Santtu Salmi

Multidisciplinary Design Optimization in an Integrated CAD/FEM Environment

Master's Thesis
in Information Technology
July 15, 2008

University of Jyväskylä

Department of Mathematical Information Technology

Jyväskylä

Author: Santtu Salmi
Contact information: ssalmi@jyu.fi
Title: Multidisciplinary Design Optimization in an Integrated CAD/FEM Environment
Työn nimi: Monitieteinen muodon optimointi integroidussa CAD/FEM-ympäristössä
Project: Master's Thesis in Information Technology
Page count: 81
Abstract: The interest toward Multidisciplinary Design Optimization (MDO) is increasing due to advances in computer hardware and parallel computing. This thesis introduces basic concepts and challenges related to design optimization and MDO. A single discipline design optimization platform is presented along with data from test runs. Groundwork is laid to prepare the expansion of the platform to an MDO platform.
Suomenkielinen tiivistelmä: Tietokoneiden ja rinnakkaislaskennan kehityksen myötä kiinnostus monitieteistä muodon optimointia kohtaan on kasvamassa. Tämä tutkielma esittää muodon optimoinnin ja monitieteisen muodon optimoinnin peruskäsitteitä ja haasteita. Yksitieteinen muodon optimointialusta esitellään testitulosten kanssa. Lopuksi luodaan suunnitelma alustan laajentamiseksi monitieteiseksi alustaksi.
Keywords: multidisciplinary, design, optimization, MDO, CAD, CAE, ModeFRONTIER, CAPRI
Avainsanat: monitieteinen, muodon, optimointi, CAD, CAE, ModeFRONTIER, CAPRI

Contents

List of Figures

List of Tables

Nomenclature

α_i	angle of ellipsoid i
$\cdot$	inner product operator
Δ	Laplace operator
$\nabla\cdot$	divergence operator
∇	gradient operator
AAO	All-at-once
API	Application Programming Interface
BEM	Boundary Element Method
BLISS	Bi-level integrated system synthesis
BVP	Boundary-Value Problem
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAPRI	Computational Analysis PRogramming Interface
CFD	Computational Fluid Dynamics
CGM	Common Geometry Module
CO	Collaborative optimization
CSSO	Concurrent subspace optimization
d	number of design variables
DOE	Design of Experiments
η	dynamic viscosity
e_0	modeling error
e_1	discretization error
e_2	numerical error
EA	Evolutionary Algorithm
f, f_i	objective function
$\mathbf{f}$	objective vector
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
$\mathbf{g}_c$	common inequality constraint vector
$\mathbf{g}_i$	discipline i inequality constraint vector
g_i	inequality constraint (function)

$\mathbf{g}$	inequality constraint vector
GA	Genetic Algorithm
GUI	Graphical User Interface
$\mathbf{h}_c$	common equality constraint vector
$\mathbf{h}_i$	discipline i equality constraint vector
h_i	equality constraint (function)
h_i	height of ellipsoid i
$\mathbf{h}$	equality constraint vector
$\mathbf{I}$	identity matrix
IDF	Individual-Discipline-Feasible
k	number of objective functions
l_i	length of ellipsoid i
MDA	Multidisciplinary Analysis
MDF	Multiple-Discipline-Feasible
MDO	Multidisciplinary Design Optimization
MDOIS	Multidisciplinary optimization based on independent subspaces
MOEA	Multiple Objective Evolutionary Algorithm
MOGA	Multiple Objective Genetic Algorithm
MOO	Multiobjective Optimization
$\mathbf{n}$	outward unit normal
NS	Navier–Stokes (equations)
NSGA-II	Non-dominated Sorting Genetic Algorithm II
Ω, Ω_x	domain in $\mathbb{R}^d$
$\partial\Omega$	boundary of Ω
PDE	Partial Differential Equation
$\mathbb{R}^d$	d-dimensional Euclidean space
ρ	density
RS	Response Surface
RSM	Response Surface Methodology
$\mathbf{s}_i^c$	discipline i surrogate coupling vector
SA	Simulated Annealing
SAND	Simultaneous Analysis and Design
SBD	Simulation Based Design
SEED	Simulation Environment for Engineering Design
SSO	Sizing and Shape Optimization
$\mathbf{u}, \mathbf{u}_x$	trial solution
U^{ad}	set of admissible designs (design space)

