Physically-Based, Real-Time Visualization and Constraint Analysis in Multidisciplinary Design Optimization

by

Yann Deremaux

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

© Yann Deremaux, MMIII. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author .....
Department of Aeronautics and Astronautics
May 19, 2003

Certified by .....
Karen Willcox
Assistant Professor
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Physically-Based, Real-Time Visualization and Constraint Analysis in Multidisciplinary Design Optimization

by

Yann Deremaux

Submitted to the Department of Aeronautics and Astronautics
on May 19, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

As computational tools becomes a valuable part of the engineering process, multidisciplinary design optimization (MDO) has become a popular approach for the design of complex engineering systems. MDO has had considerable impact by improving the performance, lowering the lifecycle cost and shortening product design time for complex systems; however, lack of knowledge on the design process is often expressed by the engineering community. This thesis addresses this issue by proposing a novel approach that brings visualization into the MDO framework and delivers a physically-based real-time constraint analysis and visualization.

A framework and methodology are presented for effective, intuitive visualization of design optimization data. The visualization is effected on a Computer-Aided-Design (CAD)-based, physical representation of the system being designed. The use of a parametric CAD model allow real-time regeneration by using the Computational Analysis PRogramming Interface (CAPRI). CAPRI is used to link a general optimization framework to the CAD model. An example is presented for multidisciplinary design optimization of an aircraft.

The new methodology is used to visualize the path of the optimizer through the design space. Visualizing the optimization process is also of interest for optimization health monitoring. By detecting flaws in the optimization definition, useless computations and time can be saved. Visualization of the optimization process enables the designer to rapidly gain physical understanding of the design tradeoffs made by the optimizer. The visualization framework is also used to investigate constraint behavior. Active constraints are displayed on the CAD model and the participation of design variables in a given constraint is represented in a physically intuitive manner.

This novel visualization approach serves to dramatically increase the amount of learning that can be gained from design optimization tools and also proves useful as a diagnostic tool for identifying formulation errors.

Thesis Supervisor: Karen Willcox
Title: Assistant Professor

Acknowledgments

Many people have helped me make this thesis possible. First and foremost, I would like to thank my advisor, Karen Willcox, for trusting me from the beginning, for giving me the freedom to attack a difficult and challenging problem, for making me discover the intricate world of MDO and for her constant help and support throughout this year and a half working with her. I am also indebted to Robert Haimes, for letting me work with CAPRI, for taking me into the guts of GV and for answering questions I should sometimes have kept for myself. My sincere appreciations also go to Curran Crawford, who gave me so many ideas and advice and with whom I shared frustrations and discoveries during my time at MIT.

The ACDL has been a wonderful place to work, but also to socialize and meet people. Garrett, on the other side of the wall, has been great teaching me Latex ("One day to learn, a lifetime to master") and giving me so many insights on the American culture. Life in the lab would have been different without the French speaking crew (David, Jérome, Hector and Matthieu) and the many special people that made each day unexpected (Mark, Victor and Vivian). Thank you to Joe, for sharing these many hockey moments that helped me stay fit and sane ! Other people at MIT also deserve a particular thank: Franck, Manu and JB.

Of course, I must acknowledged my family across the Atlantic. A lot of emails, Yahoo messenger discussions and phone calls can easily bring a piece of France to the US. And yes, Laetitia, I will graduate after you, but at least, I am finally graduating !

Finally, I could not end this section without talking about Carole. I wanted to deeply thank her for listening to me day after day, for supporting me throughout the tough times, and for accompanying me across the difficulties and the happy moments of my time at MIT.

Contents

List of Figures

List of Tables

Nomenclature

Roman

c	Mean chord
$\hat{c}$	Equality constraint vector
c_D	2D drag coefficient
$c_{D_{max}}$	maximum 2D drag coefficient
c_{D_o}	2D drag coefficient at 0 degree angle
$c_{D_{min}}$	minimum 2D drag coefficient
c_L	2D lift coefficient
$c_{L_{max}}$	maximum 2D lift coefficient
$c_{L_{min}}$	minimum 2D lift coefficient
c_{L_o}	2D lift coefficient coefficient at 0 degree angle
d	Decision variable vector
D_{fuse}	Fuselage diameter
F	Objective function
$\hat{g}$	Inequality constraint vector
g_{j_o}	Normalization quantity for the constraint g_j
L	Lagrangian function
L_{fuse}	Fuselage length
m	Number of state variables
n	Number of design variables
R	Range
$^r/_c$	Rate of climb
s	State variable vector

S_{wing}	Wing area
V_{Cruise}	Cruise velocity
V_{stall}	Stall speed
W_{span}	Wing span
x	Design vector
x^*	Optimal design vector
x_{i_o}	Normalization quantity for the design variable x_i
x^o	Initial design vector

Greek

λ	Lagrange multiplier
∂	Small perturbation

Superscripts

*	Optimum
^{o}	Initial

Acronyms

AOM	Analysis and Optimization Module
API	Application Programming Interface
CAD	Computer Aided Design
CAPRI	Computational Analysis PRogramming Interface
DAM	Design Analysis Module
GA	General Aviation
MDO	Multidisciplinary Design Optimization
OVM	Optimization Visualization Module
SQP	Sequential Quadratic Programming

Chapter 1

Introduction

In recent years, multidisciplinary design optimization (MDO) has received increasing attention throughout the system design community. MDO has enabled collaborations between industry, academia and governments to "better stimulate, predict, and optimize highly complex systems and products"[24]. With the latest developments in technology, and especially increasing computer power, a classical system design approach can now be carried out in conjunction with formal optimization methods. By casting the design problem as a formal optimization statement, computational algorithms can be used to search the design space in a rational and efficient manner, thereby increasing the effectiveness of the designer. However, MDO is often computationally intense and, as a result, generates an increasing amount of information to be handled. Finally, in many cases, MDO has rapidly become an "expert-only" area. Supporting techniques, such as visualization, can provide the engineer with information that was not previously available to sustain the design effort.

1.1 Background

1.1.1 System design and classical approach

System design has evolved, from Leonardo Da Vinci's ideas to the most modern designs, and enhancements in computers have deeply influenced the engineering and design practices. The ultimate goal switched from the idea that the most perfect design could be reached at any cost to a more rational idea. Today's design paradigm rests on the desire to achieve the best performances while minimizing the costs of design, development, fabrica-

tion and operation. However, system design can still be classified into three classical phases that have their own characteristics. The first phase is called conceptual design. The next phase, preliminary design, often decouples the different disciplines involved in the system design. Finally, detailed design will address the final stages of system design, including manufacturing issues, with the goal to release detailed plans, final performance estimations, and ultimately, production, assembly and maintenance guidance (Table 1.1 summarizes the characteristics of the different phases of the design process).

Conceptual design represents the initial phase in the design of a system. The designer needs to get a feel of the design space. Based on system requirements (including customer needs, customer requirements, derived system requirements) that were defined earlier in the project, conceptual design is based on the use of general analysis tools that can quickly analyze different design solutions. The level of detail reached during the conceptual design phase is often very sparse, but allows the designer to get many answers in a short amount of time. Consequently, conceptual design involves simple geometries and simple analysis models (for example, the Breguet Range equation in the case of an aircraft design or a Carnot cycle in the design of an engine). Typically, optimization is performed at the system level, trying to maximize the overall performance of the system. In the case of conceptual design, MDO often becomes a useful tool as it emphasizes the interactions between the disciplines. The final output of the conceptual design phase should be a single (sometimes a few) design concept(s) that is further analyzed during the next phase: preliminary design.

The preliminary design phase more deeply analyzes a conceptual idea that has been selected via the trade-studies carried out during the conceptual design phase. Rather than a broad analysis at the entire system level, the preliminary design phase encompasses the analysis of subsystems. Doing so, it decouples the disciplines, as each discipline's importance is emphasized. One can then observe a decline of interactions between individual disciplines. Robust design analysis can be added at the subsystem level. Furthermore, optimization might be performed at the subsystem level, and can focus on a specific discipline or a set of disciplines (by using MDO). The entire system's constraints are provided for a subsystem optimization. The preliminary design should provide a fine analysis at the subsystem level, but does not take into account the assembly issues, nor will get into the manufacturing perspective. One of the main weaknesses of preliminary design is the lack of overall perspective regarding the system performance, as a single-point optimized

Table 1.1: Summary of the characteristics of the different phases of system design

Phase	Characteristics
Conceptual Design	Trade-studies
	Simple overall system analysis
	Fast computation time
	Overall limited optimization
Preliminary Design	Sub-systems analysis
	Sophisticated disciplinary analysis
	Subsystem optimization
Detailed Design	Assembly details
	Fabrication issues
	Overall system assessment
	Overall and subsystems optimization

subsystem design is often favored to an overall compromised solution.

The final step in the design of a system is the detailed design. Taking place downstream, it only occurs when all the subsystems have been perfectly defined, and emphasizes the relations between all these subsystems. Assembly, as well as manufacturing will be taken into account, and final refinement of the individual subsystems will be provided. Finally, a detailed analysis should provide a better assessment of the overall performance of the system itself. The system design process flowdown is shown in Figure 1-1. Additionally, as Figure 1-1 shows, MDO can have a favorable impact on the aircraft design. This impact of MDO on the design will be detailed below. Crawford [11] interestingly notices the lack of communication between successive stages of the design, and discusses a "thrown over the wall" design approach. This issue, remaining in current design organizations, has been identified as one of the issues that the industry in general, and the MDO community more particularly, need to overcome.

1.1.2 Multidisciplinary design optimization in system design

Computational tools have grown to be an invaluable part of the engineering design process. In particular, MDO has developed into a popular approach for design of complex engineering systems. By simultaneously considering the disciplines of interest, one can "coherently exploit the synergism of mutually interacting phenomena" [14]. MDO has enjoyed successful uses in many different engineering applications, ranging from the design of aircraft, aircraft engines and spacecraft to automobiles [37, 25]. In the past, designers, or design groups, have tended to emphasize the discipline on which they were working, and

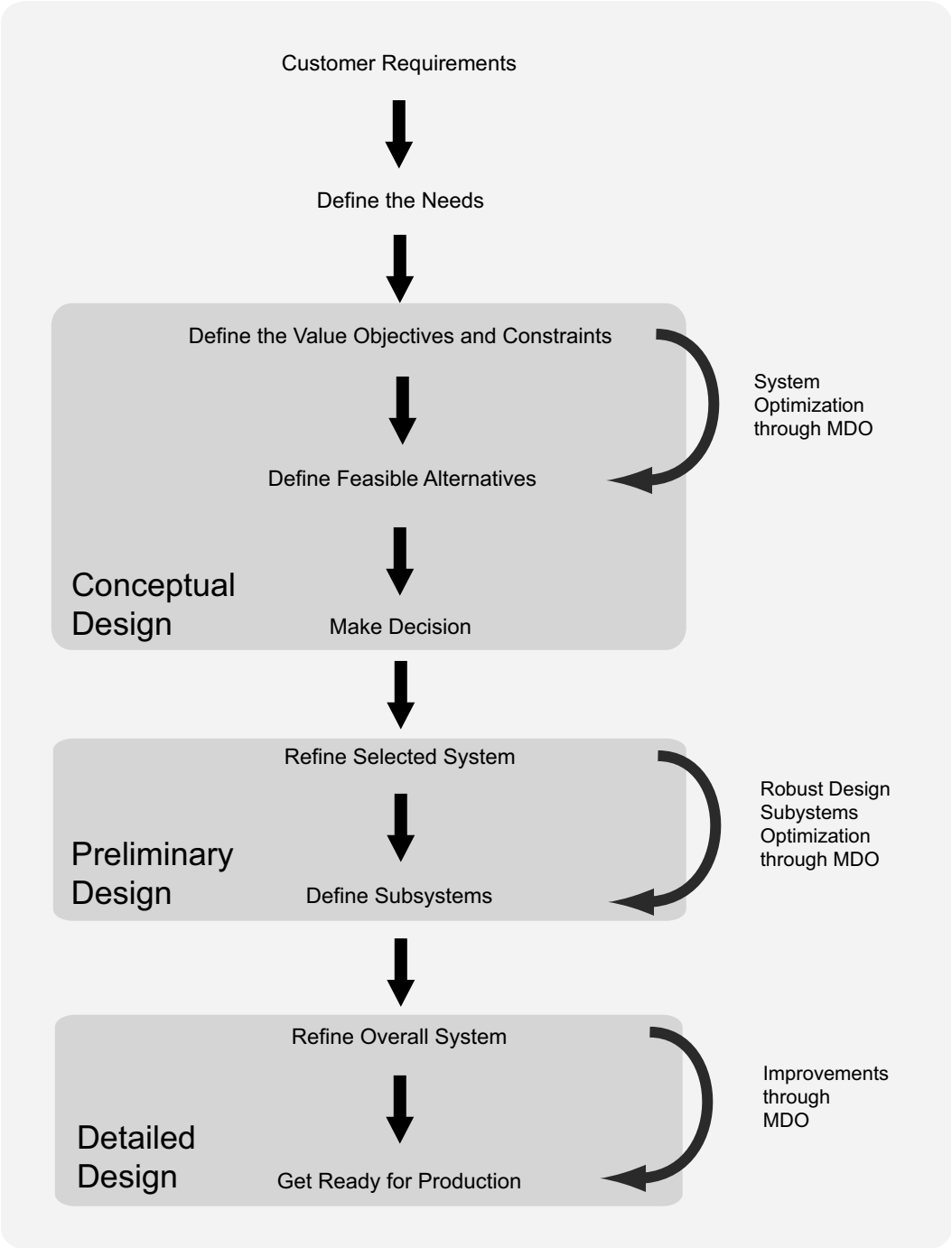


Figure 1-1: Aircraft design flowdown

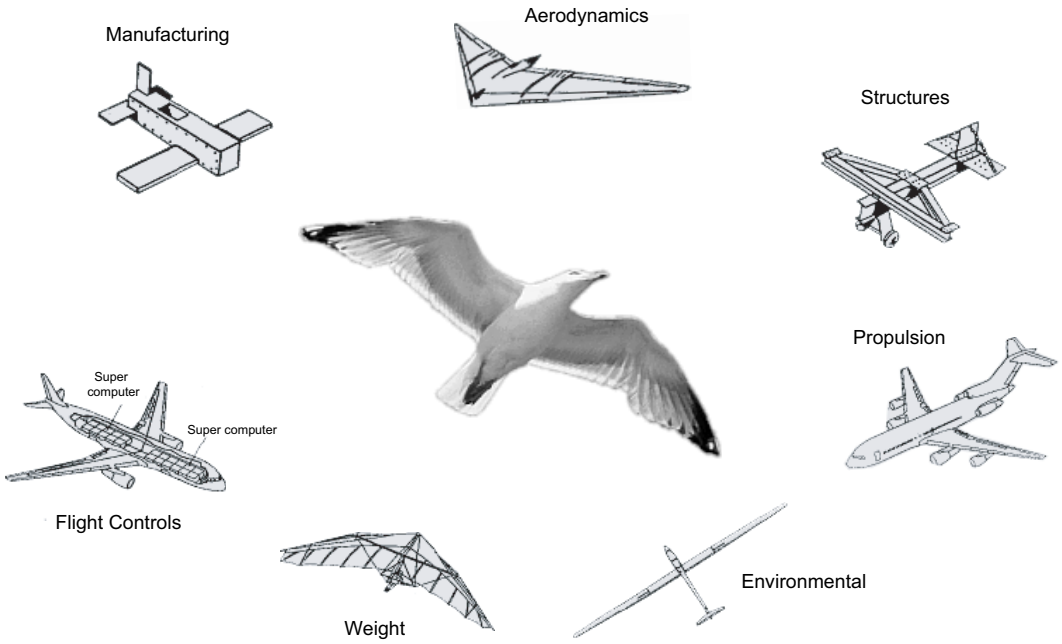


Figure 1-2: The best aircraft for different disciplines. The seagull represents the MDO optimum (Adapted from [2]).

optimized a complete system based on that discipline. In the case of aircraft design, the cartoon in Figure 1-2, adapted from [2], shows with humor which aircraft would best suit the different discipline groups. The seagull in the center is considered as the MDO design, the overall compromise, that combines the different disciplines to attain a global optimum. Korte [22] has an interesting way to define the contribution of MDO:

$$\Delta_{Design} = \left(\sum_i \Delta_{Discipline_i} \right) + \Delta_{MDO} \tag{1.1}$$

where the total amount of change in the design (for a given objective) is equal to the sum of the contributions towards this given objective from the individual disciplines plus the contributions brought by the interaction of the individual disciplines.

Recently, the field of MDO has had considerable impact by improving the performance, lowering the lifecycle cost and shortening product design time for complex systems [42, 14]. For example, the Blended-Wing-Body aircraft design team uses an extensive MDO framework, which simultaneously considers aerodynamics, structures, weights, balance, stability and controls [40]. This framework leads to improved designs as well as much faster design turnaround time. In addition, during a typical MDO run, hundreds or even thousands of

different design options might be evaluated.

To complement MDO, visualization can have a positive impact on the design. As the problems considered become more complicated, more and more data is produced by the optimizer and new ways to aid the engineer must be investigated. Bringing visualization to the design process will help in dealing with an increasing amount of information, will bring additional knowledge, and will give the designer additional insight.

1.1.3 Impact of MDO on system design

During the early stages of the design, simple tools are used, and become more and more sophisticated as the design evolves towards the detailed design. During the three stages of the design, each discipline interacts with all the others, often leading to contradictory demands. These interactions and contradictions then become more and more difficult to satisfy in later stages of the design as all the decisions made earlier come into account. Figure 1-3 shows how the total cost of a system is set in the early phases of the design, while few resources (both financial and computational) are allocated to the design. In parallel, the freedom to change the design evolves similarly to the way the total cost is set. Indeed, during the conceptual phase, the designer still has the option to explore the design space and opt for one or another option. In the preliminary phase, and furthermore in the detailed design, the decisions made earlier in the design process reduce the ability to change the design, and therefore the freedom of the designer. Finally, the knowledge available during the conceptual phase is very sparse and increases as the design progresses. Conversely, the need for knowledge is crucial at the moment when decisions influence greatly the final output of the design process. And while the designer learns more about the system during the later phase, the freedom to change the design has disappeared.

Bringing MDO in the design can lead to two main changes. While the resources allocated and the influence on the final cost do not change, MDO brings more knowledge of the system earlier in the design process. It also permits an extension of the freedom to change the system later in the design process. Figure 1-4 summarizes the benefits of MDO in the design process. Correlated to the well known idea that 65% of the final cost of a system is frozen during the conceptual design phase and 85% during the preliminary phase, while only 10% of the resources (human and financial) are dedicated to these early phases, one can easily realize the beneficial impact that MDO can have on the final cost. MDO has

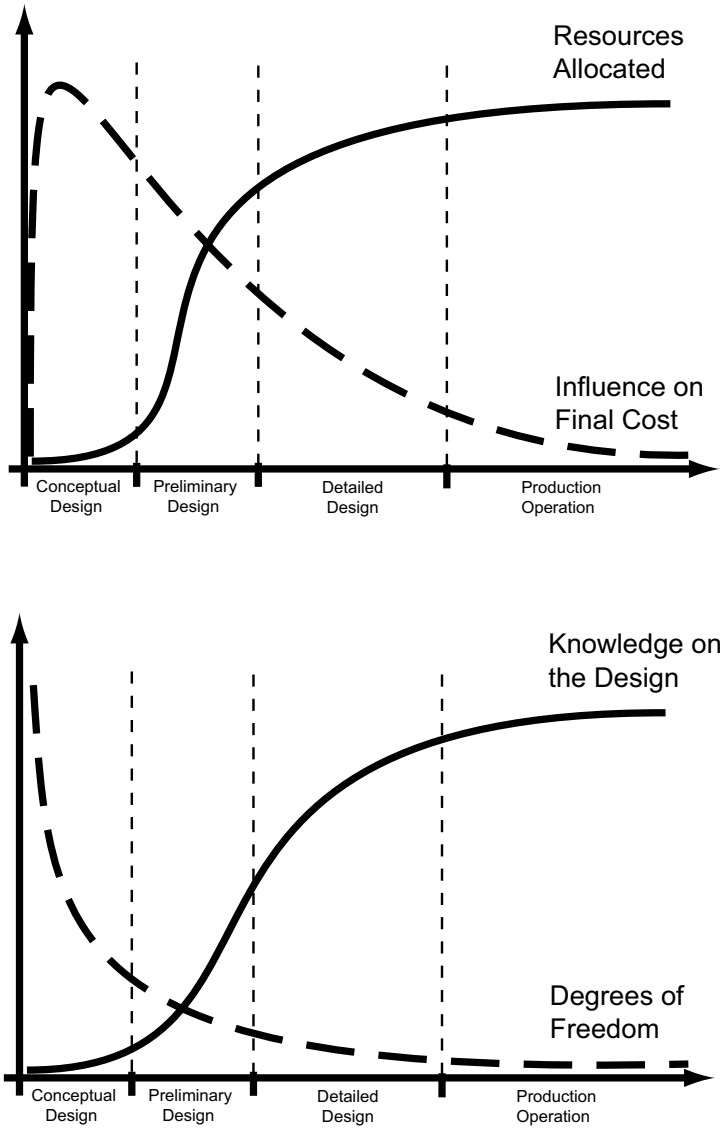


Figure 1-3: Influence of different parameters in the design process

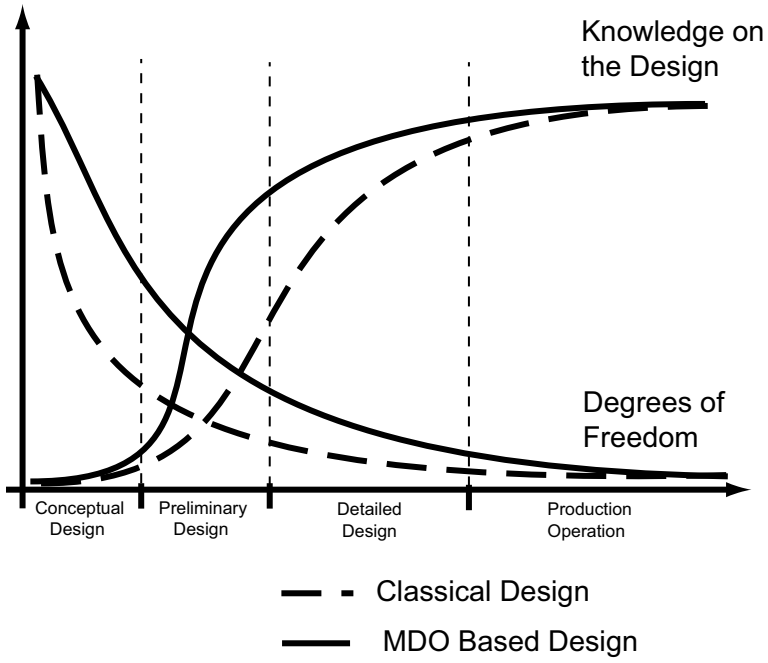


Figure 1-4: Influence of bringing MDO in the design process

therefore become a great way to improve the design of complex systems, particularly in the aerospace industry, by utilizing the interactions between different disciplines in the design process.

1.2 Motivation

MDO, while becoming more widely used, is a relatively young approach that is rapidly evolving. Many future needs can be identified, some of which are discussed in [14]. The work involved in this thesis is founded on an evaluation and an identification of the primary elements that would drive MDO towards a better acceptance and wider use, at both industry and academia levels. Existing literature was helpful in the identification process of the needs, but real input from daily MDO users represented a valuable source of information. Visualization quickly appeared as a weak point of MDO, for many reasons:

- No existing framework managed to seamlessly integrate visualization to MDO
- Amount of data to handle is increasing exponentially
- Computational resources can now deal with expenses related to visualization

- The designer wants to be involved in the design process

Sections 1.2.1 and 1.2.2 describe in more detail the process that was used to arrive to the conclusion that visualization was a crucial need.

1.2.1 MDO survey

As an initial task, it was deemed valuable to ascertain the status of MDO. The goal was to obtain, mainly from the aerospace industry, some insights on how MDO is perceived, how it has evolved and how it could be improved to become a more widely used tool. In order to gather this information, a survey was constructed and was sent out to many key players in the aerospace industry. This survey included ten questions, five of which concerned aircraft design specifically. The five other questions focused on MDO applied to aircraft design. Below are the ten questions:

(A) Aircraft design

(1) Could you summarize quickly your aircraft design activity?

(2) What are the tools you are currently using for aircraft design? What do you think about these tools (advantages, drawbacks, etc.)

(3) What aircraft design tools have you used in the past? Why did you decide to stop using them?

(4) What are the main characteristics you expect from tools for aircraft design?

(5) What could be improved on the tools you are currently using?

(B) Multidisciplinary Design Optimization

(1) Have you ever heard about MDO? What do you think about it?

(2) What do you like about MDO? What do you not like?

(3) Are you currently using MDO for aircraft design? What tools are you using in relation with MDO?

(4) What are the main characteristics of MDO (and MDO tools) that prevented you or could prevent you from using MDO?

(5) What new developments could be very beneficial to MDO? What would you expect from new MDO tools?

Eight responses were received, which, although a small number, were representative of different fields of aircraft design. Below are the names and positions of the people who submitted a response.

(1) Jean-Charles Lede, System Engineering group leader, Aurora Flight Sciences Corp.

(2) Anonymous, ED23/Structural Design Group, NASA/Marshall Space Flight Center

(3) Anonymous, Aerospace Engineer, Air Force Research Laboratory

(4) Richard Gilmore, Engineer/Scientist Specialist, Advanced Air Vehicles Aero Technology

(5) Robert Canfield, Air Force Research Laboratory

(6) Dr. Vladimir Balabanov, Senior Research and Development Engineer, Vanderplaats Research and Development, Inc.

(7) Rob Taylor, JSF Airframe Systems Engineering and Integration Team, LMCO

(8) Anonymous, LMCO

The answers received revealed a large variety of aircraft design activities, ranging from the design of UAVs to the analysis of new configurations and new concepts (Blended-Wing-Body (BWB) at Boeing, joined-wing concept in the Air Force or High-Speed Civil Transport (HSCT) at Lockheed). All the responses are gathered in Appendix A. The interesting point that was underlined was the enormous variety of tools and programs used in the design activity. From simple and widely used tools (Excel, NASTRAN, Matlab, Computer-Aided Design (CAD) software) to in-house software packages (WingMod at Boeing) or less-known more specialized tools (BOSOR, PANDA, PANAIR, ESRD StressCheck), the aircraft designer showed not only that a common tool that solves everything does not exist, but also that in-house developed tools can be favorably replaced, today, by commercial software. Characteristics that tools used in the aircraft design should possess were identified as common needs: flexibility appears to be the main driver, closely followed by ease of use and efficiency. The final request was the desire of the designer to be involved in the design, in terms of information that is communicated and/or displayed during the design.

When asked about MDO, the aircraft designer declared how MDO is a promising tool for aircraft design, but still lacks the ease of use necessary for MDO to be used, by non-MDO

experts, and during all the stages of the design. Even though tremendous improvements can be attained through the use of MDO, the difficulty to link the disciplines together, the time required to build the framework, the breadth of knowledge required to analyze the results are all issues that need to be addressed before MDO become a widely used tool. For Robert Canfield, the next step is to be able to integrate easily the tools one uses into a single framework: "It would be nice to have an MDO tool into which you could plug already existing analysis modules". Anonymous, from NASA, put the emphasis on visualization and on the role of the designer in the MDO process : "Intimately linking MDO inside high end CAD tools using the optimization links already available [would be very beneficial]. MDO tools should not require the user to be an expert in all the disciplines involved in the optimization". Finally, Anonymous, from Lockheed, reminds that "Any MDO tool should have a keen eye on the enormous complexity of aircraft design, and not take control from the discipline expert". As a conclusion, when asked what development and/or tools could be beneficial to MDO, a consensus directed towards integration and interfacing of existing analysis codes, as well as use of CAD and/or visualization emerged across the answers. This leads towards a few ideas that will be followed along the path of the research:

(1) MDO should not be a push-button process but, it the meantime, should not require a very deep expertise in optimization theory

(2) Taking into account the software or codes existing in the industry will require modularity

(3) Plugging new codes in the MDO framework should become more seamless

(4) A real need for visualization (of the results, but also of the physical system) is emerging, with the necessity to link CAD tools to the optimization tools

1.2.2 A real need for visualization in MDO

A need for visualization was highlighted by key players of the MDO community. Indeed, although the complexity of design problems handled with MDO is becoming more impressive, the ability to effectively handle data has languished. MDO frameworks commonly lack flexibility; they are often developed for one particular application by one particular person. The interface is often very unfriendly and many problem-specific attributes are hard-coded

to the tool. In particular, the information generated during an optimization run is rarely communicated in an effective way (if at all) to the designer. An optimization run does result in a solution to the specified problem, but it also provides a wealth of information about the design space.

By looking at just the optimal solution generated, the designer uses MDO as a push-button tool: specify the problem and get the best answer. However, this is not the most effective use of such a design tool. Rather than being used to eke out a 5% improvement in the design solution (where model fidelity is often an issue anyway), MDO ought to be used as a way of gaining insight to the design space, quantitatively identifying trades and finding innovative design options. Often in practical applications, it is the solution concept suggested by the optimizer but not the actual details of the design that are most interesting. For example, in Wakayama [40], through MDO it was determined that increasing the sweep of the wings could alleviate a balance problem in the Blended-Wing-Body aircraft. The details of the planform were subsequently refined using high-fidelity computational fluid dynamic analysis. In the field of turbomachinery design, recent work in developing a new design interface builds on the concept that it is insight to and understanding of the problem at hand that achieves the biggest design successes [21].

Visualization can then be added, and can become a valuable tool to the designer. Visualization, combined with MDO, is beneficial, in terms of knowledge it brings in the design, and its impact on the design cycle time. By analyzing Figure 1-5, one can realize the potential impact of visualization combined to MDO, compared to the benefit of MDO alone. By keeping more degrees of freedom for later phases of the design, and by providing the user, early, with knowledge, visualization allows for saving time and money.

Viewing the results, either visualization of the optimization, or physical display of the evolution of the system, can potentially help the engineer to:

- Further understand the impact of a design decision
- Compare different configurations
- Visualize trade-offs
- Delay important decisions for later phases of the design

As people in the industry and in the academia realized the importance of visualization,

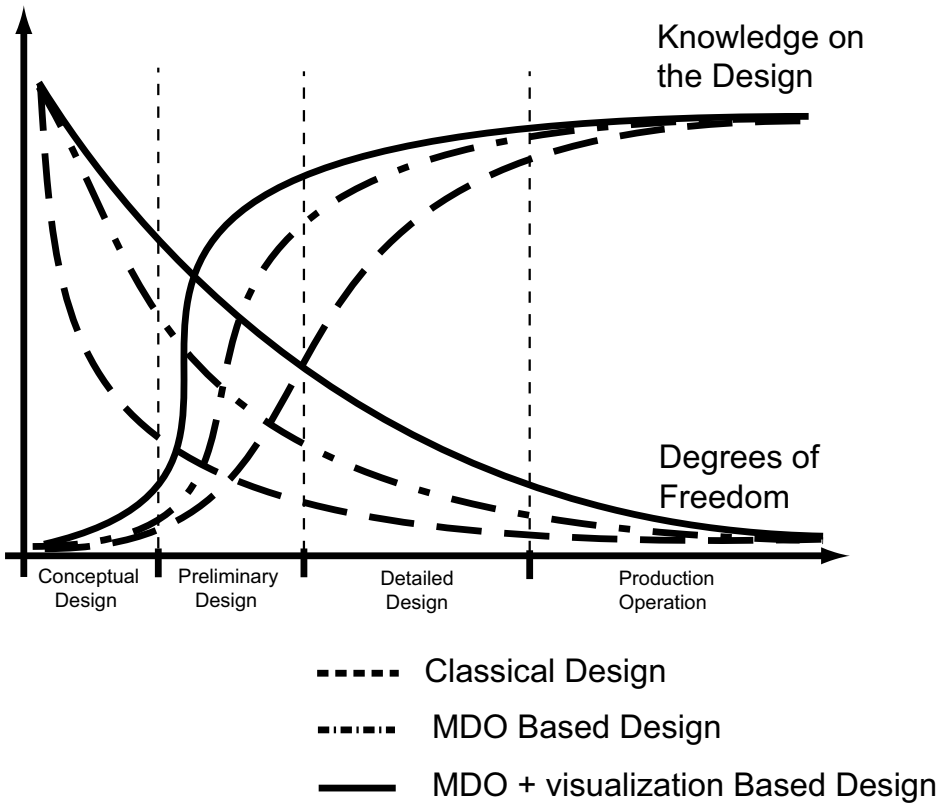


Figure 1-5: Influence of bringing MDO, and visualization in the design process

work on visualization frameworks and visualization methods emerged. Section 1.3 describes how CAD models can be used in visualization and then looks at two different approaches of system design using visualization.

