one as an **X**-variable
2. To both as a **Z**-variable
3. To either one as an **X**-variable and rerouted to the other as a coupling variable

Another aspect is decision authority: if a variable becomes an **X**-variable, only the BlackBox to which it is assigned can change it. This property is useful if the designer wants to grant a particular Blackbox control over solving a particular design problem. For example, one of the main issues with the Blended Wing-Body are the high local Mach numbers associated with the highly loaded outboard wing. As defined in section 7-6-1, the t/c of airfoil sections can be scaled using a variable called 'MaxAirfoilThickness'. Table 7-8 shows the three options for allocating this variable to 'STRUCT' or 'AERO_2D' as an **X**-variable (1 and 2), or to both as a **Z**-variable (3).

Table 7-8: Design variable allocation options for 'MaxAirfoilThickness'

Allocation	**Parameter name**	**Authority**
1	X.AERO_2D.MaxAirfoilThickness & Y.AERO_2D.MaxAirfoilThickness	BlackBox optimizer 'AERO_2D'
2	Z.global.MaxAirfoilThickness	System optimizer
3	X.STRUCT.MaxAirfoilThickness & Y.STRUCT.MaxAirfoilThickness	BlackBox optimizer 'STRUCT'

Figure 7-15 depicts a schematic of the allocation options. In the end, the tackling of the high local Mach number issue was given priority ,and so option 1 was chosen. However, there is no clear mechanism or roadmap to follow in these cases, as the effect of problem structure as large as the current one is unforeseen. As always, the designer will need experience with a particular class of problems to be able to 'make the right choices' in problem structure setup without prior knowledge of the result.

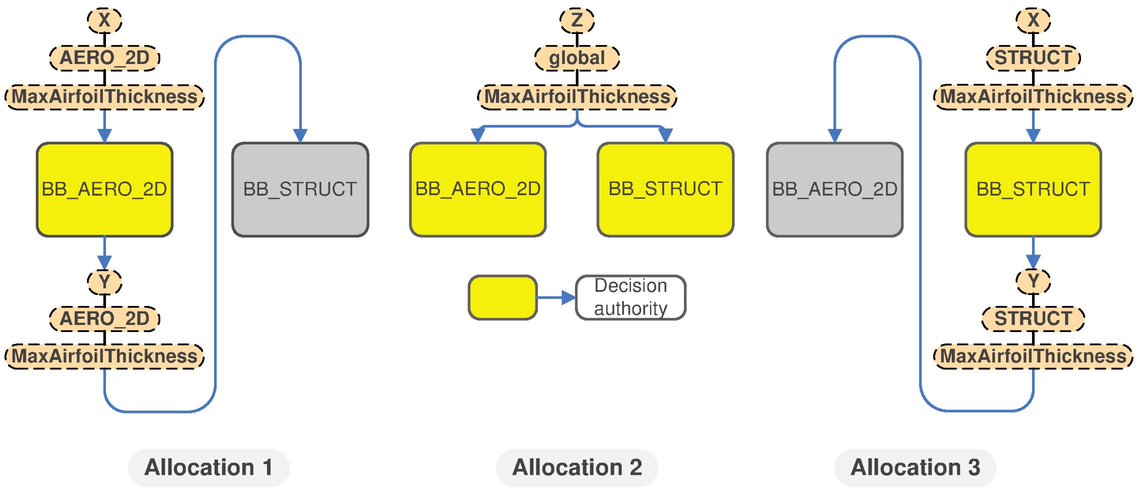


Figure 7-15: Design variable allocation options for 'MaxAirfoilThickness'

Finally, optimization tasks of the BlackBoxes should be balanced, if too many variables are assigned to one BlackBox, this is conflicting with the principle of a multilevel MDO structure and a single level structure might be more appropriate still. Figure 7-16 displays the DSM that was generated by the BLISS framework using the above variable definitions. A complete list of Blended Wing-Body design variables is given in appendix F.

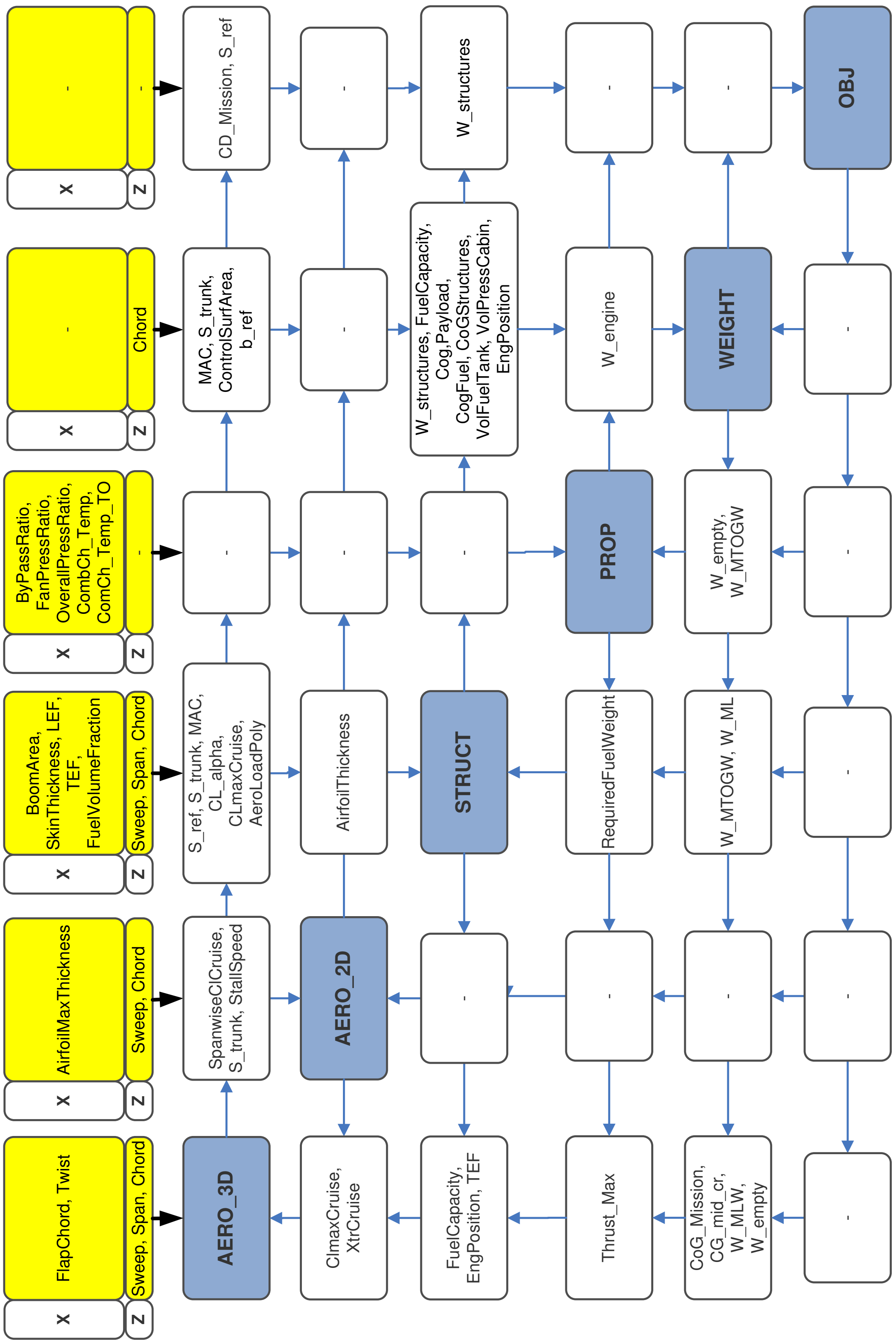

X
Z
FlapChord, Twist
Sweep, Span, Chord
AirfoilMaxThickness
Sweep, Chord
BoomArea, SkinThickness, LEF, TEF, FuelVolumeFraction
Sweep, Span, Chord
ByPassRatio, FanPressRatio, OverallPressRatio, CombCh_Temp, ComCh_Temp_TO
Chord
AERO_3D
SpanwiseClCruise, S_trunk, StallSpeed
S_ref, S_trunk, MAC, CL_alpha, CLmaxCruise, AeroLoadPoly
MAC, S_trunk, ControlSurfArea, b_ref
CD_Mission, S_ref
ClmaxCruise, XtrCruise
AERO_2D
AirfoilThickness
FuelCapacity, EngPosition, TEF
STRUCT
W_structures, FuelCapacity, Cog,Payload, CogFuel, CoGStructures, VolFuelTank, VolPressCabin, EngPosition
W_structures
Thrust_Max
RequiredFuelWeight
PROP
W_engine
CoG_Mission, CG_mid_cr, W_MLW, W_empty
W_MTOGW, W_ML
W_empty, W_MTOGW
WEIGHT
OBJ

Figure 7-16: Blended Wing-Body Design Structure Matrix

7-7-3 Constraint Allocation

When all constraints are identified and the problem is decomposed into BlackBoxes, each constraint must be allocated to a BlackBox. The most important criterion for allocation is that the constraint can be influenced by changes in design variables, so that it can be satisfied. For the larger part of the constraints, allocation is straightforward as the constraint is specifically intended for a certain discipline, e.g. 'stall speed' for aerodynamics. Results of the BBSA include the constraint sensitivities $\frac{dG}{dX}$ and $\frac{dG}{dZ}$. By reviewing $\frac{dG}{dX}$, a constraint that cannot be allocated intuitively can be put in the BlackBox where the sum of its $\frac{dG}{dX}$ is largest. It is assumed, that each constraint is sensitive to at least one design variable, as in the 'parametrization' step the models are modified such that all constraints can be evaluated.

As J. Sobieski et al. (1998) stipulates, a problem structure may be such that there are constraints $\mathbf{G}_{yz}$ that have a weak dependence on **X**. That means that the BlackBox optimizer will have difficulties in satisfying these constraints, if it is even possible to satisfy them by manipulating only **X**-variables. Therefore these constraints need to be evaluated in the System optimization as well. System constraints can be identified by checking constraint gradients w.r.t. **Z**, if these are non-zero and sufficiently large.

For computational reasons, the system optimizer uses linearized versions of $\mathbf{G}_{yz}$. Although it is possible to compute actual constraint values, this is not practically feasible for all but the smallest problems, since it requires an iterative system analysis to calculate them. The complete constraint allocation of the Blended Wing-Body is given in appendix G.

7-8 Validation

The MOB project is used to validate the structure of analysis models involved in Blended Wing-Body design and, to some extent also in B747 design. Data from Smith & Yarf-Abbasi (1999) and Qina et al. (2004) was used used to re-create the baseline MOB Blended Wing-Body in the BLISS framework. The MOB Blended Wing-Body is defined using 10 trunks on its half-span. Because for computational reasons it is desired to use 10 trunks in the BLISS optimization, the validation will also be conducted with 10 trunks, which correspond to stations in the MOB reference design, as in Figure 7-17.

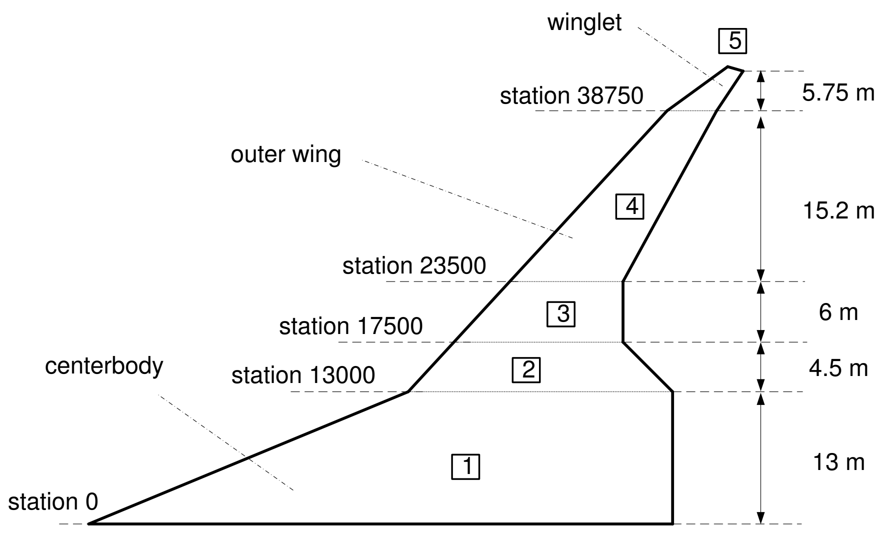


Figure 7-17: BLISS framework representation of the MOB Blended Wing-Body

With respect to the MOB Blended Wing-Body , the Blended Wing-Body design created for validation of the problem formulation is chosen such that both aircraft have equal:

- **Dimensions** at the corresponding stations (chord, thickness, twist, sweep, span, airfoils)
- **Flight conditions** (start-of-cruise at 35000 ft and Mach 0.85)
- Control surface allocation and **flap hinge line** positions
- Payload **weight**
- **Range**

Furthermore, by manual tuning of the design variables, the structural, fuel, engine and total weights are calibrated to be close to the MOB BWB values while maintianing structural feasibility. Table 7-9 shows the MOB BWB data obtained from Qina et al. (2004)[4] and Smith & Yarf-Abbasi (1999) and the corresponding the BLISS BWB values. The similarity between the MOB BWB lift and drag and its BLISS recreation is reasonable. The drag of the BLISS BWB is lower than that of the MOB BWB at a higher MTOGW, because higher fidelity methods (CFD) were used for computing the MOB BWB data.

[4]In particular those in table 5

Table 7-9: Comparison of MOB BWB and BLISS Blended Wing-Body data

Parameter	MOB BWB	BLISS BWB	% difference	Units
W_{MTOGW}	3684	4262	+15.7	[kN]
W_{OEW}	1306	1836	+40	[kN]
W_f	1262	1317	+4	[kN]
$W\{eng$	267.0	263.2	-1.4	[kN]
CL_{cr}	0.24	0.20	-16.7	[-]
CD_{cr}	0.033	0.023	-30	[-]
α_{cr}	3.0	6.9	+133	deg
$c.o.g_{cr}$	29.3	31.7	+8.2	[-]

While the BLISS Blended Wing-Body is reasonably close to the MOB one, there was a number of significant constraint violations. It appeared, that the BLISS Blended Wing-Body violated constraints on fuel required ('MissionFuel') and thrust ('Cruise_Thrust' and 'TakeOff_dist'), and the local Mach numbers on its outboard wings were too high ($\geq$ 100% violation). Figure 7-18 shows that the high outboard local Mach numbers were also a major issue in the Baseline MOB BWB, yielding a large wave drag component (see 'initial configuration').

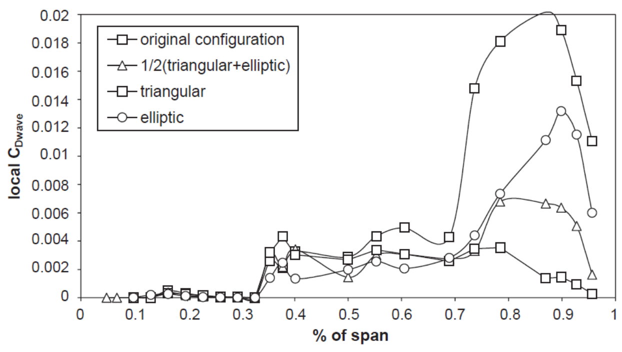


Figure 7-18: Spanwise local wave drag distribution at design C_L for M = 0.85, from Qina et al. (2004)

All in all, based on the above results the BLISS Blended Wing-Body formulation is considered as 'in the ball park' for Blended Wing-Body design. In comparing both aircraft results, the difference in modeling fidelity should be considered. One way of making a fair comparison between MOB BWB and BLISS BWB drag is to evaluate both aircraft with the same aerodynamic model. Due to time constraints, this will not be attempted.

Chapter 8

Blended Wing-Body Optimization Results

After successfully setting up and validating the BLISS framework, a final step remains in completing this research. This chapter evaluates whether the BLISS framework is able to optimize the Blended Wing-Body problem that was set up in chapter 7, using the settings and insights of chapter 6. In scale, this problem is an order of magnitude larger than the 747 problem (Table 8-1).

Vector	Number of elements
X	76
Y	110
Z	16
G	93

Table 8-1: Blended Wing-Body problem size

The Blended Wing-Body optimization was completed in two stages:

1. Section 8-1 runs the Blended Wing-Body a few cycles to indentify and resolve undesired features that were not visible in the System Analysis, but do appear in the optimization.
2. Section 8-2 discusses the final design of the Blended Wing-Body , obtained by running BLISS with the modifications made according to 8-1

Finally, section 8-3 describes how the framework could be used in design practice.