U^{fs}	set of feasible designs
w_i	discipline i analyzer
$\mathbf{x}^*$	(Pareto) optimal design vector
$\mathbf{x}_c$	common design vector
$\mathbf{x}_i$	discipline i design vector
x, y, z	Coordinates in Ω
x_i	design variable
$\mathbf{x}$	design vector
$\mathbf{y}$	objective vector
Y	feasible objective space
$\mathbf{z}$	state vector $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2)^T$
$\mathbf{z}_i$	discipline i state vector $\mathbf{z}_i = (\mathbf{z}_i^c, \mathbf{z}_i^{nc})^T$
$\mathbf{z}_i^c$	discipline i coupling state vector
$\mathbf{z}_i^{nc}$	discipline i noncoupling state vector

Preface

Recently I have been working in a project involving a design optimization platform integrated to Catia V5 CAD system using the CAPRI interface. During the time in the project I have gained some insight to design optimization and to the advantages that our optimization system offers over traditional optimization systems, especially if Multidisciplinary Design Optimization is required.

I was responsible for the technical implementation of the optimization system, and so I needed to gain some knowledge about all of the subsystems included in the optimization cycle. This includes CAD, surface (mesh) generation, volume mesh generation, solving the state equation and optimization. The test platform is a single discipline design optimization platform, but it was built from the start with the idea that it will be expandable to use multiple disciplines, and to solve multidisciplinary optimization problems (the difference between the two will be explained in the thesis). A comprehensive introduction will be presented, and no previous knowledge about design optimization is assumed.

1 Introduction

The use of computers to simulate physical phenomena is a rapidly growing field of research. Theoretically any physical process can be simulated by formulating it to a mathematical model. For the most part, computational analysis is used in engineering in order to improve a design or reduce costs. This is referred to as Computer Aided Engineering (CAE). It is closely related to Computer Aided Design (CAD), which is the standard way of creating a geometric definition of a physical object that is aimed to be manufactured.

CAE analysis has developed from validation and failure analysis to an integral part of the product design process [6]. This design methodology where simulation drives the design is referred to as Simulation-Based Design (SBD). Over the years, CAD systems and CAE analysis have evolved separately with distinct geometrical models. CAD-CAE interoperability has been a well recognized issue for a long time. A study conducted for the automotive industry in the United States estimated that money wasted to re-entering or translating data between CAD systems and downstream applications is around one billion dollars per year [64]. Alleviating this bottleneck would therefore bring significant time and monetary savings.

The desire to do multidisciplinary design optimization (MDO) with geometries obtained from CAD systems also requires the solution of this interoperability problem. In a process where hundreds or even thousands of designs are automatically generated, interactive translation and repair of geometry is out of the question. The design optimization platform has to be able to automatically generate new geometries via a valid parameterization.

Multidisciplinary design optimization (also known as multidisciplinary system design optimization) is a type of design optimization that incorporates a number of engineering disciplines (and sometimes also nonengineering disciplines). The fact that MDO incorporates various disciplines simultaneously allows it to more accurately simulate physical phenomena that interact with other physical phenomena. MDO originated from the needs of aerospace engineering. However, other engineering disciplines are following the path paved by aerospace engineers. The inherent engineering origin of MDO makes it important to be able to integrate it with CAD systems as closely as possible.

In this thesis, preliminary concepts needed to understand MDO are presented,

and the challenges in implementing MDO are studied. The CAD-CAE interoperability issue will be demonstrated and different approaches to solve it will be presented. The optimization platform of our research group will also be presented, along with its composition. Experimental data from test runs is shown from three optimization runs. So far, only single discipline design optimization runs have been made. Things are presented from an engineering viewpoint by focusing on computational analysis that is done with geometries defined by CAD systems.

2 Design Optimization

2.1 Introduction

The traditional way of designing includes costly and time consuming experiments and their analysis. To some extent we can replace traditional way of designing with computational analysis, saving considerable amount of time and money. Computational analysis can be thought of as a new tool in the design phase of a product, as some physical phenomena are very hard or even impossible to study with experiments, but rather easy to simulate with computational analysis [70]. This methodology is kind of an intermediate between theory and experiment, in which synthetic experiments are based on basic theoretical concepts.