1.3 Related work

1.3.1 Computer-Aided-Design (CAD)

Visualization can, and has, used CAD models. CAD models permit communication of design intents and geometry changes throughout the teams involved in the design. CAD models are two or three dimensional models that include geometry. Instead of having imbedded fixed dimensions, CAD modelling, based on parametric interpretation, allows the designer to set a standard geometry once upfront (contained in a feature tree, which represents the detailed chronological building process of the CAD model), and then vary the geometry endlessly as the design is changed. After changing a dimension, the regeneration of the new model occurs, by recalculating the properties of the model. Building a correct CAD model, that encapsulates the design intents, is a rigorous process that is described in detail in Section 3.1. However, it is important to make sure that the CAD model reflects the intention of the designer and capture the intended function of the part. Finally, the power of the CAD approach is that CAD modelling :

- Encapsulates the engineer's design intent
- Captures the function of a system
- Allows immediate regeneration of the geometry without cumbersome calculations and redrawing the entire system

However, even though CAD models are an efficient and powerful way to represent a system, work on visualization in MDO took different approaches that are detailed in the next section.

1.3.2 Previous work on visualization in system design

Visualization in MDO has been identified for some time as a critical need. In 1991, the AIAA MDO technical committee expressed in a white paper on current state of the art [28] that one of the challenges to overcome for MDO would be to display the path taken by

the optimizer ("presenting graphically the salient features of the design space ") in order to give the designer more confidence in the results and in the power of MDO. In the same paper, a first approach to visualization of optimization results is sketched. Assuming only two variables taken from the design vector, one can then assume these "two variables in a base plane and plot above this plane a curved surface representing the objective function which depends on the two variables and which is to be optimized". The side constraints would then define parts of the curved surface where the design is feasible and the optimum value has to be found.

Bloebaum and Winer [9] followed the idea and developed a new technique that would offer visualization to MDO problems: Graph Morphing. Graph Morphing allows the display of a subset of the design space of a system. This space can either be a two or three-dimensional representation. First, after having ranked the constraints and the design variables of a problem to identify the most significant ones, Graph Morphing considers two (or three) design variables and places them on a two-dimensional (or three-dimensional) coordinate system. The designer can then see how the variables interact with one another on curves (or surfaces). Complementing these curves (or surfaces) are curves (or surfaces) representing iso-objectives and, if applicable, curves (or surfaces) representing the boundaries of the problem. Figure 1-6 shows an example of the application of Graph Morphing to a specific problem, for a 3-dimensional example. Shown on the figure are the iso-performance surfaces, one equality constraint boundary, and two inequality constraint boundaries. Later, Winer, Samant et al.[41, 36] propose a new paradigm, Visual Design Steering. The concept is to utilize visualization (provided by Graph Morphing) as a support tool for the designer, in the analysis and optimization phases of the design. The designer can then "steer" the optimization to obtain better results. The tool allows interaction between the designer and the system, via an interface that displays the effect of any change. Figure 1-7 shows the interface (GmorphVR 2.0) developed for Visual Design Steering, that gives control to the designer over the design.

Messac and Chen developed a different approach. Physical programming [26] is another concept used to visualize the optimization process in real time. Physical programming allows "a designer to express a preference with respect to each given design metric by using the six terms highly-desirable, desirable, tolerable, undesirable, highly-undesirable and unacceptable". This being defined by the user, one can then visualize the results of

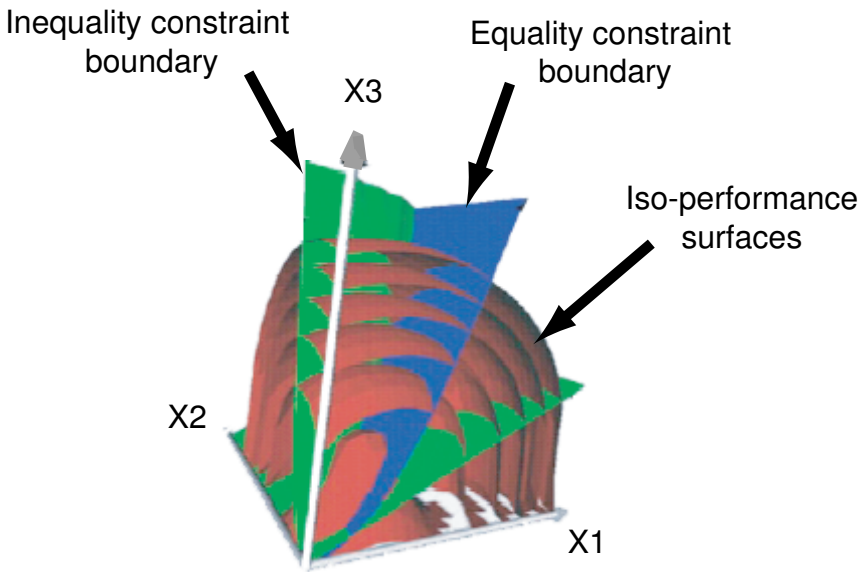


Figure 1-6: Example of the application of Graph Morphing to a specific example (modified from [41])

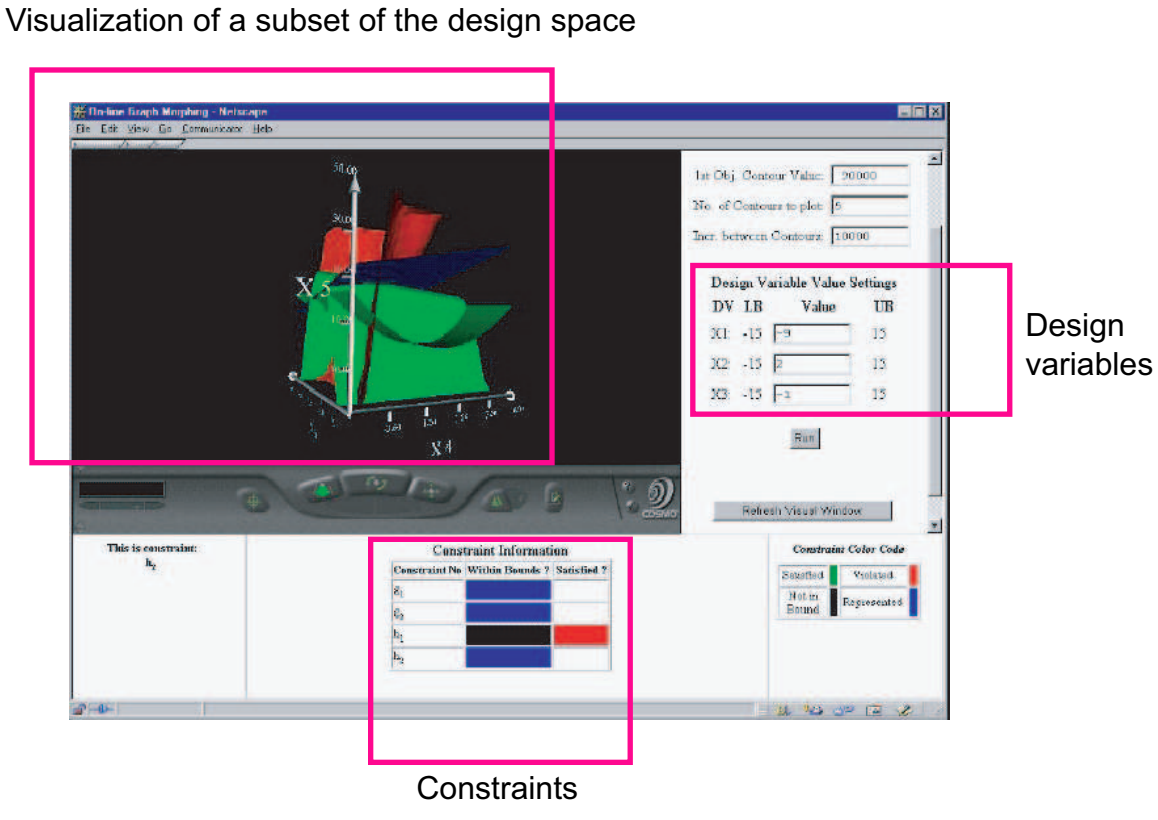


Figure 1-7: Example of the application of Visual Design Steering to a specific example (modified from [36])

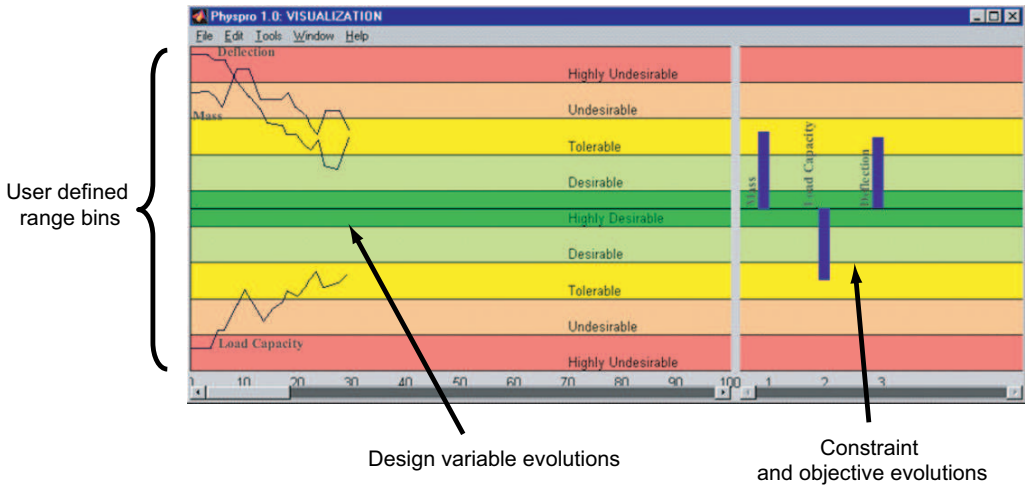


Figure 1-8: Example of the application of physical programming visualization to a specific example (modified from [26])

the optimization. Two dimensional plots and color codes are used to identify the value of a solution, and monitor in real-time the optimization process, as shown on Figure 1-8. The strengths of the physical programming approach are that the designer no longer needs to deal with numbers, but rather with subjective ranges, which give a clear representation and require less mental effort from the designer.

However, these visualization approaches have not focused on a representation of the optimization process which is linked to the physical design. Wakayama used a more physical approach to visualize optimization data. In [39], the active constraints were manually associated to physical features on the BWB, and were plotted on a planform of the BWB. A color code makes the understanding of the participation of the constraints on the actual geometry clearer. This technique was found to lend valuable insight to the designer. But, the full process being manual, it requires a lot of time and post-processing from the designer, and does not provide information that could be valuable during the optimization.

1.4 Thesis outline

From the previous sections, it appears clear that visualization is a real need for the MDO community and will become a major driver of MDO's impact on industry practises in a short-term future. Even though some work has been done, the MDO field still lacks of a technique that can efficiently link the optimization process to its effects on the real system.

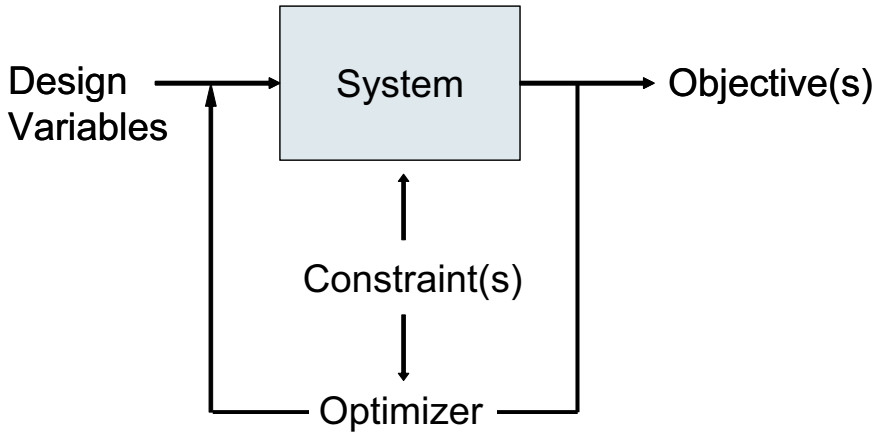


Figure 1-9: Block diagram of the optimization formulation

The purpose of the work presented here is to address this issue by developing a methodology and a framework that will allow the real-time display, using a physically-based approach, of a three-dimensional model of the system being designed. This will not only show the effect of the optimization on the shape of the system, but will also give the designer the ability to analyze, in depth, the role and effect of the constraints on the design. This novel approach, which uses a physical CAD-based model to display a more intuitive view of the optimizer design exploration, will be thereafter presented.

In this thesis, the general optimization framework is first described in Chapter 2. The emphasis is given to the mathematical description of the optimization problem and to the mathematical analysis of the constraints. It is however important to recall the overall optimization formulation. As Figure 1-9 shows, three elements can be distinguished:

(1) The design variables : values in the design that can be changed by the optimizer and that are the input of the system analysis.

(2) The objective function(s): output of the system analysis and representative of the metrics of the system. Objective function(s) are minimized by the optimizer.

(3) The constraint function(s): provide the boundaries to the design space in the analysis model, and during the optimization.

The block diagram in Figure 1-9 represents the organization of these three elements of the optimization formulation. Chapter 2 includes an analysis of the Lagrange multipliers, a

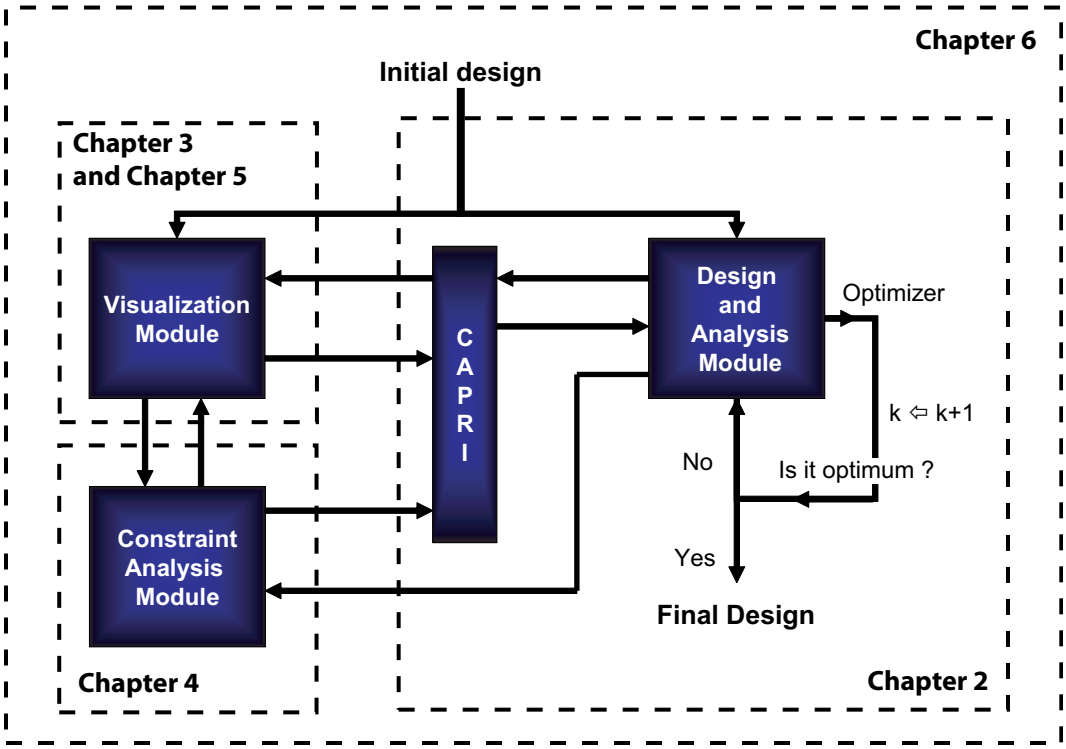


Figure 1-10: Overall architecture of the MDO framework including visualization and constraint analysis

sensitivity analysis approach, as well as the ranking of the design variables participating in a given constraint. The following section will focus on CAPRI and will describe its features that were of interest in the scope of the work. Then, two design visualization methodologies will be described in Chapters 3 and 4. The first methodology, detailed in Chapter 3, will allow visualization of the path taken by the optimizer from the initial to the final solution. The approach will be presented first, and will detail the different steps of the framework creation. The impact and value of visualization on the design will then be assessed, as a function of the complexity of the CAD model. The second approach, analyzed in Chapter 4, is a detailed visualization of constraint behavior and its impact on the design. The methodology will be described, emphasizing the sensitivity analysis method used to rank the design variables participating in a given constraint, and the way the user can use the information. Then, the three levels of information used in the visualization process will be explained. This will lead to a discussion, in Chapter 5, on the creation of the interface, including a motivation for the choice of the language, a complete description of the algorithms used and a detailed description of the interface itself. Following, in

Chapter 6, a benchmark example will present the range of utilization of the interface and framework that have been developed, including a description of the model that is used in this thesis to demonstrate the capabilities of the framework. This example, a General Aviation (GA) aircraft, is based on rather simple fidelity models that will be described. Figure 1-10 can be used to explain the structure of the thesis by linking it to the architecture of the visualization framework developed in this research. Finally, Chapter 7 will conclude this thesis with lessons learned, ideas for future works and conclusion.

Chapter 2

The context

Optimization and, more precisely, MDO have become a popular way to find improved solutions to a design. When confronted with a complex system, and therefore with a complex model, optimization helps to find improvements of the metrics of the system. It is important to note that global optimality is not mathematically guaranteed, but the goal of optimization, and the interest for the designer, is to get an improvement (reaching a local minimum) rather than finding a global optimum. Improvement can be expressed as the best design within the available means, in term of time, computer resources and money. In this search for improvement, computers have an important role. Increased computational power, by allowing a more complete search of the design space within a small amount of time, help to improve design solutions. But human interaction also has an important role. The intervention of the human in the optimization process can help to gain time, by monitoring the optimization and preventing it from exploiting the loopholes of the design formulation. It can also improve the results, as human experience can be included in the design. Human interaction can take different forms:

- Optimization framework definition
- Optimization steering by picking adequate initial design vectors and/or constraints values
- Optimization visualization, to monitor the optimization path and the health of the solution, in particular around an optimum solution.

The latter, visualization, can take different forms, as seen in Section 1.3. This includes the use of a parametric model (CAD model) to visualize the geometric evolution of the system or the use of 2D or 3D charts that can help visualize the evolution of metrics of a system in a subset of the design space.

To address the visualization need, a dual approach has been developed to fill the gap existing between the user and the optimization, by bringing physical insight into the design. Three modules are used in the approach:

- The Analysis and Optimization Module (AOM) represents the classical MDO design approach (Figure 2-1). Starting from a design vector, an analysis model computes the value of an objective vector which measures the response of the system to the design vector. The behavior of the system, as shown in Figure 1-9, is limited by some constraints. Optimization then evaluates the output response and changes the design vector to lead the solution towards an optimum. The system in this module is represented by an analysis model that represents the system with a given level of fidelity.

- The Optimization Visualization Module (OVM) allows real-time visualization of the optimization steps. A CAD model of the system is interactively updated throughout the optimization and changes are displayed in real time. The OVM architecture is described in Chapter 3.

- The Design Analysis Module (DAM) focuses on the analysis of a specific solution. It gives the designer an option to interrogate the design at a given time, and offers information on the constraint status, the participation of the design variables in both the objective and the constraints, and physically displays the results. The DAM framework is developed in Chapter 4.

Figure 2-2 shows the interactions between the three modules presented above. It is interesting to observe that the AOM is independent by itself. The OVM and the DAM are added modules that help the designer but are not required to perform the MDO analysis of the system.

The following will first describe mathematically the optimization framework being used in the thesis as well as the sensitivity analysis and the constraint analysis that represents the

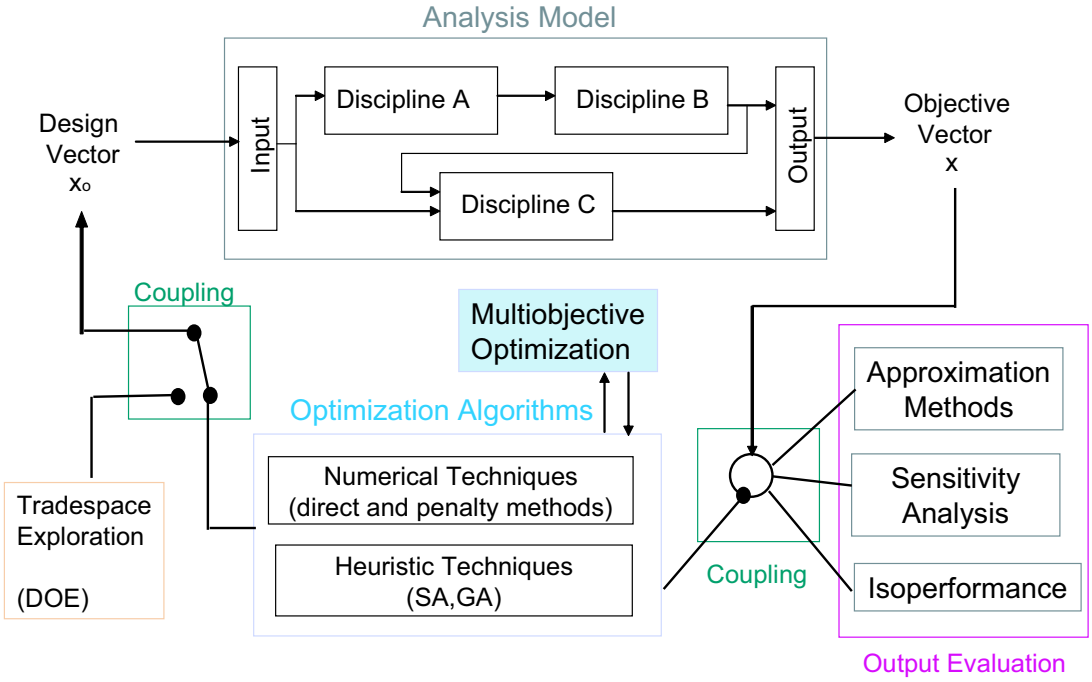


Figure 2-1: Classical MDO framework. It can include design space exploration (DoE) and post-processing

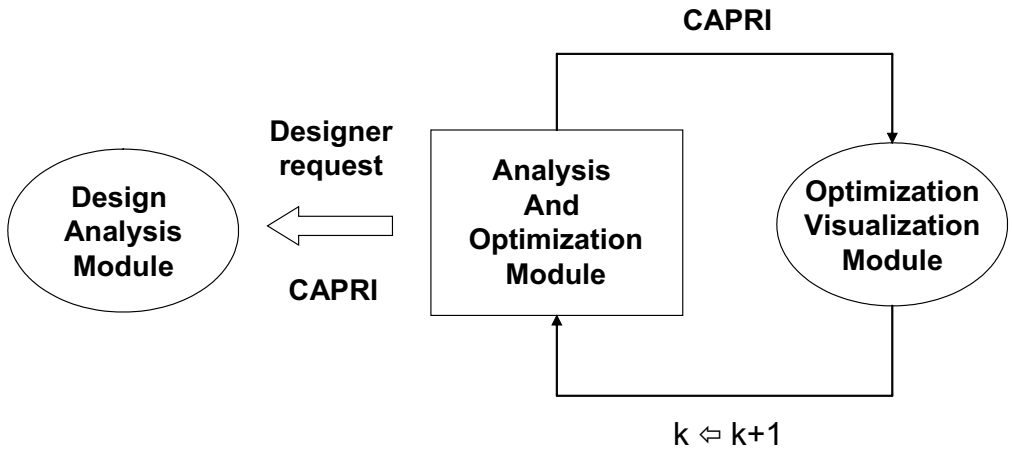


Figure 2-2: Overall architecture - Interactions between the different modules

base of the DAM (Section 2.1). As Figure 2-2 shows, CAPRI has a major role in the development of the framework by allowing information exchange, and by allowing the connection between the three modules AOM, OVM and DAM. To better understand CAPRI, Section 2.2 will detail the architecture of CAPRI and will identify its main characteristics. Finally, Section 2.3 will present the overall methodology that was developed to bring visualization into an MDO framework, and into the design process.

2.1 Optimization Framework

2.1.1 General constrained problem

Consider a general constrained problem as follows:

$$\begin{aligned} &\text{Minimize } f(x) \text{ for } x \in \Re^n \\ &\text{subject to:} \\ &\hat{g}_j(x) \leq 0,\ j = 1, ..., m_1 \\ &\hat{c}_k(x) = 0,\ k = 1, ..., m_2 \end{aligned} \tag{2.1}$$

where $f(x)$ is the objective function, x is the vector of n design variables, $\hat{g}$ is the vector of m_1 inequality constraints and $\hat{c}$ is the vector of m_2 equality constraints. A set of parameters p, inherent to the model itself, is considered a fixed value during the optimization. In addition, for clarity purposes, upper and lower bounds on design variables are considered in the formulation problem as inequality constraints.

In this work, gradient-based optimization algorithms are considered, although the visualization methodology could also be used with heuristic techniques. In general, these algorithms are iterative. Beginning with some initial guess x^o, the algorithm successively refines its current estimate of the design variables based on gradient information. In general terms, this can be written as

$$x^{k+1} = x^k + \alpha^k d^k \tag{2.2}$$

where x^k is the design vector at iteration k. The guess at iteration $k+1$ is computed by moving some scalar distance α^k in the direction d^k.

Given an initial solution x^0, different gradient-based algorithms will take different paths

through the design space. In this work, sequential quadratic programming (SQP) is used [19]. At each step, a subproblem with a quadratic objective function and linear constraints is created and solved. Optimality (represented by the optimal solution x^*) for such algorithms can be defined using the Karush-Kuhn-Tucker (KKT) conditions. If the Lagrangian function $L(x, \lambda)$ is to be written as

$$L(x, \lambda) \equiv f(x) + \sum_{j=1}^{m_1} \lambda_j \hat{g}_j(x) + \sum_{k=m_1+1}^{m_1+m_2} \lambda_k \hat{c}_k(x) \tag{2.3}$$

where λ_j is the j^{th} Lagrange multiplier, then the KKT conditions for optimality are the following:

$$\nabla f(x) + \sum_{j=1}^{m_1} \lambda_j \nabla \hat{g}_j(x) + \sum_{k=m_1+1}^{m_1+m_2} \lambda_k \nabla \hat{c}_k(x) = 0 \tag{2.4}$$

$$\hat{g}_j(x) \leq 0,\ j = 1, ..., m_1 \tag{2.5}$$

$$\hat{c}_k(x) = 0,\ k = 1, ..., m_2$$

$$\lambda_j . \hat{g}_j(x) = 0,\ \lambda_i \geq 0,\ j = 1, ..., m_1 \tag{2.6}$$

$$\lambda_k = 0,\ k = 1, ..., m_2$$

Local optimality is defined by the conditions cited above. However, a complex system can have more than one optimum solution. Global optimality is then defined by the local optimum, among all the local optima, that will give (in the case of the optimization framework defined in this section) the smallest value. Global optimality, while being mathematically defined, cannot be guaranteed unless the design space is convex, which is not the case for most engineering problems of interest.

For SQP algorithms, all iterates are feasible (that is, they satisfy the constraints in (2.1)). In that case, it may be of special interest to the designer to visualize the path taken by the optimization, represented by the sequence of iterates x^0, x^1, ..., x^k, ..., x^*, and to thus gain insight to the physical design space.

2.1.2 Constraint sensitivity analysis

Once the optimal solution, x^*, has been reached, the designer may be interested in further interrogation of the optimal design. Sensitivity analysis yields information on the

shape of the design space near the optimum and can lend valuable physical insight. Further insight can be gained by studying the set of active constraints and using sensitivity information to discern which aspects of the design are constrained by which requirements. This information can be obtained in two ways as follows.

The first approach to investigate constraint behavior is to compute the Lagrange multipliers at the optimal solution. The Kuhn-Tucker conditions (2.4) show that at the optimal solution (x^*, λ^*), if a constraint $\hat{g}_j$ is inactive then the corresponding Lagrange multiplier, λ_j^*, must be zero. That is, either $\hat{g}_j(x^*) = 0$ or $\lambda_j^* = 0$ for all $j = 1, 2, \ldots, m_1$. As a result, a non-zero value for a Lagrange multiplier directly implies the activeness of the corresponding constraint ($\lambda_j^* \neq 0 \Rightarrow \hat{g}_j(x^*) = 0$). Moreover, the value of a non-zero Lagrange multiplier gives a linear indication of the rate of change in the optimal value of the objective function as the corresponding active constraint is relaxed.

The second method of investigating constraint behavior is to compute sensitivities of each constraint with respect to each of the design variables. Using this sensitivity analysis, one can determine the relative participation of each design variable in a specific constraint. The sensitivity of constraint g_j with respect to design variable x_i is given by the partial derivative $\frac{\partial g_j}{\partial x_i}$, where the vector g now contains both the inequality constraints $\hat{g}$ and the equality constraints $\hat{c}$. In order to compare sensitivity values in a meaningful way, it is necessary to compute a relative change:

$$\frac{\Delta g_j / g_{j_0}}{\Delta x_i / x_{i_0}} = \frac{x_{i_0}}{g_{j_0}} \frac{\partial g_j}{\partial x_i} \tag{2.7}$$

where x_{i_0} and g_{j_0} are normalization quantities for design variable x_i and constraint g_j respectively. The partial derivative in (2.7) is evaluated at the point of interest (often x^*), however care must be taken when choosing the normalization quantities x_{i_0} and g_{j_0}. While the value at the optimum solution may be one choice, it is important that both x_{i_0} and g_{j_0} are chosen to be non-zero. While the choice for x_{i_0} is usually clear from physical considerations, the value of g_{j_0} is more difficult to select. For this reason, comparing sensitivity values across different constraints will not yield meaningful insight and in this work, we focus on comparing sensitivity values within each constraint.

2.1.3 Other interpretation of the Lagrange multipliers using sensitivities [29]

Sensitivities and Lagrange multipliers can be correlated using the following theory. Assume that two sets of variables are used, state variables s and decision variables d . The number of state variables, m, is equal to the number of equality constraints plus the number of active inequality constraints while the number of decision variables, $(n-m)$, is equal to the number of independent design variables, n, minus the number of state variables. The design vector x can therefore be partitioned as follows : $x = (s,d)^\top$, with $s_i = x_i;\ i = 1,..,m$ and $d_i = x_i;\ i = m+1,..,n.$ $(n-m)$ represents the number of degrees of freedom of the system. For each new active constraint, the system loses one degree of freedom.

Assuming that ∂x represents a small perturbation about any feasible point x, ∂x can be written as $\partial x = (\partial s, \partial d)^\top$. However, if decision variables can have arbitrary perturbations ∂d_i , the state variable perturbations ∂s_i must conform to feasibility.
For x^* the optimal solution, f the objective, h the set of equality constraints and active inequality constraints, and considering small perturbations (first order) near the zero value of the constraints, we have:

$$\partial f = \left(\frac{\partial f}{\partial d}\right)\partial d + \left(\frac{\partial f}{\partial s}\right)\partial s \tag{2.8}$$

Similarly,

$$\partial h = \left(\frac{\partial h}{\partial d}\right)\partial d + \left(\frac{\partial h}{\partial s}\right)\partial s \tag{2.9}$$

We can rewrite (2.9) as

$$\partial s = \left(\frac{\partial h}{\partial s}\right)^{-1}\partial h - \left(\frac{\partial h}{\partial s}\right)^{-1}\left(\frac{\partial h}{\partial d}\right)\partial d \tag{2.10}$$

Finally, using (2.8) and (2.10), we get:

$$\partial f = \left(\frac{\partial f}{\partial d}\right)\partial d + \left(\frac{\partial f}{\partial s}\right)\left(\frac{\partial h}{\partial s}\right)^{-1}\partial h - \left(\frac{\partial f}{\partial s}\right)\left(\frac{\partial h}{\partial s}\right)^{-1}\left(\frac{\partial h}{\partial d}\right)\partial d \tag{2.11}$$

Using z, a new unconstrained function which would be equivalent to the original function f if the state variables had been eliminated, we now define the constrained (or reduced)

gradient of f as:

$$\left(\frac{\partial z}{\partial d}\right) = \left(\frac{\partial f}{\partial d}\right) - \left(\frac{\partial f}{\partial s}\right)\left(\frac{\partial h}{\partial s}\right)^{-1}\left(\frac{\partial h}{\partial d}\right) \tag{2.12}$$

In that case, (2.11) can be rewritten:

$$\partial f = \partial z = \left(\frac{\partial z}{\partial d}\right)\partial d + \left(\frac{\partial f}{\partial s}\right)\left(\frac{\partial h}{\partial s}\right)^{-1}\partial h \tag{2.13}$$

At the optimal point x^*, $\left(\frac{\partial z}{\partial d}\right) = 0$, then

$$\partial f(x^*) = \partial z(x^*) = \left(\frac{\partial z}{\partial h}\right)^*\partial h \tag{2.14}$$

where $\left(\frac{\partial z}{\partial h}\right)^* = \left(\frac{\partial f}{\partial s}\right)^*\left(\frac{\partial h}{\partial s}\right)^{*\ -1}$.
At a stationary point that is a minimum, we may define :

$$\lambda^\top = -\left(\frac{\partial f}{\partial s}\right)\left(\frac{\partial h}{\partial s}\right)^{-1} \tag{2.15}$$

leading to the relation :

$$\left(\frac{\partial z}{\partial h}\right)^* = -\lambda^\top \tag{2.16}$$

Equation 2.16 shows that the sensitivity coefficients of the objective relative to the constraint (i.e. the rate of change of the optimal values of the objective relative to small changes in the values of the constraints) are given by the opposite of the Lagrange multiplier vector. It is important to note that this solution is only valid for small changes, due to a first-order approximation valid locally near x^*.