8-1 Preliminary BLISS Run

The Blended Wing-Body design process starts from an initial point, which has the same configuration as the MOB, but not the same initial design vector. This was done on purpose, because the goal of this project is to evaluate BLISS in the conceptual design phase. Intuitively, this calls for an initial design with unknown characteristics, not with one that has gone through an extensive optimization process already. Following the results of chapter 6, the design is optimized using small move limits of 3% and linearized constraints for both **X** and **Z** and a BBSA step of 1e-8. All constraints are evaluated both at BlackBox and System level. The initial point is infeasible, with 'CriticalMach' being the constraints with the largest violation. Despite its large size, the results show that BLISS is able to decrease the objective value of the Blended Wing-Body problem:

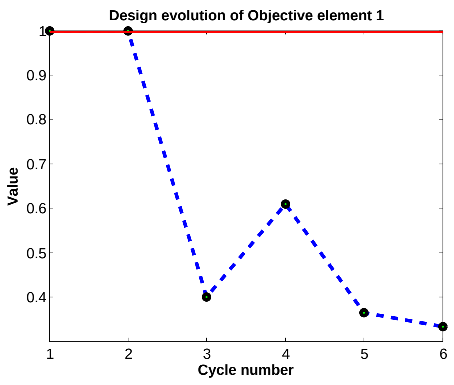
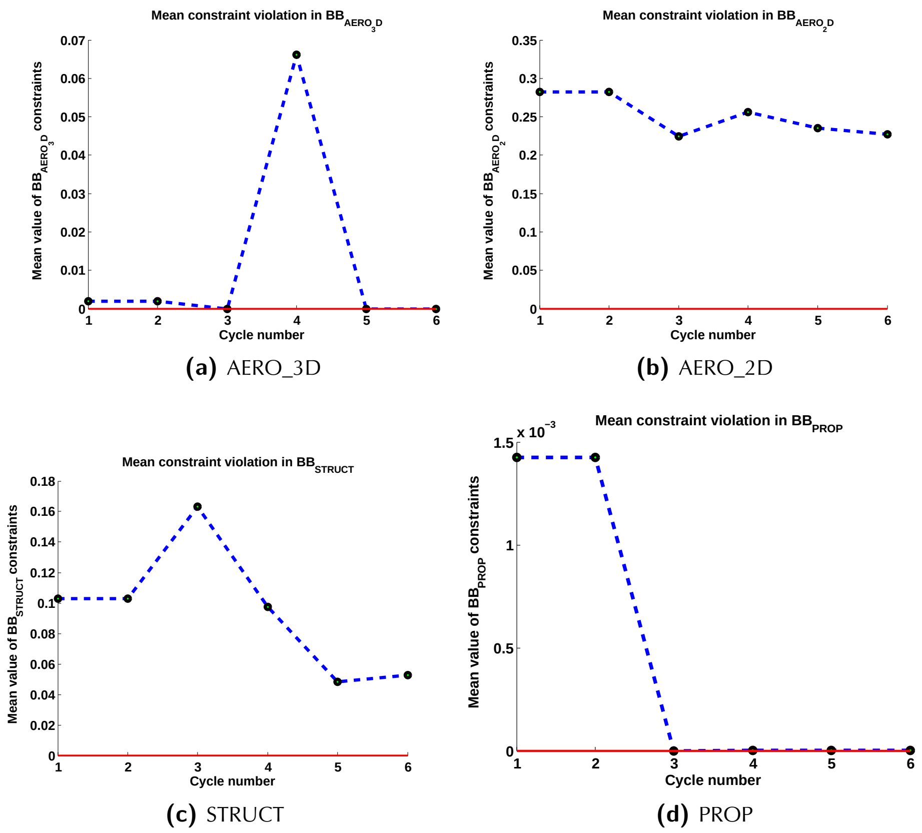


Figure 8-1: Blended Wing-Body objective history

Figure 8-2 shows that the problem feasibility increases each cycle and in each individual BlackBox. It is clearly visible that 'AERO_2D' was the most difficult BlackBox to optimize, as it is the last one to become feasible and constraint violation decreases only gradually. This is due to the constraint on maximum local Mach number, related to high local lift coefficients on the outboard wing, which are identified as well in Roman et al. (2000) and Morris et al. (2004).



(a) AERO_3D (b) AERO_2D (c) STRUCT (d) PROP

Figure 8-2: Blended Wing-Body Mean Constraint Violations per BlackBox

After a few cycles, it appeared that each cycle, the optimizers downsized the wingbox 'BoomArea' and 'SkinThickness' with the maximum amount allowed by the move limits. Meanwhile, design sensitivities showed (as expected) that these two design variables had a large effect on the shear and normal stress constraints. Prediction was, that this trend would continue until both constraints approached zero. With move limits of 3%, many cycles would be needed to 'walk' to this point. Therefore, the wingbox was downsized manually between cycles to accelerate this process. This method of design modification is only advisable if:

1. The design variables in question are coupled to a small set of constraints and states (structural weight, in this case). If not, changing them manually will turn around the entire design.
2. The runtime of the BLISS problem should be accelerated.

Eventually, BLISS will find the aforementioned point itself, only with a much larger time investment. Manual modifications are crude considering that BLISS takes into account sensitivities of the objective to all variables. Hence uninterrupted BLISS runs are preferable if time permits.

The results overview is continued with the development of some common design characteristics in Figure 8-3 throughout the BLISS procedure. The L/D value significantly increases towards the end of the optimization and is close to the one given for the baseline Blended Wing-Body in Liebeck et al. (1998). This result depends of course on the drag model that was used, and L/D will almost certainly decline if a more advanced drag model is used, with for instance the influence of viscosity and turbulence. In the conceptual stage however, this underestimation is considered allowable. Naturally, as more detailed and accurate design knowledge on the Blended Wing-Body is gained, this knowledge can be transported back to the conceptual phase by applying suitable correction factors to the design states. However, this is considered beyond the scope of this thesis.

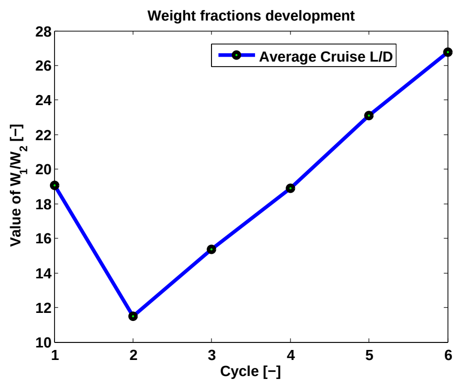
From Figure 8-3, it appears that the thrust-to-weight ratio of the Blended Wing-Body becomes quite large compared to the order of magnitude it has in Liebeck (2004). By close inspection of the code of the 'PROP' BlackBox, it appeared that an inner design loop of the engine mass flow was still present. This mass flow was sized according to achieve sufficient cruise thrust. Because BLISS could not control the mass flow, the engine weight and maximum thrust became higher than necessary (as could be derived from large negative constraint values in BB_PROP). The presence of inner design loops is highly undesirable (see section 7-6), so the following actions were taken to remove it:

1. Engine design mass flow was made an X-variable in BB_PROP
2. Design mass flow and combustion chamber temperature were sized manually to values for which all constraints of BB_PROP were satisfied

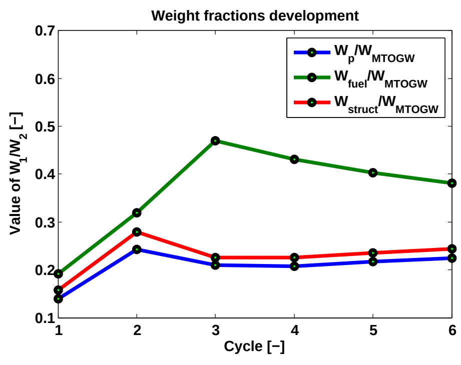
With this modification, the Blended Wing-Body BLISS run was continued. Figure 8-4 gives the evolution of the spanwise local lift coefficient[1] C_l.

The changes per BLISS cycle in C_D and C_L acros the entire mission (5 flight conditions) is visualized by Figure 8-8. It shows that C_D is not only decreasing overall, but also that the mission profile of the drag coefficient becomes flatter, until it resembles a straight line. This is caused by the way the objective formulated, as a minimum mission average drag instead of minimum drag at one cruise condition. Thus, the BLISS algorithm is indeed able to identify how the design variables should be changed to achieve lower drag across the entire mission.

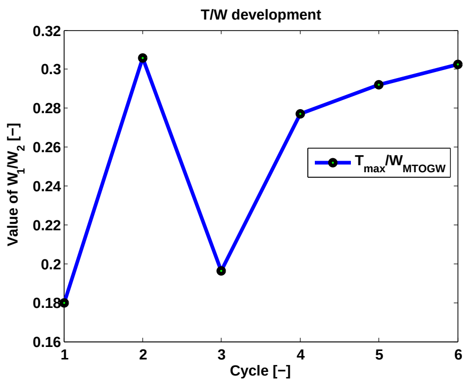
[1]Note that, although the lift distribution is a smooth curve in Tornado, the 'AERO_2D' BlackBox only 'sees' the lift distribution at the 6 points where an airfoil is defined.



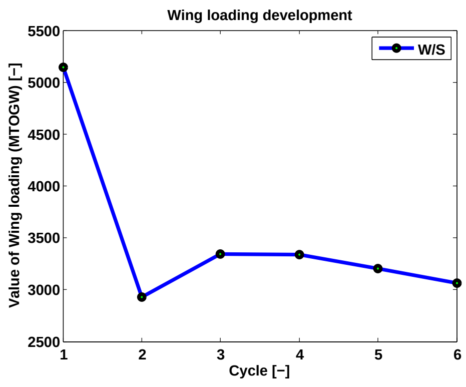
(a) Mission lift-to-drag ratio



(b) Weight fractions



(c) Thrust-to-weight ratio



(d) Wing loading at MTOGW

Figure 8-3: Development of common design characteristics

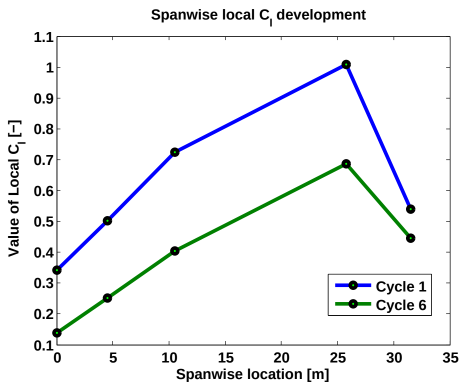
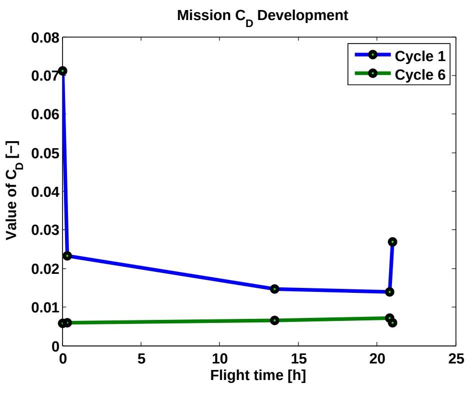
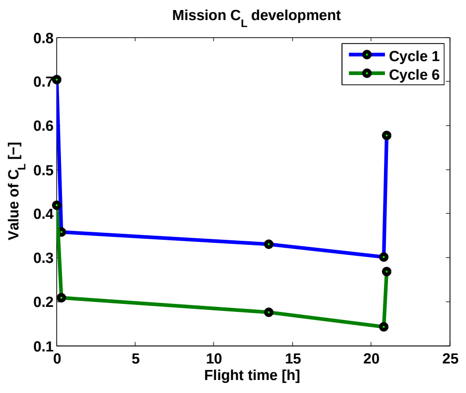


Figure 8-4: Development of spanwise $C_{l,cr}$



(a) Drag across mission



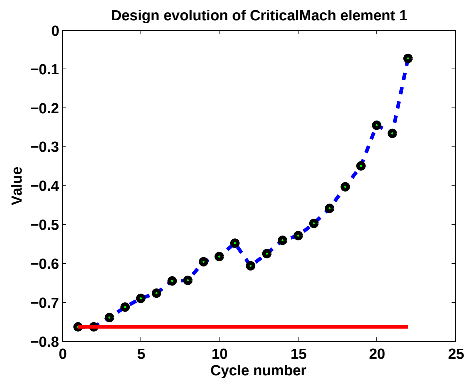
(b) Lift across mission

Figure 8-5: Development of mission C_D and C_L

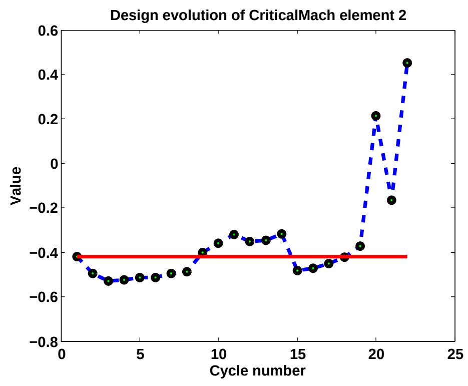
8-2 Final BLISS Run

The first attempt to optimize the Blended Wing-Body shows a typical trait of MDO optimization: problem formulations get exponentially more complex if problem size increases, but still the optimization results can become unreliable because of small flaws in the problem formulation or analysis models. For the final BLISS run, the design is started using the same settings as the preliminary run, and the design vector of its last completed cycle. After 22 cycles the design had became feasible except for the critical Mach number constraint on the outboard wing (trunk 4).

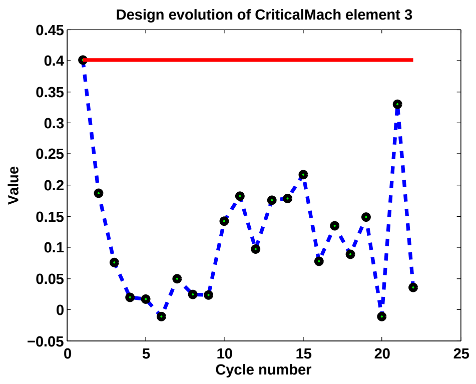
This is in accordance with the results from the MOB project, where the outer wing had to be redesigned to avoid a strong shock wave occuring there (Qina et al. (2004)). Similar to the BLISS Blended Wing-Body , the MOB centerbody was shock-free. The progression of the critical Mach constraint is hinting that the BLISS framework is not going to find a way to satisfy it completely (figure 8-6): starting at cycle 19 the constraint value at trunk 4 is jumping from 0 to 1.5 back and forth, trunks 2 and 3 show similar tendencies.



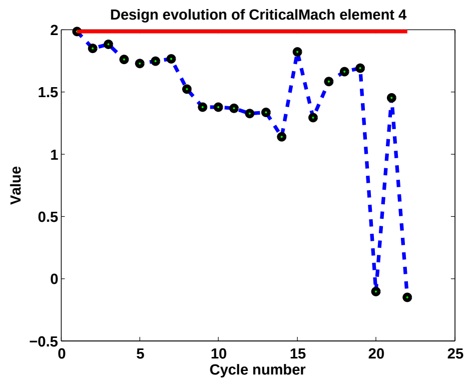
(a) Centerbody (trunk 1)



(b) Wing (trunk 2)



(c) Wing (trunk 3)



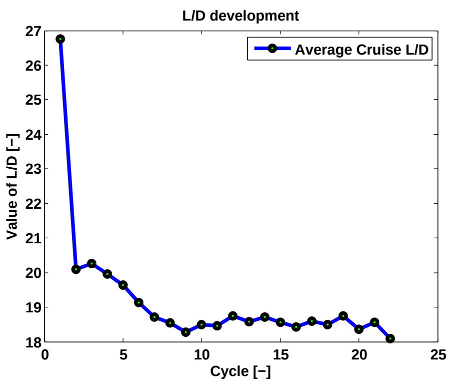
(d) Wing (trunk 4)

Figure 8-6: Development of 'CriticalMach' constraint

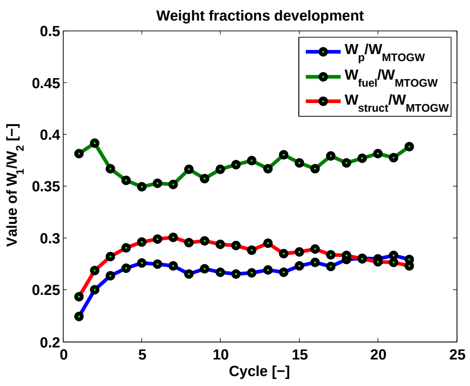
From the figure, one can deduce how BLISS attempts to satisfy element 4 of the constraint by moving the issue to the centerbody, where the 'CriticalMach' constraint value goes from -0.75 to almost 0. In the process, the constraint is also violated on trunks 2 and 3. BLISS manages on the other hand to drive down the value of element 4 significantly by increasing the sweep angle of the fourth wing trunk. The centerbody does not have Mach issues, and consequently BLISS decreased the sweep and chord there to locally increase lift at the centerbody and compensate for the lift that is lost by adding outboard sweep. So BLISS:

- Has more opportunity to tackle design challenges locally as the number of design variables increases
- Is able to drive constraint values to zero at a certain location (trunk 4) by making modifications elsewhere (the centerbody)

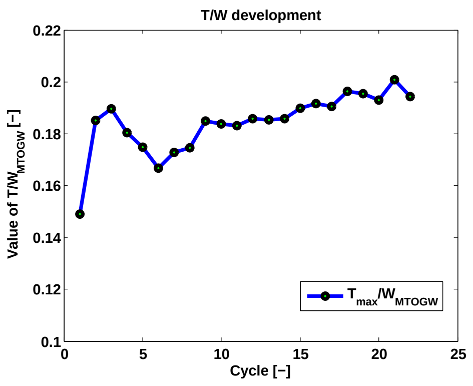
In a two-trunk Blended Wing-Body , BLISS would still have to resize the outboard trunk, which is now half of the wing. A designer observing this trend could erroneously assume that having high local Mach numbers is an issue for the complete outboard wing. This shows again that one should not blindly rely on results of a single MDO run. Instead, once BLISS runs into problems during its iterations, the designer should investigate them them further and appropriately modify the problem formulation, which in the above case boils down to increasing the number of wing trunks. Figure 8-7 again shows the development of common design parameters during the final optimization run.



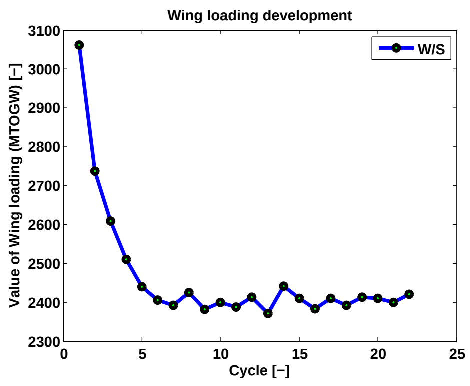
(a) Mission lift-to-drag ratio



(b) Weight fractions



(c) Thrust-to-weight ratio



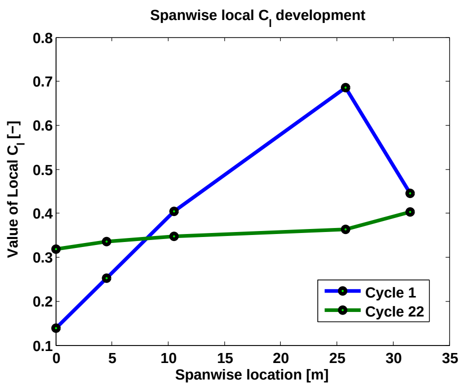
(d) Wing loading at MTOGW

Figure 8-7: Development of common design characteristics

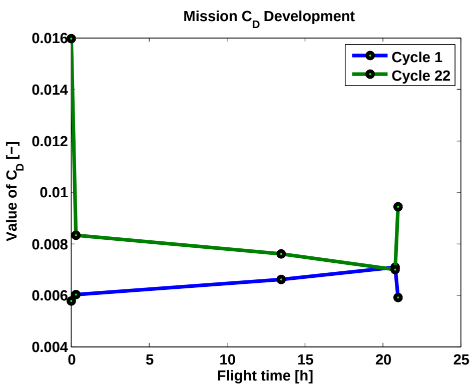
Part of the L/D increase in Figure 8-7 is also due to reduced trim drag, caused by decreasing control surface size: across the optimization process, BLISS decreases their size with 10%. In the final run, BLISS decreases lift on the outboard wing to reduce local Mach numbers there (Figure 8-8a), leading to a decrease in overall lift (C_L) and an increase in drag(Figures 8-8b and 8-8c). Hence, to solve the local Mach problem, BLISS is forced to decrease objective value.

This chapter is concluded with a short discussion on the planform changes BLISS implemented in its 22 uninterrupted cycles. Four views are shown in Figure 8-9[2]. These Figures show how the framework decreases sweep and chord of the centerbody to locally increase spanwise loading while the opposite occurs at the outboard wing.

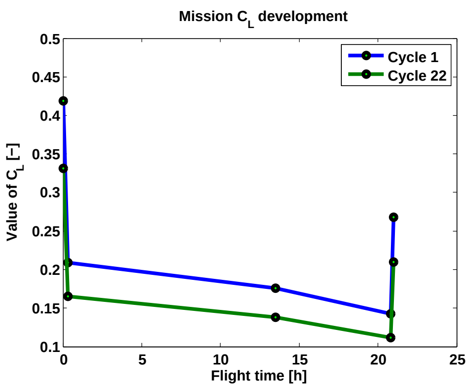
[2]grey lines = initial, red lines = final, aerodynamic planform, green lines = final, wingbox, solid: aerodynamic contours



(a) Spanwise lift distribution

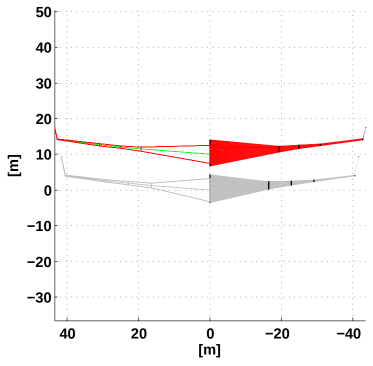


(b) Drag across mission

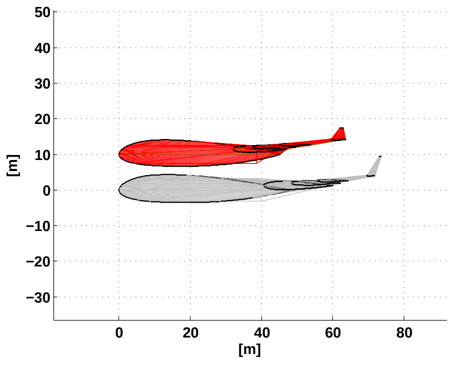


(c) Lift across mission

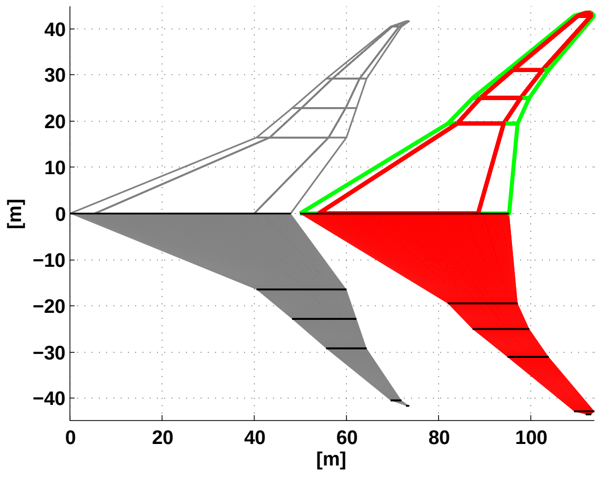
Figure 8-8: Development of mission C_D and C_L



(a) Front view



(b) Side view



(c) Top view

Figure 8-9: Blended Wing-Body planform changes

8-3 Framework Utilization

In the previous section, focus was on obtaining meaningful results on the design of a Blended Wing-Body . In this one, it is analyzed what steps were made to come to the Blended Wing-Body results, and an attempt is made to generalize these steps to make them applicable to other complex MDO problems as well. After finishing the first version of the Blended Wing-Body problem structure and validating it with the MOB, a BLISS run of multiple cycles was attempted. Although the framework was running properly, a number of unforeseen flaws in the Blended Wing-Body implementation emerged. In the path from defining the research goal to the final Blended Wing-Body design, the validation cases of chapter 6 and the Blended Wing-Body design formulation underwent countless revisions. These were needed to let the BlackBoxes pass on the the correct information to each other, cover analysis model assumptions with constraints. In the process, a number of problem characteristics frequently caused erroneous or inconsistent results optimization results. They can largely be avoided with the use of a checklist, which is included checklist in appendix E-1. The Blended Wing-Body case has demonstrated how the BLISS optimizers tried to solve the design challenge of the critical local Mach number outboard. BLISS reduced this local Mach number a great deal, the question that remains is whether the designer agrees on the suggested design changes or not. During utilization of the BLISS MDO framework in this research, four major steps were identified:

1. The MDO problem is defined by the designer
2. The framework computes large amounts of sensitivity data and optimizes the design variables accordingly
3. In the results, the designer identifies new design challenges he wishes to adress and implements them in the framework
4. The framework is run again

These steps are repeated until either the designer is satisfied with the results, or a limit on available design time is reached. Inherent to these steps is the notion that the designer is still responsible for making design choices on a macroscopic scale (e.g. design configuration), which calls for the creativity associated with a human designer. The framework on the other hand is in charge of making changes on a much smaller scale, requiring extensive and tedious 'data mining' for which todays computers are highly suitable. Figure 8-10 represent a flowchart of this process formulated for a general MDO problem.

The Blended Wing-Body case shows that using the BLISS framework could provide the designer with answers to a broad array 'what-if' questions even with a complex design case. A few examples:

- **What if a different objective is used?**
  - Design for maximum range
  - Design for minimum fuel consumption
  - Design for minimum weight
  - Design for a weighted combination of the above
- **What is the effect on the design of in- or excluding a given discipline?**
  - Aeroelasticity
  - Noise & pollutant emissions
  - Direct Operating Cost
- **What if the design requirements change?**
  - Regional Blended Wing-Body with medium range, 200 pax. (Liebeck et al. (1998))
  - High-subsonic cruise speed, $M_{cr} = 0.95$ (Roman et al. (2003))
  - Single or double deck passenger cabin
  - Design of a family of aircraft ranging from 250-550 pax. (Liebeck et al. (1998))

- **What if the initial point for optimization should change?**
  - Design of a 20 trunk Blended Wing-Body design instead of 10 trunks
  - Design of a Blended Wing-Body with a vertical tail on the centerbody
  - Executing a range of optimization runs using different principal design parameters (chord, sweep, span, t/c)

Once the Blended Wing-Body BLISS formulation is in place, the above questions can be answered with relative ease, by deriving a problem formulation for each question. Although these new problem formulations will incorporate different requirements, objectives or parametrization, on a more fundamental level they are still very similar to the original formulation. Using the traditional approach, such a time consuming problem formulation is absent, but on the other hand the entire conceptual design process should be re-run for each question that is to be answered.

The BLISS framework should be seen as a tool that can help the designer to design, not a tool that designs all by itself. The framework can tune a number of parameters that would be impossible to do by hand, but before it can do this, the designer needs to use his design experience to create a BLISS problem formulation that represents the actual aircraft he wishes to design. Otherwise, a very promising conceptual design may turn out to be a mishap in the preliminary design phase, where it is much more costly to implement major design changes. Furthermore, the BLISS framework may not find a solution if the initial point of the optimization is too crude: if you insert a brick, BLISS is not going to come up with an aircraft. In conclusion, a good dose of engineering judgment is still of primary importance when using this tool: the optimization results are only as good as the designer that created them.

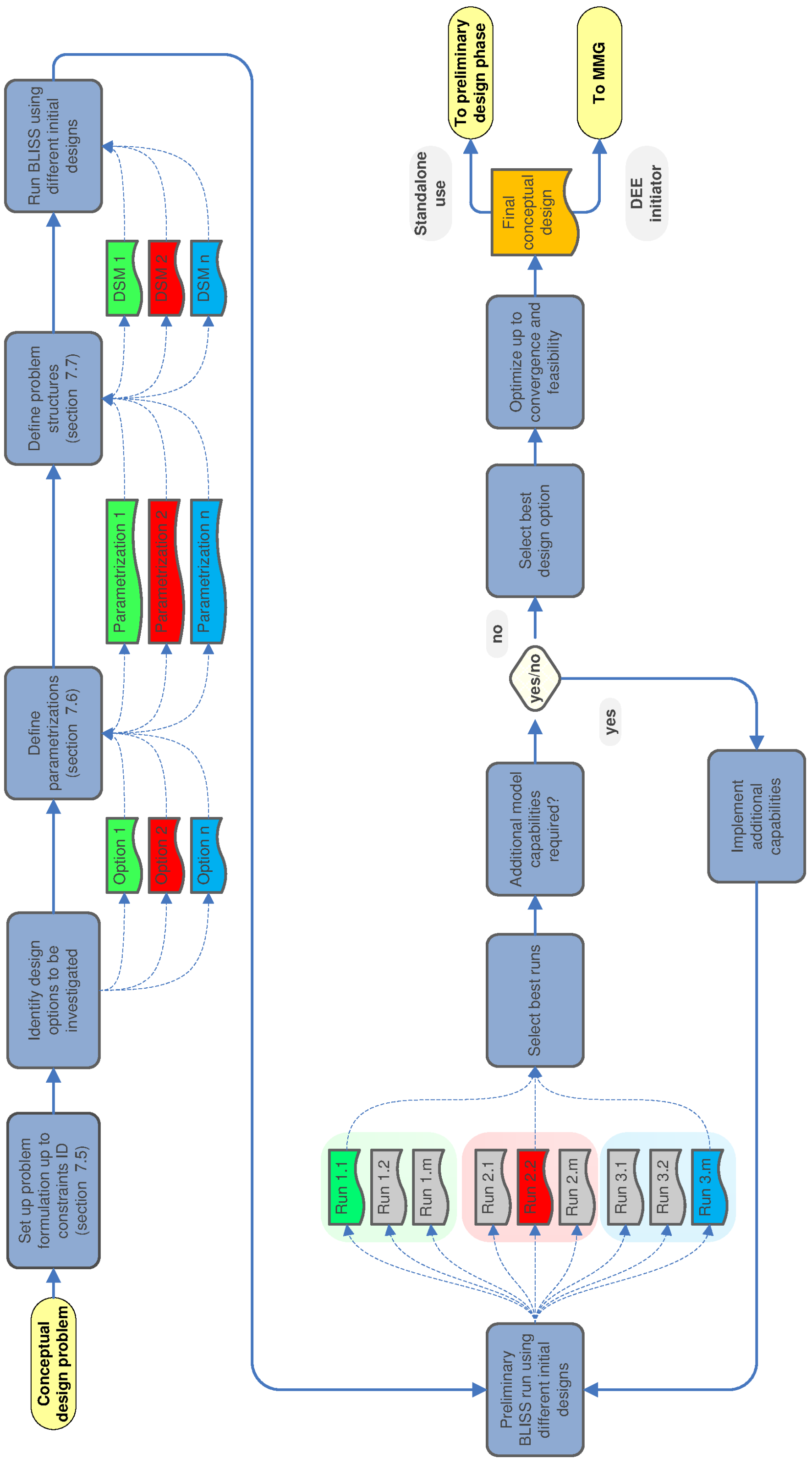

Conceptual design problem
Set up problem formulation up to constraints ID (section 7.5)
Identify design options to be investigated
Option 1
Option 2
Option n
Define parametrizations (section 7.6)
Parametrization 1
Parametrization 2
Parametrization n
Define problem structures (section 7.7)
DSM 1
DSM 2
DSM n
Run BLISS using different initial designs
Preliminary BLISS run using different initial designs
Run 1.1
Run 1.2
Run 1.m
Run 2.1
Run 2.2
Run 2.m
Run 3.1
Run 3.2
Run 3.m
Select best runs
Additional model capabilities required?
yes/no
yes
no
Implement additional capabilities
Select best design option
Optimize up to convergence and feasibility
Final conceptual design
Standalone use
DEE initiator
To preliminary design phase
To MMG

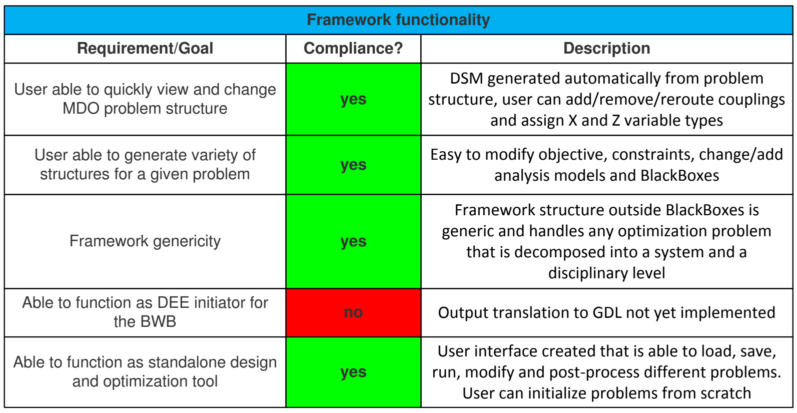
Figure 8-10: BLISS utilization plan

Chapter 9

Conclusions

MDO is emerging methodology, for which best practices and procedures need to be developed by trying different approaches and gaining experience , as happened originally in the early 20th century with the traditional aircraft design process. This research set out to create a framework which facilitates MDO using the BLISS strategy. MDO was envisaged as a way of dealing with complex design problems with strong couplings between the various design disciplines, and with no or little information available from statistics and handbook methods, based on design experience from the past. Therefore, the framework was given the task of solving a Blended Wing-Body design case. Besides the fact that the Blended Wing-Body aircraft is a highly integrated design, the magnitude of the design case formulated is, with almost 100 design variables larger than MDO sample cases in literature. In building up and validating the framework and the Blended Wing-Body design case, many interesting aspects of MDO problem structuring and optimization were found.