This gives valuable information on the quality of a constraint. If $\| - \lambda^\top \|$ is large, the effect of the constraint on limiting the value of the objective is also large. Conversely, if $\| - \lambda^\top \|$ is small, the presence of the constraint is not very restrictive to the objective value. However, the λ contains two pieces of information:

- The direction of the gradient of z relative to the gradient of h.
- The magnitude of the gradient of z.

As Figure 2-3 shows, in the case of two constraints, the impact of slightly moving one constraint can not be completely determined by the sole magnitude of the gradient of z with respect to the constraint. Two things can vary: While $\frac{\partial z}{\partial h}$ is available through the

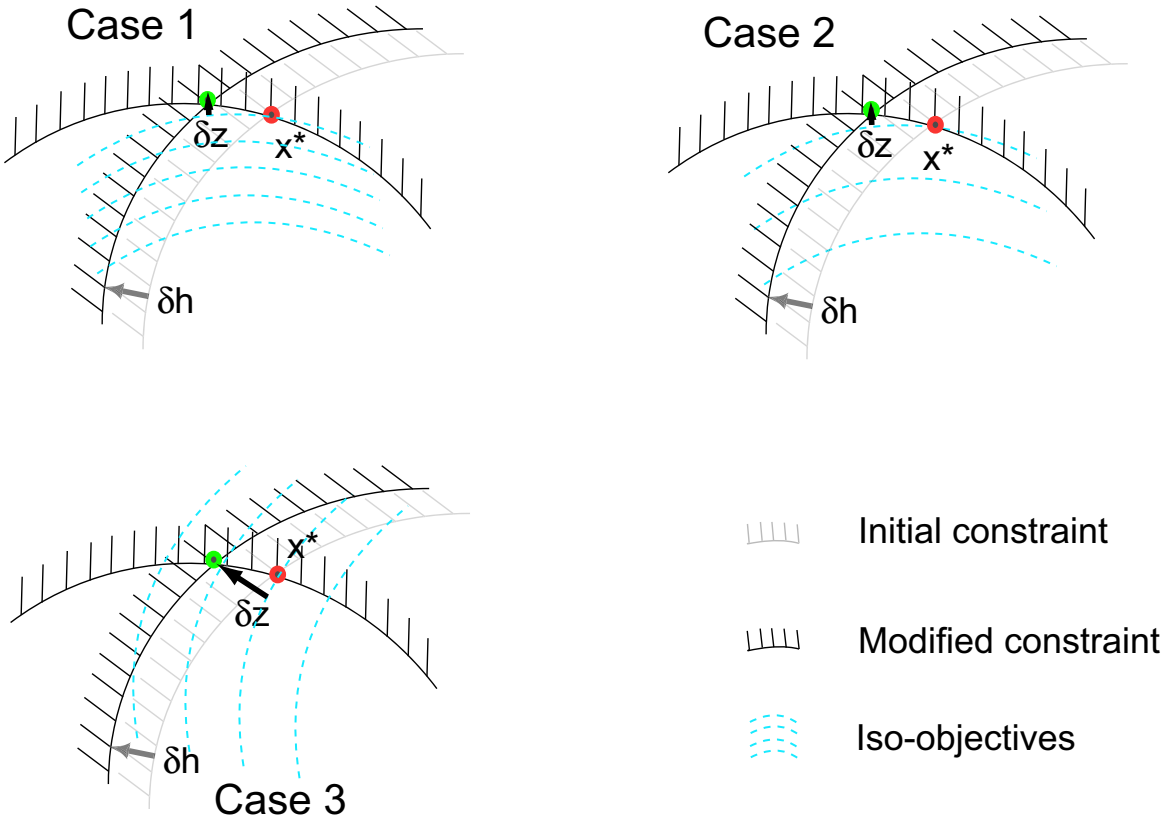


Figure 2-3: Different examples that show the difference between the magnitude and the direction of the gradient

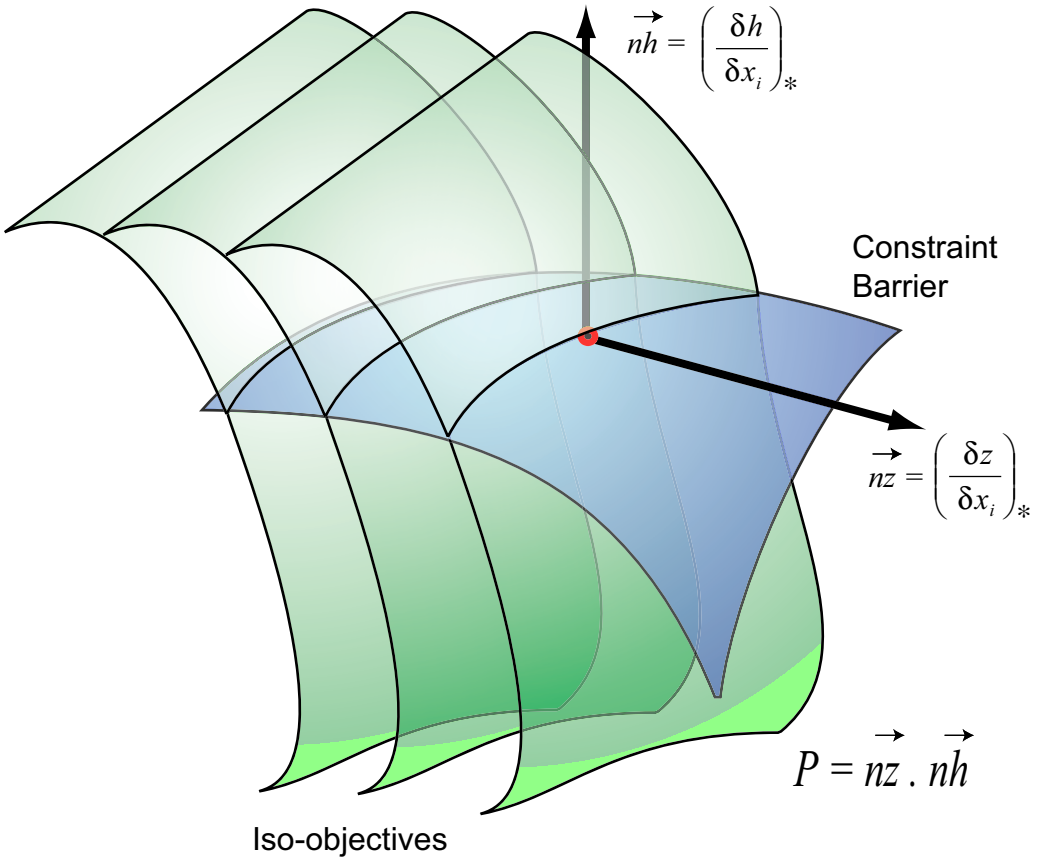


Figure 2-4: Dot product application to the 3D case

Lagrange multiplier, it does not bring all the information to analyze the design. In Case 1 and Case 3, the orientation of the iso-objectives and the orientation of the constraint are identical. Therefore, Case 1 can be compared to Case 2. The information carried in the Lagrange multiplier is sufficient. Comparing Case 1 and Case 3 by just using the Lagrange multiplier is impossible: the orientations of the iso-objectives are different. In this case, additional information can be added by looking at the dot product between the normal to the iso-objective and the normal to the constraint barrier, as illustrated on Figure 2-4 for a 3D environment.

With $\overrightarrow{nh}$ and $\overrightarrow{nz}$ normalized, the evaluation of $P = \overrightarrow{nh}.\overrightarrow{nz}$ (with $\overrightarrow{nh} = \left(\frac{\partial h}{\partial x}\right)^*$ and $\overrightarrow{nz} = \left(\frac{\partial z}{\partial x}\right)^*$) gives precious information, as shown on Figure 2-5. For a small P, one can expect that moving the constraint will not have a big impact on the value of the objective (Case 2 of Figure 2-3). Conversely, for P close to one, moving a constraint will bring a significant improvement in the value of the objective (Cases 1 and 3).

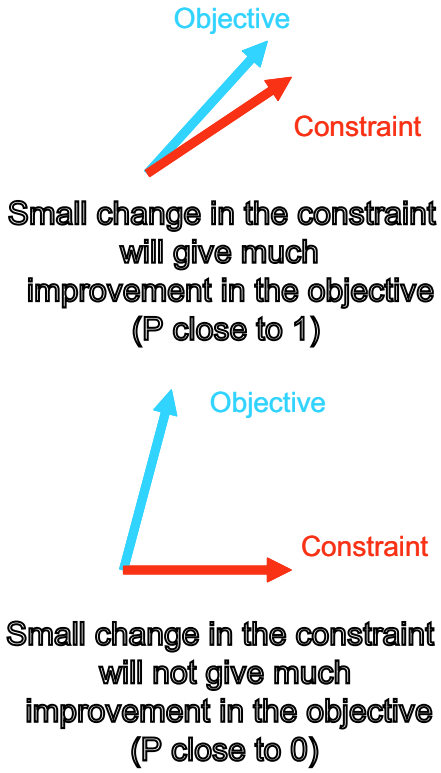


Figure 2-5: Effect of constraint changes on the objective as a function of the dot product P

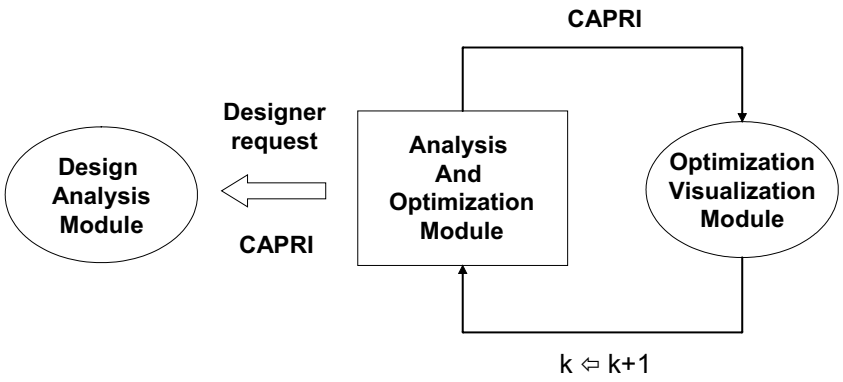


Figure 2-6: Architecture of the framework developed for visualization

2.1.4 Optimization/visualization framework

As was seen above, three mathematical pieces of information have been identified that are of considerable interest to the designer: the sequence of iterates, $\{x^k\}$, the Lagrange multipliers at the optimal solution, λ_j^*, and the sensitivities $\frac{\partial g_j}{\partial x_i}$. Two modules will be developed to visualize this information in a physical manner. Figure 2-6 shows the architecture of the framework that has been developed to link the three modules: while the optimization takes place, the Optimization Visualization Module (OVM) allows the real-time visualization of the optimization process, i.e. the sequence of iterates during the optimization. The Design Analysis Module (DAM) allows the designer to interrogate a particular solution. It displays the active constraints, show the physical impact of the constraints on the design, and demonstrate how the constraints drive the design variables and the objective function. These visualization modules should satisfy a set of requirements. First, the modules should be flexible. They must couple easily with different geometric representations of the physical system and with different optimization frameworks. Also they must be able to handle a broad range of problems. Second, the visualization modules should not substantially increase the computation time of the optimization run. To help meeting the latter requirement, all the communications and information exchanges between the different modules and the CAD model use the Computational Analysis PRogramming Interface (CAPRI) that is further described in Section 2.2.

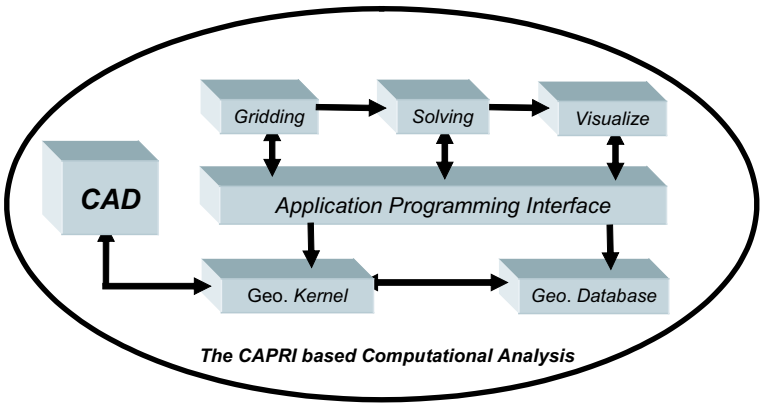


Figure 2-7: The CAPRI based Computational Analysis Suite

2.2 CAPRI

2.2.1 Overview of CAPRI

CAPRI [17] is a CAD vendor-neutral API. This middleware provides appropriate programming access for analysis suites that require direct access to the CAD model and can be used when the desired analysis does not have a direct CAD connection. The CAPRI programmer and/or user need not be a CAD operator.

Figure 2-7 shows that by allowing access to the geometry from within all the analysis sub-modules (grid generators, solvers and post-processors) such tasks as meshing, solver-based node adaptation and general geometry queries become simpler and consistent. The connection to the geometry is made through an API, which isolates the analysis suite from the geometry kernel, which avoids any loss of solid geometry information in a translation. The Geometry Viewer of CAPRI will also be used. It is not an integral part of the API, but can be thought of as another module of the software suite. It can be used as the visual front-end for CAPRI.

For the work described in this thesis, all CAD access was performed through CAPRI after a CAD part was constructed in a "parametric" sense (called the Master-Model). Pro/Engineer was used as the back-end, but any supported CAD system could have been applied to the work presented here. In this case, CAPRI accesses the CAD data in a native manner through Pro/Toolkit (Pro/Engineer's internal API).

CAPRI's API avoids the complete Computational Geometry (CG) perspective while

maintaining full functionality. This simplification of the data definition and API provides ease of software generation, without regard for special cases.

2.2.2 CAPRI's main capabilities

The important functions found in CAPRI that were used are summarized below:

(1) Support Manifold Solids. By only supporting solid geometry, problems in trimming surfaces do not exist. If handled properly, the geometry need not be fixed or modified.

(2) Direct CAD Access. The data exposed through CAPRI exists in the CAD system. There is no geometry translation, thus avoiding the errors and other problems associated with CAD model translation.

(3) Dual View of the Geometry. Both the CG and a discrete view of the solid are available through CAPRI. A complete, closed, conformal tessellation[16] of the geometry found in the CAD system is exposed on a surface-by-surface basis. This provides a proper foundation for the type of functions that are required for visualization task at hand.

(4) Master-Model Manipulation. By allowing the specification of both the parameter values (that define the geometry construction) and suppression of nodes of the "feature-tree", different instances of the part (or assembly) can be constructed. This portion of CAPRI allows for both geometric parameter studies as well as full design optimization.

(5) Shape Modification. It has been found that the parametric Master-Model view is inadequate for shape optimization (shapes tend to be too complex to drive in this manner). CAPRI allows for the direct manipulation of specific curves that are the basis for the operations of blending, lofting, extruding and rotating. For example, an application then has the ability to change the shape of a wing by adjusting the curves that define the airfoil shapes at various sections.

2.3 Visualization and constraint analysis methodology

2.3.1 MDO framework requirements

The methodology that is being developed in this thesis, as part of a more general MDO framework, ought to satisfy requirements. Many of these requirements were described

carefully by Salas and Townsend [35] and the most significant requirements in the scope of this work can be repeated verbatim for clarity purposes:

(A) Architectural design requirements

  (1) A framework should provide a Graphical User Interface that is intuitive

  (2) A framework should be extensible and should provide support for developing the interfaces required to integrate new processes in the system

  (3) A framework should support collaborative design

(B) Problem formulation requirements

  (1) A framework should support the user in incorporating legacy codes [...] and proprietary codes [...] into the MDO problem formulation

  (2) A framework should allow the user to integrate discipline analyses with several optimization methods [...]

(C) Problem execution requirements

  (1) A framework should be able to execute multiple process in parallel

  (2) A framework should support user interaction (steering) during the design cycle

(D) Information access requirements

  (1) A framework should provide database management

  (2) A framework should provide the capability to visualize intermediate and final optimization and analysis results

  (3) A framework should provide monitoring capability for viewing the status of an execution, including the system status

Meeting these requirements is not an easy task, but the methodology presented in the next section tries to incorporate them rigorously.

2.3.2 Methodology

As was shown before, a close interaction between three modules has been created, supported by the capabilities of CAPRI. Information is exchanged between the AOM, the OVM

and the DAM. However, management of the information, mainly at the mathematical level, between these modules, and during the optimization process, is a challenge. The following methodology has been developed in order to face this challenge and will be used in order to implement the different modules.

(1) Define the optimization problem in terms of identification of the design variables, of the objective(s) and of the constraint(s). As for any optimization problem, it is of major importance to give extra care to the characterization of the system: what is input or output entirely depends on the viewpoint from which the system is being observed. A clear distinction between design variable and design parameter is important at this stage. The distinction between constraint and objective must also be carefully analyzed, especially since the definition of a system appears intimately linked to the different backgrounds or disciplines to which the people contributing to the definition belong. Finally, identifying the objective function that is of importance is a challenge, and will most likely greatly influence the outcome of the optimization.

(2) Track the evolution of the design variable values during the optimization, as well as values of constraint and objective functions. As the optimization progresses, the design vector will be changed by the optimizer. It is important to monitor these changes. Instantaneously tracking the variations of the design variables will allow the exchange of data between the optimizer and the CAD model. A posteriori, it will allow the user a retrospective analysis of the optimization, following a more classical approach (two or three-dimensional plots of the evolutions of the variables).

(3) Use Lagrange multipliers to determine activeness of the constraints at a given design point. As demonstrated in Section 2.1.2, a non-zero Lagrange multiplier will express the activeness of a constraint. Notice that an equality constraint is by definition always active.

(4) Use sensitivity analysis to determine the participation of the design variables in the different constraints. This step is really important in preparation of the display. It will enable a constraint to be related to a specific physical feature on the system. However, the sensitivity analysis is often very dependent on the normalization quantities.

(5) Rank predominant variables for each constraint. Based on the normalization tech-

nique used, it is possible to determine the design variables that have a significant participation in a given constraint. Significant participation will imply that a slight change in the value of the constraint can lead to a big change in the value of the design variable for a new feasible design. Inversely, it will also imply that a small change in the value of the design variable will cause a serious violation of the constraint.

(6) Allow data exchange between optimizer loop and visualization module using CAPRI. This step will be described in Chapter 3. CAPRI, by allowing direct access to the CAD geometry, will allow the exchange of data in both directions. The new information issued by the optimizer will be sent to the CAD model. A new tessellation will be created, and will allow the display of the updated model. In the other direction, information of the CAD model (dimensions, geometry) can be retrieved by the AOM for different purposes (high-fidelity analysis, data, etc.).

(7) Display optimization path and constraint information. Discussed in Chapter 3 and 4, the display of the optimization path and the constraint information is a novel and potentially valuable tool to the user in the design process.

This high level framework methodology will be used in designing the graphical user interface, and the data exchange process throughout the progress of the work.

2.4 Conclusion

This chapter briefly established the mathematical foundations of the optimization and visualization problems. The constraint analysis, using both Lagrange multipliers and sensitivity analysis allowed better understanding of the importance of a specific constraint and gave an idea of the participation of the design variables in given constraints. An overview of the architecture of the optimization/visualization framework that will be further developed was given. CAPRI was then presented as the main interface between the different modules in this same optimization/visualization framework. Main characteristics of CAPRI were given. Finally, a visualization and constraint analysis methodology was presented, and will offer the reader a baseline for the work covered in this thesis.

In the following chapters, the methodology is decomposed into two parts. Chapter 3

focuses on the methodology used to display the optimization process in real-time. It also analyzes the value of visualization in the design process. Chapter 4 emphasizes the constraint analysis and how constraints can be physically displayed.

Chapter 3

Real-Time Visualization

As discussed earlier, one of the issues facing MDO is the disconnection between the designer and the physical representation of the system being designed. An optimizer will take full advantage of gaps in the formulation. For instance, if a constraint is omitted or cast incorrectly, an optimizer will often return a physically unreasonable design or will fail to converge. However, when the optimizer output is communicated to the designer by a list of numbers, it is difficult to quickly observe and diagnose this behavior. Offering a more physical representation to the designer by visualizing optimization data on a CAD model will enable the following:

(1) Have a physical representation of the system being optimized.

(2) Improved understanding of the tradeoffs that are made in the optimization design process.

(3) Assist in the early stages of optimization problem formulation and implementation.

(4) Easily determine the path taken by the optimizer through the design space and interrogate individual design options.

(5) Verify assumptions regarding the physical configuration of the system. In many cases, the designer may not be aware that these assumptions were made.

(6) Simplify the downstream management of the assembly and manufacturing tasks by identifying problems upfront. For example, the optimizer could decrease the thickness

of a part to a unreasonable value that would prevent the part from being manufactured. Monitoring the design evolution would help to identify this more quickly.

3.1 Approach

The methodology developed to visualize the progress of an optimization routine can be summarized by the following four steps: creation of a CAD model, parameterization of the CAD model, validation of the CAD model, and linking of the CAD model to the optimization framework.

3.1.1 CAD model description

The first step in developing the real-time visualization module is to create a robust CAD model. The CAD model should reflect the design approach and encapsulate the trends of the final design. It should capture the aspects of the physical system that are most likely to evolve throughout the optimization and/or should be communicated to the designer. At the least, it should include the key parameters of the design. MDO is often used for conceptual or preliminary design, hence a model that roughly represents the proportions of the object will, in general, lend sufficient insight to the designer. However, a higher fidelity CAD model may be beneficial for other issues, such as assembly and manufacturing concerns. Finally, it should be noted that it is important to recognize the limitations of a coarse CAD model and use caution when utilizing information, such as a mesh, as input to a higher fidelity model.

3.1.2 CAD model parameterization

Parameterization of a CAD model requires insight and input from the designer. The parameterization should be carried out in the context of linking the CAD model to an optimization framework, however, it is not necessary that all design variables map to a different parameter in the CAD model. Instead, simple relations can be defined, which allow the design variables to map to a reduced set of CAD model parameters. Another issue is that the parameters used in the CAD model may not be geometrically linked with any particular design variable. Especially for complicated representations, there may be a number of CAD model parameters which are hidden from the optimization, that is, they

are set by relations within the CAD model and cannot be accessed or modified from the optimization framework. Even though this approach may simplify the CAD model, it is advised to use extra parameters in order to suppress such hidden definitions and create a more robust model.

Different parameterization approaches can be conceived, which lead to the same geometric representation. For example, consider a simple tapered wing. Two parameterization approaches can be considered, as shown in Figure 3-1. The first approach defines the wing using the following nine parameters: wing area, wing span, taper ratio, sweep angle, dihedral angle, thickness distribution and the X, Y, Z position of a point of the wing. This could also be complemented by the choice of specific airfoils.

The second approach puts the emphasis on the airfoils rather than on the wing itself. One is located at the root, the other at the tip, but both are defined and located by their own set of parameters. In this case the ten parameters are the spatial location (X, Y, Z) of one point at the leading edge of each airfoil, the chord length and the thickness ratio, both defined at the root and at the tip.

The two approaches give a same result, as shown in Figure 3-1, but largely differ by the fact they involve different sets of design variables, and allow different level of flexibility. For example, the second approach permits the addition of more airfoils between the root and the tip. However, the second approach is less physical than the first one, that may result in a loss of understanding by the designer.

3.1.3 CAD model validation

Once the CAD model has been created, it needs to be tested for different values of the parameters. One should make sure that a particular value of a parameter does create a geometry that the CAD system cannot generate. The range of variation of the parameters should at least include the projected feasible design space. This is important to ensure that the coupled optimization-visualization framework is robust.

3.1.4 Optimization-visualization link

As Figure 3-2 shows, the main idea is to have two sequences of separate actions running in parallel (a double-threaded approach) that can communicate with each other and share resources. The double-threaded approach allows concurrent processes, and therefore

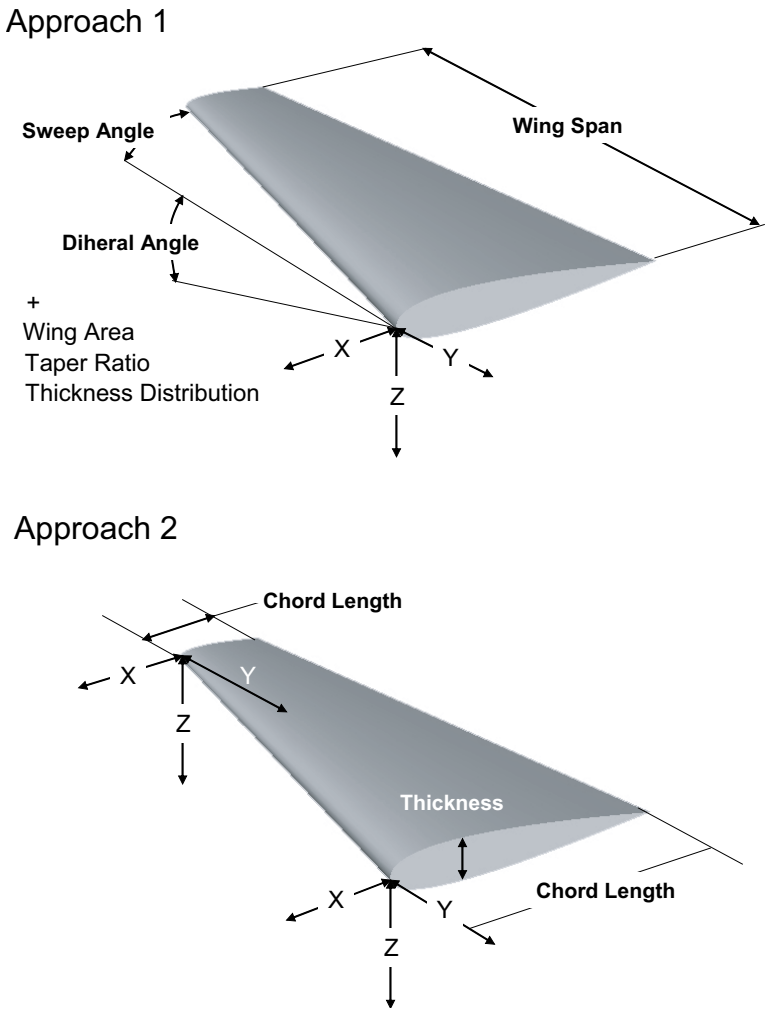


Figure 3-1: Two different approaches for the parameterization of a tapered wing.

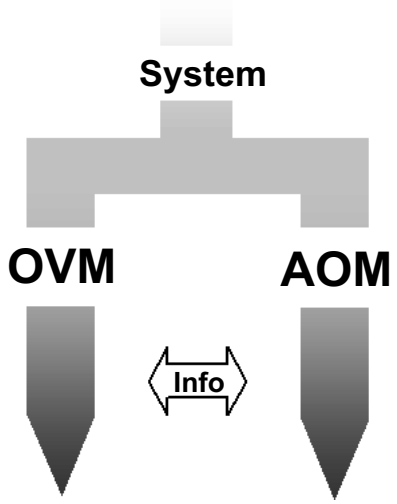


Figure 3-2: Double-threaded approach. First thread carries analysis and optimization module. Second thread carries dynamic visualization module

real-time visualization, and is particularly interesting and efficient for computer-intensive problems. This step was necessary in order to ensure the continuous update of the CAD model while the optimization is running. The link between the two sequences, the OVM and the AOM, is carried out via CAPRI. CAPRI is also the interface between the CAD model and its definition, the model analysis tools, and the visualization module. It allows for information to be shared between the three components, as shown in Figure 3-3. On the right of the figure, the design module executes the optimization loop and runs the analysis routines. The analysis routines are defined by the user, and can be of different fidelities. The main goal of the architecture is to allow different types of routines to be plugged in on the architecture. This modularity is essential to MDO, and was identified as a main requirement by Salas and Townsend [35]. Another characteristic of the framework is the flexibility to interface with different optimization methods. While in the present case, gradient search methods are used for optimization, one can easily switch to other methods, such as heuristic methods, without major changes. Modularity being a key component, objective and constraint functions are written such as they can be used by any type of optimization codes. On the left, the visualization module dynamically displays the evolution of the system as the optimization proceeds. As the schematic shows, the OVM gets the

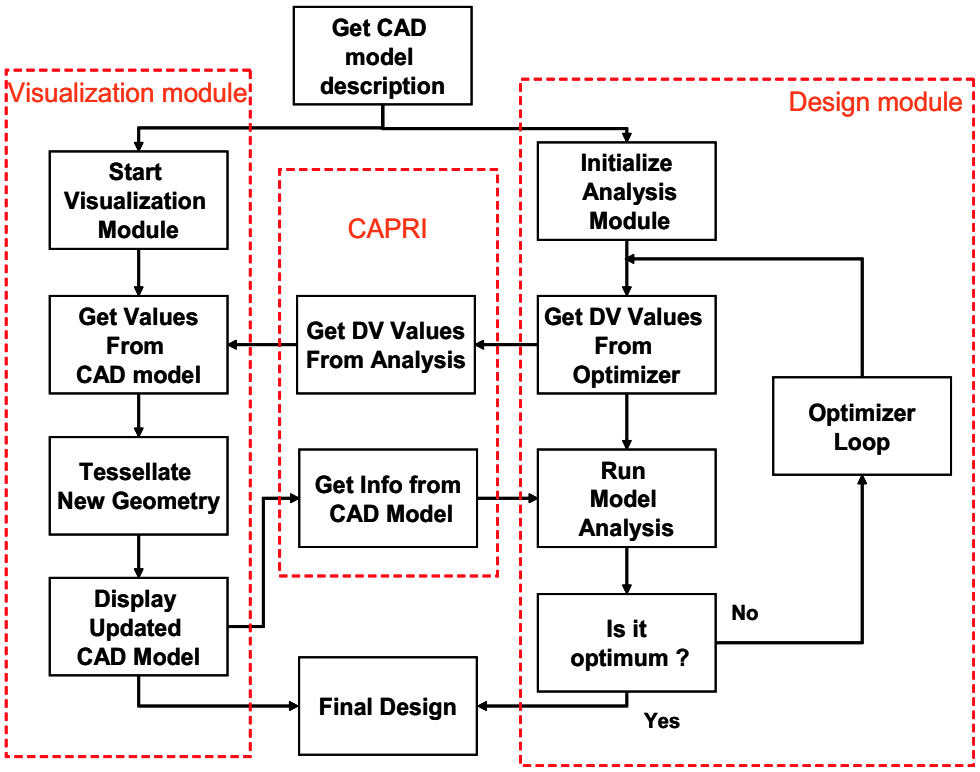


Figure 3-3: Approach of the visualization problem

information from the AOM via CAPRI. Using the tessellation routine of CAPRI, the new geometry, which encapsulates the changes resulting from the optimization, is regenerated and can be displayed. One of the major advantages of this technique is that the tessellation created by CAPRI can be used by the AOM. By using CAPRI, the user is provided with a water-tight geometry, based on the configuration at the time of the regeneration. The tessellation can then be used as input for a Computational Fluid Dynamics (CFD) analysis, or a Finite Element Method (FEM) analysis for example.

We can also notice that different options must be given to the user during this phase. A major requirement of a MDO framework is the capability to track, a posteriori, the histories of chosen objective functions, constraints or design variable values. To meet these requirements, the framework allows the user to save all the numerical data, and/or snapshots of the design at each iteration. This information can then be reviewed outside of the loop, and can provide a better understanding of issues, such as what might have gone wrong. A limit to this is the size of the problem. While this is applicable to small to medium scale problems, large problems requires a selection of the information that is saved.