Table 9-1: Goals and requirements compliance: BLISS framework functionality



Framework functionality		
Requirement/Goal	**Compliance?**	**Description**
User able to quickly view and change MDO problem structure	yes	DSM generated automatically from problem structure, user can add/remove/reroute couplings and assign X and Z variable types
User able to generate variety of structures for a given problem	yes	Easy to modify objective, constraints, change/add analysis models and BlackBoxes
Framework genericity	yes	Framework structure outside BlackBoxes is generic and handles any optimization problem that is decomposed into a system and a disciplinary level
Able to function as DEE initiator for the BWB	no	Output translation to GDL not yet implemented
Able to function as standalone design and optimization tool	yes	User interface created that is able to load, save, run, modify and post-process different problems. User can initialize problems from scratch

Now, conclusions on both topics can be presented. First of all, tables 9-1 and 9-2 summarize the requirements identified for the BLISS framework and checks if the framework complies to them. Table 9-3 does the same for the Blended Wing-Body design case formulation and solution. In the research subgoal definition, it was stipulated that the conceptual design tool should be flexible, adaptable and accurate. Tables 9-1 and **??** show that these sub-goals have indeed been fulfilled.

Table 9-2: Goals and requirements compliance: BLISS framework performance

Framework performance		
Requirement/Goal	**Compliance?**	**Description**
Able to move from infeasible to feasible region	yes	Demonstrated by BWB and B747 problems, which started from an infeasible point.
BLISS implementation mathematically equal to to single optimizer approach	yes	Demonstrated for analytical test problem
Validate BLISS for different problems, structures, settings	yes	Convergence for all validation cases, ideal settings problem dependent, B747 solution similar to the actual aircraft
Able to realize objective improvement and/or feasibility	yes	Demonstrated for BWB and all validation testcases with complete constraint linearization
Robustness in convergence to final solution	yes	Final BWB BLISS run executed without intermediate modifications

Table 9-3: Goals and requirements compliance: Blended Wing-Body

Blended Wing Body design case		
Requirement/Goal	**Compliance?**	**Description**
Create roadmap for setting up large aircraft design problems in BLISS	yes	Step-by step procedure developed to start a design problem. Subjective choices are involved
Create feasible conceptual BWB design	partial	All constraints satisfied save for local Mach constraint on outboard wing
Tackle BWB-specific design challenges	partial	Some challenges could not be modeled and are hence not accounted for (3D flow effects, control power required, directional stability & control)
Reproduce known BWB design trends	yes	Stall and high local Mach issues concentrate at outboard wing, significant trim drag
Compute BWB solution on a normal PC	yes	Duration of 4 hours per cycle, 22 cycles to final design
Show that multilevel MDO can be find useful solutions for highly coupled problems with many independent variables (100+)	yes	BWB design case larger than known multilevel MDO testcases from literature

This research has attempted to show that while MDO problems require a large deal of preparation time, an immense amount of useful design information can be extracted from them once a problem is running. Also after the problem setup,the designer must take care that he/she continuously validates every modification that is made. From the design formulation process, it became clear indeed that MDO is a process of many iterations of validating analysis models, checking the variable definitions they pass onto each other, checking the complete MDA, changing the DSM, adding or interchanging models and so on.

The BLISS framework currently developed supports these activities and helps designers to maintain a helicopter view over the design problem. In the mean time, the tedious task of finetuning the design variables within the problem structure to optimize the system objective is performed by the framework.

Chapter 10 Recommendations

Moving from an idea to a working piece of software of respectable size is a large step. While the BLISS frameowork created during this research is able to converge and generate promising results for the Blended Wing-Body , that does not make it a fully-functional yet. Besides mathematical formulation or consistent implementation, a large portion of the success of the program in practice depends on its usefullness for an engineer, who has a complex (design) optimization problem he/she wants to decompose and solve. During the research, various directions for future work could be identified, of which the principal ones are summarized in this chapter.

- **Make the framework into competely functional standalone software**
  - Implementation of an error handling system that catches and rethrows errors, and makes the user aware of the source of error can improve usability for an engineer who knows his way around MDO, but is not necessarily acquainted with the inner workings of the BLISS framework.
  - Implementation of an interface in which the user can make all kinds of changes to the problem structure at hand (change variable type, change couplings) without having to manually modify m-files, which is error-prone.
- **Exploit parallel computing possibilities**
  - On a single 2011 PC, the Blended Wing-Body design is iterated in a matter of days, running all its components in series. It is expected that BLISS runtime for large problems can significantly be reduced by using its parallel computing capabilities in the BBSA and BBOpt components. To run in parallel mode,the framework requires changes in the implementation either on a single multicore PC or a network of PC's.
- **Increase the scope of the BLISS framework**
  - Make the BLISS framework compatible with a MultiModel Generator such as the Dar-Wing. The current analysis models used in the BLISS framework were custom-made. This allowed control over their implementation, their parameter definitions and conventions. Unfortunately, creating parametrizations for these models models this way has proven to be extremely time consuming. The MGG is dedicated to efficiently generating consistent parametrizations of a myriad of aircraft shapes
  - Implement a translation from the BLISS Blended Wing-Body model to GDL to allow it to pass the BLISS Blended Wing-Body to the DEE. Using the design optimized by BLISS, the DEE can continue design work on a more detailed level using its MMG, converger and evaluator

- **Extend the number of framework validation cases**
  - Investigate the effects of changing the problem structure for larger problems (now only for an analytical three-variable problem)
  - Idem for the allocation of design variables and division of the analysis models across BlackBoxes
  - Create benchmark cases where BLISS optimization results are compared to high-fidelity codes

- **Blended Wing-Body Model Refinement**
  - Add directional stability & control to adress the design challenges the Blended Wing-Body has on this area
  - Model 3D aerodynamics beyond VLM (VSAero/quasi 3D for viscous drag)
  - Integrate Flight Mechanics Model into the problem formulation to asses handling qualities, trimmability, eigenmotions etc. already in an early conceptual stage (Dircken (2008))
  - Integrate a model that accounts for the weight of the passenger cabin,
  - Include control actuator representation for dealing with control power issues

Bibliography

Anderson, J. (2001). *Fundamentals of aerodynamics*. McGraw Hill.

Balling, R., & Sobieszczanski-Sobieski, J. (1996). Optimization of coupled systems: A critical overview of approaches. *AIAA JOURNAL, 34*.

Belie, R. (2002). Non-technical barriers to multidisciplinary optimization in the aerospace industry. *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*.

Bertin, J. J., & Smith, M. L. (1998). *Aerodynamics for engineers*. Prentice Hall.

Boeing 747-400 airplane characteristics for airport planning. (2009, June 25). `http://rgl.faa.gov/Regulatory_and_Guidance_Library/`.

Carpentieri, G., Tooren, M. van, Morino, L., & Bernardini, G. (2004). Sequential multi-disciplinary optimization for the conceptual design of a blended-wing-body aircraft. *International Conference on Computational & Experimental Engineering*.

Cleanera. (2010, December 27). `http://nl.wikipedia.org/wiki/Luchthaven_Schiphol`.

Dircken, F. (2008). Controllability of blended wing body aircraft [Graduation Thesis].

Drela, M. (1989). Xfoil: An analysis and design system for low reynolds number airfoil. *Conference on Low Reynolds Number Airfoil Aerodynamic*.

Drela, M., & Giles, M. (1987). Viscous-inviscid analysis of transonic and low reynolds number airfoils. *AIAA Journal*, *10*(25).

E. Obert, P. ir., & R. Slingerland, D. ir. (2007). *Aerodynamic design of transport aircraft*.

Far. (2009, June 25). `http://rgl.faa.gov/Regulatory_and_Guidance_Library/`.

Frediani, A., Gasperini, M., Saporito, G., & A.Rimondi. (2003). Development of a prantl-plane aircraft configuration.

Geuskens, F., Koussios, S., Bergsma, O., & Beukers, A. (2008). Non- cylindrical pressure fuselages for future aircraft. *Structures, Structural Dynamics and Materials Conference*(AIAA 2008-1907).

Hamann, R., & Tooren, M. van. (2004). *Systems engineering & technical management techniques*.

Hendrich, T. (2009). Literature study on mdo in relation to conceptual design of the blended wing body.

Ho, Y., & Pepyne, D. (2002). Simple Explanation of the No-free-lunch Theorem and its Implications. *Journal of Optimization Theory and Applications*(115).

Icao: Act global. uniting aviation on climate change. (2011, April 2). `http://www.icao.int/Act_Global/`.

Katz, J., & Plotkin, J. (2001). *Low speed aerodynamics*. Cambridge Aerospace Series.

Kok, H. (2008). Quantitative assessment of a distributed propulsion system featuring boundary layer ingestion turbofan engines applied to a blended wing body aircraft. *DUT thesis work, faculty of Aerospace Engineering, DAR chair*.

Koning, J. H. (2010). *Development of a kbe application to support aerodynamic design and analysis: Towards a next-generation multi-model generator*. Graduation Thesis, Delft University of Technology, Delft.

Kurzk, J. (1995). Advanced user-friendly gas turbine performance calculations on a personal computer. *ASME*(ASME 95-GT-147).

Laban, M., Arendsen, P., Rouwhorst, W., & Vankan, W. (2002). A computational design engine for multidisciplinary optimisation with application to a blended wing body configuration. *American Institute of Aeronautics and Astronautics*(AIAA 2002-5446).

Liebeck, R. H. (2004). Design of the Blended-wing-body Subsonic Transport. *Journal of Aircraft*, *41*(1).

Liebeck, R. H., Page, M., & Rawdon, B. (1998). Design of the Blended-wing-body Subsonic Transport. *Journal of Aircraft*(AIAA A98-16309).

The mathworks, matlab and simulink for technical computing. (2009, September 1). `http://http://www.mathworks.com/`.

Megson, T. (1999). *Aircraft structures for engineering students*. Elsevier Butterworth Heinemann.

Melin, T. (2000). A vortex lattice matlab implementation for linear aerodynamic wing applications. *Royal Institute of Technology (KTH)*.

Morris, A., Arendsen, P., Rocca, G. L., Laban, M., Voss, R., & Honlinger, H. (2004). Mob - a european project on multidisciplinary design optimisation. *International Congress of the Aeronautical Sciences*.

Mulder, J., Staveren, W. van, Vaart, J. van der, & Weerdt, E. de. (2006). *Flight dynamics*.

Perez, R. E., Liu, H. H. T., & Behdinan, K. (2001). Applying the design structure matrix to system decomposition and integration problems: A review and new directions. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, *48*(3).

Perez, R. E., Liu, H. H. T., & Behdinan, K. (2004). Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design. *American Institute of Aeronautics and Astronautics*(AIAA 2004-4537).

Philips, W., & Snyder, D. (2000). Modern adaptation of prandtl's classic lifting-line theory. *Journal of Aircraft*, *37*(4).

Qina, N., Vavalleb, A., Moignea, A. L., Labanc, M., Hackettb, K., & Weinerfeltd, P. (2004). Aerodynamic considerations of blended wing body aircraft. *Progress in Aerospace Sciences 40*((2004) 321⊠343).

Raymer, D. P. (2002). *Enhancing Aircraft Conceptual Design using Multidisciplinary Optimization*. Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden.

Raymer, D. P. (2006). *Aircraft design: A conceptual approach*. American Institute of Aeronautics and Astronautics.

Rocca, G. L., Krakers, L., & Tooren, M. van. (2002). Development of an icad generative model for blended wing body aircraft design. *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*(AIAA 2002-5447).

Rocca, G. L., & Tooren, M. V. (2009). Knowledge-based engineering approach to support aircraft multidisciplinary design and optimization. *Journal of Aircraft*, *46*(6).

Roman, D., Allen, J. B., & Liebeck, R. (2000). Aerodynamic Design Challenges of the Blended-wing-body Subsonic Transport. *American Institute of Aeronautics and Astronautics*(AIAA 2000-4335).

Roman, D., Gilmore, R., & Wakayama, S. (2003). Aerodynamics of High-subsonic Blended-wing-body Configurations. *American Institute of Aeronautics and Astronautics*(AIAA 2003-554).

Rommens, M. (2008). Component weight estimation module. *DUT Aerospace faculty 'DAR exercise'*.

Ruijgrok, G. (1996). *Elements of aircraft performance*. Delft University Press.

Schut, E., & Tooren, M. van. (2008). Engineering primitives to reuse design process. *American Institute of Aeronautics and Astronautics*(AIAA 2008-1804).

Smith, H., & Yarf-Abbasi, A. (1999). The mob blended body reference aircraft. (G4RD-CT1999-0172).

Sobieski, I., & Kroo, I. (1996). Aircraft design using collaborative optimization. *American Institute of Aeronautics and Astronautics*.

Sobieski, J., Agte, J., & Jr., R. S. (1998). Bi-level integrated system synthesis (bliss). *National Aeronautics and Space Administration*(NASA/TM-1998-208715).

Soremekunt, G., Gurdal, Z., Haftkas, R. T., & Watson, L. T. (1996). Improving genetic algorithm efficiency and reliability in the design and optimization of composite structures. *American Institute of Aeronautics and Astronautics*(AIAA-96-4024-CP).

Tooren, M. van, Gerwen, D. van, & Steenhuizen, D. (2009). *Advanced design methodologies*.

Tooren, M. V., Rocca, G. L., Laan, A. H. van de, Berends, J., Cerulli, C., Schut, J., et al. (2005). Knowledge based engineering supported design.

Torenbeek, E. (1982). *Synthesis of subsonic airplane design*. Delft University Press.

Torenbeek, E. (1992). *Development and application of a comprehensive, design-sensitive weight prediction method for wing structures of transport category aircraft* (No. Report LR-693).

Tosserams, S., Hofkamp, A. T., Etman, L. F. P., & Rood, J. E. (2010). A specification language for problem partitioning in decomposition-based design optimization. *Structural and Multidisciplinary Optimization*, *42*(5).

Visser, W. (1995). Gas turbine engine simulation at nlr. *CEAS Symposium on Simulation Technology*.

Wakayama, S., & Kroo, I. (1998). The Challenge and Promise of Blended-wing-body Optimization. *American Institute of Aeronautics and Astronautics*(AIAA 1998-4736).

Weck, O. de, Agte, J., Sobieski, J., Arendsen, P., Morris, A., & Spieck, M. (2007). State-of-the-art and future trends in multidisciplinary design optimization. *American Institute of Aeronautics and Astronautics*(AIAA 2007-1905).

Appendix A SQP Algorithm

Sequential Quadratic Programming consists of two steps:

1. Finding a search direction $\mathbf{S}_q$ using the current design vector $\mathbf{X}_{q-1}$
2. Finding a new estimate $\mathbf{X}_q$ for the design vector by solving for α

The optimal search direction $\mathbf{S}_q^*$ is found by making a quadratic approximation of the objective function J and a linear approximation of the constraints g at the point $\mathbf{X}_{q-1}$ (see equations A-1 and A-2). As with all gradient-based algorithms, it is required that the objective function is smooth.

$$J(\mathbf{S}) = J(\mathbf{X}_{q-1}) + \nabla J(\mathbf{X}_{q-1})^T \mathbf{S} + \frac{1}{2}\mathbf{S}^T\mathbf{B}\mathbf{S} \tag{A-1}$$

$$g_i(\mathbf{S}) = g_i(\mathbf{X}_{q-1}) + \nabla g_i(\mathbf{X}_{q-1})^T \mathbf{S} \leq 0 \; for \; i = 1, \ldots, m \tag{A-2}$$

Equations A-1 and A-2 form a quadratic programming problem, which can be solved for S^*. This S* is then used as the new search direction. The B matrix is an approximation to the Hessian of the Lagrangian of the original, unapproximated problem. It is initialized as the identity matrix. A method to update B is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method which will not be elaborated here.

When the search direction $\mathbf{S}_q$ for iteration q is found, $\mathbf{X}_q$ is only a function of the scalar α and thus, the objective J also becomes a function of α and the resulting optimization problem is one-dimensional. The objective is augmented with a penalty function with parameter u_i, which must make sure that none of the constraints are violated during the search for the optimum α^*. The formulation of the augmented objective ϕ is given in equation A-3.

$$\phi(\alpha) = J(\mathbf{X}(\alpha)) + \sum_{i=1}^{m} u_i \; max\left[0, g_i(\mathbf{X}(\alpha))\right] \tag{A-3}$$

After finding α^*, it is used together with S_q to form $\mathbf{X}_q$:

$$\mathbf{X}_q = \mathbf{X}_{q-1} + \alpha^* S_q \tag{A-4}$$

This two-step procedure is iterated until the changes in **X** between cycles are small, and the process is said to be converged.

The built-in optimization routine of Matlab , `fmincon`, uses the Sequential Quadratic Programming algorithm to find a solution to a constrained minimization problem. It can handle both inequality and equality constraints, and bounds imposed on design variables. The `fmincon` routine also uses BFGS to update B. Sources for this overview of SQP were M. van Tooren et al. (2009) and *The Mathworks, Matlab and Simulink for Technical Computing* (2009).

Appendix B

Discipline Model Theory

B-1 Aerodynamics

B-1-1 Tornado

The Vortex Lattice Method (VLM) is extensively treated in literature, amongst others in Bertin & Smith (1998). It will now be briefly explained here. The VLM is essentially an extension of Prandtl's lifting line theory explained in Philips & Snyder (2000). This theory states that the inviscid, incompressible flow over a wing can be represented by one spanwise vortex filament and two trailing vortices starting at the wing tips and extend into infinity. All filaments have strength Γ. This arrangement is a so-called horseshoe-vortex. The velocity dV induced by a horseshoe vortex segment dl (or in fact any arrangement of vortex filaments) at a point which is located r away from dl, is given by the Biot-Savart law:

$$d\vec{V} = \frac{\Gamma_n \left(d\vec{l} \times \vec{r}\right)}{4\pi \left|\vec{r}\right|^2} \tag{B-1}$$

This can than be integrated to equation B-2, which gives the induced velocity of a vortex segment of arbitrary length (see Bertin & Smith (1998)).

$$\vec{V} = \frac{\Gamma_n}{4\pi} \frac{\vec{r}_1 \times \vec{r}_2}{\left[\vec{r}_1 \times \vec{r}_2\right]^2} \vec{r}_0 \cdot \frac{\vec{r}_1}{\vec{r}_1} - \frac{\vec{r}_2}{\left[\vec{r}_2\right]} \tag{B-2}$$

The nomenclature for equation B-2 is given in Figure B-1.

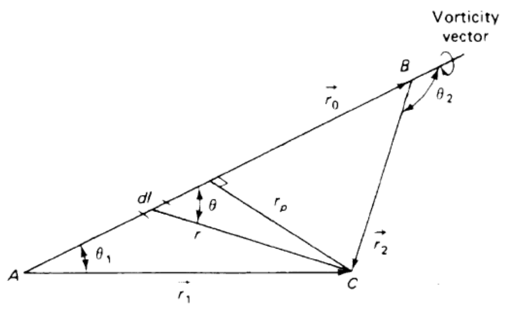


Figure B-1: Nomenclature for calculating the velocity induced by a finite vortex segment

In the VLM, a lifting surface (e.g. a finite wing or a horizontal tail) is divided into panels which each contain three vortex segments, together forming a horseshoe vortex starting at the panel quarter-chord point. Since a lifting surface is a solid object, there can be no flow normal to it. To prevent this from happening, each horseshoe vortex has a so-called collocation point where this boundary condition has to be satisfied (see Figure B-2 from Melin (2000)). This is done by balancing the velocity component at a collocation point normal to the panel surface induced by all vortex segments, with the normal free stream velocity component, which can be calculated using the flight condition (angle of attack, sideslip angle, air density).

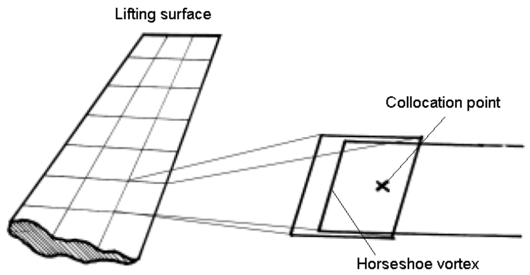


Figure B-2: Lifting surface division for the VLM

The unknown vortex strengths Γ can be solved with equation B-3 for an arbitrary number of panels. Here, w_{ij} is the velocity through panel j induced by vortex i, Γ_i is the vortex strength of vortex i and b_i is the flow through panel i determined by the flight condition.

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,n} \end{bmatrix} \cdot \begin{bmatrix} \Gamma_1 \\ \vdots \\ \Gamma_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \tag{B-3}$$

With the vortex strengths from equation B-3, the Kutta-Joukowski equation B-4 can be used to calculate the force vector acting on each panel, where l is the length of the vortex segment crossing that panel. All force vectors together determine both lift and induced drag. Zero lift (viscous drag) is determined in the Xfoil model (see B-1-2).

$$\vec{F} = \rho \left(\vec{V}_{ind} \times \vec{\Gamma} \right) \cdot l \tag{B-4}$$

The Tornado program extends the traditional VLM explained above. It can handle a wide varieties of geometries, including control surfaces, twist, dihedral and sweep, and allows multiple lifting surfaces to be used. Each lifting surface can be divided into partitions, making it possible to model wing trunks.In addition, it provides the option of using a free wake, meaning that the wake realigns with the free stream some distance downstream of the TE. This is done with a method described in Katz & Plotkin (2001), which will not be elaborated here. Each trunk can be assigned an inboard and outboard airfoil profile. Tornado extracts the camberline of these profiles (but not thickness) to create the vortex lattice.

Finally, the zero lift drag coefficient CD_0 is estimated using the Eckerts method of Raymer (2006). This method calculates the flat-plate estimation for the friction coefficient (B-29d) for each trunk using its Reynolds number based on its mean geometric chord. If however this number is lower than a cut-off value, the cut-off value is used.

$$c_{f_{turb}} = \frac{A}{\log Re^{2.58}}(1+0.144M^2)^{0.65} \tag{B-5a}$$

$$c_{f_{lam}} = \frac{1.328}{\sqrt{Re}} \tag{B-5b}$$

When transition occurs at a chordwise position X_{tr}, the total friction coefficient becomes:

$$c_{f,tot} = X_{tr}c_{f_{lam}} + (1-X_{tr})\,c_{f_{turb}} \tag{B-6}$$

The friction coefficient is then corrected with a so-called 'form-factor' FF to account for the shape of the wing trunk using its maximum thickness t_{max} and chordwise maximum thickness position $X_{t_{max}}$, assumed to be located either on the inboard or outboard airfoil. The form factor is calculated with $c_{f,tot}$, the trunk wetted area S_{wet} and S_{ref}:

$$FF = \left(1 + \frac{0.6}{x_{t_{max}}\,(t_{max})} + 100(t_{max})^4\right)\left(1.34M^{0.18}\cos\lambda\right)^{0.28} \tag{B-7}$$

With FF found, C_{D0} becomes

$$C_{D0} = \sum_{i=1}^{ntrunks} c_{f,tot_i}FF_i\frac{S_{wet,i}}{S_{ref}} \tag{B-8}$$

Fianlly, Tornado calculates the maximum lift coefficient by running its VLM method while gradually increasing AOA. If locally on the span the lift coefficient becomes larger than a threshold value, the aircraft is considered to be stalled. In the Blended Wing-Body model, this value is supplied as the smallest $C_{l_{max}}$ value of all airfoils, which is computed by Xfoil.

B-1-2 Xfoil

In the 2D aerodynamics BlackBox, Xfoil analysis provides the 2D drag coefficients of the Blended Wing-Body airfoils. The program uses a linear-vorticity stream function panel method for inviscid, incompressible modeling of 2D flow around an airfoil. This formulation is corrected for compressibility using the Karman-Tsien compressibility correction. To account for viscous behaviour (boundary layer, local separation, bubbles), Xfoil uses a two-equation lagged dissipation integral BL formulation. Giving a full explanation of Xfoil is beyond the scope of this project, but elaborate information on the theory behind Xfoil can be found in Drela (1989) and Drela & Giles (1987).

Xfoil is run to see if drag divergence does not occur during cruise flight. For that, the minimum c_p value found from Xfoil airfoil analysis of the cruise condition is compared into the following equation for the critical pressure coefficient[1] from Anderson (2001):

$$C_{p,crit} = \frac{2}{\gamma M_{cr}}\left[\left(\frac{1+\frac{\gamma-1}{2}M_{cr}^2}{1+\frac{\gamma-1}{2}}\right)^{\frac{\gamma-1}{\gamma}}\right] \tag{B-9}$$

When a high sectional lift coefficient is fed to Xfoil, its viscous module may not converge. When this happens, the calculation is repeated with a lower C_l until Xfoil converges. This prevents Xfoil from crashing the entire BLISS routine.

[1]The highest (least negative) pressure coefficient for which flow across the wing becomes locally supersonic

B-1-3 Trim & Balance

Formulae from Mulder et al. (2006) are used to compute the required angle-of-attack and control surface deflection at each flight condition. All quantities in this section are calculated for an arbitrary flight condition 'i'.

The lift required to balance weight at condition *i* is found as:

$$C_{L_{req}} = \frac{W_i}{q_i S} \cos \gamma_i \tag{B-10}$$

where

$$W_i = W_{f,i} + W_p + W_{empty} \tag{B-11}$$

Since the lift is a function of angle-of-attack in the aerodynamic data calculated by Tornado, the required angle-of-attack α_{req} can be found as:

$$C_L = C_L(\alpha) \Rightarrow C_{L_{req}} = C_L(\alpha_{req}) \tag{B-12}$$

With this angle, the derivatives C_{m_0}, C_{m_δ} and C_{m_a} can be found, which are necessary for calculating the required elevator deflection angle δ_e:

$$\delta_e = \frac{-1}{C_{m_\delta}} \left[C_{m_0} + C_{m_{thrust}} + C_{m_a}(\alpha_{req} - \alpha_0) \right] \tag{B-13}$$

This delfection angle is assumed equal for all TE control surfaces. In reality, this restriction is not imposed, but devising a deflection schedule for the individual trunk control surfaces is considered beyond the scope of this project. Now that ... is found, the trim drag can be calculated by taking the components of C_{x_δ} and C_{z_δ} in the direction of flight:

$$C_{D_{trim}} = -\left(C_{x_\delta} \cos \alpha_{req} + C_{z_\delta} \sin \alpha_{req} \right) \delta_e \tag{B-14}$$

The total drag coefficient now becomes:

$$C_D = C_{D_i} + C_{D_0} + C_{D_{trim}} \tag{B-15}$$

The required thrust is calculated as:

$$T_i = q_i S C_D + W_i \sin \gamma_i \tag{B-16}$$

In equation B-13, $C_{m_{thrust}}$ is found as :

$$C_{m_{thrust}} = \frac{T_i l_{thrust}}{q_i S c_{mac}} \tag{B-17}$$

where l_{thrust} is the thrust arm with respect to the aircraft c.o.g.

B-2 Structures

B-2-1 Structural Strength

The Blended Wing-Body is modeled structurally using the principle of structural idealization, which is suitable for thin-walled structures. This principle is accurately described in Megson (1999). Using this principle, the calculations normal stresses and shear flows within the BWB structure is simplified to such an extent that they become analytical. In structural idealization, the following assumptions are made:

- The actual structural elements (skin, stringers, ribs, spars) are represented by booms and plates
- The booms replace the stringers and carry all normal stresses σ
- The plates replace the aircraft skin and carry all shear stresses τ
- Boom cross-sectional area and plate thickness are constant across trunk span
- Torsion moments are small w.r.t. bending moments and can be neglected
- The external shear force due to lift is acting at the quarter-chord line
- The structural and content weight are linearly distributed

Figure B-3 shows an actual wing cross-section with loads and its idealized counterpart.

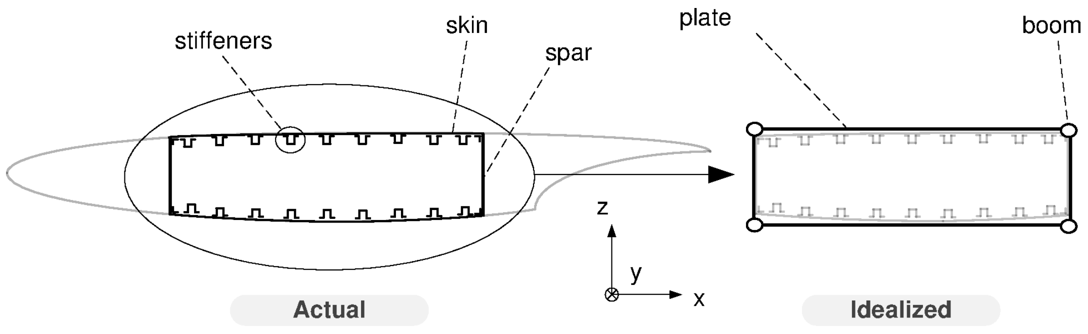


Figure B-3: Idealization of wingbox (cross-section view)

The idealized and actual wingbox cross-sections have these traits in common:

- Same total cross-sectional area
- Same enclosed area
- Same average thickness

Each boom has a specified cross-sectional area, and it is assumed that the normal stress across this area is constant for a given section. Figure B-4 taken from Megson (1999) gives an idea of the effect of idealization on a square panel. It can be seen from this figure, that idealization greatly simplifies structural modeling. However, this approximation will come at the cost of a decreased accuracy of the calculations. In the Blended Wing-Body and B747 structural models, the number of booms in the idealized wingbox is set to four.

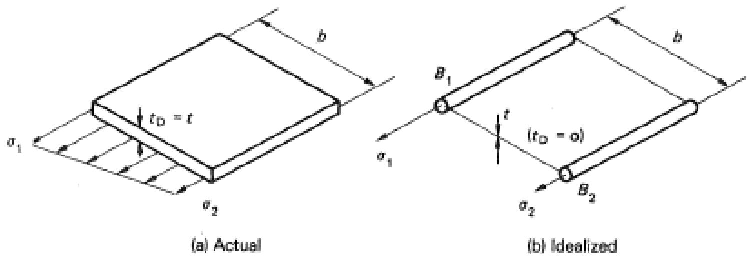


Figure B-4: Idealization of a panel

Since normal stresses are only carried by the booms, the shear flow q_{ij} between two booms i and j will be constant (see equation 9.74 of Megson (1999)). The moments of inertia of an idealized section are entirely due to the boom areas, as for this purpose the effective skin thickness is assumed to be zero.

According to Megson (1999), equation B-18 can be used for calculating the shear flows in an idealized section with given normal and shear loads.

$$S_x\eta_0 - S_y\xi_0 = \oint q_b p \frac{ds}{t} + 2Aq_{s,0} - \sum_{r=1}^{m} P_{x,r}\eta_r + \sum_{r=1}^{m} P_{y,r}\xi_r \tag{B-18}$$

where $S_x = 0$ and S_y is acting through the airfoil local center of pressure, assumed to be at 25% of the local airfoil aerodynamic chord.

The geometry belonging to equation B-18 is given in Figure B-5 from Megson (1999).