3.2 Impact of visualization on the design

3.2.1 Rate of learning vs. complexity

Computation time, even with the dramatic increase in computer power, remains an important issue. Applications of MDO are tending towards consideration of more complex systems, use of higher fidelity tools and use of more complex optimization techniques. Venkatara and Haftka identified three types of complexity [38]:

(1) Model complexity, inherent to the size of the problem, and related to the number of design variables and constraints.

(2) Analysis complexity, related to the level of fidelity of the models used. Fidelity ranges from low-fidelity, empirical models to medium-fidelity models based on a simplified approach, such as beam structural models or panel methods for aerodynamics. High-fidelity models, such as CFD and finite element structural analysis, are typically used in MDO for post-analysis and limited optimization [8].

(3) Optimization complexity, related to the type of optimization: linear or nonlinear, deterministic or probabilistic.

However, the visualization problem does not depend on any of these three types of complexity. Rather, it is completely dependent on the complexity of the CAD model. The tessellation that occurs at each step of the optimization for visualization requires computational resources proportional to the complexity of the shape to be created. This leads to a trade-off between the need for information and the ratio between the time needed for the combination of analysis and optimization and the time required for the visualization while accounting for the time available.

Rubbert [34] evaluates the quality of a design solution by the amount that can be learned over a certain period of time.

$$(Rate\ of\ Learning) = \left(\frac{Learning}{Cycle}\right) * \left(\frac{Cycle}{Time}\right) \tag{3.1}$$

The first term, $\left(\frac{Learning}{Cycle}\right)$, represents the amount of information that is gathered during a design cycle. The second term, $\left(\frac{Cycle}{Time}\right)$, represents the number of cycles that can be effectively carried out in a given amount of time. The product of these two terms can give

the designer an idea of the amount of information that can be obtained over a given amount of time. The higher the rate of learning, the more valuable the information for the designer. However, in order to evaluate the value of bringing visualization in the design process, we need to think in terms of incremental change brought by the visualization. Rubbert uses the following terms: $\left(\Delta\frac{Learning}{Cycle}\right)$, which represents the new information that the visualization will bring to the designer and $\left(\Delta\frac{Cycle}{Time}\right)$, which represents the additional number of cycles that can be executed in a given amount of time. In the case of the visualization, this term is negative since bringing visualization into the design will increase the amount of time necessary per cycle. Ignoring higher order terms, the ratio of the incremental value of visualization to the value of previous information can be written as:

$$\frac{\Delta Value_{Viz}}{Value} = \left(\frac{\Delta\frac{Learning}{Cycle}}{\frac{Learning}{Cycle}}\right) + \left(\frac{\Delta\frac{Cycle}{Time}}{\frac{Cycle}{Time}}\right) \tag{3.2}$$

3.2.2 Effect of the visualization and the complexity of the CAD model on the learning rate

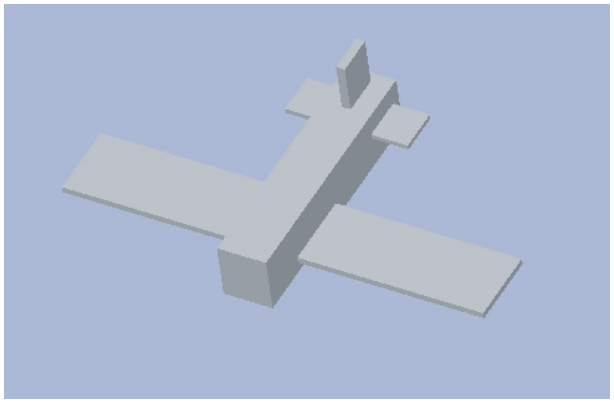
Using this approach, the effect of complexity of the CAD model on the rate of learning can be analyzed. Complexity of the CAD model has been decomposed into three levels:

(1) Low complexity: the CAD model encapsulates only the general ideas of the design at the simplest level. In the case of a GA aircraft, the CAD model of the system to be designed looks like an enhanced box as shown in Figure 3-4 (a).

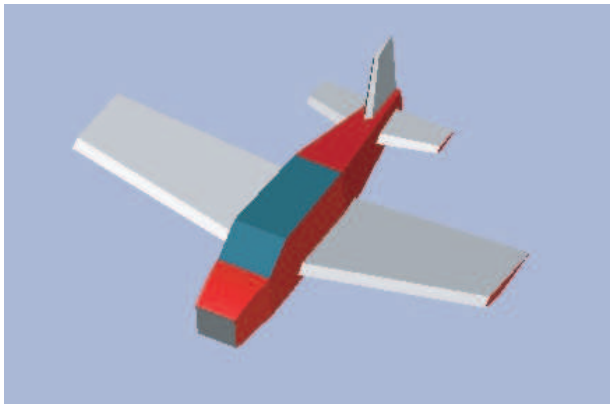
(2) Medium complexity: the CAD model encapsulates the details of the design that are necessary for physical understanding. However, features such as fillets and corners are not considered unless they bring added value to the designer (Figure 3-4 (b)).

(3) High complexity: the CAD model is an exact representation of the physical system, including details such as fillets. This CAD model could be used directly for CFD analysis or in the manufacturing process (Figure 3-4 (c)).

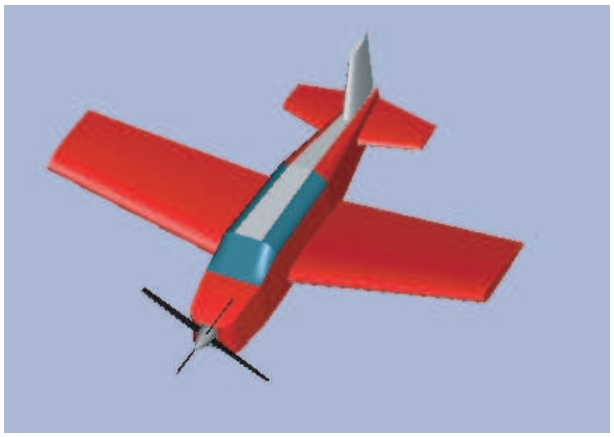
"No CAD model" will represent the status of the framework when no visualization is used.



(a) Low complexity



(b) Medium complexity



(c) High complexity

Figure 3-4: Different complexity levels for a CAD model of a General Aviation aircraft

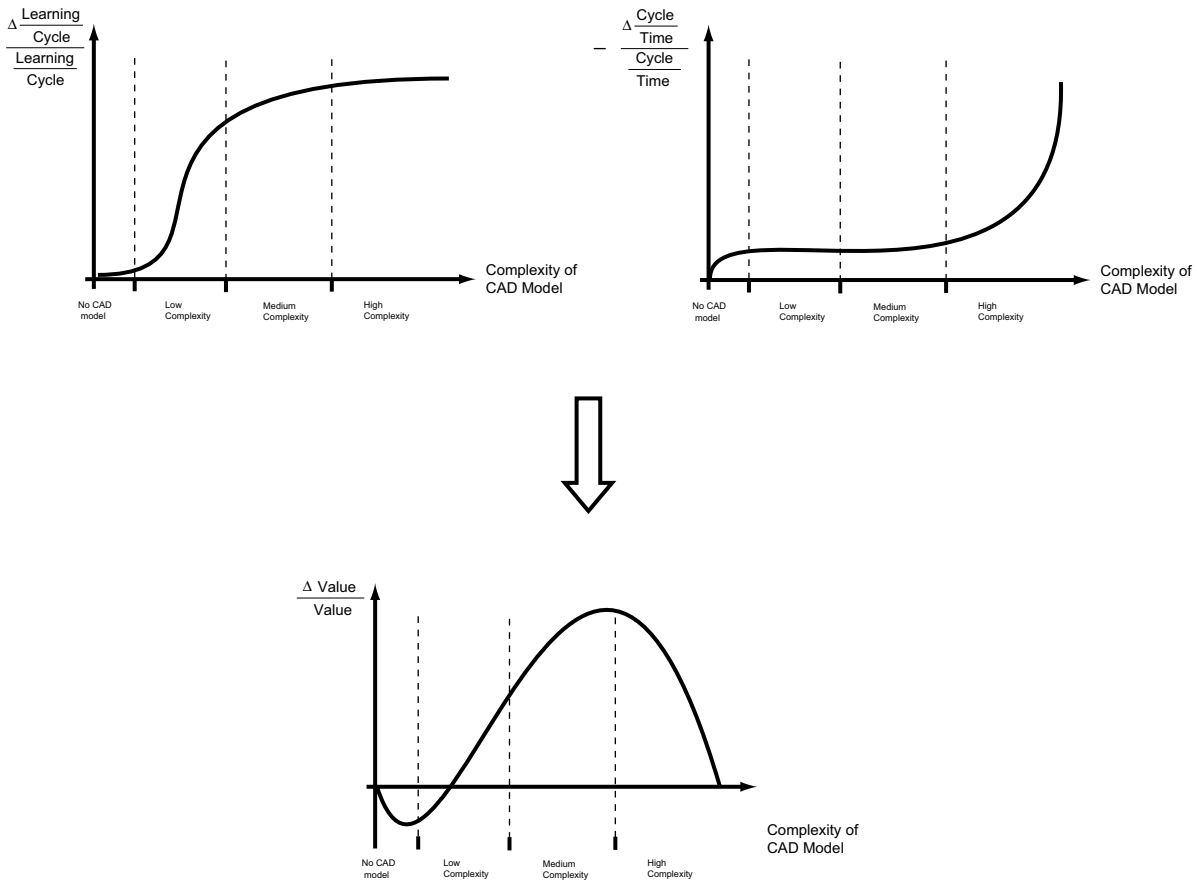


Figure 3-5: Effect of the complexity of the CAD model on the value of visualization

Figure 3-5 shows the amount of added information received per cycle, the extra number of cycles that can be computed over a given period of time, and, finally, the added value that the visualization brings, as functions of the complexity of the CAD model (refer to equation (3.2)). It can be seen that the amount of added information received per cycle is largely dependent on the complexity of the CAD model. No visualization obviously brings no additional information. However, as soon as a CAD model is used simultaneously with the optimization, the engineer's insight is significantly improved. As complexity increases, the amount of details becomes excessive, often without relevance or impact for the user. While being significant for the manufacturing process, excessive details do not bring extra information in the optimization process. Therefore, in Figure 3-5, a plateau can be observed in the amount of added information as the CAD model complexity becomes high. Moreover, the extra details that are not of significant importance for the designer will prove to be a time penalty factor. While the tessellation process is not significantly affected by changes of

complexity in the low-end part, it is drastically complicated by small details and increased complexity in the high-end. Tessellation is a process that can take a significant amount of time when processing complicated geometry, and is necessary in the scope of CAPRI (Note however that high complexity analysis methods (CFD, FEM) might use the tessellation). For example, while a box will be tessellated in a very short time, very sharp edges found in high complexity models can lead to long processing times. This can explain the fact that, on the upper right corner of Figure 3-5, the number of cycles for a certain amount of time encounters a sharp decrease when the complexity of the CAD model reaches "unreasonable" levels. Using equation (3.2), one can now plot the value of adding visualization versus the complexity of the CAD model. The bottom part of Figure 3-5 shows that, when using no CAD model, or a low fidelity CAD model, using visualization is a burden. However, as soon as the complexity of the CAD model becomes an enabling factor to better understanding of the geometry and of the optimization process, the value of visualization is obvious. Notice that for very high complexity, the value decreases, as the time taken to visualize cannot compensate the extra information brought by the model.

In conclusion, Figure 3-5 demonstrates the value and the need of having a CAD model that balances accuracy and computation time. Such a model should be accurate enough to represent reality, but should not include details superfluous to the designer's needs.

3.2.3 Complexity and value of visualization

As was shown, complexity of the CAD model is a big driver in the value of the information brought by the visualization module. The bottom line is that visualization is a huge help when coupling with a medium-high fidelity CAD model, that includes all the main characteristics of the desired product, but does not encapsulate useless details. Regarding the complexity of the analysis model and/or the complexity of the optimization it appears once again that the value of the visualization is high for medium complexity problems. It brings sufficient information to help the designer without adding significant extra computation time. For low complexity problems, even though the information brought is crucial, the ratio between the time needed for the visualization and the time needed for the computation without the visualization is very high, and minimizes the advantages. For high complexity models, having a visualization module does not change anything in the total computation time, but the designer probably need less information about that design space at this level.

Indeed, high complexity analysis models and/or probabilistic optimization are often used to refine the previous results around an optimum.

3.3 Conclusion

This chapter gave a detailed overlook at the methodology required to provide an engineer with physical insight of the design optimization process. This methodology spans all aspects of the problem: selecting an appropriate CAD model complexity, careful buildup of this model, parameterization, and a linking between optimization and visualization using CAPRI. This combination of optimization and visualization provides valuable information to the engineer, and can be used with great advantage during the optimization, as was demonstrated in the last part of the chapter, with adequate CAD model complexity.

However, such information provided by the real-time visualization of the optimization steps can successfully be complemented by a novel approach: constraint analysis and physically based visualization of constraints. The approach combines a mathematical analysis of the constraints, using both Lagrange multipliers and sensitivity analysis. It leads to an efficient way to link constraints active during the optimization to physical features on the system. This methodology is then used directly in relation with the CAD model, and provides the designer additional information, that could be used to both improve design and reduce design cycle time.

Chapter 4

Constraint Analysis and Visualization

Obtaining a clear picture of the design space is a difficult issue to address. Displaying optimization progress through the design space is one approach that was discussed in Chapter 3. As will be discussed in this chapter, further insight may be gained by a more detailed investigation of constraint behavior. In particular, the designer may be interested in discerning which constraints are active, which constraints affect which parts of the system, and how much certain constraints drive the design. Effective visualization of this information is not an easy task. The methodology must be developed in such a way that it is flexible, applicable to general problems and automated. As discussed earlier, Winer and Bloebaum [41] use Graph Morphing to display the constraints on a 3D representation of a subset of the design space. Here, a novel, more intuitive approach is used, which visually ties the constraints to physical features of the system. Such an approach was performed manually for a Blended-Wing-Body optimization result, and was found to lend valuable insight to the designer [39]. The DAM developed here allows physical visualization of the constraints to be performed automatically in real time. This chapter describes the methodology that is developed to allow the detailed analysis of the constraints. This includes first a sensitivity analysis approach, and then the linking of physical features of the system to elements of the optimization that leads to the constraint visualization.

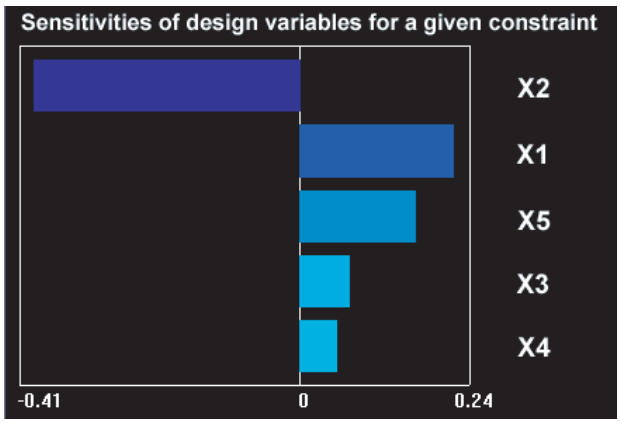


Figure 4-1: Effect of the five design variables of the GA aircraft model on the wing bending stress constraint.

4.1 Constraint sensitivity analysis

The first step of the approach is to determine which constraints are active at a given design solution. This can be easily achieved by finding the constraints with non-zero Lagrange multipliers. The relative participation of the design variables in each constraint is then determined using (2.7). This sensitivity analysis results in a matrix containing the normalized sensitivities of each constraint with respect to each of the design variables, where a row corresponds to a particular constraint and a column corresponds to a particular design variable. It is important to note that the normalization allows a good comparison of the effect of each design variable within a given constraint (going across a row of the matrix). However, there is no single way to scale each constraint, thus preventing the effective comparison of the relative importance of each constraint (going down a column of the matrix).

Once the importance of each design variable within a specific constraint has been calculated, it is possible to rank the design variables by comparing the sensitivities defined by equation (2.7). This ranking will give the designer insight as to which design variables have an effect on a constraint, and their relative importance. Figure 4-1 depicts how such information might be displayed using a bar graph. In this case, the normalized sensitivities of a particular constraint with respect to five design variables are plotted. As the figure shows, variables x_2, x_1 and x_5 are particularly important for this constraint. While the plot shown in Figure 4-1 can provide useful information to the designer, it does not instantaneously lend physical insight. Moreover, if the number of design variables and constraints is large, studying a large number of such graphs with many bars might be time-consuming

and ineffective. This approach should therefore be complemented with a more intuitive approach, which displays the information on a physical model of the system.

4.2 Constraint Visualization

4.2.1 Types of constraints

Different types of constraints can be identified. The type of constraint will directly affect the way it needs to be displayed.

The first type of constraint that can be encountered is a constraint that is directly linked to a geometric feature of the model. For example, if the designer considers a panel approach for the model, certain constraints can be directly related to a specific panel.

The most common type of constraint is one that is not directly linked to any geometric parameter. In this case, the sensitivity analysis discussed earlier allows the identification of the design variables that drive the constraint towards its bounds. The constraint is therefore linked to the physical model through the design variables.

4.2.2 A multi-level approach

A multi-level approach will be used in order to display the constraints. Each constraint will be linked to a set of design variables. Each design variable will contain three levels of information. The first level of information refers to the feature with which the design variable can be associated. The second level is the type of the design variable. Different types can be distinguished, including length, area, diameter, angle, ratio, and user defined. Finally, the third level represents the real meaning (for the designer) of the design variable, defined by its name. These three levels of information are depicted in Figure 4-2. L, S, A, D, R are the five different types that the design variables can carry, representing respectively Length, Surface, Angle, Diameter and Ratio. A sixth category, Other, can be added, and would represent all the types that cannot be included in the previous five, such as non-geometric design variables (e.g. speed, temperature or pressure).

Each level is represented by a graphical display. The first level of information is depicted by highlighting the specific feature on the CAD model representing the object to be optimized. The second level and third level are accessed by clicking on a given constraint, and then by selecting the design variable for which more information is needed. Doing so,

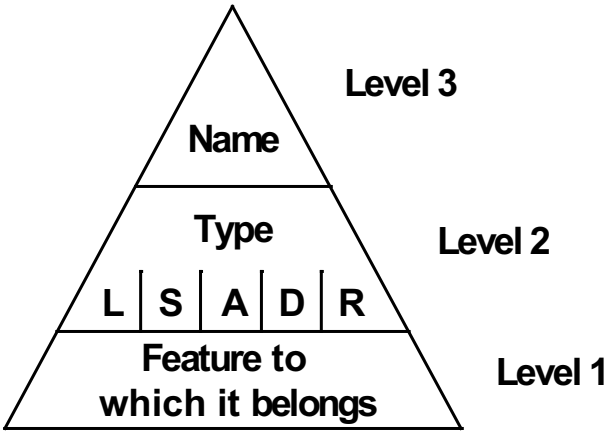


Figure 4-2: Multi-level approach for constraint visualization

the designer can choose, in order to perfectly identify all the design variables involved, to list the important design variables.

The entire process can be summarized as follows:

(1) Identify the active constraints using a Lagrange multiplier approach.

(2) Identify the design variables that can be associated to a specific active constraint using sensitivity analysis, rank them, and determine the significant design variables that drive the design for a given constraint.

(3) Make the distinction between a design variable that is related to a physical element and a design variable that is not linked to the geometry.

(4) Highlight the geometric features corresponding to the identified design variables.

(5) At the designer's discretion, display the second level of information in a separate window and reveal the type of design variables. Display non-geometric variables in another window.

(6) Display the names of the constraints and/or design variables.

This visual information is also complemented by a list that transmits the status of any constraint, as well as the design variables related to it, at the chosen solution point.

4.3 Conclusion

This chapter has hereby presented the reader with a novel approach and new methodology that offers even more insight on the design. The approach can be summarized in a few steps:

- Determination of the participation of the design variables for each of the constraint
- Linking of the design variables to physical features of the system
- Linking of the constraints to the system via the significant design variables.
- Use of a three level approach that provides the designer with valuable information on the constraint behavior and on the relation between the constraints and the system.

The approach is limited to systems for which the design variables can be associated to physical features (geometric design variables). This excludes designs of systems for which all the design variables are abstract (e.g. design and optimization of a HSCT Engine Cycle [33], where all the design variables are pressure ratios, temperatures or flows). However, many engineering systems have geometric inputs and will be very well suitable for the methodology.

Used jointly with the real-time visualization approach, the constraint analysis methodology offers new horizons to the engineer to better understand the behavior of the optimizer, and therefore refine, with great insight, the optimization framework. The implementation of the interface will embody the two approaches, by encapsulating the needs and requirements that have been expressed earlier. Its implementation is detailed in Chapter 5.

Chapter 5

Visualization and constraint analysis user interface

In the previous two chapters, a new methodology for visualization and constraint analysis was presented. This methodology allows the linking of the different modules (namely the AOM, OVM and DAM) in order to provide the designer with valuable information on the optimization process, on the system design behavior, and on the status of the constraints involved in the optimization. Based on the ideas about real-time visualization and constraint analysis detailed in Chapter 3 and Chapter 4, a user interface needs to be built around the framework to be used during the design of a system. As was shown earlier, the interface needs to encapsulates various ideas, but should also satisfy different requirements. This chapter explicitly describes the implementation of the interface, based on the methodology developed in this thesis, but also the needs and requirements identified previously. It also gives a detailed description of the interface. It finally emphasizes the role of the user in the preparation of the interface for an effective use.

5.1 Interface overview

The goal of the interface is to use the methodology described in Chapter 3 and Chapter 4 to bring visualization into the design process. First, the interface must allow real-time visualization of the optimization path, as described in Chapter 3. Second, the interface must enable visual navigation of the design constraints, using the approaches discussed in Chapter 4. The interface needs to meet the requirements that were expressed in Section 2.3.1, even

though many of these were directly answered by the methodology itself (by the use of multi-threading and modularity, as well as by the visualization approach). Finally, control over the two different parts of the interface must be given to the designer, as well as monitoring capabilities for the optimization and the interface itself. The complete framework that describes the interface will be decomposed into three distinct modules that will be detailed in Section 5.2.2.

This being said, the goals of the user interface can now be described. The first goal of the interface is to allow the user a visualization of the system being designed. This will be achieved by bringing CAD software capabilities into the interface. As a result, a 3D model display of the system will allow the user to monitor the evolution of the design. This function will be achieved by linking the CAD model to the AOM via CAPRI. As explained earlier, real-time capability will offer real-time monitoring of the optimization process, as well as health monitoring functionality. Real-time will be achieved by exploiting the double-threading capabilities of CAPRI. The next goal is to provide the user with information on the constraints, computed by the constraint analysis module. This includes information on the activeness of the constraint (both on a textual representation and a physically-based representation of the activeness), participation of the different design variables in a given constraint and sensitivity analysis.

5.2 Implementation of the interface

Now that the requirements and needs of the interface have been identified, the following step becomes the implementation. Some work has been done previously by Haimes [15], later completed by Darmofal and Haimes [12]. That work represents the foundation of the GV interface that is being provided with CAPRI, and give some interesting leads on how to visualize information. However, using this work as the base of the user interface required here was not clear cut. The following section analyzes the choice of the implementation language.

5.2.1 Choice of the implementation language

Building an interface often requires a specific language. GV was developed with some built-in interface capacities, but could potentially be limited for certain graphical applica-

Table 5.1: decision criteria in the choice of the implementation language

Criteria	coefficient	TCL/Tk	JAVA	GV
Speed	3	0	2	5
Portability/ Multiplatform	3	4	5	4
Compatibility with Capri/GV	3	3	2	5
Time required to code	3	4	3	4
Ease of programming	2	3	5	4
Look of interfaces	2	4	4	3
Help available	2	2	5	4
Ease of transferal to new developers	1	2	5	2
Number of references	1	2	5	1
Multithreading	1	2	4	3
Integration to web page	1	2	4	1
Results		63	82	83

tion. In order to be able to develop a very personalized interface, other languages could be used. Java [3] and Tcl/Tk [4] are probably the most common in industry, and present many advantages. In order to determine which one is the more suitable, a trade-study was conducted, based on literature resources [31], advice from knowledgeable people [1] and personal experience. A list of criteria was developed, with associated different levels of importance associated. Finally, three options were considered. In all cases, GV and CAPRI will be used for the 3D display, as the 3D viewing capability and interfacing between AOM and OVM will exploit CAPRI technology. The rest of the interfacing could be implemented using different languages:

- Java
- Tcl/Tk
- GV itself (based on C code), using the built-in functions, and developing new ones.

Table 5.1 summarizes the results of the analysis of the most suitable language for the implementation of the user interface. Coefficients were used to express the importance of each criteria. Scores, for each criteria range from 0 to 5, 0 being poor and 5 excellent.

As the table shows, even though many of the results are subjective, using GV as the core of the interface seems to be the best solution. Using Java brings portability and ease of web integration, but GV brings speed and compatibility with the existing code that cannot be neglected. Tcl/Tk appears to be not widely used and documented and, more importantly, too slow to be a major competitor to either Java or GV. It was therefore

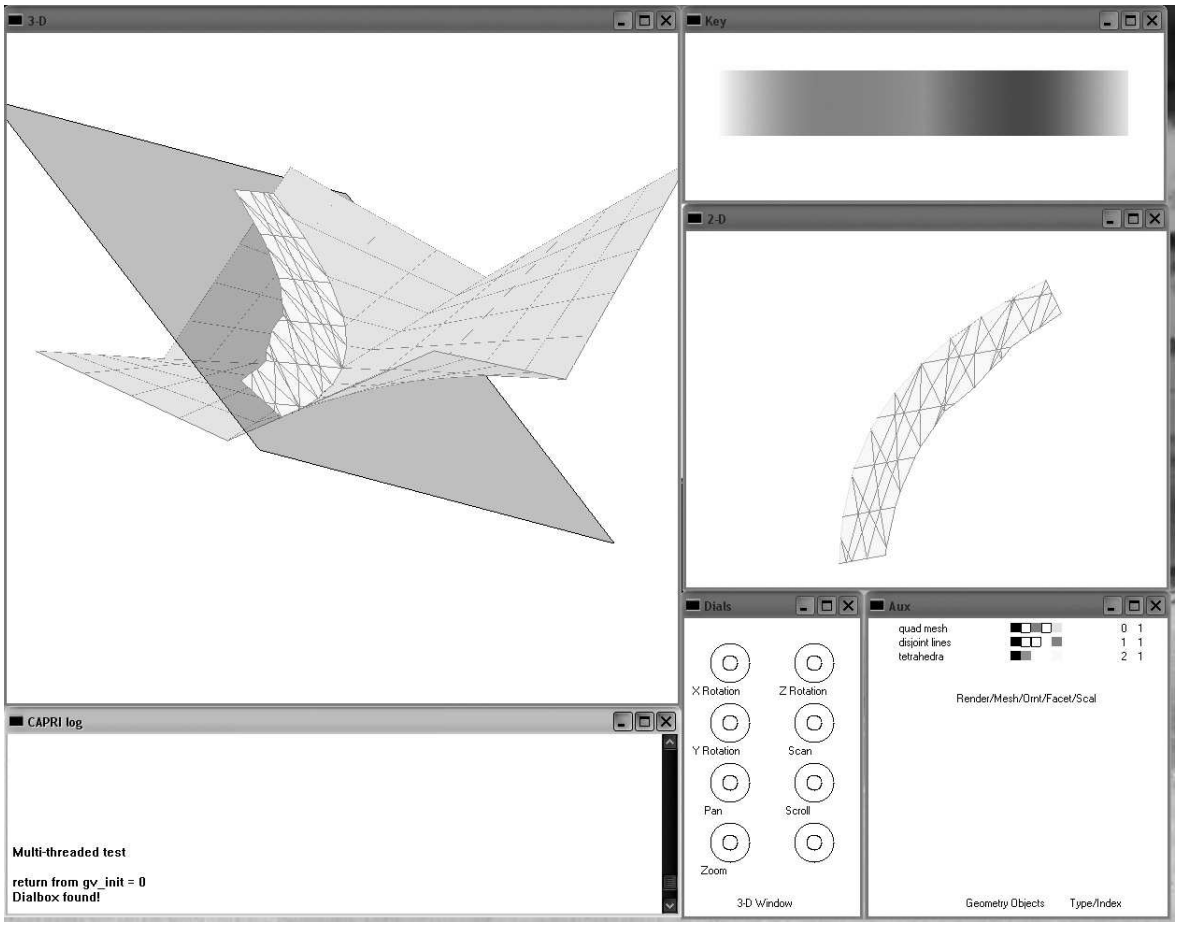


Figure 5-1: Example of the GV interface before any remodeling

decided to implement the interface as an add-on to the existing GV capabilities. This could be later augmented/modified to a Java based interface. However, the interest to make a proof of concept, fast, efficient and portable leads us towards GV.

5.2.2 Architecture of the interface

Based on the decision to use GV as a base for the implementation of the interface, many functions are already existing, as well as windows that will display information. A glance at the existing GV interface (Figure 5-1) shows the existing modules. Even though some existing modules will be reused, modules like the "Aux" module (lower right corner of Figure 5-1), which gives control over the graphical representation of the elements (faces, edges, points) of the model, or the "Key" module, which gives a color scale used in the 3D representation, need to be complemented by new modules that will carry the functions

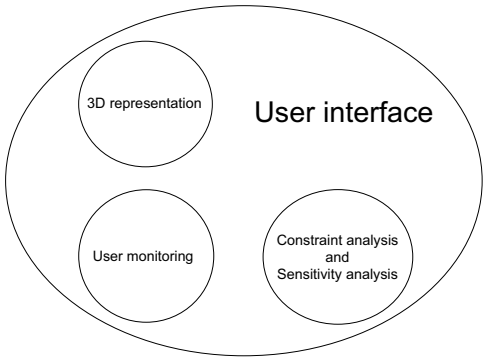


Figure 5-2: Module architecture required in the user interface

needed by the new framework. Figure 5-2 shows briefly the main modules needed to satisfy the needs and requirements of the interface. The "3D representation" module is already existing. However, the 3D display of the CAD model needs to be complemented with a function that highlights specific features, as explained earlier. The "user monitoring" module, mainly based on the existing log window, will display all the information related to the optimization (design variables, constraints or objective function values), but also on the health of the visualization (tessellation errors, visualization errors, etc.). The final module, "Constraint analysis and sensitivity analysis", will be developed from scratch, and should encapsulate all the ideas developed in Chapter 4.

The list below gives a more detailed overview of the function of the three modules

Module 1 : 3D representation

- 3D representation of the object being optimized
- Real-time visualization of the optimization process steps
- Highlighting of one of the active constraints chosen by the designer

Module 2 : User monitoring

- Initial interface between the user and the framework: initial conditions, options, etc.
- Logging of all the errors associated to the visualization (tessellation status, regeneration status, display errors)

- Interface between the optimization and the user: display of the status of the optimization, values of variables and parameters.

Module 3 : Constraint analysis and Sensitivity analysis

- Description of the constraints, description of the design variables participating in each constraint
- Bar chart highlighting the design variables as active
- Physically-based representation of the activeness of the constraints

Having established the architecture, the next step is the actual implementation.

5.2.3 Interface implementation

As described earlier, the user interface was created using the GV interface that accompanies CAPRI. GV is a set of C routines that are then compiled into a library. In order to allow more flexibility if any changes were necessary into the basic GV, only two files are modified, and are given direct access to the rest of the GV codes. These two files, here "gvevent.c" and "gvdraw.c" allow the definition of specific events. A third file, "decvar.h" allows the declaration of all the variables that are used in the implementation of the interface.

User monitoring

The user monitoring module is the first one to be implemented, using the "CAPRI log" window provided by GV. The textual information displayed in that window can be controlled anywhere in the program, and does not need to be compiled in the GV library. The "CAPRI log" window allows the user, when starting up the program, to choose the options of the framework as shown is Figure 5-3. The designer can choose to disable the visualization, can choose between different optimizers (SQP or Simulated Annealing), can decide whether to create a file output that will contain the data of the optimization, and can finally decide to take snapshots of the 3D configuration at each iteration. This final option will allow the user to later create a movie of the optimization history using software such as Movie Maker 2 from Microsoft® [5].