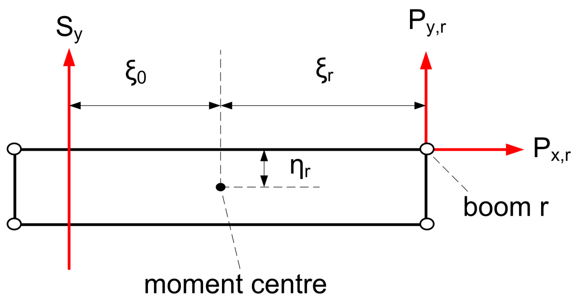


Figure B-5: Idealized wing cross-section subjected to shear and normal loads

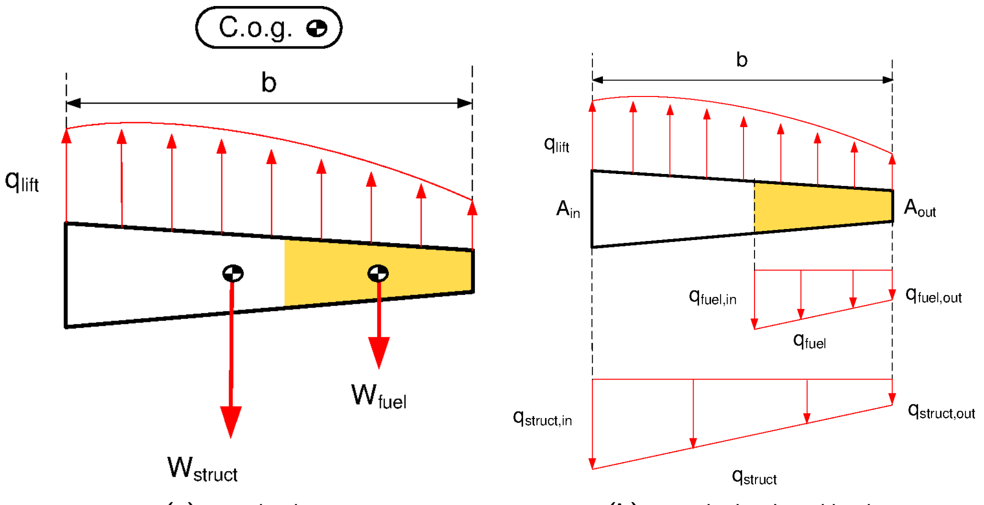
The internal bending moment M and shear force V in a spanwise cross-section is built up from two components:

- Aerodynamic forces (lift)
- Inertial forces (structural, fuel and payload weight)

Each contribution is modeled as a distributed spanwise load. The aerodynamic loads are fed to the structural model directly from Tornado. For the inetial loads, it is assumed that their distribution q is linear. This distribution is calculated by 'spreading' the total load (e.g. total structural weight in a trunk) across the trunk span, where the ratio between the inboard and outboard value of q is equal to the ratio of the respective cross-sectional areas, and the total magnitude equal to the point load it represents. So for instance:

$$\frac{q_{struct,in}}{q_{struct,out}} = \frac{A_{in}}{A_{out}} \tag{B-19}$$

Figure **??** shows how this is done for structural and fuel weight.



(a) Point loads **(b)** Linearly distributed loads

Figure B-6: Inertial loads acting on a Blended Wing-Body trunk

By integrating twice, M and V can be calculated from q as:

$$\begin{aligned} V(x) &= -\int q(x)dx \\ M(x) &= \int V(x)dx \end{aligned} \tag{B-20}$$

B-2-2 Structural Weight

The Torenbeek method is a method for determining the structural weight of the wing of a conventional transport aircraft. Detailed information is found in Torenbeek (1982). The method divides the wing structure in a primary and secondary. Only the primary part is load-carrying. The primary part is then subdivided into:

- The actual structural elements (skin, stringers, ribs, spars) are represented by booms and plates
- Optimum weight, the amount material needed to carry al the loads
- Non-optimum weight, several weight penalties for presence of cut-outs, joints etc.

The wing weight subdivision is given in Figure B-7, taken from Torenbeek (1982)

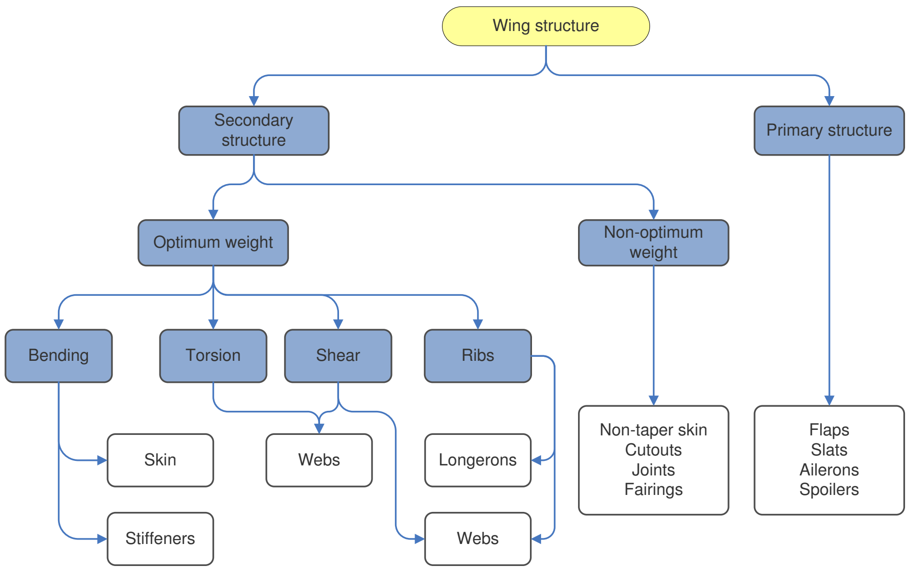


Figure B-7: Wing weight subdivision

For each component, Torenbeek (1982) defines a method of finding its weight. These methods can be:

- **Empirical**: based entirely on statistical data, related to conventional transport A/C
- **Semi-empirical**: based on both statistical data and design geometry of a conventional transport A/C

Table B-1 qualitatively describes the method used for each component in Figure B-7.

#	Weight component	Method type	Description
1	Material required to resist bending, shear	Semi-empirical	Material weight of equivalent structure resisting actual loads, accounting for gust loads
2	Material required to resist torsion	Empirical	Educated guess of stress level increase due to torsion
3	Rib weight	Semi-empirical	Function of mean profile thickness
4	Fixed LE and TE weight	Empirical	Fixed weight per m^2 from statistics
5	Control surface weight	Empirical	Fixed weight per m^2 from statistics

Table B-1: Methods for structural component weight estimation

The fuel weight is calculated with the fuel density and the fuel volume V_f:

$$W_f = \rho_f V_f g \tag{B-21}$$

The first time structural weight is calculated, it is started from a fixed initial value. Since running Torenbeek changes the structural weight, changing the spanwise bending moment and shear force due to structural weight, Both Torenbeek and the idealized structures method are iterated until the structural weight converges.

B-2-3 Loads & Center of Gravity

Since the wingbox has constant taper across each trunk, it is assumed that fuel, payload and structural weight manifest themselves as linearly distributed loads. Aerodynamic loads are take from the aerodynamics discipline and can have any shape. It is assumed that both the structural and fuel- or payload c.o.g. of a wing trunk lie on the centroid of the structure and the fuel or payload present in the trunk (see Figure B-6).

B-3 Propulsion

The engine model used for BWB optimization was originally created for a thesis specifically on BWB engine design by Kok (2008). To assess the efficiency of three conventional, large turbofans with respect to many small engines employing BLI[2], a Gasturbine Simulation Tool (GST) was developed in Matlab and coupled to a genetic algorithm (GA). This tool is extensively explained and validated in Kok (2008), appendix B.

B-3-1 Gasturbine Simulation Tool

This tool was modified and used within the BLISS propulsion BB. Due to time constraints, it was convenient to leave the structure for generating the population for the GA intact, and let it define a population of one individual.

The GST analyzes the engine performance at two kinds of user-defined operating conditions:

- One design point, typically the cruise condition. Cruise mass flow and fuel required are iterated until the desired cruise range is attained
- A number of off-design conditions, typically climb, take-off and descent. the off design module is iterated until a predefined error function converges to zero using Newton-Rahpson iteration (see Kok (2008))

For both condition types the engine is considered to be in a steady state. The cruise range is found using the Breguet range euqation:

$$R_{cr} = V_{cr} C_t \frac{L}{D}_{cr} ln \frac{W_{MTOGW}}{W_{MTOGW} - W_{fuel}} \tag{B-22}$$

Design and off-design conditions are defined by the user by their Mach number, flight altitude and combustion chamber temperature. The GST builds the engine with four independent variables:

- By-pass ratio
- Fan pressure ratio
- Overall pressure ratio
- Combustion chamber temperature

The station numbering in the GST tool is given in Figure B-8.

[2]BLI: Boundary Layer Ingestion

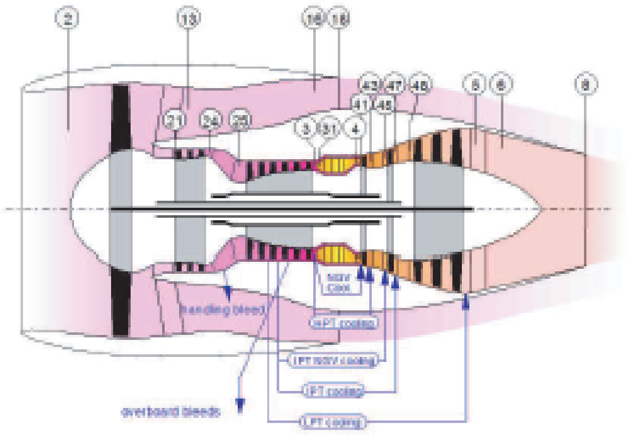


Figure B-8: Engine station numbering

The Kok (2008) research validated the GST with two other tools, GSP and GasTurb and came up with satisfactory results. In the design point, the maximum installed thrust difference between GST and GSP/Gasturb was found to be 0.7%. In the off-design conditions (climb, take-off and descent) this difference was 4%.

B-3-2 Airfield performance

Besides the engine characteristics, the propulsion BB also calculates the airfield performance of the BWB. The BWB is checked for:

- Sufficiently large climb angle, OEI
- Balanced field length, AEO

First of all, the take-off stall speed $V_{stall,TO}$ is calculated as:

$$V_{stall,TO} = \sqrt{\frac{W}{S}\frac{2}{\rho}\frac{1}{C_{L_{max}}}} \tag{B-23}$$

Then, the minimum climb speed V_{climb} and the lift-off speed V_{LOF} according to FAR regulations are defined as:

$$V_{climb} = 1.2V_{stall,TO} \tag{B-24a}$$
$$V_{LOF} = 1.1V_{stall,TO} \tag{B-24b}$$

To check the FAR OEI requirements, the take-off thrust with OEI $T_{TO,OEI}$ is calculated in equation B-25. It is assumed that $T_{TO,OEI}$ is also the maximum attainable thrust.

$$T_{TO,OEI} = T_{TO}\frac{N_{eng} - 1}{N_{eng}} \tag{B-25}$$

Raymer (2006) provides a formula for averaging the thrust throughout the take-off phase:

$$T_{av} = T_{TO} \frac{5 + BPR}{4 + BPR} \tag{B-26}$$

The maximum climb angle γ_{climb} is given equation B-27 from **?**.

$$\gamma_{climb} = \text{arcsin} \frac{(T_{TO,OEI} - D_{climb})}{W_{MTOGW}} \tag{B-27}$$

Now, the balanced field length is calculated according to Raymer (2006) by the empirical equation:

$$\begin{aligned} BFL = \left[\frac{0.863}{1 + 2.3G_{climb}}\right] &\left[\frac{\frac{W_{MTOGW}}{S_{ref}}}{\rho_{airfield} g_0 C_{L_{climb}}} + h_{obstacle}\right] \\ &\frac{1}{\left(\frac{T_{av}}{W_{MTOGW}} - U\right) + 2.7} + \left[\frac{655}{\frac{\rho_{airfield}}{\rho_0}}\right]^{0.5} \end{aligned} \tag{B-28}$$

with:

$$G_{climb} = \gamma_{climb} - \gamma_{min,FAR} \tag{B-29a}$$
$$U = 0.01CL_{max} + 0.02 \tag{B-29b}$$
$$C_{L_{climb}} = \frac{W_{MTOGW} \cos \gamma_{climb}}{\frac{1}{2} \rho_{airfield} V_{climb}^2 S_{ref}} \tag{B-29c}$$
$$h_{obstacle} = 35\text{ft} \tag{B-29d}$$

B-4 Weight

The aircraft weight is divided into structural weight W_{struct}, systems weight W_{sys}, fuel weight W_f, engine weight W_{eng} and payload weight W_p:

$$W_{ac} = W_{struct} + W_f + W_p + W_{sys} + W_{eng} \tag{B-30}$$

The MTOGW is the aircraft weight with full fuel tanks and maximum payload. All weight components are constant except the fuel weight, which will vary during the flight.

The systems weight is approximated with the empirical method of Raymer (2006)[3]. Since the locations of the aircraft systems are not determined, the overall systems c.g. is assumed to be equal to the empty weight c.g.

[3]This method is formulated in British units

$$W_{fuelsystem} = 2.405 (V_t)^{0.606} \left(1 + \frac{V_i}{V_t}\right)^{-1.0} \left(1 + \frac{V_p}{V_t}\right) (N_t)^{0.5} \tag{B-31a}$$

$$W_{flightcontrols} = 145.9 (N_f)^{0.554} \left(1 + \frac{N_m}{N_f}\right)^{-1.0} (S_{cs})^{0.20} \left(I_y 10^{-6}\right)^{0.07} \tag{B-31b}$$

$$W_{instruments} = 4.509 K_r K_{tp} (N_c)^{0.541} N_{en} (L_f + B_w)^{0.5} \tag{B-31c}$$

$$W_{hydraulics} = 0.2673 N_f (L_f + B_w)^{0.937} \tag{B-31d}$$

$$W_{electrical} = 7.291 (R_{kva})^{0.782} (L_a)^{0.346} (N_{gen})^{0.10} \tag{B-31e}$$

$$W_{avionics} = 1.73 W_{uav} (N_t)^{0.983} \tag{B-31f}$$

$$W_{furnishings} = 0.0577 (N_c)^{0.1} (W_{cargo})^{0.393} (S_f)^{0.75} \tag{B-31g}$$

$$W_{anti-ice} = 0.002 W_{dg} \tag{B-31h}$$

$$W_{handlinggear} = 3.010^{-4} W_{dg} \tag{B-31i}$$

where:

Table B-2: Systems weight variables

Variable	Description	Value	Units
L_a	Electric routing distance, generators - cockpit - avionics	Equal to 2x root chord	ft
L_f	Fuselage total length	Equal to root chord	ft
S_f	Fuselage wetted area	variable	ft^2
S_{cs}	Total control surface area	variable	ft^2
B_w	Wing span	variable	ft^2
K_r	reciprocating engine correction factor	1	-
K_{tp}	turboprop engine correction factor	1	-
N_{en}	Number of engines	3	-
N_m	Number of mechanical functions of controls		-
N_p	Number of persons on board	400	-
N_c	Number of crew	3	-
N_f	Number of functions performed by controls		-
N_{gen}	Number of generators	3	-
N_t	Number of fuel tanks	8	-
R_{kva}	Systems electrical rating	60	V
V_i	Integral tank volume	varaiable	ft^3
V_t	Total tank volume	equal to V_i	ft^3
V_p	Protected tank volume	equal to V_i	ft^3
W_{dg}	Design gross weight	variable	N
I_y	Moment of inertia around y-axis	variable	lb ft^3

The W_{struct} is calculated with the method of **?** using the boom areas, plate thicknesses and dimensions in the structural discipline. W_p is considered fixed and equal to the payload weight of a large conventional airliner. The payload is located only in the centerbody of the BWB. Fuel is stored only in the outboard wing, and not in the winglets and centerbody.

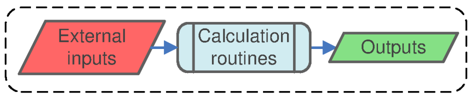
The aircraft center of gravity x_{cg} is calculated by taking the moments of all separate weight components with respect to one reference point:

$$W_{ac} x_{cg} = W_{struct} x_{struct} + W_f x_f + W_p x_p + W_{sys} x_{sys} + W_{eng} x_{eng} \tag{B-32}$$

The center of gravity in z-direction is calculated analogously to equation B-32. It is assumed that the c.g. y-position is always on the aircraft centerline. The systems weight will be predicted using a handbook method such as Torenbeek (1982), under the assumption that, though the BWB is a new configuration, it will still have the same systems layout as a conventional aircraft (e.g. hydraulics, de-icing, airconditioning). A project which was made within the Aerospace DAR chair will serve as a reference. This project (see Rommens (2008)) offers four weight estimation methods in digital format, among which the one of Torenbeek (1982). Only the parts predicting the systems weight of an aircraftv will be used, since the methods are based on empirical data of conventional aircraft, and applying them to the BWB would not be realistic.

Appendix C
Discipline Model Flowcharts

C-1 Aerodynamics



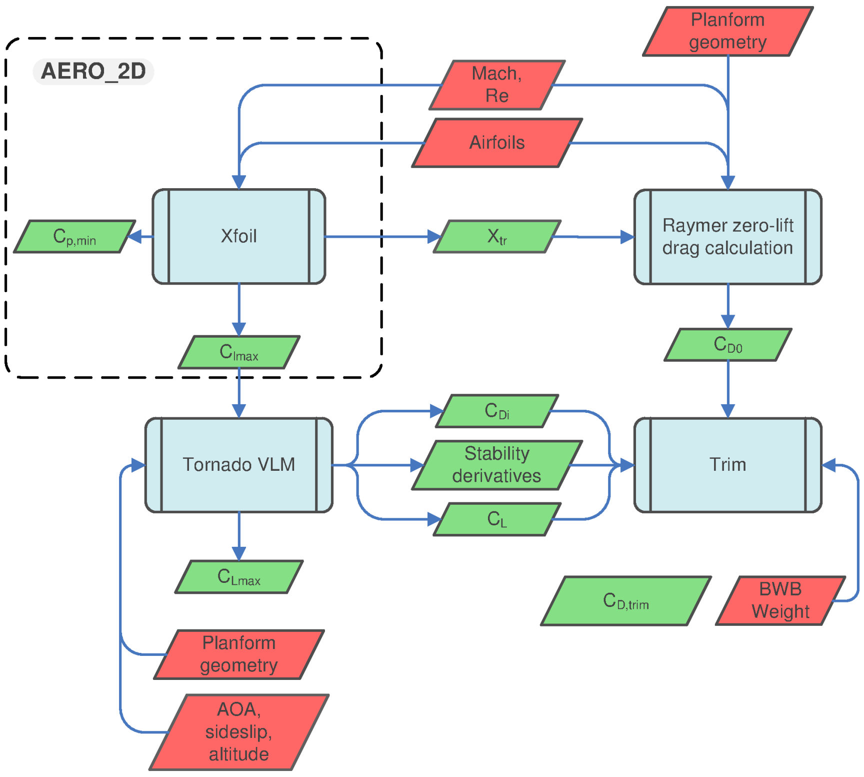


Figure C-1: Flowchart of aerodynamics discipline

C-2 Structures

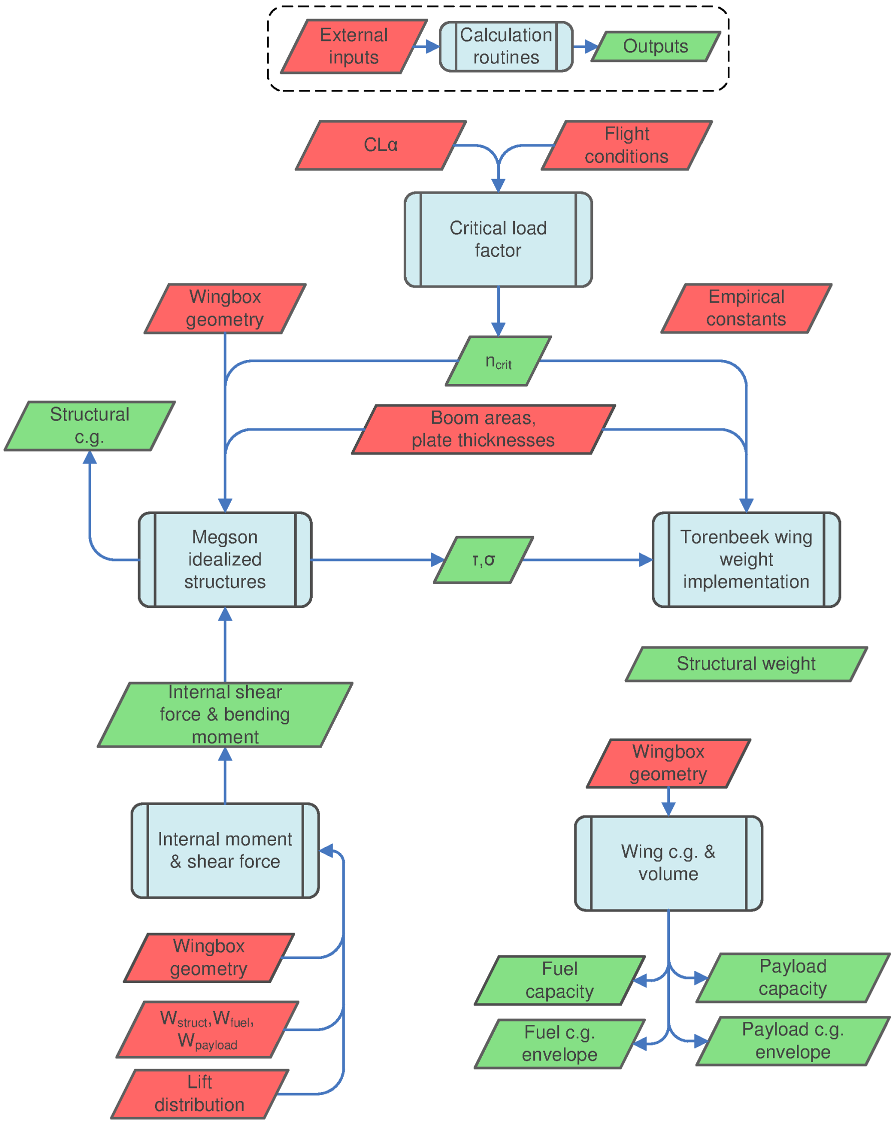


Figure C-2: Flowchart of structures discipline

C-3 Propulsion

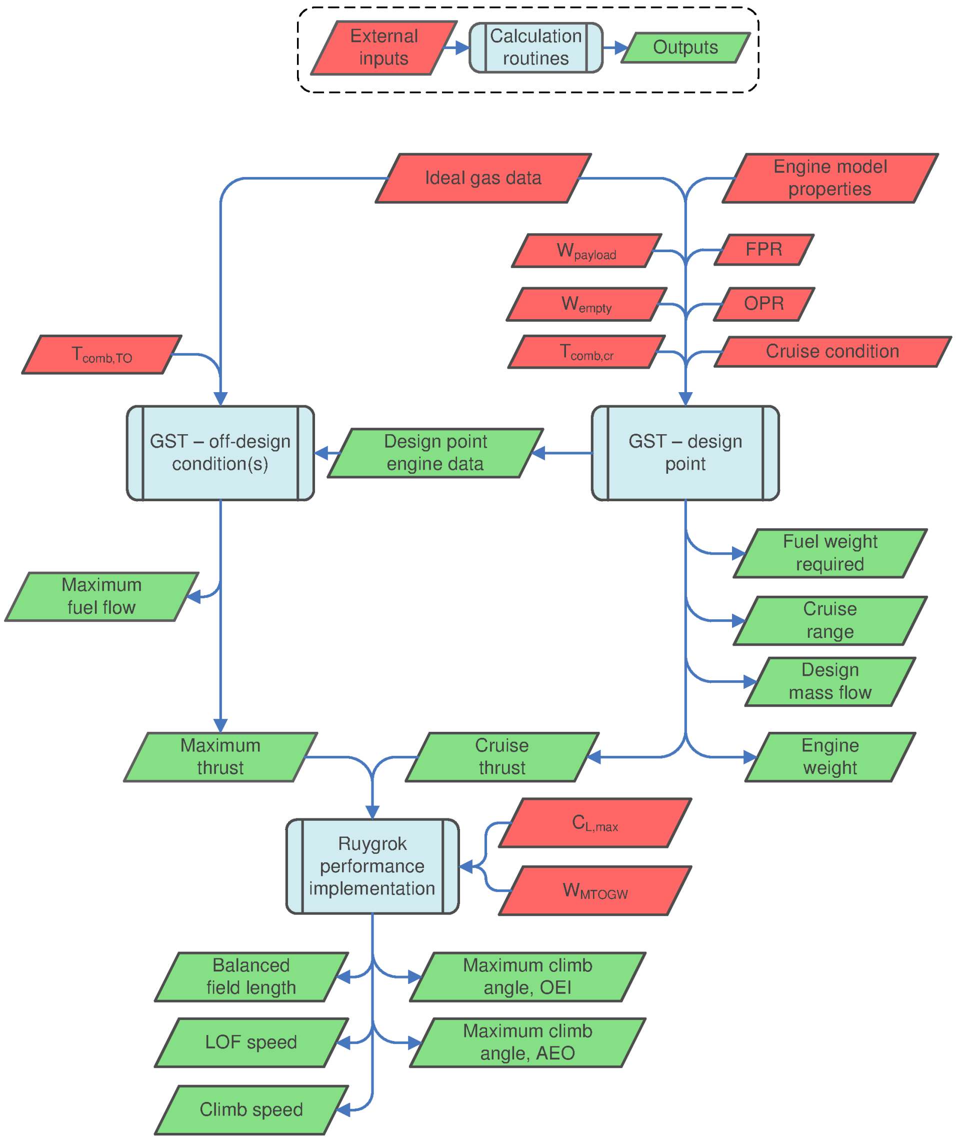


Figure C-3: Flowchart of propulsion discipline

C-4 Weight

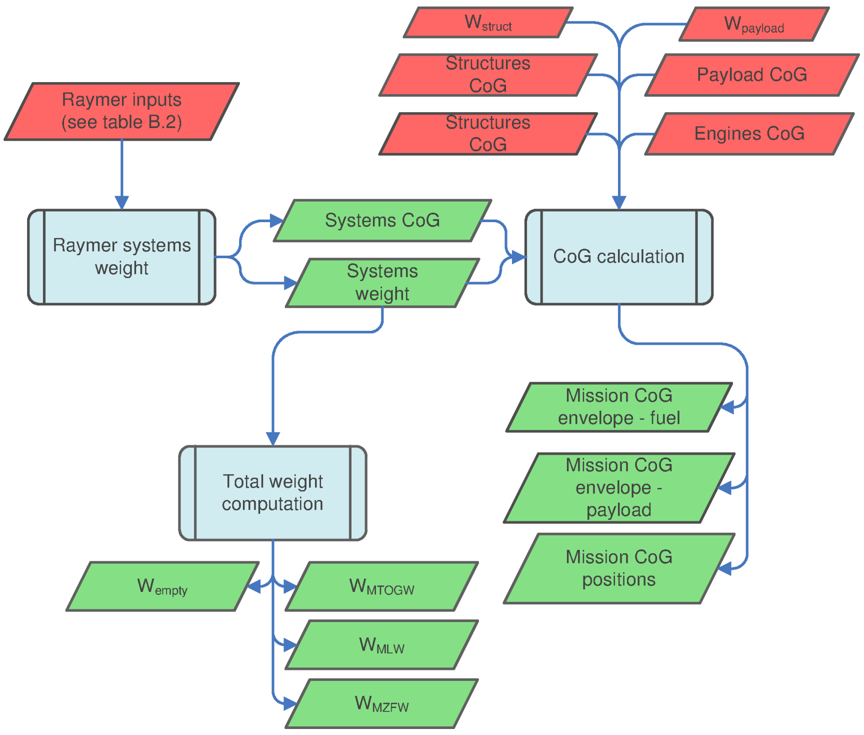


Figure C-4: Flowchart of weight estimation discipline

Appendix D

BLISS FrameWork Manual

This chapter is a summary of the most important capabilities of the BLISS framework. For a more elaborate overview, please consult the README file.

D-1 BLISS Main Menu

A screenshot of the BLISS main menu:

```
"""""""""""""""""""""""""""""""""""""""""""""""""""""""
"  BLISS MDO problem solver  Version 1                "
"                                                     "
"  Developed by Tijl Hendrich, 2010                   "
"""""""""""""""""""""""""""""""""""""""""""""""""""""""

 Type '?' for command list....

        BLISS: ?

"""""""""""""""""""""""""
"  Available commands   "
"""""""""""""""""""""""""

 Problem definition.

        [prob]. Choose BLISS problem
        [init]. Restart current BLISS problem

 Problem structure.

        [DSM ]. Create Design Structure Matrix
        [sdef]. Define new problem structure
        [scon]. Check structure consistency
        [BBIO]. Show required BlackBox I/O
        [side]. Print side constraints

 BLISS settings.
```

```
       [cset]. Display current settings
       [nset]. Set new run settings
       [lset]. Load existing run settings

 BLISS optimization run.

       [run ]. Run complete BLISS routine
     [SA  ]. Run System Analysis
     [BBSA]. Run BlackBox Sensitivity Analysis
     [SSA ]. Run System Sensitivity Analysis
     [BBOp]. Run BlackBox Optimization
     [SysO]. Run System Optimization
     [DXDZ]. Update design vector
       [rloa]. Load BLISS results file
       [rsav]. Save current BLISS results
       [reve]. Revert BLISS result file to previous cycle
       [smod]. Change BLISS problem variables and/or structure

 BLISS post-processing.

       [outp]. List available outputs
       [prin]. List current problem variables
       [feas]. Show feasibility information
       [summ]. General run results
       [stat]. Current problem status
       [hist]. Problem variable history
       [sens]. Sensitivity plot

 Miscellaneous.

       [clea]. Clear current results, settings, problem
       [keyb]. Keyboard access

       [exit]. Exit BLISS
```

D-2 Starting BLISS

1. Start Matlab , min. version 2008b
2. Set the Matlab working directory to the main BLISS directory (C:/../../BLISS)
3. Type 'BLISS' to start the program
4. Type '?' to see a list of available commands

D-3 Defining a New Problem

Prerequisites:

1. The problem folder is located in the 'BLISS' main folder
2. The problem has the same name as the folder it is located in
3. All necessary models are located in this folder (subfolders allowed)
4. All settings (e.g. mesh sizes, calculation methods, tolerances) are located in a file 'BB_settings' in the problem folder
5. All constraints are located in a file 'BB_constraints' in the problem folder

6. The formulation process of chapter 7 was followed to identify the desired problem structure
7. Each model is wrapped in Matlab using the '[**Y**,**G**] = BB_(model name)(X,Y,Z,CONSTANTS)' format.

Steps towards definition (manual):

1. Define BLISS settings in the 'SETTINGS' array by using the '***SetBLISSConfig***' command. The framework will request desired values for each setting from the user.
2. Constant parameters are defined in the 'CONSTANTS' array as 'CONSTANTS.(*constant-name*) = value'
3. If the problem is aircraft related, flight conditions are defined in the 'FCOND' array as 'FCOND.(*condition-name*) = value'
4. Define variable names that are inserted as X-variables in X_{in}.(*BBname*) for each BlackBox *BBname*. Names in X_{in}.(*BBname*) must correspond to names inside $BB\_$(*BBname*))
5. Define Z-variable input names in Z_{in}.(*BBname*). Names in Z_{in}.(*BBname*) must correspond to names inside each BlackBox it is used
6. Define Y-variable output names in Y_{out}.(*BBname*). Names in Y_{out}.(*BBname*) must correspond to names inside BlackBox it is used *BBname*
7. Define couplings between BlackBoxes using the command '***DefineCouplings***'. Only variable names in Y_out can be used. The framework will ask for couplings from each discipline to each discipline. They are stored in the format *couplings.(to).(from)*.
8. Initialize X, Y and Z using the command '***InitializeXYZ***'. The program will prompt for values for each the variable names in step 3, 4, and 5. The user is responsible for assigning each variable the correct number of elements.
9. Now, the BLISS program will execute a system analysis. If this system analysis is able to converge, the BLISS algorithm can be started. Otherwise, errors are present in the problem initialization and the user should review them.