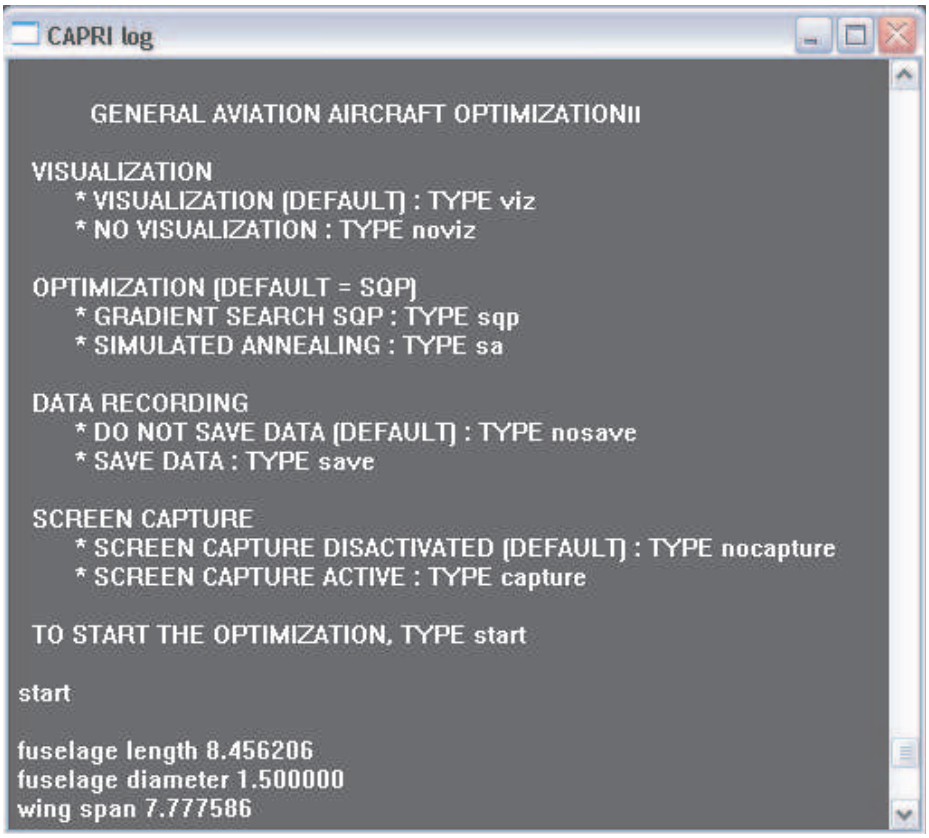


Figure 5-3: User menu

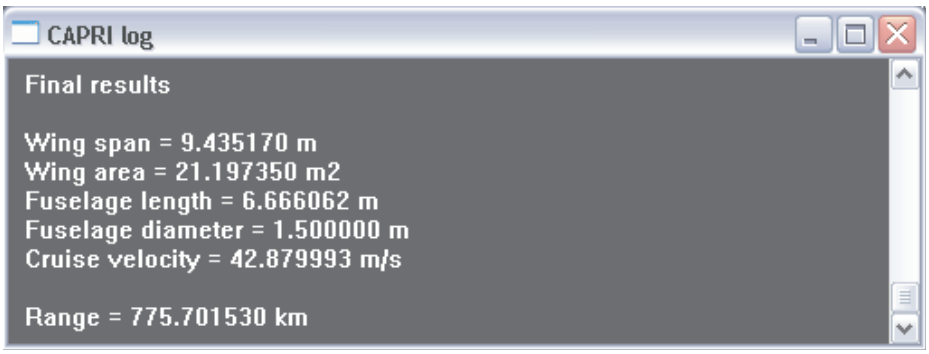


Figure 5-4: Monitoring panel

When all the options are input, the "CAPRI log" window becomes the interface between the user and the framework to report valuable information, status and/or errors. This window allows the constant interaction between the user and the interface. It will textually display the options given to the user as well as the main information of the optimization such as the number of iterations, the value of the different design variables, the value of the objective. The errors that can occur during a run will also be gathered in this monitoring panel. Figure 5-4 shows the display of design variable information in a specific example that will be discussed later.

3D visualization

The 3D visualization module is based on the GV capabilities. Using the geometric data from the CAD model, the 3D visualization module will display, after a tessellation of the new geometry that is made available by CAPRI, the 3D representation of the system. Figure 5-5 shows the 3D representation of a GA aircraft for example. This 3D visualization can be complemented by a 2D view, entirely developed in the early version of GV, which gives the engineer the capability to operate cuts of the model. However, due to space concerns, this functionality is kept hidden in the final version of the interface.

Constraint analysis and Sensitivity analysis

In the case of the constraint analysis and sensitivity analysis module, the events that are implemented are the creation of new windows, as well as the definition of all the events that should occur in the window. In particular the window called "constraints" is created, as shown on Figure 5-6.

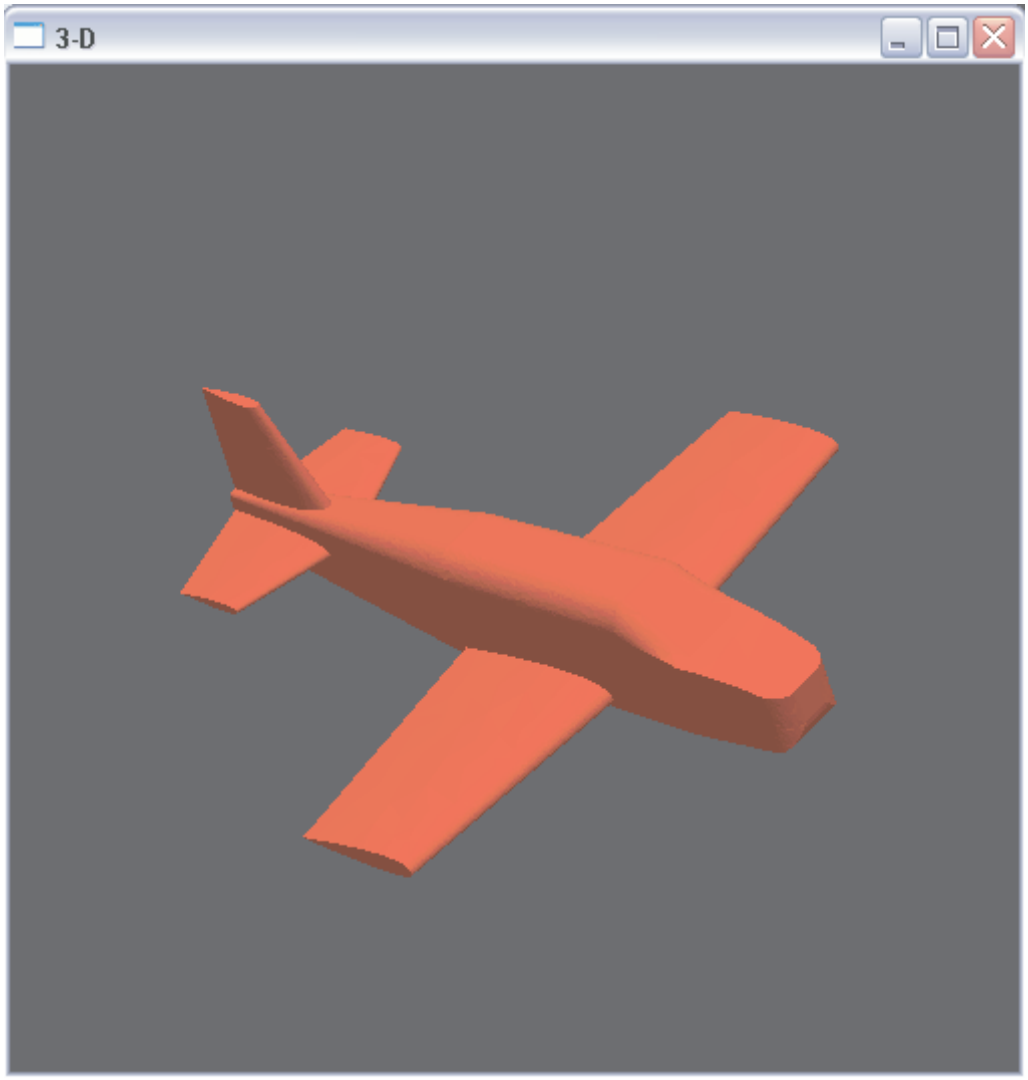


Figure 5-5: 3D visualization module - constraints not displayed

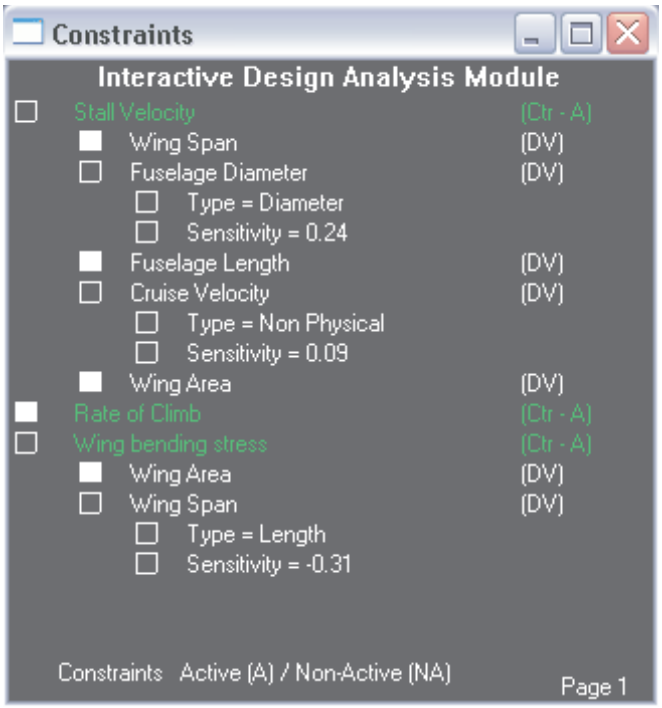


Figure 5-6: Constraint analysis module

A tree displays the first level of information, i.e. the constraints that are being evaluated in the design, and the status of each of them (active or non active). A color code helps the user to have a quick overview of the status of the design. By clicking on the small boxes to expose or not the branches of the tree, the user has the ability to navigate through the tree. By doing so, the user chooses to expose the next levels of information. Namely, the design variables that have a significant participation in the given constraint are displayed. The next and final level displays information related to the design variables (sensitivities in the constraint and type). A link exists between the "constraints" window and the "3D" window. By selecting a constraint, one can then visualize, on the "3D" window, highlighted features that are representative of the features that are impacted by the selected constraint. Figure 5-7 shows the "3D" window in the case of a GA aircraft example. An advanced algorithm allows the color of faces on the 3D model, corresponding to the physical feature(s) to be displayed, to be changed to illustrate the activeness of constraints. The lighter colors represent the active constraint, in that case, rate of climb. Finally, one can note that if the information available cannot fit on one page, the user can access the other page by using the keys "u" and "d".

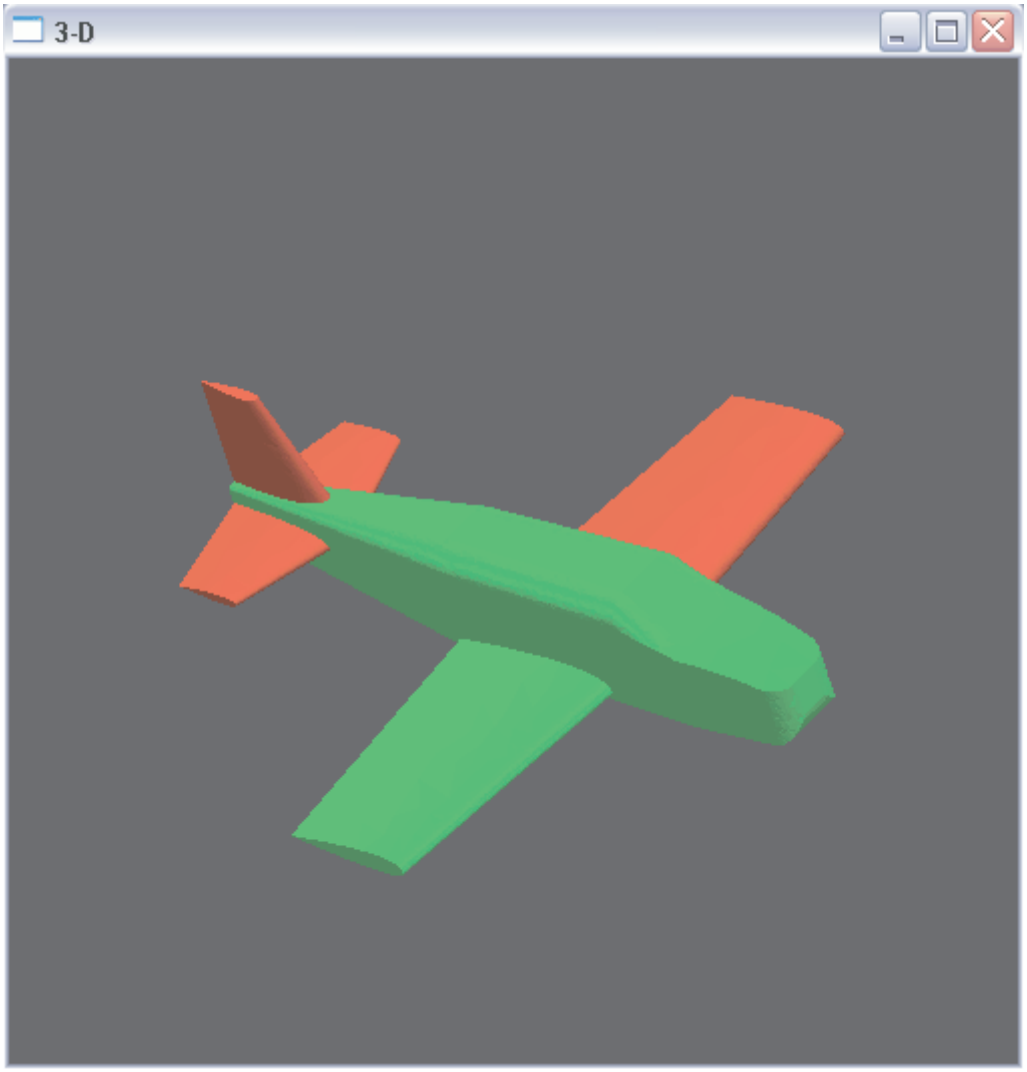


Figure 5-7: 3D visualization module - rate of climb constraint displayed by highlighting right wing and fuselage

The constraint sensitivity analysis window (called "Sensitivities" on Figure 5-8) will visually collect the information for a given constraint. It displays, using bar charts and following a classic approach, the sensitivity of each design variable related to a specific constraint. It will only take into account the main major design variables (if the corresponding sensitivity is above a threshold defined by the user) affecting in a given constraint (and therefore participating to the design at a given point). The number of main major design variables that are displayed can be chosen, depending on the scale and complexity of the problem.

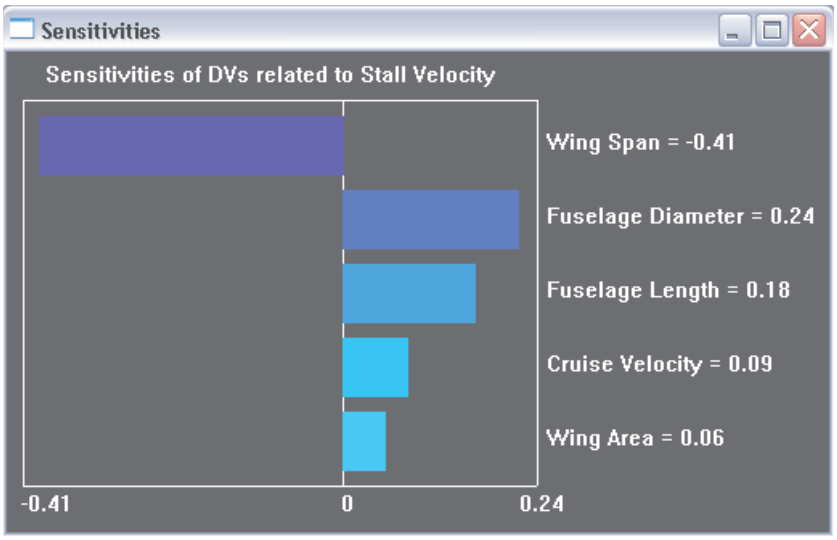


Figure 5-8: Constraint sensitivity analysis module

5.2.4 Interface implementation

All the modules presented above form the global user interface for visualization and constraint analysis. This interface can be seen on Figure 5-9, at a given state, for a given example. We can notice the presence of a last window that was not detailed earlier. the "Dials" window electronically simulates the control panel that designers often used in the past, and gives full control on the 3D model (or 2D model) in term of rotation, scaling, or displacement. However, the use of the interface being shown on Figure 5-9 requires some preliminary work by the user in order to define the way the problem is set up, and establish a link between features on the model and face numbers. This preliminary work is described in Section 5.3.

5.3 User input

In order to prepare the framework to work efficiently, the user needs to do a few preparative operations. These operations, mainly gathered in the file "gvevent.c" consist in declaring the optimization framework and getting information from the CAD model for linking to the DAM.

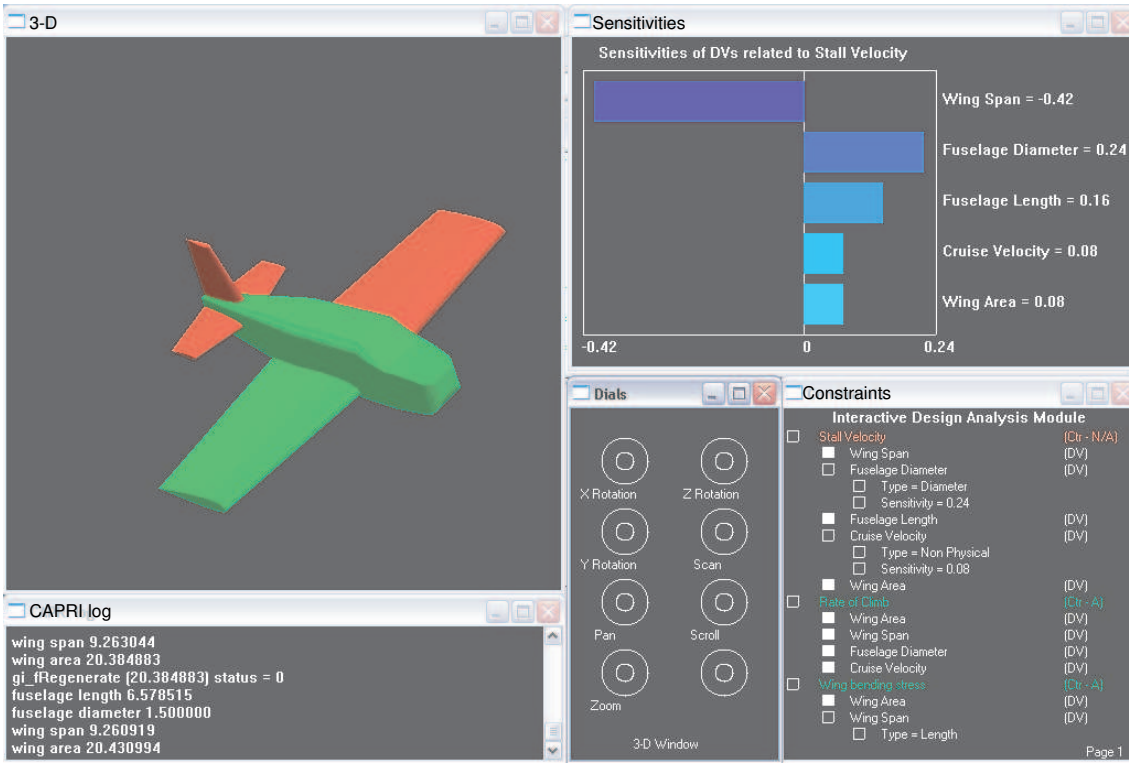


Figure 5-9: Optimization framework

5.3.1 Definition of the optimization framework

The first step is to define, in the program, the optimization framework. This includes defining the design variables, defining the constraints and defining upper and lower bounds for both design variables and constraints. Finally, the objective needs to be defined. When the design variables have been outlined, the designer needs to associate each design variable with a name and a physical feature on the system to be designed. When a design variable cannot be associated to a physical feature, it is assigned the feature "0". Finally, each design variable is associated with a type:

- Non physical (type 0)
- Length (type 1)
- Surface (type 2)
- Diameter (type 3)
- Angle (type 4)

- Ratio (type 5)
- Other (type 6)

As the whole framework has been written in C, these operations will be executed before compilation. Regarding the constraints, the user needs to name all the constraints, in order to display correctly the name during the optimization.

5.3.2 Definition of the physical features

The next step is then to link the features that have been defined earlier with real elements of the 3D model. That will let the framework display the correct physical elements when needed. This is probably the most cumbersome manual operation. The idea is to identify all the faces of the CAD model, and associate the faces with a feature. When the geometry is imported into CAPRI, each face is given a number that will now be used. This needs to be done manually, but could be enhanced or automatized in later versions of the framework. A feature, imbedded into CAPRI and illustrated on Figure 5-10, allows the user to identify the number of the faces by dragging the mouse pointer over the model. This will help during the process.

5.4 Conclusion

The whole framework and interface have now been put in place, and are ready to be used in real applications. As was demonstrated, the framework can successfully display, in real time, the evolution of a CAD model. The new information is transferred from the optimizer to the visualization module via CAPRI. In the meantime, a module allows the analysis of the design, especially at the constraint level. This gives, at the designer's discretion, insight to the design. Indeed, the user can choose to physically display, on the 3D model of the system, the active constraints, and the way they are linked to the physics of the system. The interface that has been created implements all the concepts that were presented and gives the engineer a novel and useful tool to explore the design and the optimization process. In the next chapter, an example will be presented to give an overview of the capabilities of the framework in a real case. The GA aircraft has been selected for its simplicity and the link that can be made between optimization setup and physical representation.

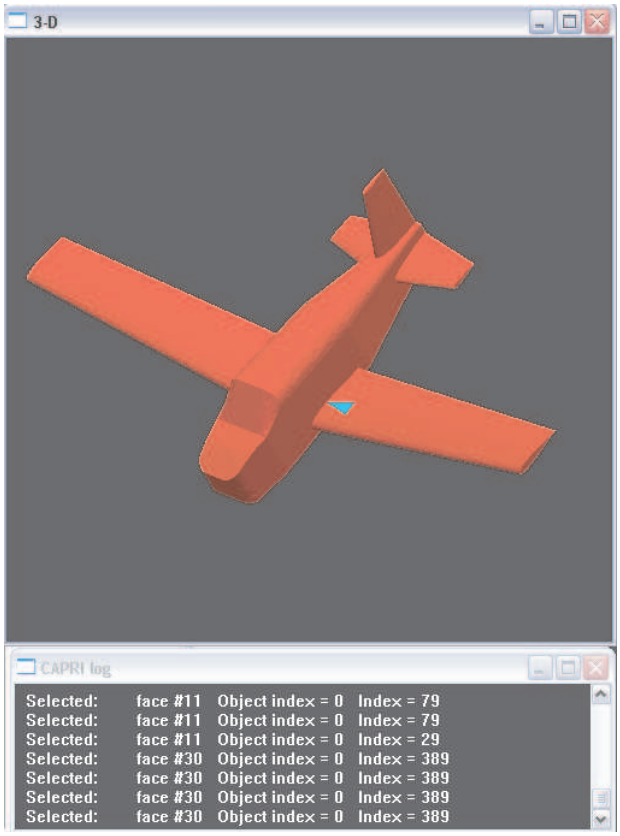


Figure 5-10: Identification of the features

Chapter 6

Application to a design case: the General Aviation (GA) Aircraft

In Chapters 2, 3 and 4, the framework was defined. In Chapter 5, implementation of the interface was discussed. The next step is to apply these to a design case. As noted earlier, the strength of this approach resides in cases that can be linked to geometric parameters, since the whole approach is based on a physical representation of the system and its design constraints. By meeting this criteria, the GA aircraft case is a good candidate. This chapter describes the analytical model and the CAD model that were developed for the design, and recalls the user input required by the framework. It then describes in depth the use of the framework and the interface in the scope of the design. It finally demonstrates the value of the approach.

6.1 GA aircraft model

To fulfil these objectives, a simple General Aviation (GA) aircraft model will be considered in this work. Although simple, the model should have the right trends to effectively conduct an optimization, and apply the visualization and constraint analysis framework with satisfying results. This framework is based on low-fidelity analysis models. The aerodynamics are modelled empirically using DATCOM data [13], the structural analysis is based on a simple beam model and the weights model is based on empirical relations [30, 27, 32]. The models will now be described in depth.

Table 6.1: Summary of the design variables, constraints and objective of the General Aviation aircraft design problem.

Design variables	Wing span
	Wing area
	Fuselage diameter
	Fuselage length
	Cruise velocity
Constraints	Stall velocity
	Rate of climb
	Wing bending stress
	Fuselage diameter
Objective function	Max. Breguet range

6.1.1 Model overview

Table 6.1 summarizes the design variables, constraints and objective function used. In parallel with the five design variables identified above, a set of parameters, fixed during the optimization, is used and characterizes other aspects of the aircraft. Table 6.2 summarizes the parameters used in the model.

Table 6.2: Summary of the design parameters of the General Aviation aircraft design problem.

Parameters	Value	SI
Standard atmosphere density, rho	1.225	[kg/m^3]
Ultimate loading, Nzult	5.7	[−]
Engine weight, Wen	70	[kg]
Number of engines, Nen	1	[−]
Power of each engine, Pen	90000	[W]
Number of passengers, Npass	2	[−]
Passenger weight, Wpass	100.00	[kg]
Payload weight, Wpay	100	[kg]
Propulsive efficiency, eta	0.85	[−]
Specific fuel consumption, SFC	5E-06	[m^{-1}]
Max lift coefficient, Clmax	1.7	[−]
Oswald factor, e	0.8	[−]
Design altitude, h	0	[m]

In addition to these parameters, assumptions on the geometry are made, as illustrated in Figure 6-1. For example, the wing is assumed to be of taper ratio equal to 1, with no break. The same assumptions are made for the vertical and horizontal stabilizer. In addition, the fuselage is assumed to be cylindrical for the forward section, while the rear section is conical. Finally, the location of the wing, stabilizers, as well as the size of the stabilizers are calculated empirically, in relation to the design variables, using the relations

shown in Figure 6-1.

On the analysis side, the model that is used for the GA aircraft design is a preliminary design model, which includes simple weight models, analytical aerodynamic model and simple rate of climb relations. The range evaluation will be based on the Breguet range equation. The main goal of this model is to give a fairly accurate estimate of the design of a GA aircraft, while still being fast for computation time purposes. When the framework and the interface is completed, a more accurate model can be developed and could involve higher fidelity models. The N^2 diagram, shown in Figure 6-2, represents the way the different modules interact with each other. It should be noted that feedback loops in this diagram (as well as internal iterations) prevent the designer from coming up with an analytical solution for the range evaluation. The next section will detail the different modules that are involved in the GA model.

6.1.2 Detailed overview of the analysis modules

Aerodynamics Module

The aerodynamics module has a detailed model of the various drag contributions. The program evaluates, for each part of the aircraft, the friction drag, the induced drag, and the interference drag, based on the geometry of the airplane.

Wing drag The first step is to evaluate the friction drag of the wing, taking into account the wing area, the aspect ratio, the force applied on the wing and the flight speed. The program decomposes the wing into a series of airfoils, uniformly spread along the span according to the user definition. Using a lookup table associated to one or more specific airfoils (see Figure 6-3 below), the program reads the value of $c_{L_{min}}$ (minimum 2D lift coefficient), $c_{D_{min}}$ (minimum 2D drag coefficient), c_{L_o} (2D lift coefficient a 0 degree angle), c_{D_o} (2D drag coefficient a 0 degree angle), $c_{L_{max}}$ (maximum 2D lift coefficient) and $c_{D_{max}}$ (maximum 2D drag coefficient) of the airfoil for two Reynolds numbers (one preceding and one following the actual value of the Reynold's number at which the design is executed) and two Mach numbers (one preceding and one following the actual value of the Mach number at which the design is executed). It then interpolates between these values for the actual Reynolds and Mach number.

From these interpolations, at a given c_L (2D lift coefficient) (associated to the appro-

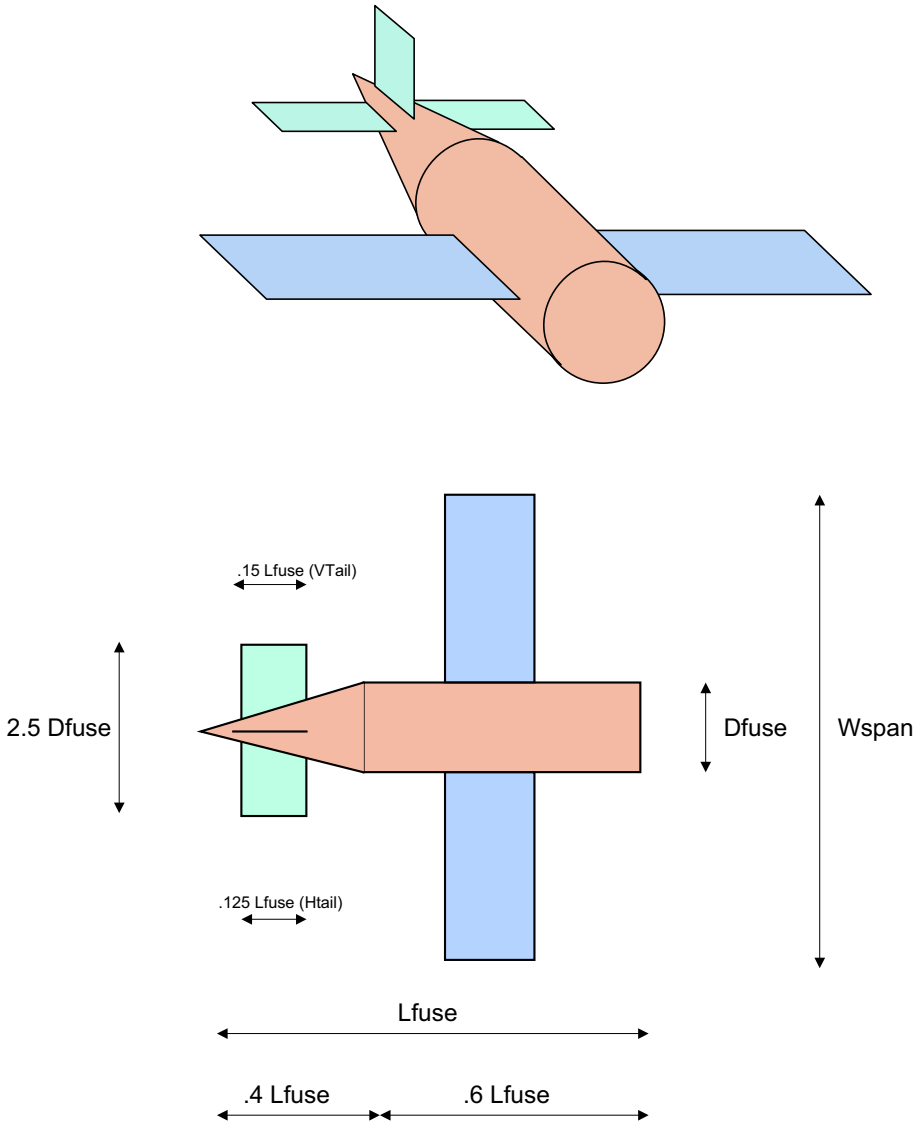


Figure 6-1: Geometry assumptions used in the GA model

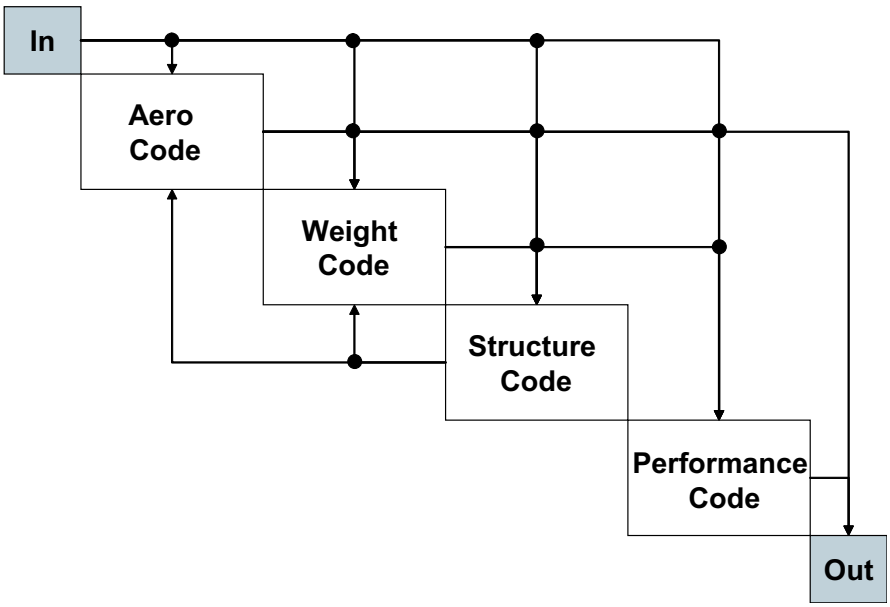


Figure 6-2: N2 diagram for the General Aviation Aircraft model

priate airfoil), the program evaluates the c_D (2D Drag coefficient) of the airfoil, using the data read in the lookup table and using a panel approach. More information on the method can be found in Anderson, Abbot and Hoerner [7, 6, 18]. Each c_D is then multiplied by the corresponding surface area ($dy \times c$ where c represents the chord and dy the width in between two airfoils spread along the wing). Repeating the operation for all the airfoils along the wing, a global friction drag area coefficient SCd_f is obtained as follows .

$$SCd_f = \sum_{k=1}^{na} c_D \times dy \times c$$

with na the number of airfoils along the wing. The lift distribution is assumed to be elliptical for the problem. The wing-induced drag area coefficient SCd_i is simply computed using the following equation:

$$SCd_i = \frac{S_{wing}.CL^2}{\pi.e.AR}$$

where S_{wing} is the wing area, CL the design lift coefficient, e the Oswald factor and AR the aspect ratio. As a reminder, $AR = \frac{W_{span}^2}{S_{wing}}$ with W_{span} the wing span. Finally, the wing induced drag area coefficient SCd_{in} is evaluated at 3% of the induced drag area

number of airfoils		2	
NACA2412			
correction		0.1	
Number of Mach		2	
Number of Re		2	
CLMIN		2.00E+06	6.00E+06
	0.05	-1.077753	-1.013492
	0.15	-1.012251	-1.040213
CDMIN			
	0.05	0.014862	0.009092
	0.15	0.012542	0.009018
CL0			
	0.05	0.387783	0.334919
	0.15	0.39971	0.35007
CD0			
	0.05	0.005266	0.005224
	0.15	0.005373	0.005241
CLMAX			
	0.05	1.721101	1.710477
	0.15	1.470753	2.058985
CDMAX			
	0.05	0.023462	0.015989
	0.15	0.017088	0.022409

NACA12	12% thickness	airfoil	data	
correction	0.1			
Number of Mach	4			
Number of Re	4			
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	-0.73	-0.90	-1.00	-1.55
0.2	-0.73	-0.80	-1.00	-1.55
0.4	-0.73	-0.80	-1.00	-1.35
0.55	-0.73	-0.70	-1.00	-1.15
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	0.0170	0.0174	0.0136	0.0165
0.2	0.0170	0.0150	0.0140	0.0175
0.4	0.0200	0.0165	0.0150	0.0150
0.55	0.0220	0.0152	0.0165	0.0125
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	0.00	0.00	0.00	0.00
0.2	0.00	0.00	0.00	0.00
0.4	0.00	0.00	0.00	0.00
0.55	0.00	0.00	0.00	0.00
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	0.0160	0.0111	0.0057	0.0046
0.2	0.0162	0.0112	0.0058	0.0046
0.4	0.0175	0.0113	0.0059	0.0048
0.55	0.0190	0.0116	0.0063	0.0050
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	0.74	0.90	1.00	1.55
0.2	0.74	0.80	1.00	1.55
0.4	0.74	0.80	1.00	1.35
0.55	0.74	0.70	1.00	1.15
	1.50E+05	3.00E+05	1.00E+06	4.00E+06
0	0.0170	0.0174	0.0136	0.0165
0.2	0.0170	0.0150	0.0140	0.0175
0.4	0.0200	0.0165	0.0150	0.0150
0.55	0.0220	0.0152	0.0165	0.0125

Figure 6-3: An example of the look-up table used for the various drag coefficient

coefficient[10]. The wing drag area coefficient SCd_{wing} can then be calculated as:

$$SCd_{wing} = SCd_f + SCd_i + SCd_{in}$$

Horizontal and vertical stabilizers drag The same calculations are carried out for the horizontal and vertical tail, using a second lookup table. Friction drag, induced drag and interference drag are evaluated the same way they were evaluated for the wing (4% of the induced drag is considered here for the interference drag). The stabilizers drag area coefficient SCd_{stab} can therefore be computed.