D-4 Generic Problem Initialization File

A generic example of a problem formulation is found in the file *start_new_problem.m*, which is repeated below. With the help of this file, and the appropriate models wrapped in BlackBoxes, a BLISS MDO problem can be created.

```

%% Template file for creating an initialization file for the BLISS
%% framework.
%% Warning: this file cannot be run!


%% All locations where user input is required, are marked with a cell
%% separator. Other locations are not to be changed.


%% BLISS settings
SETTINGS.disc_names    = {'name BlackBox 1' 'name BlackBox 2' 'etc' 'OBJ'}; % BB names,
    order determnies order of SA iteration. must always contain OBJ
SETTINGS.X_move_limit = xx; % number > 0, 1 == 100%
SETTINGS.Z_move_limit = xx; % number > 0, 1 == 100%
SETTINGS.BBSA_step     = xx; % number << 1, e.g. 1e-8
SETTINGS.BBSA_method   = xx; % forward or central
SETTINGS.BBSA_order    = xx; % 1 (forward) or 2 (central)
SETTINGS.constraint_method_BB = xx; % 1 = linearized constraints, 0 = actual constraints
SETTINGS.constraint_method_Sys = xx; % 1 = linearized constraints , 0 = actual
    constraints
SETTINGS.TolY = xx; % number <<1
SETTINGS.StopTolerance = xx;  % number <<1


SETTINGS.BLISS_filename = 'enter name here for saved designs';

SETTINGS.BB_Opt_options = optimset('use any of the fmincon settings , see matlab help');
    % BB optimizer options
SETTINGS.Sys_Opt_options = optimset('use any of the fmincon settings , see matlab help');
     % System optimizer options

SETTINGS.G_yz = []; % constraint indices to be evaluated as system constraints. leave
    empty if unknown
%% DONT CHANGE
struct2var(SETTINGS);

%% Define X and Z varnames (BB input) and Y varnames (BB output)

% input X
X_in.(BlackBox_1) = {'variable name 1' 'variable name 2' 'variable name 3'};
X_in.(BlackBox_2) = {'variable name 1' 'variable name 2'};
X_in.(etcetera) = []; % BB without X variables

% output Y
Y_out.(BlackBox_1) = {'variable name 1' 'variable name 2' 'variable name 3'};
Y_out.(BlackBox_2) = {'variable name 1' 'variable name 2'};
Y_out.(etcetera) = {}; % BB without Y variables don't exist

% input Z
Z_in.(BlackBox_1) = {'variable name 1' 'variable name 2' 'variable name 3'};
Z_in.(BlackBox_2) = {'variable name 1' 'variable name 2'};
Z_in.(etcetera) = []; % BB without Z variables
Z_in.global = {'all Z variable names'};


%% DONT CHANGE
% Define couplings
couplings = [];
% set each coupling to empty first
for i = 1:length(disc_names);
    for j = 1:length(disc_names);

        to = char(disc_names(i));
        from = char(disc_names(j));
        couplings.(to).(from) = []; % couplings are absent unless specified otherwise

    end
end

%% Nonzero couplings
% Format: couplings.(to BlackBox 1).(from BlackBox 2) = couplings.BlackBox1.BlackBox2
couplings.(to_BlackBox_1).(from_BlackBox_2)   = {'coupling variables here, choose from
    Y_out.BlackBox2'};
couplings.(to_BlackBox_2).(from_BlackBox_1)   = {'coupling variables here, choose from
    Y_out.BlackBox1'};

% couplings to objective BB
couplings.OBJ.(to_BlackBox_1) = {'coupling variables here, choose from Y_out.BlackBox1'
    };
couplings.OBJ.(to_BlackBox_2) = {'coupling variables here, choose from Y_out.BlackBox2'
    };

%% DONT CHANGE
% create design structure matrix
[DSM] = create_DSM(X_in, Z_in,couplings,disc_names,0);
```

```
% don't remove objective
Y_out.OBJ = {'Objective'};
clear temp



% Display BB in- and output filenames
for i = 1:length(disc_names);

    from = char(disc_names(i));
    Y_in.(from) = [];

    % Y
    for j = 1:length(disc_names);
        to = char(disc_names(j));
        Y_in.(from) = [Y_in.(from) couplings.(from).(to)];
    end

    Y_in.(from) = unique(Y_in.(from));

    disp(' ')
    disp(['INPUT BB_',from])
    disp('--------------------------------------------------------------')
    disp(['These X-variables are assigned to BB_',from,': '])
    disp(' ')
    disp(char(X_in.(from)'))
    disp(' ')
    disp(['These Y-variables are assigned to BB_',from,': '])
    disp(' ')
    disp(char(Y_in.(from)'))
    disp(' ')
    disp(['These Z-variables are assigned to BB_',from,': '])
    disp(' ')
    disp(char(Z_in.(from)'))
    disp(' ')
    disp(' ')
    disp(' ')
    disp(['OUTPUT BB_',from])
    disp('--------------------------------------------------------------')
    disp(['These Y-variables must be assigned in BB_',from,': '])
    disp(' ')
    disp(char(Y_out.(from)'))
    disp(' ')

end

% Add defining vars to STRUCTURE array
STRUCTURE.couplings = couplings;
STRUCTURE.disc_names = disc_names;
STRUCTURE.X_in = X_in;
STRUCTURE.Y_in = Y_in;
STRUCTURE.Y_out = Y_out;
STRUCTURE.Z_in = Z_in;

%% Initial values

X.(BlackBox_1).(var_name_1) = [xx1 xx2 xx3 xx4];
X.(BlackBox_1).(var_name_2) = [xx1 xx2];
X.(BlackBox_1).(var_name_3) = xx;

Y.(BlackBox_1).(var_name_1) = xx;
Y.(BlackBox_1).(var_name_2) = xx;
Y.(BlackBox_1).(var_name_3) = xx;

X.(BlackBox_2).(var_name_1) = xx;
```

```
X.(BlackBox_2).(var_name_2) = xx;

Y.(BlackBox_2).(var_name_1) = xx;
Y.(BlackBox_2).(var_name_2) = xx;

Z.global.(var_name_1) = [xx1 xx2];
Z.global.(var_name_2) = [xx1 xx2];
Z.global.(var_name_3) = [xx1 xx2];

%% Define side constraints
clear X_bound Z_bound

X_bound.(BlackBox_1).bu = ['Put lower bounds here'];
X_bound.(BlackBox_1).bl = ['Put upper bounds here'];

X_bound.(BlackBox_2).bu = ['Put lower bounds here'];
X_bound.(BlackBox_2).bl = ['Put upper bounds here'];

Z_bound.global.bu = ['Put lower bounds here'];
Z_bound.global.bl = ['Put upper bounds here'];

%% DONT CHANGE
% System Analysis - first run
[Y,G, NoLoops] = SA(X,Y,Z,CONSTANTS,FCOND, STRUCTURE,TolY);

% convert couplings to numeric formats
[couplings_local] = define_local_couplings(Y,disc_names,couplings);
[couplings_global] = define_global_couplings(Y,disc_names,couplings_local);

% Define orders of magnitude
% based on initial values
DefineOrderofMagnitude;

% Check for doubly defined or unused vars
CheckXYZCONSTANTS;

% print problem variables
PrintProblem;

% find problem size
FindProblemSize;

% Set move limits
SetMoveLimits


% Estimate problem duration
[Duration] = BLISS_time_est(X,Y,Z,CONSTANTS,FCOND, STRUCTURE, couplings_global,
    BBSA_order);
```

D-5 Running BLISS

First Run

1. Choose a problem ('***prob***')
2. Define ('***nset***') or load ('***lset***') settings
3. Run ('***run***') BLISS, insert desired number of cycles and output array name
4. Indicate if intermediate changes should be possible
5. BLISS runs for the desired number of cycles or until the stopping criterion is met

Continued Run

1. Load the desired results file ('***rloa***')
2. Choose '***run***' and insert the name of the output file to-be-continued
3. If it exists BLISS indicates how many cycles the output file has run already
4. Insert desired number of cycles
5. BLISS runs for the desired number of cycles or until the stopping criterion is met

D-6 Problem Structure Modification

The problem structure modification is conducted manually by the user.

General procedure:

1. Create a Design Structure Matrix with '***DSM***' and view it in Excel
2. Any change in structure can be made, as long as the set of names going in and out of each BlackBox remains the same. If not, appropriate changes must be made in the BlackBoxes themselves.
3. After modification, run a System Analysis to see if the problem structure is still correct, and review the DSM if the desired structure was obtained

Changing Side Constraints

1. BBSA must be finished
2. Use command '***feas***'
3. Select design variable(s) to which an unsatisfied constraint has a large sensitivity compared to the rest of the design vector
4. Change these variable(s) (''), increase if sensitivity is negative
5. Run SA and evaluate effect of change ('***stat***')

Changing Design vector

1. BBSA must be finished
2. Use command '***feas***'
3. Select design variable(s) to which an unsatisfied constraint has a large sensitivity compared to the rest of the design vector
4. Change these variable(s) (''), increase if sensitivity is negative
5. Run SA and evaluate effect of change ('***stat***')

D-7 Manually Increasing Problem Feasibility

1. BBSA must be finished
2. Use command '***feas***'
3. Select design variable(s) to which an unsatisfied constraint has a large sensitivity compared to the rest of the design vector
4. Change these variable(s) (''), increase if sensitivity is negative
5. Run SA and evaluate effect of change ('***stat***'). Repeat steps if needed.

D-8 Viewing Results

Cyclewise values of **X**, **Y**, **Z** and **G** can be viewed with the command '***hist***' or '***ShowDesignHistory***' outside the BLISS program:

1. Choose if variables should be normalized to their initial values or not
2. Choose desired output array
3. Choose desired items to plot. Each item that is plotted, is also saved in a separate m-file.
4. Choosing 'all' means that nothing is plotted, but the design history of each variable is stored separately

A screenshot:

```
-------------
Design history
-------------

Use normalized values in graphs? (y/n)n

Available output arrays:

    FinalBWB: [1x6 struct]

Output name? (default = OUT) FinalBWB

1 = X
2 = Y
3 = Z
4 = Objective
5 = Constraints
6 = DphiBB
7 = DphiSys
8 = Settings
9 = All
Which variable type do you want to view?
```

Appendix E

BLISS Problems Checklist

Table E-1: BLISS troubleshooting checklist

#	Item	Remarks
1	Bounds have good proportion?	Each single side constraint should be in proportion with the design vector element it belongs to
2	All constraints in G_{yz}?	Advisable for an initial optimization attempt
3	Consistent problem structure (DSM)?	Also checked by BLISS framework itself
4	Objective sensitive to the desired variables?	If not, include variables in objective function
5	No 'selective' or 'grouping' constraints?	Constraints that use maximum, minimum, or mean values of state and/or design variables
6	Unconstrained BlackBoxes do not pass on design variables to other BB?	-
7	Initial point is reasonably feasible?	E.g. max(G) = 0.5, limits calculation times
8	no internal design loops in BB's?	Design loop inside a BB controlling parameters that significantly influence the overall design

Appendix F

BWB Design and State Variables

Table F-1: BWB local design variables (X)

BlackBox	**X variables**	**Number of elements**	**Remarks**
AERO_3D	FlapChord	5	Control surface size as fraction of trunk chord (in- and outboard)
	Twist	6	Defined at trunk interfaces
AERO_2D	AirfoilMaxThickness	6	Defined at trunk interfaces
STRUCT	BoomArea	20	A trunk has 4 booms, each boom can have a different boom area
	SkinThickness	20	A trunk has 4 plates each boom can have a different skin thickness
	LEF	6	Leading Edge Fraction, defined at trunk interfaces
	TEF	6	Trailing Edge Fraction, defined at trunk interfaces
	FuelVolumeFraction	1	Fraction of wingbox volume used for fuel
PROP	ByPassRatio	1	
	FanPressRatio	1	
	OverallPressRatio	1	
	CombCh_Temp	1	Cruise thrust setting (combustion chamber temperature)
	ComCh_Temp_TO	1	Take-off thrust setting, also maximum setting (combustion chamber temperature)

Table F-2: BWB state variables (Y)

BlackBox	Y variables (out)	Number of elements	Remarks
AERO_3D	CD_Mission	5	One value for each flight condition
	CL_Mission	5	One value for each flight condition
	CL_alpha	1	Lift curve slope of the whole aircraft
	CLmaxCruise	1	CL_max at cruise altitude
	S_ref	1	Sum of all trunk areas
	S_trunk	5	Trunk surface area
	ControlSurfArea	5	
	b_ref	1	Sum of all trunk spans
	MAC	1	Mean Aerodynamic Chord
	AeroLoadPoly	11	Polynomial coefficients, fitted to lift distribution
	Thrust_Mission	5	Sufficient thrust at each flight condition
	StallSpeed	1	V_min at cruise condition
	SpanwiseClCruise	5	
AERO_2D	ClmaxCruise	1	Arifoil maximum 2D lift coefficient, defined at trunk interfaces
	AirfoilThickness	6	AirfoilMaxThickness' passed onto 'STRUCT' BB as Y-variable
	XtrCruise	5	Transition point per airfoil as fraction of local chord
STRUCT	CoGStructures	3	CoG of empty aircraft
	EngPosition	3	
	TEF	6	
	VolFuelTank	1	Fuel in wingbox, including correction factor to account for space occupied by structure, systems etc.
	VolPressCabin	1	Payload in centerbody wingbox, including correction factor to account for space occupied by structure, systems etc.
	W_structures	1	
	FuelCapacity	1	
	CogPolyFuel_x	3	Polynomial of total momentary fuel fraction vs. CoG x-position
	CogPolyFuel_z	3	Polynomial of total momentary fuel fraction vs. CoG z-position
	CogPolyPayload_x	3	Polynomial of total payload fraction vs. CoG x-position
	CogPolyPayload_z	3	Polynomial of total payload fraction vs. CoG z-position
PROP	Thrust_Max	1	
	W_engine	1	Weight of one engine
	RequiredFuelWeight	1	Required to match design range
WEIGHT	CG_mid_cr	3	
	CoG_Mission	15	Defined for each flight condition
	W_MLW	1	Equal to 70% MTOGW
	W_MTOGW	1	
	W_empty	1	
OBJ	Objective	1	

Table F-3: BWB global design variables (Z)

BlackBox	Z variables	Number of elements	Remarks
global	Sweep	5	Quarter chord sweep
	Span	5	Trunk span
	Chord	6	Chord length, defined at trunk interfaces

Appendix G
BWB Constraints

Table G-1: Blended Wing-Body constraints

BlackBox	Constraints	Number of elements	Remarks
AERO_3D	Thrust_trim	5	Sufficient thrust at each flight condition
	lift_approach	1	
	max_fus_width	1	Maximum fuselage width
	real_CLmax	5	Trimmed CL_max must be smaller than actual CL_max
	real_Elevator	5	TE control surfaces can only deflect up to a limit to balance the aircraft
	real_FlapChord	5	Limit to TE control surface size, control surface and wingbox may not overlap
	stall_AoA	5	Sufficient margin between flight and stall AOA during entire mission
	stall_speed	1	Sufficient margin between flight and stall speed during entire mission
AERO_2D	ClCruise_2D	5	Airfoil 2D lift must match local lift coefficient at cruise obtained with Tornado
	CriticalMach	4	Critical Mach number for local supersonic flow never exceeded during mission
	real_ToverC	6	Upper limit of t/c
STRUCT	MissionFuel	1	Required fuel capacity corresponds to actual capacity in wingbox
	PayloadVolume	1	Payload fits in centerbody
	Sigma	16	Normal (boom) stresses below maximum value, including critical load factor
	Tau	16	Shear (skin) stresses below maximum value, including critical load factor
PROP	BladeHeight	1	See Kok (2008)
	OEI_climb	1	FAR One Engine Inoperative required climb gradient
	TakeOff_dist	1	Balanced field length according to Raymer (2006)
	hpc_exit	1	See Kok (2008)
	hpt_PR	1	See Kok (2008)
	ipt_PR	1	See Kok (2008)
	lpt_inlet	1	See Kok (2008)

Appendix H

747 Design and State Variables

Parameters with no remarks have the same definitions as their Blended Wing-Body equivalents.

Table H-1: 747 local design variables (X)

BlackBox	**X variables**	**Number of elements**	**Remarks**
AERO	Chord	2	
	Twist	2	
STRUCT	WingboxThickness	2	
	SkinThickness	4	
	LEF	2	
	TEF	2	
	BoomArea	4	
	TankVolumeUsed	1	Equivalent to FuelVolumeFraction in the Blended Wing-Body

Table H-2: 747 state variables (Y)

AERO	CL	1	
	CD	1	
	CL_a	1	
	CL_max	1	
	S_ref	1	
	S_segment	1	Equivalent to S_trunk
	Scs	1	
	b_ref	1	
	c_mac	1	Equivalent to MAC
	Chord	2	
	poly_aero	6	Equivalent to AeroLoadPoly
STRUCT	CoG_structures	3	
	Vol_fuel_tank	1	Equivalent to VolFuelTank
	W_structures	1	
	fuel_capacity	1	Equivalent to FuelCapacity
	fuel_cg_x	3	
	fuel_cg_z	3	
	WingboxThickness	2	
WEIGHT	CG_mid_cr	3	
	CoG	3	
	W_MLW	1	
	W_MTOGW	1	
	W_empty	1	
OBJ	Objective	1	

Table H-3: 747 global design variables (Z)

BlackBox	**Z variables**	**Number of elements**	**Remarks**
global	Sweep	1	Quarter chord sweep
	Span	1	Trunk span

Appendix I
747 Constraints

Parameters with no remarks have the same definitions as their Blended Wing-Body equivalents.

Table I-1: 747 constraints

BlackBox	Constraints	Number of elements	Remarks
AERO	max_t_c	2	See \BWB equivvalent
	realistic_CL	1	See \BWB equivvalent
	stall_margin_crit	1	See \BWB equivvalent
STRUCT	WingDeflection	1	Max. 10% tip deflection
	max_normal_stress	4	See \BWB equivvalent
	max_shear_stress	4	See \BWB equivvalent
	range	1	Fuel consumption based on 747-400 range and max. fuel capacity