Fuselage drag Once again, the drag evaluation process consists in evaluating the friction drag and the induced drag. The friction drag takes into account the fineness of the fuselage (length over diameter). The process used for the fuselage friction drag area SCd_f estimation is developed in the DATCOM [13]:

$$\begin{aligned} Cd_{friction} \times fric\_factor \quad = \quad & fric\_a2 \times \ln(Re)^2 + \\ & fric\_a1 \times \ln(Re) + \\ & fric\_a0 + \\ & (fric\_b1 \times \ln(Re) + fric\_b0) \times M \end{aligned}$$

where M is the mach number, and:

$$fric\_a2 = 5.9712$$

$$fric\_a1 = -244.16$$

$$fric\_a0 = 2686.27$$

$$fric\_b1 = 4.3797$$

$$fric\_b0 = -91.082$$

$$fric\_factor = 100000$$

Finally:

$$Scd_f = \frac{Cd_{friction}}{fric\_factor}\left(1 + \frac{60}{fineness^3} + 0.0025.fineness\right) * Swetfus$$

where $Swetfus$ is the wetted area of the fuselage. The induced drag area coefficient SCd_i of the fuselage is then computed as:

$$Scd_i = CSfus.\alpha^2 + 0.6.\alpha^4.Fineness^2$$

where $CSfus$ is the fuselage cross-section area and α the angle of attack at the design point, evaluated by calculating the wing forces for longitudinal equilibrium assuming no pitching moment due to thrust to avoid coupling during cruise. The fuselage drag area coefficient SCd_{fuse} can finally be calculated as:

$$SCd_{fuse} = SCd_f + SCd_i$$

In conclusion, adding up all the drag areas coefficient SCd, one can easily calculate the lift to drag ratio:

$$\frac{L}{D} = \frac{S_{Wing}.CL_{flight}}{SCd_{wing} + SCd_{stab} + SCd_{fuse}}$$

Structural module

A simple structural module is used to evaluate the maximum bending stresses at the root of the wing. Figure 6-4 shows the geometry assumptions made for the model. This module assumes an elliptical distribution of lift on the wing, and outputs the bending moment at the root as well as the maximum bending stress. This is done by integrating the lift distribution over the span of the wing and applying an equivalent lumped force at the center of gravity of this distribution, located by l_1. This results in a bending moment due to the lift. Added to this bending moment is the moment due to the weight of the wing, and the weight of fuel of the wing. The weight of fuel and the weight of the wing are assumed to be applied at half the span of the wing (located by l_2).

$$l_1 = \frac{4}{3\pi}\left(\frac{W_{span}}{2} - \frac{D_{fuse}}{2}\right)$$

$$l_2 = \frac{1}{2}\left(\frac{D_{fuse}}{2} + \frac{W_{span}}{2}\right)$$

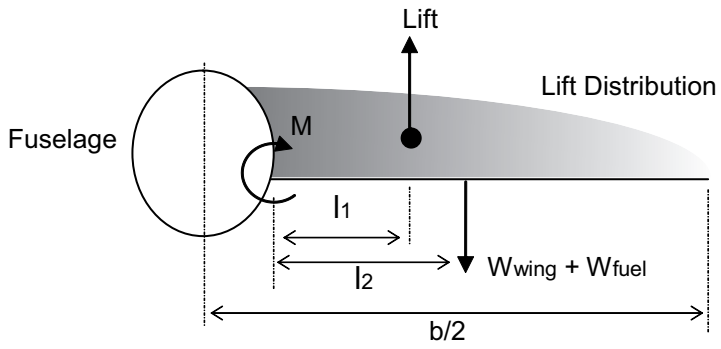


Figure 6-4: Elliptical lift distribution on one wing with lumping model

Therefore, the bending moment M_{root} at the root can be evaluated using:

$$M_{root} = L.l_1 - W_{wing}.l_2 - W_{fuel}.l_2$$

where L is the total lift of the aircraft, W_{wing} the weight of the wing and W_{fuel} the weight of fuel carried in the wing.

The wing spar is considered to be a carbon fiber hollow tube of constant circular cross-section, with a fixed thickness. The external radius is r_o and the internal radius r_i. The second moment of area I of such a section is given by:

$$I = \frac{\pi \left(r_o^4 - r_i^4\right)}{4}$$

Finally, the maximum bending stress σ_{max} is determined using:

$$\sigma_{max} = \frac{M_{root}.\left(r_o/2\right)}{I}$$

The outer radius of the spar is assumed to be equal toone quarter of the average thickness of the wing. For a 10% thick airfoil with a 1m chord, this would yield a spar diameter of 5cm. For obvious structural reasons, the value of σ_{max} is bounded by the yield strength σ_E (corrected with an appropriate safety factor of 1.5) of the carbon fiber material. ($\sigma_E = 2.0GPa$).

Weight evaluation module

The weight evaluation module calculates an empty weight and a total weight (GTOW) based on the GA aircraft weight model given by Raymer [30]. By using the geometrical simplifications shown by Figure 6-1, the gross take-off weight $GTOW$ is calculated using the equations and parameters shown in Figures B-1, B-2 and B-3 in Appendix B. In addition, the fuel weight is approximated by idealizing the fuel tank (the wing) as a cylinder of elliptical section. The volume of such a cylinder is given by:

$$Vol = \frac{\pi.c.b/2}{4}$$

where c is the chord of the wing and b its span. Because of the structure, the control mechanism, and other equipment, it was assumed that only one quarter of the wing volume would be filled with fuel. The weight evaluation module then iterates on the weight calculation until it converges to a stable value (some weight calculations are based on previous weight data). Notice that the weight evaluation module gives a detailed estimation of the different component weights.

Performance evaluation module

Most of the models presented in this section were adapted from Raymer, Ojha and Roskam [30, 27, 32]. The performance evaluation module calculates first the fuel available for cruise, assuming a safety margin of 15%. It then calculates the range (MaxRange) using the Breguet Range Equation:

$$R = \frac{V_{Cruise}.L/D.\eta}{g.SFC} log(\frac{W_{Total}}{W_{Total} - W_{FuelCruise}}) \tag{6.1}$$

where V_{Cruise} is the cruise velocity, L/D the lift over drag ratio, η the propulsive efficiency, SFC the Specific Fuel Consumption, W_{Total} the Total Gross Take-Off Weight (GTOW) and $W_{FuelCruise}$ the weight of fuel burnt during the cruise.
The performance evaluation module also evaluates the stall speed (Vstall) at sea level at GTOW and given CLmax.

$$V_{stall} = \sqrt{\frac{2.g.W_{Total}}{\rho.S_{wing}.Cl_{max}}} \tag{6.2}$$

Rate of Climb module

The model that was developed is based on a simple aero model.
We first need to calculate a cruise lift coefficient:

$$C_L = \frac{W_{Total}.g}{\frac{1}{2}\rho.V_{Cruise}^2.S_{Wing}}$$

The drag coefficient C_D has been computed earlier. As a result, the force created by the drag is calculated as follow:

$$D = \frac{1}{2}\rho.V_{Cruise}.S_{Wing}.C_D$$

Moreover, assuming an engine power P_{en}, and a propulsive efficiency ρ_{prop}, the available thrust is the following:

$$T = \frac{P_{en}\rho_{prop}}{V_{Cruise}}$$

The rate of climb ${}^{r}/_{c}$ can finally be calculated as the amount of power available, once the airplane is balanced on level flight.

$${}^{r}/_{c} = \frac{(T-D).V_{Cruise}}{W_{Total}g}$$

Note that one limitation needs to be taken into account: ${}^{r}/_{c} = 0$ if $C_L \geq C_{L_{Max}}$

6.1.3 Pro/Engineer (Pro/E) CAD model

In order to carry out the visualization, a CAD model has to be implemented. The CAD model, designed using Pro/E, is required to be easily linked to the analysis model. As expressed in Section 6.1.1, the five design variables chosen to describe the GA aircraft model are the wing span, the wing area, the fuselage length, the fuselage diameter, and the cruise velocity. The first four design variables will be used as parameters in Pro/E and will drive the CAD model. These four parameters will then be linked to the dimensions of the CAD model by relations embedded in the CAD model. The Pro/E model created is of high complexity, but can easily be modified into a medium-high complexity CAD model, by suppressing superfluous features, such as the propeller. Figure 6-5 shows the CAD model that was created and Figure 6-6 displays the feature tree associated with the CAD model. It can be seen that the CAD model rendering does not match the simplicity

of the geometry assumptions. The optimization model itself is simple (five design variables, three constraints and a single objective). The CAD model, by the way it has been built, presents a geometry that resembles current GA aircraft, but also offers freedom to the designer. Many parameters (including wing position, taper ratio, break location, airfoil type, stabilizer shape, fuselage shape) are encapsulated in the CAD model but given zero value. That leaves the capability to add design variables in the optimization definition for example, without being obliged to build a new CAD model

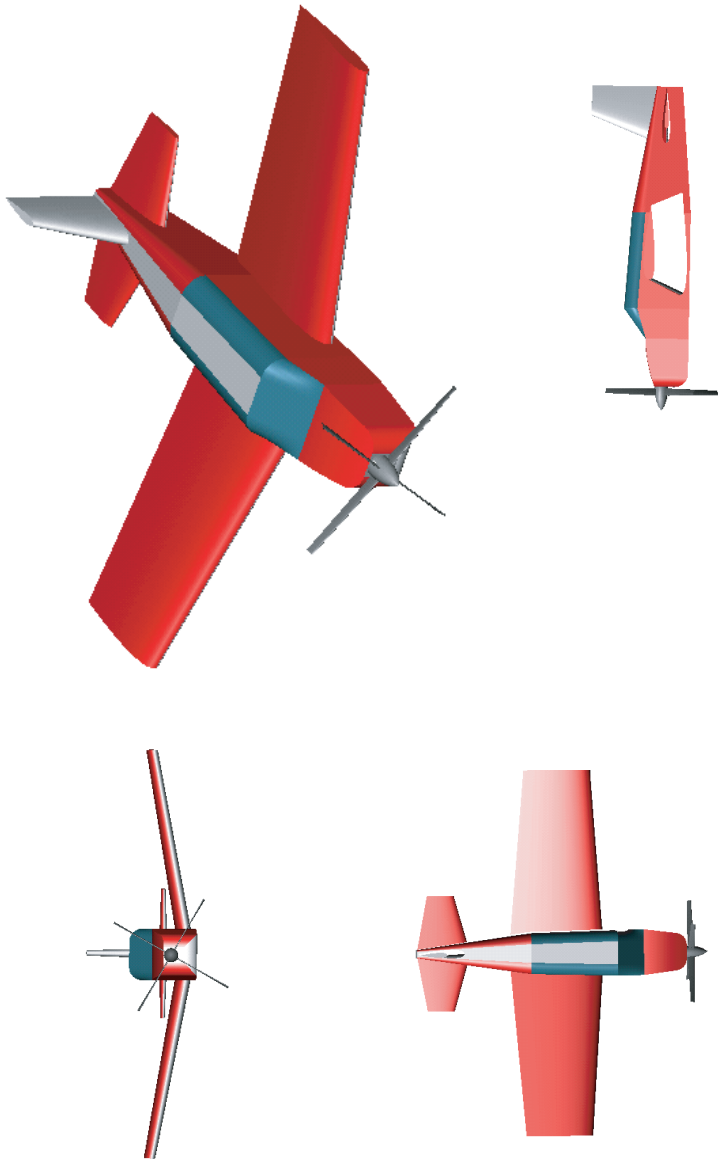


Figure 6-5: Different views of the GA aircraft CAD model created using Pro/E

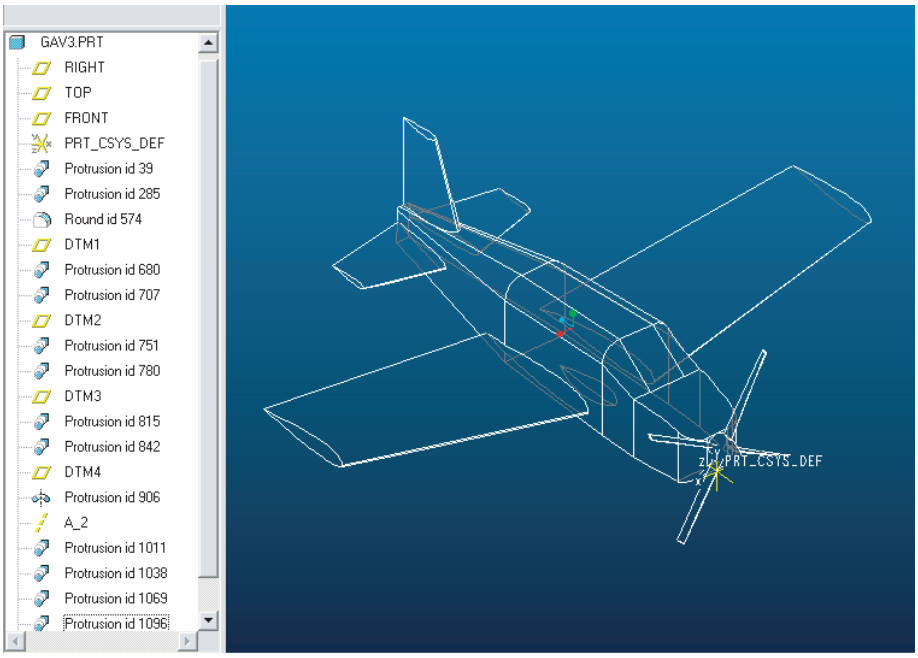


Figure 6-6: Snapshot of the Pro/E window, showing a wireframe representation of the CAD model, as well as the feature tree

6.1.4 Summary

The different underlying models have been discussed, and a CAD model, the main element of the visualization framework, has been presented. These represent the core of the overall approach. The complete GA aircraft model can now be used in any type of optimization, but will be mainly used for testing and validating the framework that has been presented earlier.

6.2 Design space and classical optimization

6.2.1 Design space exploration

Using existing data for GA aircraft [20], values for the constraints were determined. Table 6.3 summarizes the values used for the constraints. Note that a fourth constraint, a bound on the minimum fuselage diameter, was used.

In order to get some insights about the design space, investigation was made into the effect of the design variables on the value of the objective function, and into how the constraints would bound the design. Figure 6-7 shows a two-dimensional slice of the design

Table 6.3: Summary of the constraint value and type for the General Aviation aircraft design problem.

Name	Value	Units	Type
Stall velocity	27	$m.s^{-1}$	max
Rate of climb	2	$m.s^{-1}$	min
Wing bending stress	1500	MPa	max
Fuselage diameter	1.5	m	min

space. It maps the objective function value (best range in the upper right corner) within the permissible range of two design variables, namely wing span and wing area. The three other design variables are assumed constant. Fuselage diameter is set to 1.5 m, fuselage length to 6.6 m and cruise velocity to 43 $m.s^{-1}$ It can give a first insight on the narrow shape of the design space, bounded by three constraints. To satisfy all the constraints, the design point must be located above the rate of climb constraint boundary, but below the bending stress boundary. Finally, the stall velocity constraint diminishes further the design space by limiting the feasible designs to the right part of the previously delimited design space. This show clearly the direction that the optimization might take, trying to reach the upper corner of the design space. However, this representation of the design space does not include the effects of the other design variables, in particular the cruise velocity that expands the feasible region as it is decreased.

6.2.2 GA aircraft optimization

With some understanding of the design space, optimization can now be applied to the design problem. The CFSQP optimization algorithm was used [23] here, but the use of heuristic algorithms (genetic algorithm or simulated annealing) is possible. The initial design point is feasible and was based on Cessna-172 data [20]. Both the initial design point and the resulting optimal solution data are shown in Table 6.4.

Using the contour graph of the design space slice, it is possible to plot the trajectory of the optimization. Figure 6-8 describes the behavior of the optimization for the eight most significant iterations. Since an SQP algorithm is used, the trajectory, identified by the arrow on Figure 6-8, stays within the feasible design space. It can be seen that both wing area and wing span are increased until the rate of climb and bending stress constraints become active. However, notice again that this graph supposes constant cruise velocity and

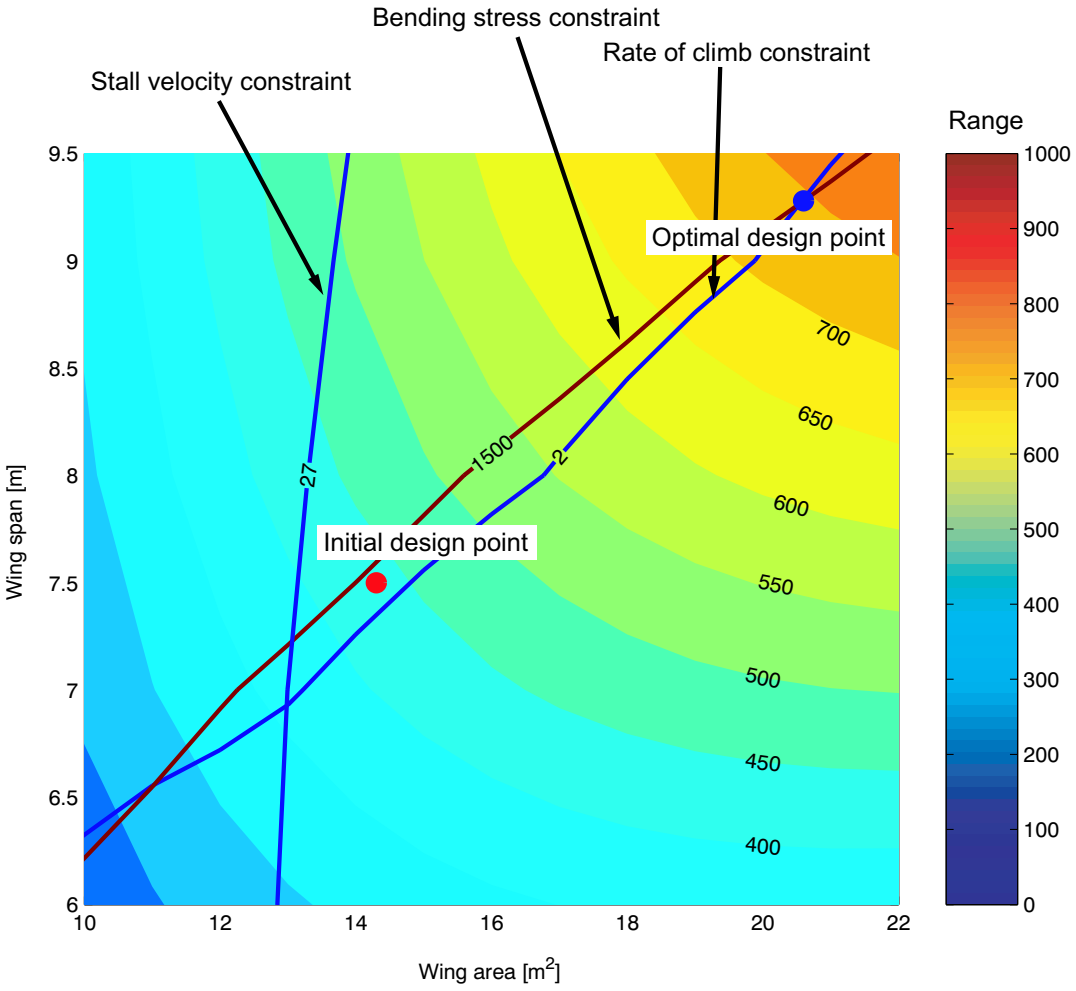


Figure 6-7: Design space representation, for given cruise velocity and fuselage shape

Table 6.4: Summary of the results obtained for the initial point and the optimal solution for the General Aviation aircraft design problem.

Type	Name	Init	Opt	Units
Design variables	Wing span	7.5	9.4	m
	Wing area	14.3	21.2	m^2
	Fuselage diameter	1.5	1.5	m
	Fuselage length	8.2	6.6	m
	Cruise velocity	40.1	42.9	$m.s^{-1}$
Constraints	Stall velocity	26.9	23.8	$m.s^{-1}$
	Rate of climb	2.15	2	$m.s^{-1}$
	Wing bending stress	1482	1500	MPa
Objective	Breguet range	373	775	km

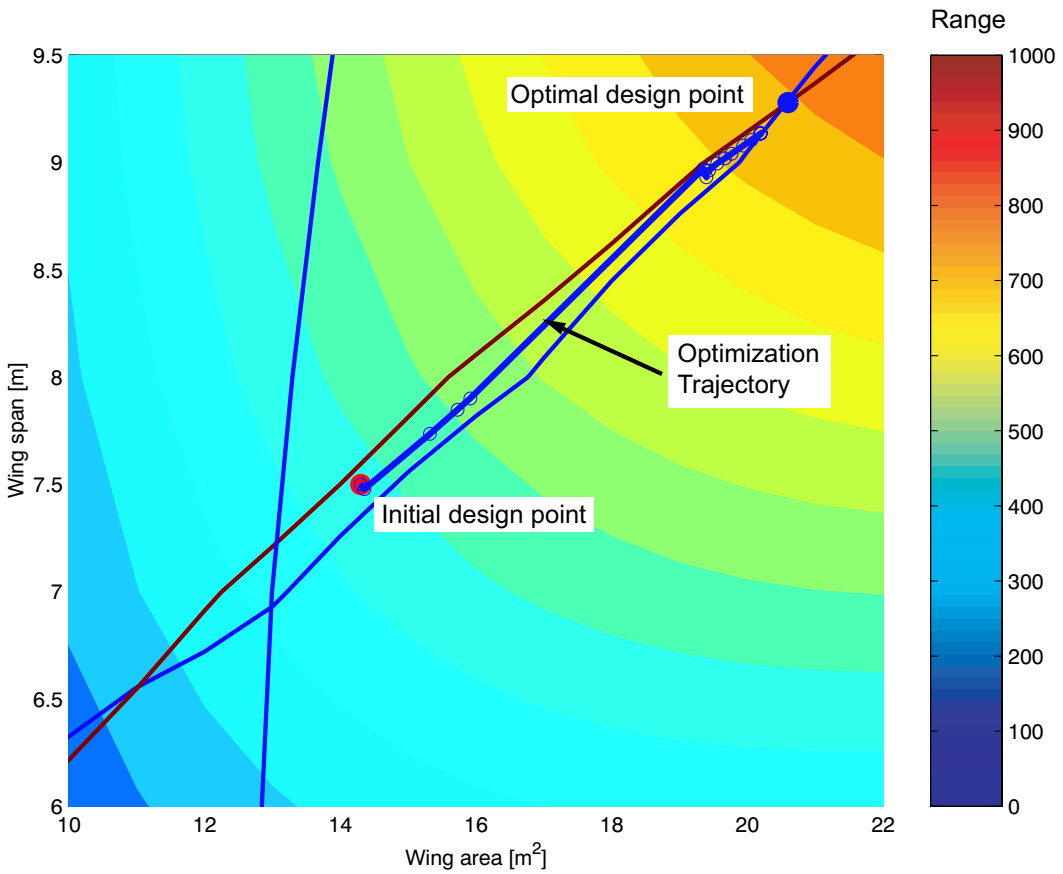


Figure 6-8: Trajectory of the optimization inside the feasible design space

fuselage dimensions, which is not true in the optimization. The narrowness of the design space makes the optimization very quick (8 iterations to get an acceptable final solution).

It is also very instructive to take a closer look at the optimization. Figures 6-9, 6-10, 6-11 show the evolution of the five design variables throughout the iterations, while Figure 6-12 illustrates the improvement in range during the optimization. Finally, Figures 6-13 and 6-14 show the values of the constraints. Figure 6-15 shows a radar plot of the data for three iterations. This an alternative means for displaying optimization information that tries to give some insight to design tradeoffs. Notice that the values of the design variables have been normalized using the value at the initial design point.

While the radar plot in Figure 6-15 is an effective way to visualize a limited set of optimization data, it is easy to see how a large number of design variables or large number of iterations would make this technique inappropriate. Moreover, the radar plot does not tie optimization results directly to the physical system being designed. The next step is

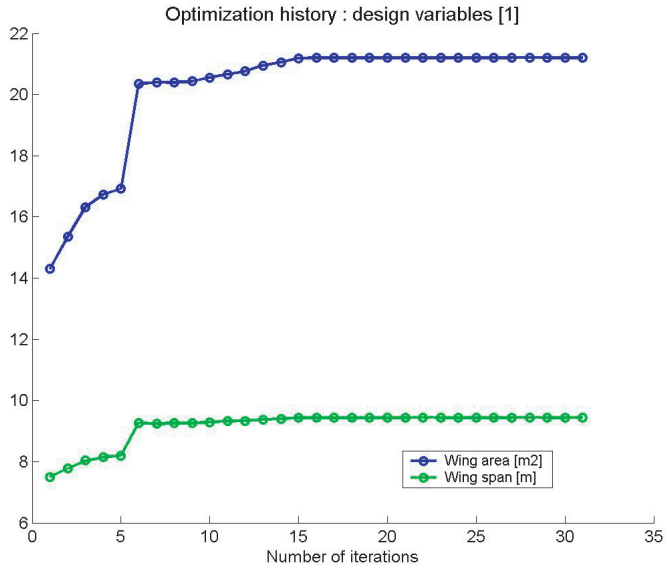


Figure 6-9: Design variables (wing span and wing area) evolution during the optimization

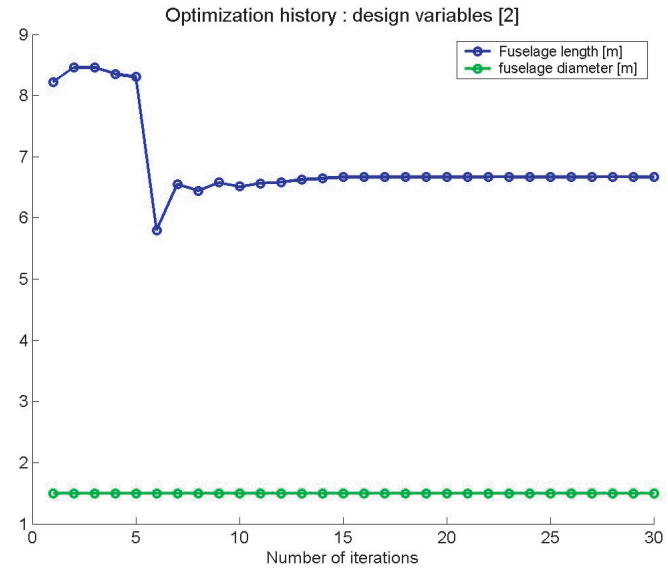


Figure 6-10: Design variables (fuselage diameter and fuselage length) evolution during the optimization

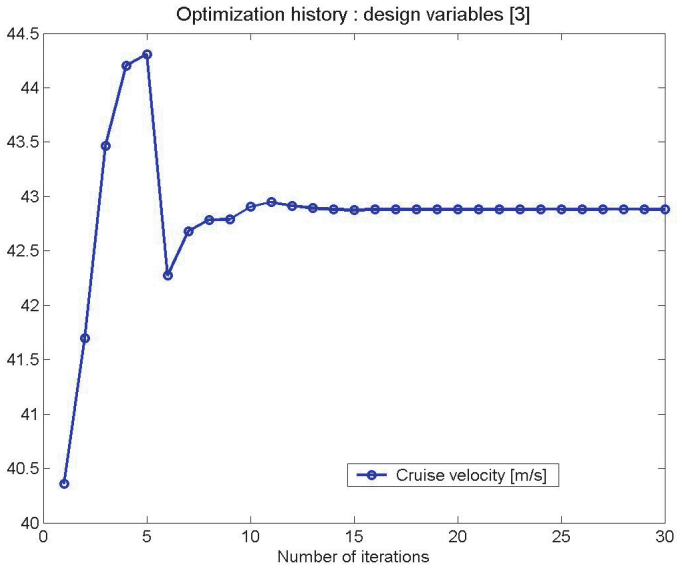


Figure 6-11: Design variables (cruise velocity) evolution during the optimization

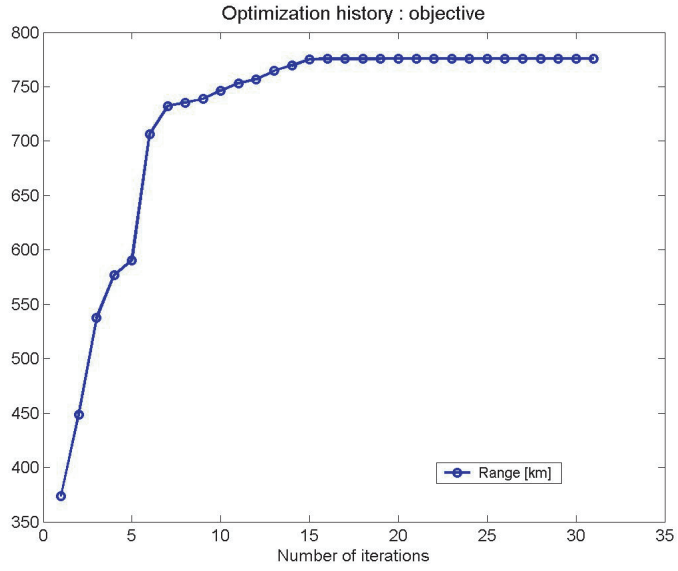


Figure 6-12: Objective (range) evolution during the optimization

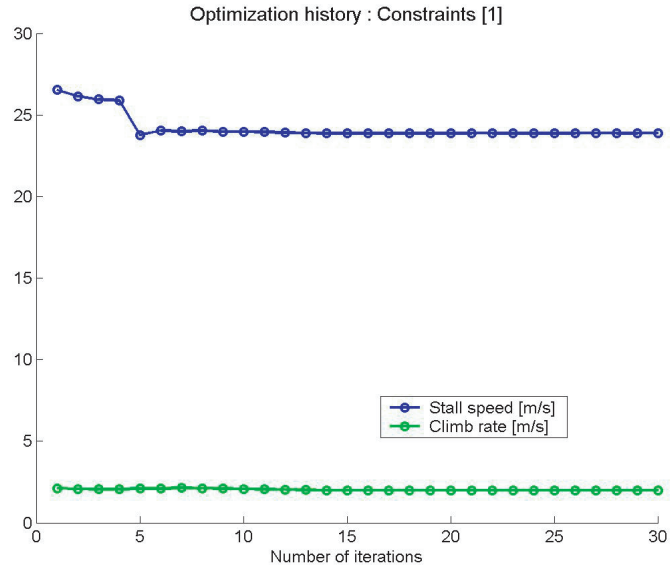


Figure 6-13: Constraints (rate of climb and cruise velocity) evolution during the optimization

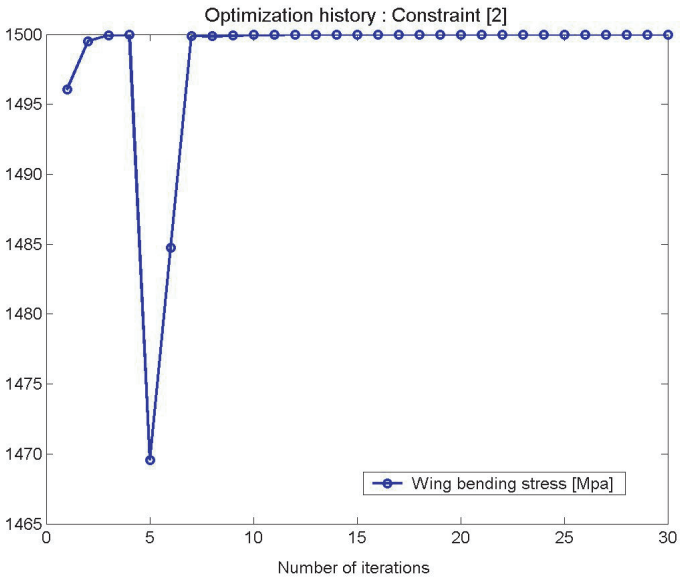


Figure 6-14: Constraints (Wing bending stress) evolution during the optimization

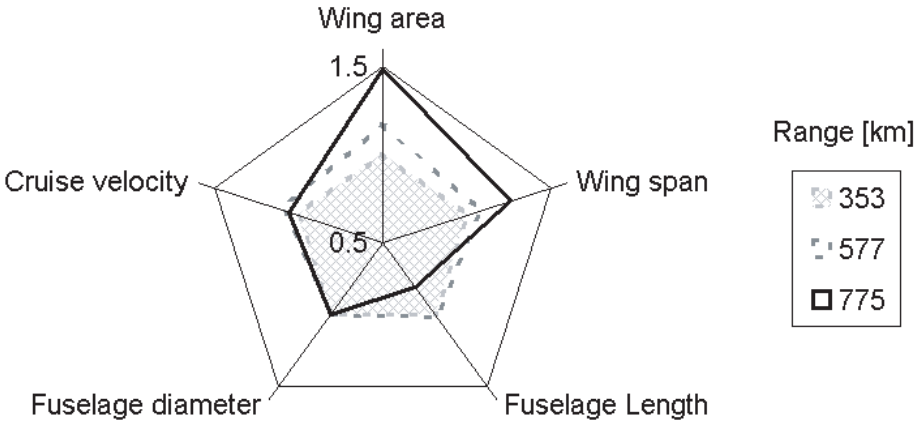


Figure 6-15: Radar plot showing three iterations (First, fifth and final steps of the optimization)

to repeat this optimization using the visualization framework that was developed in this research. Investigation will be made into how valuable the information brought by the visualization and constraint analysis framework is to the engineer, and how it can help the designer understand the behavior of the optimization, the trade-offs that are being made, and the constraint influences on the design.

6.3 Optimization and constraint analysis visualization

The visualization interface, developed to link the OVM and the AOM, will now be used in the GA aircraft model. However, a few manipulations, as described in Section 5.3, are necessary in order to ready both the model and the interface.

6.3.1 User input

First, it is necessary to define the different elements of the model (design variables and constraints) and their relation to the physical features of the model. In the case of the GA aircraft, the design variable "wing span" will be associated with the feature "right wing" that will be assigned a number "1". The feature "fuselage diameter" will be assigned to the feature "fuselage", that will be assigned the number "2". All the design variables that cannot be linked to a physical feature on the system will be associated to a feature numbered

Table 6.5: List of the design variables and the associated features in the case of the GA aircraft

Design variable name	Type	Feature name	Number
Wing area	2	Right wing	2
Wing span	1	Right wing	2
Fuselage diameter	3	Fuselage	1
Fuselage length	1	Fuselage	1
Cruise velocity	0	N/A	0

Table 6.6: List of the features and the related faces on the CAD model

Feature Name	Number	Face number
Right wing	2	26 to 28
Fuselage	1	1 to 25
Left wing	3	29 to 31
Horizontal and vertical stabilizer	4	32 to 41

"0". Table 6.5 illustrates this for the GA aircraft example.

Second, each feature that has been created needs to be linked to faces that compose the CAD model. These faces, which can be defined using the GV function described in Section 5.3, will help displaying efficiently the adequate feature. Table 6.6 illustrates the association between a feature and all the faces on the CAD model. At the code level, this is being done as shown in appendix C, on Figures C-1 and C-2. Figure C-1 shows that different inputs are necessary:

- The number of features to be created : "nbr_feat"
- The number of faces per feature i: "nbr_face_feat[i]"
- The face number for each of the feature i: "ass_face_feat[i][j]" (with j varying from one to "nbr_face_feat[i]")

This step is crucial to the process, and must be carefully executed for better results in the visualization. Figure C-2 shows how the design variables and constraint are defined in the program.

6.3.2 Optimization path monitoring

A CAD representation of the aircraft was developed in Pro/Engineer and linked to the optimization framework using CAPRI. The model has been correctly defined in the code

itself. The interface is now ready to be used and tested in the case of the GA aircraft. The first approach to be analyzed is the real time visualization. One use that can be of interest for a designer is the optimization path monitoring. This monitoring is essential to understand the behavior of the optimizer, and detect the reason for choices made during the optimization.

At each iteration, the current design was viewed in real time using the CAD model. Figure 6-16 shows snapshots of the design at the initial and optimum solutions. From superimposing the two designs, it can be seen clearly how the optimizer chose to change the design: the fuselage length has decreased, while the wing span and wing area have increased. These physical changes are instantly evident to the designer using the visualization. It replaces a long list of values, that lack immediate meaning, by a 3D representation of the system being designed. Figure 6-25 show the values of the design variables, objective, and constraints, for the optimization in the case of the GA aircraft. Figures 6-17 to 6-24 illustrate the complete optimization process, using the snapshots that were saved at each iteration. Notice that the diagram representing the evolution of the range was added a posteriori. One can contrast the different modes of information transferal represented by the table in Figure 6-25 and the sequence of plots in Figures 6-17 to 6-24.

An increase in wing span improves the L/D of the aircraft, and thus the range. An increase in wing area allows more fuel to be carried. A decrease in fuselage length results in a decrease in empty weight. Similarly, the fuselage diameter does not increase from its lower bound due to both weight and drag considerations. From Table 6.4 it can also be seen that the cruise velocity was increased; however, this is a design variable that is not visualized on the CAD model. Since the GTOW tends to decrease during the optimization, the stall velocity limit rapidly becomes inactive. Figure 6-26 summarizes the different tradeoffs and the constraints that need to be considered, when looking at the Breguet range equation.

While the optimization progress presents valuable information to the designer, in order to fully understand the tradeoffs, one must also consider the impact of the constraints as discussed next.

6.3.3 Constraint analysis and visualization

In order to illustrate the visual constraint analysis in the case of the GA aircraft design, the constraint limiting the rate of climb was analyzed and is shown in Figure 6-27. This

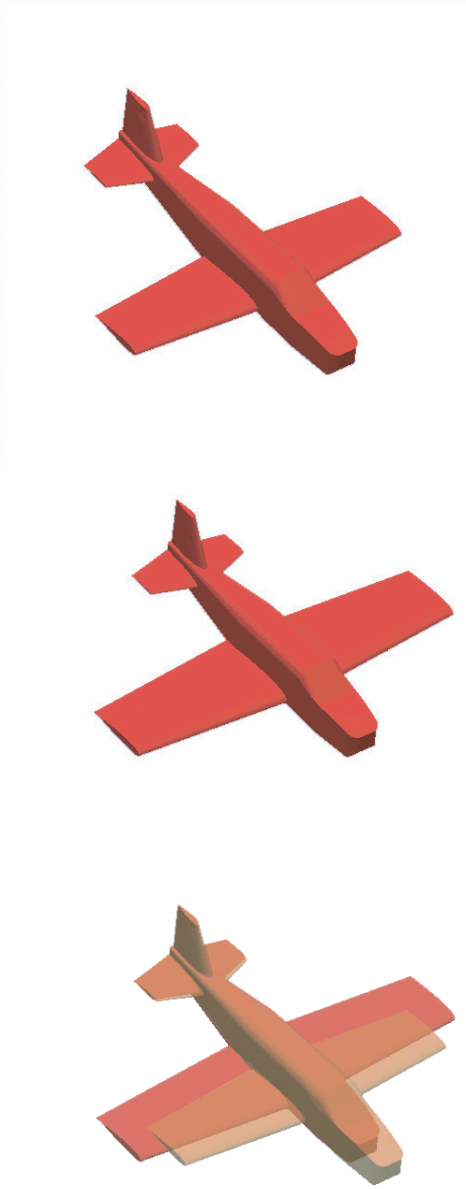


Figure 6-16: Snapshots of the design taken for the CAD model. Top: the initial design solution; middle: the final design solution; bottom: the initial and final design solutions are superimposed. The bottom plot clearly shows the design tradeoffs chosen by the optimizer

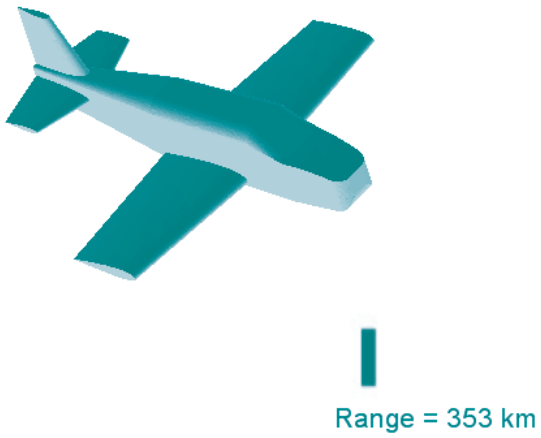


Figure 6-17: First iteration of the optimization of the GA aircraft

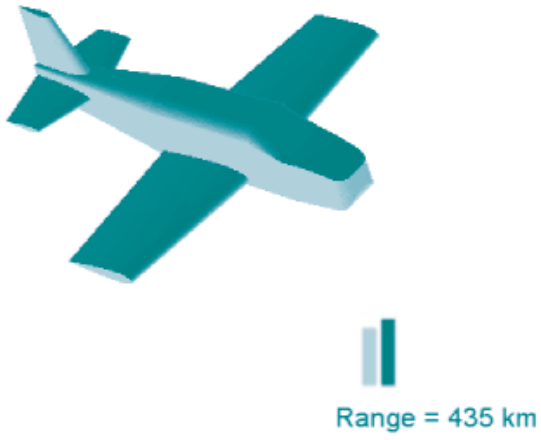


Figure 6-18: Second iteration of the optimization of the GA aircraft

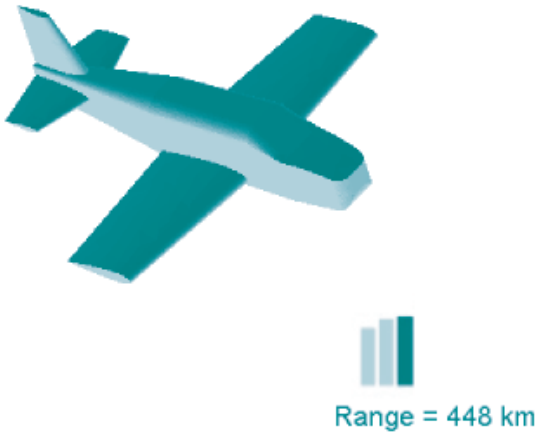


Figure 6-19: Third iteration of the optimization of the GA aircraft

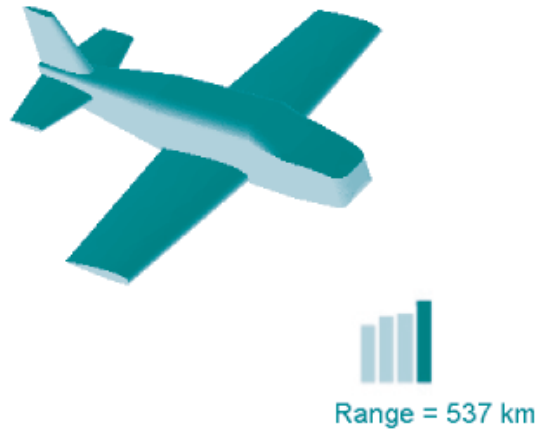


Figure 6-20: Fourth iteration of the optimization of the GA aircraft

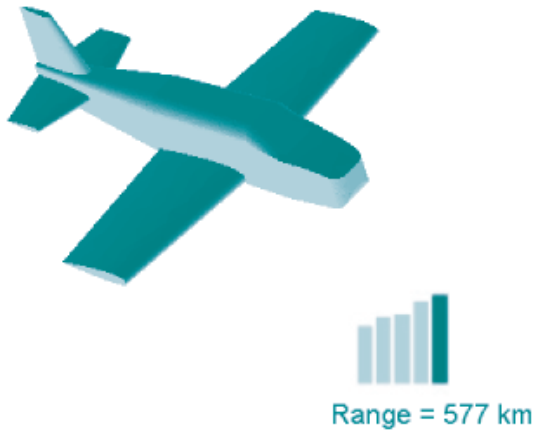


Figure 6-21: Fifth iteration of the optimization of the GA aircraft

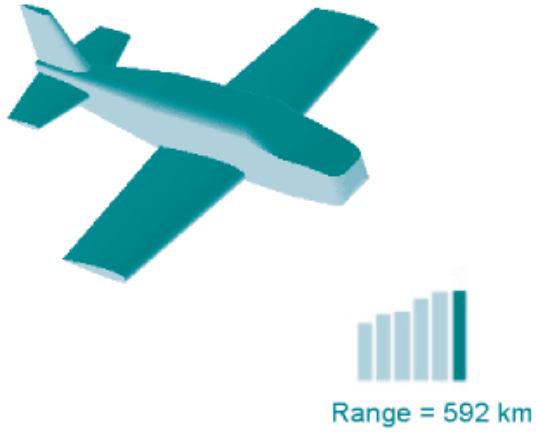


Figure 6-22: Sixth iteration of the optimization of the GA aircraft

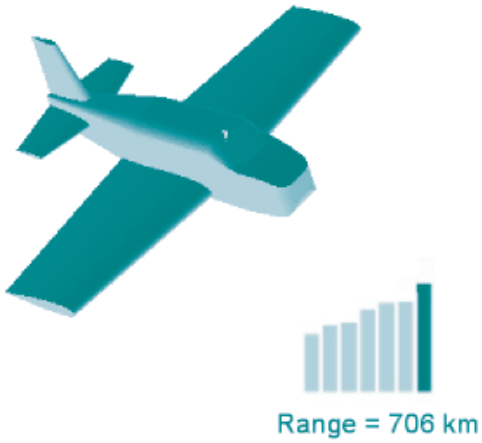


Figure 6-23: Seventh iteration of the optimization of the GA aircraft

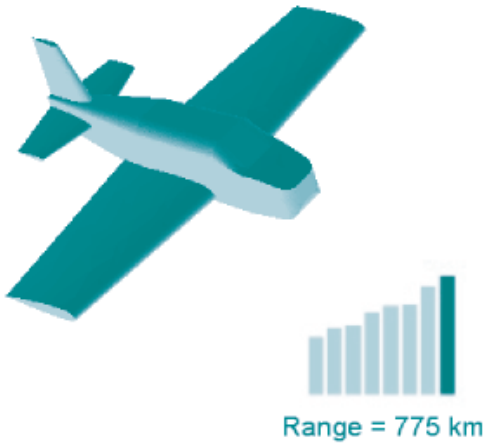


Figure 6-24: Final iteration of the optimization of the GA aircraft

x1	x2	x3	x4	x5	j	h1	h2	h3
14.3	7.5	8.22	1.5	40.36	373.929	1496.1	2.1	26.5
15.3525	7.7776	8.4562	1.5	41.6989	448.6569	1499.5	2.1	26.1
16.3226	8.0349	8.4556	1.5	43.4678	537.2606	1499.9	2.1	26
16.7268	8.1473	8.348	1.5	44.2036	577.1332	1500	2.1	25.9
16.916	8.2006	8.3056	1.5	44.3077	590.3721	1469.6	2.1	23.8
20.3491	9.2714	5.7991	1.5	42.2743	706.359	1484.7	2.1	24
20.3957	9.2311	6.5524	1.5	42.6802	732.2775	1499.9	2.1	24
20.3849	9.263	6.4424	1.5	42.7854	735.4066	1499.9	2.1	24
20.431	9.2609	6.5785	1.5	42.7882	738.8649	1499.9	2.1	24
20.5513	9.2955	6.5135	1.5	42.9053	746.4595	1500	2.1	24
20.6678	9.3183	6.564	1.5	42.9479	753.1838	1500	2.1	24
20.7638	9.3394	6.5866	1.5	42.9133	756.7762	1500	2	23.9
20.9421	9.3787	6.6233	1.5	42.8918	764.4262	1500	2	23.9
21.0582	9.4044	6.6438	1.5	42.8812	769.4419	1500	2	23.9
21.1906	9.4336	6.6656	1.5	42.8761	775.302	1500	2	23.9
21.1974	9.4351	6.6665	1.5	42.8798	775.6999	1500	2	23.9
21.1975	9.4352	6.666	1.5	42.8797	775.7015	1500	2	23.9
21.1973	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9
21.1974	9.4352	6.6661	1.5	42.88	775.7015	1500	2	23.9

Figure 6-25: Optimization values, for the five design variables, the objective and the tree constraints

constraint, even though it is physical, is not linked to a specific geometric feature. As a result, sensitivity analysis gives the designer valuable information on the design variables that participate strongly in the constraint.

The user interrogates the constraint using the tree in the constraint analysis module window (lower right of Figure 6-27). When the rate of climb constraint is selected, the sensitivity information is displayed. As shown by the bar graph in the figure, the sensitivity analysis suggests that the constraint is serving to limit the wing span and fuselage diameter and increase the wing area. At the same time, the physical display of the constraint is activated in the 3D visualization window. The appropriate features identified by the sensitivity analysis are highlighted, as in Figure 6-27, where the fuselage and wing of the GA aircraft are highlighted. This novel approach to physical constraint visualization, combined with the classic sensitivity bar chart, allows a designer to quickly gain insight to the design space and, if necessary, refine the optimization formulation. The full value of physical constraint visualization would be realized on a more complex example with many constraints and design variables. A similar analysis on other active constraints shows that, as expected, a further increase in wing span is limited by bending stress restrictions.

Figure 6-26: Trade-offs that lead to an augmentation of the objective function: Breguet range

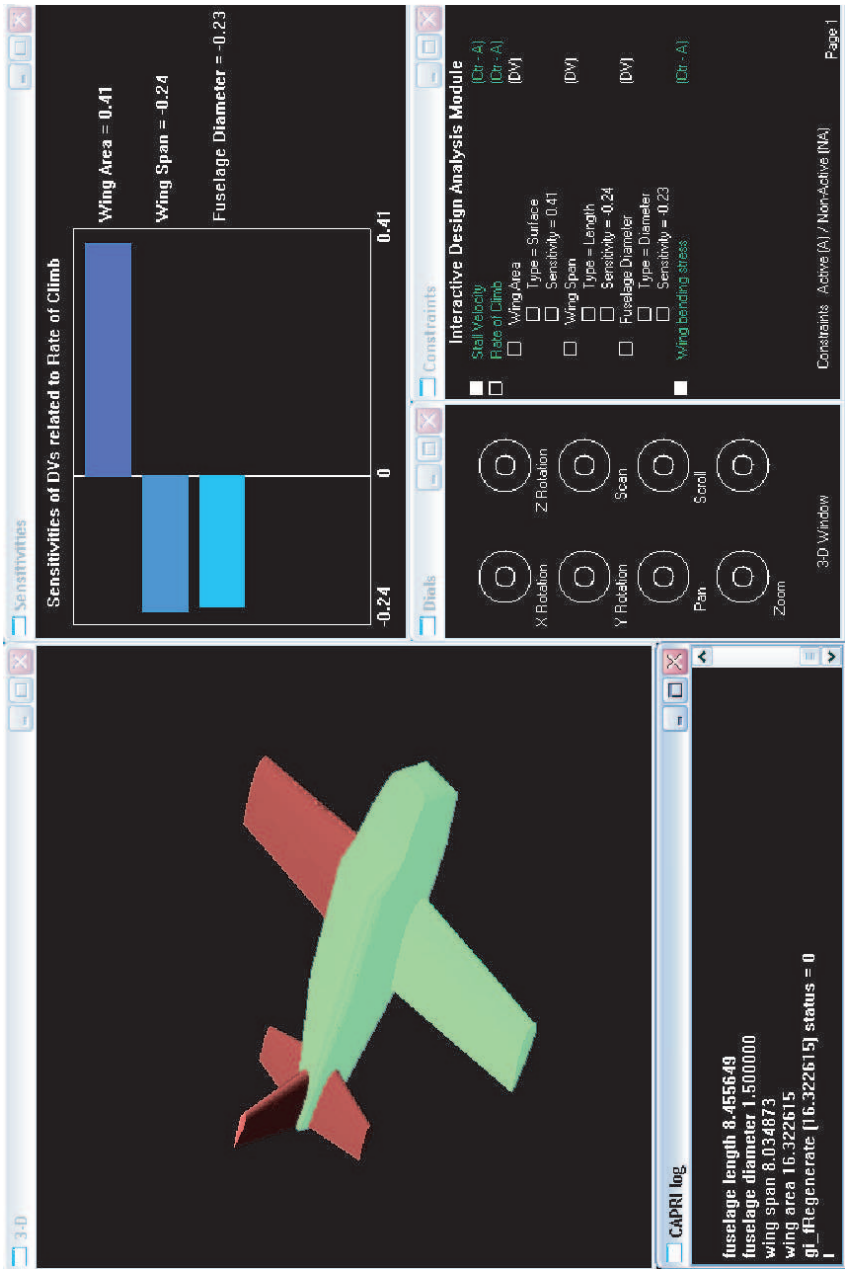


Figure 6-27: User interface for the real-time visualization and physical display of constraint analysis during the optimization process

Figure 6-28: The constraint on the fuselage diameter was forgotten. The optimization drives the design towards a fuselage diameter of zero. The CAD model would not regenerate at the next iteration.

6.3.4 Optimization health monitoring

While the previous example showed the value that can be gained from physically visualizing optimizer progress, another use of this methodology might be to monitor the health of an optimization run. This allows the designer early detection of errors in the formulation that lead the optimizer toward a physically unreasonable design solution, and may be an invaluable tool in the initial formulation of a complex MDO problem. Two examples are shown here for the GA aircraft example. Figure 6-28 represents a design iterate of an optimization for which the constraint on the fuselage diameter was forgotten. As a result, the weight and aerodynamic analyses try to drive the value of the fuselage diameter to its minimum, in this case zero. Figure 6-29 shows the case of a bad constraint on bending stress. In this case, the design is driven towards an infinite wing span in order to improve the lift-to-drag ratio and thus the range. In each case the optimizer would find, after several iterations, an optimum that is not physically reasonable. For a more complicated system, these wasted iterations could be costly and time-consuming. Visualization therefore allows the designer to detect a flaw in the analysis model, and stop the optimization early.

Figure 6-29: The constraint on the wing bending stress was forgotten. The optimization drives the design towards an infinite wing span.

6.4 Conclusion

The visualization and constraint analysis has been hereby tested and applied to a real design example: the GA aircraft design. It has proved its value on a simple example, by giving more physical insight to the designer about the optimization and its behavior, by making clearer the interactions of the constraints with the design evolution, and by giving the engineer a way to steer the design by better understanding how it works. It also appears to be a valuable tool to detect flaws in the design, and thereby to save money and time, both important metrics in today's industry.

Chapter 7

Conclusions and recommendations

The main objective of this research was to implement a visualization framework that can enhance the design process of complex engineering systems. Enhancing the design process, in the scope of an MDO framework, can take different forms. Visualization has been identified in this thesis as a need for the MDO community, as well as an efficient way to bring physical insight to the designer. The present methodology, based on the physical visualization of the system being optimized, is supported by the use of CAD modelling. Using a parametric approach, CAD modelling offers endless opportunities to change the design and modify the metrics that drive it. A framework was created, composed of three modules. The Analysis and Optimization Module (AOM) represents the classical MDO design approach. The Optimization Visualization Module (OVM) allows real-time visualization of the optimization steps, by the use of a physical representation of the system using a CAD model. It provides the designer with optimization path monitoring as well as optimization health monitoring. The final module, the Design Analysis Module (DAM) focuses on the analysis of a specific solution. It helps the designer to understand the importance of the constraints in driving the system towards a given optimum. It eventually helps the engineer to steer the design in order to further improve the final solution. CAPRI was used as the link between the three modules, by allowing geometry data exchange. The link between the different modules is created in such a way to preserve flexibility, mainly for the use of different codes in the AOM (such as legacy codes used in the industry).

The implementation of the interface itself, which serves as the way to actually use the methodology, was explained in detail. This included the identification of the needs,

requirements and goals. The language that would be used was discussed, in order to identify the most efficient way to implement a working interface. Each of the windows composing the interface was detailed, as well as the interaction between the different windows. Finally, the user input was discussed.

It is crucial to remind the reader of the importance of the user in the optimization definition, in the CAD model build-up and in the framework preparation. The CAD model and the optimization set-up both capture design intents, and not only vary from one system to another, but also can change the optimum solution that is obtained by the optimizer. Furthermore, the user's aid is required in the current version of the interface in order to link the "mathematical" elements of the optimization to the physical features on the design.

A simple problem that involves the design of a GA aircraft was then considered. Monitoring of the optimizer sequence of iterates was shown to lend physical insight to design trade-offs, without a time-consuming study of data files or numerical output. This real-time visualization was also shown to be useful for monitoring the health of an optimization run and for early detection of formulation errors. Visualization of constraint behavior was also demonstrated and shown to provide physical insight to the designer.

While a working methodology was established, it is of importance to note that this framework represents the first step in a more complex approach. It demonstrated the feasibility of physically-based, real-time approach of visualization and constraint analysis in the scope of a MDO framework. However, more work can be done to implement more advanced notions, such as the ones presented in Section 2.1.3. Constraint analysis was proven to have a major impact on the way a designer can understand and later steer the design and the optimization. But the techniques used in the framework developed in this research only included the use of sensitivity analysis. Even though sensitivity analysis provides the user with valuable information, the use of Lagrange multiplier information, as well as the calculation of dot products between the gradients of the objective and of the constraints, will lead to even more physical insights and understanding of the optimization.

An important area of future work is the adaptation of new models to the existing framework. The GA aircraft design case was used to benchmark and validate the approach. New cases need to be applied, principally larger scale problems. While the methodology and the the interface demonstrated their efficiency in the case of a small scale problem, it

is clear that the analysis of larger scale problems will need more investigation, and require to make decisions.

First, larger problems imply a lot more design variables and constraints. The novel methodology developed in this research allows the visualization of a physical model regardless of the size of the problem. The visualization of the CAD model is not dependent on the number of design variables or constraints. However, the display of information on the design, such as constraint analysis, will imply displaying a large amount of information. This issue can be solved by only displaying the information of greatest interest, which can be determined either computationally or by the user. Both computer power and monitor space are factors that need to be taken into account.

Secondly, larger problems will need an even larger amount of preparatory work from the designer, which might become overwhelming. The automation of the process of associating faces of the CAD model to specific features might become necessary. The user would be helped in this task by the creation of another interface that associates directly faces to features. This interface would replace the burdensome task of coding the information, as seen in Figure C-1. The enhancement of CAPRI could also alleviate the user's task by automatically imbedding the information in the CAD model and sharing it with the framework.

Finally, the use of many different analysis codes will test to great extent the modularity of the framework. Before using legacy codes, it is important to make sure that the linking between the code and the framework can be easily carried out, without disrupting the information flow and the optimization process.

Once again, one should remember that design cannot be a "push-button" only process. Design requires the intervention of the human, and can be improved from the knowledge and the expertise the designer can bring into it. Within this idea, it is clear that visualization offers a huge potential for improvement in designing large and complex engineering system. The framework proposed in this work is general, but only represents the first step towards a fully integrated design framework, that would embed visualization as well as design aids (sensitivity analysis, constraint analysis). This research has developed a methodology that can help visualizing, physically, the optimization process and can analyze the constraints of the optimization. It also used a concrete example to demonstrate the validity of the

framework and open the methodology for larger scale problems.

Appendix A

Aircraft Design and MDO Survey

A.1 Contributors

- *Jean-Charles Lede [JCL]*, System Engineering group leader, Aurora Flight Sciences
- *Anonymous [AN1]*, ED23/Structural Design Group, NASA/Marshall Space Flight Center
- *Anonymous [AN3]*, Aerospace Engineer, Air Force Research Laboratory
- *Richard Gilmore [RG]*, Engineer/Scientist Specialist, Advanced Air Vehicles Aero. Technology
- *Robert Canfield [RC]*, Air Force Research Laboratory
- *Dr. Vladimir Balabanov [VB]*, Senior Research and Development Engineer, Vanderplaats Research and Development, Inc.
- *Rob Taylor [RT]*, JSF Airframe Systems Engineering and Integration Team, Lockheed Martin Aeronautics Co.
- *Anonymous [AN3], LMCO*

A.2 Survey

A.2.1 Aircraft design

Could you summarize quickly your aircraft design activity?

[JCL]: Usually, it starts with a mission requirement: range, endurance, altitude, payload weight and other constraints, operation requirement (vertical take-off, air launch, sub-launch, etc...). A very rough sizing and configuration selection is done with rule of thumb (payload fraction, propulsion fraction, L/D guess, engine selection and nominal performance, etc...).

Then the first rough drawings are made and a initial design point is estimated more carefully: weight buildup by systems, drag build-up, engine performance model, simplified mission simulation. This model is setup to allow variations of all the main parameters and perform sensitivity analysis.

A full optimization is conducted for large programs fully funded and for large airplanes that are very performance sensitive and difficult to "tweak" between the prototype and the production vehicles.

[AN1]: I have not worked aircraft, but launch vehicles. Personal design experience involves composite and metallic intertanks, composite and metallic tanks, and vehicle integration. Currently working on a hi-fidelity (preliminary $\rightarrow$ detail), multidisciplinary design team. Responsible for geometry definition and supply of derivative data including mass properties, simplified analytical models, etc.

[AN2]: My current focus is on aeroelastic Joined-Wing concept design.

[RG]: I run a multidisciplinary design code called "WingMOD" which combines a vortex lattice code with simple structures, controls, and weights modules to optimize wing planforms. I primarily support the Blended-Wing-Body project, optimizing geometries for a variety of missions and performing trade studies. I also write code to update the analysis modules in the optimizer. I'm currently attempting to add an economic module.

[RC]: As an Air Force officer, I have been the project engineer for development of MDO software for aircraft design. I've also funded studies of applications of this software to aircraft design by aircraft manufacturers. As an associate professor of aerospace engineering, I am currently investigating the design of a joined-wing aircraft for a military sensor-craft mission.

[VB]: 1989-1993: Research Engineer at Central Aerohydrodynamic Institute (TsAGI), Russia. This is NASA-like research institute. Worked in static aeroelasticity division. Participated in designing several commercial aircrtaf.

1996: Intern at Boeing. Loads and Dynamics division, HSCT program, Renton, WA.

Participated in HSCT preliminary design.

1997-present: Development of general purpose optimization tool VisualDOC. Its application to several practical aircraft and non-aircraft design projects.

[RT]: Bonded composite joint design and analysis

[AN3]: I am primarily involved with the structural analysis and design of unmanned aerial vehicles serving in reconnaissance missions. In particular, I am examining the impact of various technologies on the aircraft and evaluating their performance.

What are the tools you are currently using for aircraft design? What do you think about these tools (advantages, drawbacks, etc)?

[JCL]: Most of the analysis is conducted in EXCEL with inputs from airfoil analysis (XFOIL or MSES), 3D aero (VSAERO or AVL), propeller analysis if required (XROTOR), manufacturers engine models. The advantages are a great flexibility. The drawbacks are that the model can easily become overwhelming and not always easy to conduct multi variable optimization.

[AN1]: On center, a plethora of tools. Some of the biggies include:

UG CAD, Veribest (Mentor Graphics) ECAD, OTIS, POST, Gridgen, FDNS, IGRIP, Patran, NASTRAN, ZAERO, ROCETS, BOSOR, PANDA, Hypersizer, STK, MP06 (Excel), SINDA.

Conceptual/Preliminary design uses some other tools, including CONSIZE, INTROS, and some operations and cost estimating relationships.

One of the big holes in multidisciplinary design/analysis is a good mesh interpolator between geometry based analytical disciplines. Being able, for example, to map CFD results onto a shell thermal model and onto a beam loads model, then getting results from the thermal and loads models mapped onto a stress model.

[AN2]: ASTROS – great for aeroelastic optimization.

NASTRAN – MSC is expensive

PANAIR – Hard to find expert advise – not easy to run/debug

Adaptive Modeling Language (AML) – Object-Oriented design modeling tool with native geometry and meshing capability. Is indispensable for unique integrated design modeling applications.

[RG]: The tool I use is "WingMOD". It is a collection of intermediate fidelity analyses

hooked up to an optimizer. The primary advantage is the ability to optimize planforms quickly. Once a baseline has been established, quick trade studies can be run to see the effects of various factors on wing planform. Another advantage is flexibility (any database variable can be the objective, and any input variable can be a design variable). There are two primary drawbacks: difficulty of use and fidelity of analysis. The WingMOD database contains order 10,000 variables and does not have an appealing user interface. The analysis is quick, but ultimately higher fidelity tools must be used to analyze the final design.

[RC]: Adaptive Modeling Language (AML), Automated Structural Optimization System (ASTROS), MSC.Nastran, PanAir, SDRC/I-DEAS, Design Optimization Tools (DOT), and Matlab.

They are adequate for the conceptual studies we are doing. Integration of the tools (data transfer and communication) and tailoring of the software to the specific problem (defining different objectives, having to optimize across several different models, etc.) are the biggest drawbacks.

[VB]: CAD packages.

Many advantages: including ease of communication, multiuser access, etc.

Drawbacks:

It is not easy to transfer complex models from one CAD to another. Not all the parametrization could be transferred.

All the changes to the model have to be done manually. It is very hard to incorporate programmatic changes in the model. For exmaple, linking CAD to optimization package is not trivial

[RT]: MSC PATRAN/NASTRAN—prevalent in aerospace industry; very comprehensive and useable tool for structural analysis at many levels (global to detail); includes capabilities for multidisciplinary analysis and design—aero, thermal, dynamic, optimization; customizable; interfaces well with other modeling and analysis tools;

ESRD StressCheck—p-method FEM solver for linear and nonlinear structural analysis; better at detail part analysis; parametric geometry and mesh capability; handbook capability allows designer to have pre-verified finite element model and resize for given scenario

In-house detail stress analysis tools—various tools of varying quality for specific analysis of detail structural loading and design scenarios

CATIA—comprehensive CAD tool; gets the job done and works; slow to implement newer

CAD capabilities; interface is not as intuitive for new users as other CAD systems;

[AN3]: In short, MSC/NASTRAN. I use NASTRAN for static aeroelasticity (linear aerodynamic loads), structural optimization, random gust analysis, and flutter analysis. The tool is very comprehensive, but does have a steep learning curve.

I've also used PATRAN/FLIGHTLOADS to build the static aeroelasticity model for NASTRAN. It too is very comprehensive.

I've also used an in-house code, ACAD, to build the FEM. It is particularly geared toward aircraft design, but doesn't have some of the detailed features that CATIA has. I'm not much of a CAD user, so I can't really comment much on the advantages/disadvantages of ACAD.

What aircraft design tools have you used in the past? Why did you decide to stop using them?

[JCL]: The previous version of optimization tools was based on a very large, difficult to use and to keep under configuration control C program.

In addition to the former limitations, the model could only run on computers with a C-compiler (now more common but not then). Other then that, it was a useful tool very powerful despite it's so-so optimization routine.

[AN1]: Usually commercial products and/or in-house developed code.

[AN2]: Pro-Engineer for geometric modeling. CAD software is too restrictive and is hard to integrate with other analysis packages.

[RG]: I have used WingMOD almost exclusively since I've worked here (2 years).

[RC]: CONMIN and Fortran. I stopped developing my own software when commercial tools were available.

[RT]: Pro/Engineer, Ideas, Altair Hypermesh/Optistruct, ASTROS, FLOPS, ANSYS, GENESIS, MathCAD, Matlab, Adaptive Modeling Language

Use depends on current job function and requirements. On a program assignment (my current job), tool use depends almost entirely on program management decisions on what will be supported for the program. This means prevalent, accepted tools that engineers are familiar with will be used. In an R&D environment, tool possibilities would be more liberal and newer, less mature tools could be used. Much of my tool experience comes from an academic research environment and simply does not fit in my current job function or my

company uses a competing product.

[AN3]: In my Ph.D. research, I used ASTROS since it is a public code, and I had access to the source code. Since working in industry, I haven't been using it, since it doesn't have some of the same comprehensive features as NASTRAN. It also isn't used as extensively as NASTRAN in industry. It is a good academic tool, however.

What are the main characteristics you expect from tools for aircraft design?

[JCL]: Flexibility. Aurora designs a very wide range of aircraft. Some for which rule of thumbs, linear assumptions, etc. work, some for which it doesn't even come close. We need to be able to change pretty much anything...

[AN1]: Accuracy. Reliability. Ability to share data with other tools.

[AN2]: I have gone the AML route (www.TechnoSoft.com). You can read about it there.

[RG]: I see a spectrum of tools. At one end are the high fidelity slow tools. At the other end are the low fidelity fast tools. There is always a trade-off between speed and fidelity. Both ends of the spectrum are equally important. Fast tools are needed for early development, while more accurate tools are needed for more details design work.

Some desired general properties are:

Easy to use. User friendly interface.

Flexible. Methods and algorithms change often.

[RC]: Reliability (can find a solution without much intervention)

Ease of use (can define my problem within confines of program's features)

Efficiency (reasonable turnaround time)

[VB]: Ease of use. Programmatic access to main features.

[RT]: Provide timely information to make decisions

Communicate information (not data) effectively and efficiently

Operate at a level of fidelity appropriate for the given phase of the design process

Interface effectively with other tools and processes within the design organization

[AN3]: - Accuracy

- As much as is possible, ease of use and clarity of inputs

- Good documentation

- Modifiable

What could be improved on the tools you are currently using?

[JCL]: The compatibility between the multi-variable, multi-constraint optimizer and the performance estimation routines.

[AN1]: As mentioned above, mesh interpolation.

[AN2]: We are integrating geometric modeling, analysis to produce rapid cost and weight estimates for rapid performance assessments.

[RG]: WingMOD needs a better user interface. Higher fidelity analyses could be incorporated, but they would likely penalize the speed too much.

[RC]: Integration with other tools being used.

[VB]: Eliminating disadvantages from question 2.

[RT]: Anything to facilitate the items in (4)—parametrics, associativity, data exchange, user interface, among others

[AN3]: Not sure, yet.

A.2.2 Multidisciplinary Design Optimization

Have you ever heard about MDO? What do you think about it?

[JCL]: Yes. It's great and critical for high performance vehicle such as high altitude UAV.

[AN1]: Yes. Supportive.

[AN2]: Typically, MDO is great if you already understand the design mechanics and want to squeak out the last few percent of performance.

[RG]: I work with MDO on a daily basis. It is a great idea. Optimizers can find solutions you would never have thought of. They make finding solutions quicker. Optimizers can also find unrealistic solutions by taking advantage of the shortcomings in the analyses, so one must be wary of that.

[RC]: Yes, it has great promise for improving designs.

[VB]: It is a powerful tool, but there are no established methods that are clearly useful for large practical programs at the final stages of the design.

[RT]: Yes. Noble cause, yet designers must carefully assess when it actually adds value to a program. Tools must integrate with product development processes to become truly valuable in aircraft design. Otherwise, MDO is an analytical process that can only verify design decisions or identify the need for late-cycle design decisions. MDO capabilities in

NASTRAN are headed in the right direction; they integrate fairly well with aircraft design if engineers and engineering managers would learn to exploit them effectively.

[AN3]: Yes. The concept of optimizing aircraft (or parts of aircraft) in consideration of multiple (and possibly conflicting) disciplines is very good. It is probably the only way to design a truly "optimal" airplane.

What do you like about MDO? What do you not like?

[JCL]: I like the fact that it includes effect of the airplane design main disciplines (structure static and dynamic, aero, controls) in a single model instead of in several external loops.

It can be difficult to setup the model-the optimizer will take advantage of any loophole there may be so everything has to be pretty tightly modeled.

[AN1]: MDO has huge potential benefits for providing major system level improvements in cost and performance. It should be applied at every possible opportunity.

But, it is not placed in the hands of those who could make the best application of it. MDO is traditionally performed by analysts, having largely grown out of a need for aeroelastic analysis. Therefore it's often operating on meshes and morphing them, and working in the guts of the analytical tools. Those who can best apply it, but not develop it, are designers, not analysts. People who are already accustomed to dealing with all the disciplines and have a top level understanding of many disciplines without getting particularly deep in any of them. Additionally, the powerful parametric CAD packages available these days are begging to be tied to optimization, and often are, but not for multiple disciplines.

[AN2]: When MDO works, it is like analysis with a unique solution. MDO will never replace design because MDO always requires more extensive prior knowledge in order to set it up.

[RG]: Pros: Combines disciplines that don't usually work together, thus allowing you to achieve an optimum quicker. Able to consider many more factors than a human designer when making design decisions. Quick. Cons: Finicky. Takes advantage of shortcomings of analyses.

[RC]: It can explore designs and identify the best ones, when I do not have the time to do it by conventional means.

[VB]: Single discipline optimization and MDO should be more actively used at the early stages of the design process.

[RT]: MDO addresses improves system capability by addressing interactions between disci-

plines that would not be captures by single disciplinary optimization or analysis. I believe that insufficient focus has been given to how MDO interfaces with the design process and how it actually addresses the needs of the aircraft design process. Define the process (and hence the needs of the process) and then design the tools to meet those needs. MDO sometimes seems to be viewed as an end in itself.

[AN3]: Like-The idea of all disciplines contributing simultaneously to the design of the aircraft in an integrated fashion. Dislike-not sure

Are you currently using MDO for aircraft design? What tools are you using in relation with MDO?

[JCL]: I wouldn't call it MDO but the models we use include aero, structure, and some controls.

The next big step for the high flyers would be to have some aeroelasticity capability.

[AN1]: No, we're focusing on Multidisciplinary Design/Analysis through the use of a good tool set and Product Lifecycle Management. In the past I've used ADS, DOT, and GA's as optimization tools.

[AN2]: Mostly structural optimization – ASTROS for starters.

[RG]: Yes. WingMOD.

[RC]: See #2.

[VB]: Whenever I can. The tool: VisualDOC

[RT]: Unfortunately, my current job function does not call for MDO.

[AN3]: NASTRAN-Structural optimization in consideration of aeroelastic (aero-structural interaction) constraints.

What are the main characteristics of MDO (and MDO tools) that prevented you or could prevent you from using MDO?

[JCL]: Lack of time. Everything has to be done before we can bring in a complex tool on line. The needs for it is also limited

[AN1]: - typically need a lot of in-depth coding to link the disciplines. Very hard and time-consuming to tie the disciplines together.

- mesh interpolation not readily available.

[AN2]: No comment

[RG]: Difficult to use. Must have breadth of knowledge to interpret results.

[RC]: Learning curve and the need to tie together many different analysis programs.

[VB]: It is not the MDO or MDO tools. There are plenty of good optimization tools that could do good job at MDO. For example: VisualDOC, iSight, LMS Optimus, etc.
It is the attitude of management that does not like changes. And does not like trying new things.

[RT]: Insufficient integration and interface with accepted standard design and analysis tools. Insufficient awareness of capabilities and value of MDO tools.

[AN3]: I imagine that a potential road block to using MDO techniques, might be the size of the problems that it is used on. In other words, does the MDO tool become too computationally or user intense when used on a realistic aircraft design problem ?

What new developments could be very beneficial to MDO? What would you expect from new MDO tools?

[JCL]: Again, having the aeroelestacity modeled is critical for the design of high flyers. Being able to change the environment is also important. On that note, heat exchangers have a significant impact on HALE as well.

[AN1]: - intimately linking MDO inside high end CAD tools using the optimization links already available
- MDO tools should not require the user to be an expert in all the disciplines involved in the optimization.

[AN2]: Read our AIAA/SDM paper.

[RG]: More computing power would be beneficial. It would be nice to have an MDO tool into which you could plug already existing analysis modules.

[RC]: MDO tools need to be developed for specific applications, or built into computer programs that are already doing simulations for design.

[VB]: The main benefit could be convincing management (and sometimes design engineers) to try new things and not to be fixed on the in-house tools.

[RT]: Improved integration and interface with accepted standard design and analysis tools. Improved definition of value-added provided by MDO tools to aircraft design.

[AN3]: Still being pretty new in the industry, I don't know that I can say yet, but probably that any tool should have a keen eye on the enormous complexity of aircraft design (or its

various parts). Also that the tool does not take "control" from the discipline expert. The disciplinarians must remain "in the loop."

Appendix B

GA aircraft weight model

General-Aviation Weights

$$W_{\text{wing}} = 0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{\cos^2 \Lambda} \right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100\ t/c}{\cos \Lambda} \right)^{-0.3} (N_z W_{dg})^{0.49} \tag{15.46}$$

$$W_{\substack{\text{horizontal} \\ \text{tail}}} = 0.016 (N_z W_{dg})^{0.414} q^{0.168} S_{ht}^{0.896} \left(\frac{100\ t/c}{\cos \Lambda} \right)^{-0.12} \times \left(\frac{A}{\cos^2 \Lambda_{ht}} \right)^{0.043} \lambda_h^{0.02} \tag{15.47}$$

$$W_{\substack{\text{vertical} \\ \text{tail}}} = 0.073 \left(1 + 0.2 \frac{H_t}{H_v} \right) (N_z W_{dg})^{0.376} q^{0.122} S_{vt}^{0.873} \left(\frac{100\ t/c}{\cos \Lambda_{vt}} \right)^{-0.49} \times \left(\frac{A}{\cos^2 \Lambda_{vt}} \right)^{0.357} \lambda_{vt}^{0.039} \tag{15.48}$$

$$W_{\text{fuselage}} = 0.052\ S_f^{1.086} (N_z W_{dg})^{0.177} L_t^{-0.051} (L/D)^{-0.072} q^{0.241} + W_{\text{press}} \tag{15.49}$$

$$W_{\substack{\text{main landing} \\ \text{gear}}} = 0.095 (N_l W_l)^{0.768} (L_m/12)^{0.409} \tag{15.50}$$

$$W_{\substack{\text{nose landing} \\ \text{gear}}} = 0.125 (N_l W_l)^{0.566} (L_n/12)^{0.845} \tag{15.51}$$

$$W_{\substack{\text{installed engine} \\ \text{(total)}}} = 2.575\ W_{en}^{0.922} N_{en} \tag{15.52}$$

$$W_{\text{fuel system}} = 2.49\ V_t^{0.726} \left(\frac{1}{1 + V_i/V_t} \right)^{0.363} N_t^{0.242} N_{en}^{0.157} \tag{15.53}$$

$$W_{\substack{\text{flight} \\ \text{controls}}} = 0.053\ L^{1.536} B_w^{0.371} (N_z W_{dg} \times 10^{-4})^{0.80} \tag{15.54}$$

$$W_{\text{hydraulics}} = 0.001\ W_{dg} \tag{15.55}$$

$$W_{\text{electrical}} = 12.57 (W_{\text{fuel system}} + W_{\text{avionics}})^{0.51} \tag{15.56}$$

$$W_{\text{avionics}} = 2.117\ W_{uav}^{0.933} \tag{15.57}$$

$$W_{\substack{\text{air conditioning} \\ \text{and anti ice}}} = 0.265\ W_{dg}^{0.52} N_p^{0.68} W_{\text{avionics}}^{0.17} M^{0.08} \tag{15.58}$$

$$W_{\text{furnishings}} = 0.0582\ W_{dg} - 65 \tag{15.59}$$

Figure B-1: GA aircraft weight model equations from Raymer [30]

Weights Equations Terminology

A = aspect ratio
B_h = horizontal tail span, ft
B_w = wing span, ft
D = fuselage structural depth, ft
D_e = engine diameter, ft
F_w = fuselage width at horizontal tail intersection, ft
H_t = horizontal tail height above fuselage, ft
H_t/H_v = 0.0 for conventional tail; 1.0 for "T" tail
H_v = vertical tail height above fuselage, ft
I_y = yawing moment of inertia, lb-ft^2 (see Chap. 16)
K_{cb} = 2.25 for cross-beam (F-111) gear; = 1.0 otherwise
K_d = duct constant (see Fig. 15.2)
K_{door} = 1.0 if no cargo door; = 1.06 if one side cargo door; = 1.12 if two side cargo doors; = 1.12 if aft clamshell door; = 1.25 if two side cargo doors and aft clamshell door
K_{dw} = 0.768 for delta wing; = 1.0 otherwise
K_{dwf} = 0.774 for delta wing aircraft; = 1.0 otherwise
K_{Lg} = 1.12 if fuselage-mounted main landing gear; = 1.0 otherwise
K_{mc} = 1.45 if mission completion required after failure; = 1.0 otherwise
K_{mp} = 1.126 for kneeling gear; = 1.0 otherwise
K_{ng} = 1.017 for pylon-mounted nacelle; = 1.0 otherwise
K_{np} = 1.15 for kneeling gear; = 1.0 otherwise
K_p = 1.4 for engine with propeller or 1.0 otherwise
K_r = 1.133 if reciprocating engine; = 1.0 otherwise
K_{rht} = 1.047 for rolling tail; = 1.0 otherwise
K_{tp} = 0.793 if turboprop; = 1.0 otherwise
K_{tpg} = 0.826 for tripod (A-7) gear; = 1.0 otherwise
K_{tr} = 1.18 for jet with thrust reverser or 1.0 otherwise
K_{uht} = 1.143 for unit (all-moving) horizontal tail; = 1.0 otherwise
K_{vg} = 1.62 for variable geometry; = 1.0 otherwise
K_{vs} = 1.19 for variable sweep wing; = 1.0 otherwise
K_{vsh} = 1.425 if variable sweep wing; = 1.0 otherwise
K_{ws} = $0.75[1 + 2\lambda)/(1 + \lambda)]$ $(B_w \tan\Lambda/L)$
K_y = aircraft pitching radius of gyration, ft ($\cong 0.3L_t$)
K_z = aircraft yawing radius of gyration, ft ($\cong L_t$)
L = fuselage structural length, ft (excludes radome, tail cap)
L_a = electrical routing distance, generators to avionics to cockpit, ft
L_d = duct length, ft
L_{ec} = length from engine front to cockpit—total if multiengine, ft
L_f = total fuselage length
L_m = length of main landing gear, in.
L_n = nose gear length, in.
L_s = single duct length (see Fig. 15.2)
L_{sh} = length of engine shroud, ft
L_t = tail length; wing quarter-MAC to tail quarter-MAC, ft
L_{tp} = length of tailpipe, ft
M = Mach number

Figure B-2: GA aircraft weight model parameters

N_c = number of crew

N_{cr} = 1.0 if single pilot; = 1.2 if pilot plus backseater; = 2.0 pilot and copassenger

N_{en} = number of engines

N_f = number of functions performed by controls (typically 4–7)

N_{gen} = number of generators (typically = N_{en})

N_l = ultimate landing load factor; = $N_{gear} \times 1.5$

N_{lt} = nacelle length, ft

N_m = number of mechanical functions (typically 0–2)

N_{mss} = number of main gear shock struts

N_{mw} = number of main wheels

N_{nw} = number of nose wheels

N_p = number of personnel onboard (crew and passengers)

N_s = number of flight control systems

N_t = number of fuel tanks

N_u = number of hydraulic utility functions (typically 5–15)

N_w = nacelle width, ft

N_z = ultimate load factor; = 1.5 × limit load factor

q = dynamic pressure at cruise, lb/ft^2

R_{kva} = system electrical rating, kv · A (typically 40–60 for transports, 110–160 for fighters & bombers)

S_{cs} = total area of control surfaces, ft^2

S_{csw} = control surface area (wing-mounted), ft^2

S_e = elevator area, ft

S_f = fuselage wetted area, ft^2

S_{fw} = firewall surface area, ft^2

S_{ht} = horizontal tail area

S_n = nacelle wetted area, ft^2

S_r = rudder area, ft^2

S_{vt} = vertical tail area, ft^2

S_w = trapezoidal wing area, ft^2

SFC = engine specific fuel consumption—maximum thrust

T = total engine thrust, lb

T_e = thrust per engine, lb

V_i = integral tanks volume, gal

V_p = self-sealing "protected" tanks volume, gal

V_{pr} = volume of pressurized section, ft^3

V_t = total fuel volume, gal

W = fuselage structural width, ft

W_c = maximum cargo weight, lb

W_{dg} = design gross weight, lb

W_{ec} = weight of engine and contents, lb (per nacelle), $\cong 2.331\, W_{engine}^{0.901}\, K_p K_{tr}$

W_{en} = engine weight, each, lb

W_{fw} = weight of fuel in wing, lb

W_l = landing design gross weight, lb

W_{press} = weight penalty due to pressurization, $= 11.9 + (V_{pr} P_{delta})^{0.271}$, where P_{delta} = cabin pressure differential, psi (typically 8 psi)

W_{uav} = uninstalled avionics weight, lb (typically = 800–1400 lb)

Λ = wing sweep at 25% MAC

Figure B-3: GA aircraft weight model parameters

Appendix C

User input in the case of the GA aircraft

```
/*****************************************************************************

        Module to assignate faces to specific features

        Yann Deremaux - October 2002

        This module will associate each feature
        that the designer wants to create to all the faces
        that compose the feature in GV.
        In needs to be done by hand, until Capri allows
        better handling of the information

*****************************************************************************/


void associate_feature (void)

        {
        int i;
        // features required for the GA aircraft=
        // wing
        // fuselage
        // others
        nbr_feat=4;

        //Number of faces per features
        //fuselage
        nbr_face_feat[1]=25;
        //right wing
        nbr_face_feat[2]=3;
        //left wing
        nbr_face_feat[3]=3;
        //horizontal and vertical stabilizer
        nbr_face_feat[4]=9;

        //numeros of the faces for each feature
        //fuselage
        for (i=1;i<=nbr_face_feat[1];i++)
        {
        ass_face_feat[1][i]=i;
        }
        //right wing
        ass_face_feat[2][1]=26;
        ass_face_feat[2][2]=27;
        ass_face_feat[2][3]=28;
        //left wing
        ass_face_feat[3][1]=29;
        ass_face_feat[3][2]=30;
        ass_face_feat[3][3]=31;
        //horizontal and vertical stabilizer
        for (i=1;i<=nbr_face_feat[4];i++)
        {
        ass_face_feat[4][i]=31+i;
        }

        }

 /*****************************************************
```

Figure C-1: User input for the GA aircraft that associates features to faces on the CAD model

```
/*************************************************************************

        Module to declare the design variables and constraints

        Yann Deremaux - October 2002

        This module will associate to each design variable
        a feature it belongs to (according to the number
        used in the declaration of the features),
        a type (non physical (type 0), length (type 1), surface (type 2)
        diameter (type 3), angle (type 4), ratio (type 5) or other (type 6),
        and finally its name. This will be gathered in a structure.
        It will also assignate a name to each constraint.


*************************************************************************/

void declare_DV(void)
{
        // first design variable = wing area
        DV[1].feature=2;
        DV[1].type=2;
        strcpy(DV[1].DVname,"Wing Area");

        // second design variable = wing span
        DV[2].feature=2;
        DV[2].type=1;
        strcpy(DV[2].DVname,"Wing Span");

        // third design variable = Fuselage Diameter
        DV[4].feature=1;
        DV[4].type=3;
        strcpy(DV[4].DVname,"Fuselage Diameter");

        // fourth design variable = Fuselage Length
        DV[3].feature=1;
        DV[3].type=2;
        strcpy(DV[3].DVname,"Fuselage Length");

        // fourth design variable = Fuselage Length
        DV[5].feature=0;
        DV[5].type=0;
        strcpy(DV[5].DVname,"Cruise Velocity");
        }

void declare_ctr(void)
{
        // first design variable = wing area
        strcpy(CTR[1].ctrname,"Stall Velocity");

        // second design variable = wing span
        strcpy(CTR[2].ctrname,"Rate of Climb");

        // third design variable = Fuselage Diameter
        strcpy(CTR[3].ctrname,"Wing bending stress");

        }
```

Figure C-2: User input for the GA aircraft that associates design variables to a name, a type and a physical feature, and constraints to a name

Bibliography

[1] Personal discussion with Robert Haimes about GV.

[2] http://adg.stanford.edu/aa241/AircraftDesign.html, June 2003.

[3] http://java.sun.com/, June 2003.

[4] http://www.tcl.tk/, June 2003.

[5] http://www.microsoft.com/windowsxp/moviemaker/downloads/moviemaker2.asp, June 2003.

[6] I. Abbott and E. von Doenhoff. *Theory of Wing Sections*. Dover Edition, 1959.

[7] J. Anderson. *Fundamentals of Aerodynamics, 3rd edition*. McGraw Hill, 1999.

[8] P. Bartholemew. The role of MDO within aerospace design and progress towards an MDO capability. AIAA-98-4705, 1998.

[9] C.L. Bloebaum and E.H. Winer. Design visualization by graph morphing for multidisciplinary design optimization. Conference proceedings of first international conference on Engineering Design and Automation (EDA '97), Bangkok, Thailand, 1997.

[10] E.E. Covert, editor. *Thrust and Drag : Its Prediction and Verification (Progress in Astronautics and Aeronautics, Vol 98)*. AIAA, New York, NY, 1985.

[11] C.A. Crawford. An integrated CAD methodology applied to wind turbine optimization. Master's thesis, Massachusetts Institute of Technology, June 2003.

[12] D.L. Darmofal and R. Haimes. Preliminary development of a visual environmnent for multidisciplinary optimization. AIAA paper 96-1942, presented at the 27th Fluid Dynamics Conference, New Orleans, LA, June 1996.

[13] R. D. Finck and D. E. Hoak. USAF stability and control DATCOM. USAF Contract F33615-76-C-3061, April 1978.

[14] J. P. Giesing. A summary of industry MDO applications and needs. AIAA Paper 98-4737, June 1998.

[15] R. Haimes. pV3: A distributed system for large-scale unsteady cfd visualization. AIAA Paper 94-0321, presented at the 32th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1994.

[16] R. Haimes and G. Aftosmis, M. On generating high quality 'water-tight' triangulations directly from CAD. Proceedings of the 8^{th} International Conference on Numerical grid generation in Computational Field Simulations. Honolulu, Hawaii, June 2002.

[17] R. Haimes and G. Follen. Computational Analysis PRogramming interface. Proceedings of the 6^{th} International Conference on Numerical grid generation in Computational Field Simulations. University of Greenwhich, September 1998.

[18] S.F. Hoerner. *Fluid Dynamic Drag*. Hoerner Fluid Dynamics, 1965.

[19] R. Horst and Pardalos P. M., editors. *Handbook of Global Optimization*, chapter Quadratic Optimization, pages 217–269. Kluwers Academic Publishers, London, 1995.

[20] P. Jackson. *Janes All the World Aircraft*. Janes Information Group Ltd, Surrey, U.K., 1997-98.

[21] J. P. Jarret, W. N. Dawes, and P. J. Clarkson. Accelerating turbomachinery design. GT-2002-30618, Proceedings of the ASME Turbo Expo 2002, The Netherlands, June 2002.

[22] J.J. Korte, Weston R.P., and Zang T.A. Multidisciplinary optimization method for preliminary design. presentation at AGARD Interpanel (FDP+PEP) Symposium "Future Aerospace Technology in the Service of the Alliance, Paris, France., April 1997.

[23] C. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints, 1994.

[24] K. Lewis. Multidisciplinary design optimization. Aerospace America, 2002 Air—Space the year in review, December 2002.

[25] L.D. Liaw, R.I. DeVries, and D.L Cronin. An MDO-compatible method for robust design of vehicles, systems, and components. AIAA Paper 98-4786, 1998.

[26] A. Messac and X. Chen. Visualizing the optimization process in real-time using physical programming. AIAA-98-39707, 1998.

[27] S. K. Ojha. *Flight Performance of Aircraft*, chapter 12. AIAA Education Series, 1995.

[28] AIAA Technical Committee on Multidisciplinary Design Optimization. White paper on current state of the art, January 1991.

[29] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design*. Cambridge University Press, Cambridge, United Kingdom, 2000.

[30] D. P. Raymer. *Aircraft Design: A Conceptual Approach*, chapter 15. AIAA Education Series, third edition, 1999.

[31] H.J. Reekie, C. Hylands, and E.A. Lee. Tcl and Java performance. http://ptolemy.eecs.berkeley.edu/ cxh/java/tclblend/scriptperf/scriptperf.html, December 2002.

[32] J. Roskam. *Airplane Aerodynamics and Performance*. DARCorporation, Lawrence, Kansas, 1988.

[33] B. Roth and D.N. Mavris. Analysis of advanced technology impact on HSCT engine cycle performance. AIAA paper 99-2379, presented at the 35th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Los Angeles, California, June 1999.

[34] P.E. Rubbert. *On The Pursuit of Value for CFD*. Caughey, D.A and Hafez, M.M., 1998.

[35] A.O. Salas and J.C. Townsend. Framework requirements for MDO application development. AIAA paper 98-4740, 1998.

[36] A. Samant, P. Shah, and E.H. Winer. Visual design steering to aid decision-making in optimal design. AIAA-2002-4815, presented at the $9^{th} AIAA/USAF/NASA/ISSMO$ Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, September 2002.

[37] J. Sobieszczanski-Sobieski and R.T. Haftka. Multidisciplinary aerospace design optimization - Survey of recent developments. AIAA Paper 96-0711, presented at the 34th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1996.

[38] S. Venkataraman and R. T. Haftka. Structural optimization: What has Moore's law done for us ? AIAA-2002-1342, Presented at the 43^{rd} AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference in Denver, CO., April 2002.

[39] S. Wakayama. Blended-Wing-Body problem optimization setup. AIAA Paper 2000-4740, presented at the $8^{th} AIAA/USAF/NASA/ISSMO$ Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, September 2000.

[40] S. Wakayama and I Kroo. The challenge and promise of Blended-Wing-Body optimization. AIAA Paper 98-4736, presented at the $7^{th} AIAA/USAF/NASA/ISSMO$ Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, September 1998.

[41] E.H. Winer and C.L. Bloebaum. Visual design steering for optimization solution improvement. AIAA-2000-4815, presented at the $8^{th} AIAA/USAF/NASA/ISSMO$ Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, September 2000.

[42] T. A. Zang and L. L. Green. Multidisciplinary design optimization techniques: Implications and opportunities for fluid dynamics research. AIAA paper 99-3798, presented at the 30^{th} AIAA Fluid Dynamics Conference, Norfolk, VA, June 1999.