The behavior of a phenomenon in a system depends upon the geometry of the system, the property of the material or medium, and the boundary and initial conditions [45]. With design optimization the goal is to find the optimal shape or topology of this system, and sometimes also the optimal values of material and physics parameters. If only the shape of the system is optimized while other properties of the system remain constant, we can use the term optimal shape design or shape optimization [34].

2.2 Types of Design Optimization

Sizing and shape optimization (SSO) are the most mature and researched disciplines of design optimization [7]. The difference between them is that with sizing optimization we are only changing the sizes of parameterized components (e.g. thickness), where as in shape optimization we can also have more complex parameterized components such as splines.

The relative newcomer to design optimization is topology optimization. Topology optimization defines the geometry with cells. The computational area is specified by the size and number of these cells, and within these limitations it is free to form any conceivable geometry. Therefore, topology optimization is not confined to the restrictions of the initial geometric shape of the design. For example, with shape optimization no additional holes can be created during the optimization process (see figure 2.1) [7]. However, the drawback of topology optimization is that it

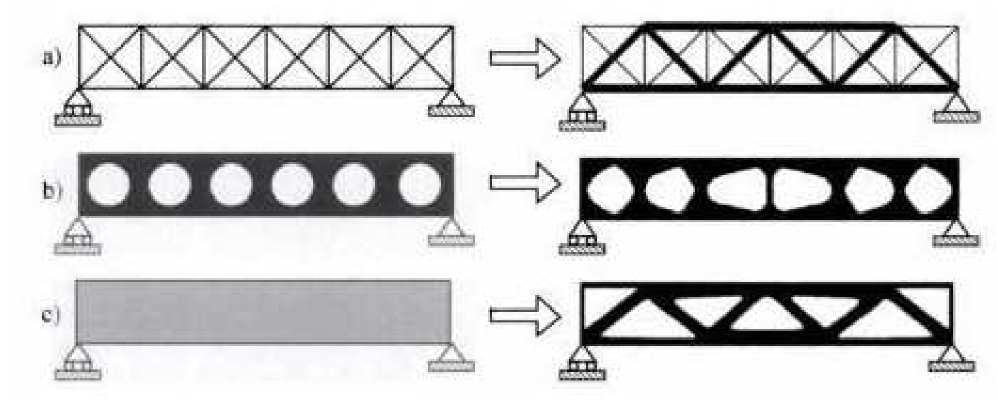

Figure 2.1: The initial and final designs of three types of design optimization: a) sizing optimization b) shape optimization and c) topology optimization.

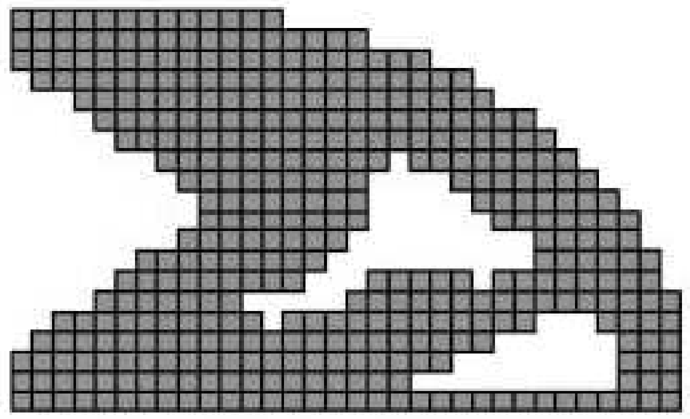

Figure 2.2: Cells of a 2D topology optimization problem.

always leads to a nonsmooth structural geometry [62], as can be seen from figure 2.2. Most engineering applications require a smooth geometric shape, especially for manufacturing. To overcome this, methods that convert the nonsmooth geometry of topology optimization to a smooth geometry exist. To take this a step further, topology and shape optimization can be combined by using topology optimization to find an initial design that will be used as the starting point of shape optimization [62]. However, the conversion process is the bottleneck of this kind of combined optimization.

2.3 Definition

Definition 2.1 *We call a design optimization problem, the following optimization problem [57]:*

$$\begin{aligned} \textit{Find } \boldsymbol{x} &\in U^{ad} \subset \mathbb{R}^d, \\ \textit{to minimize } f(\boldsymbol{x}), \\ \textit{subject to } \boldsymbol{g}(\boldsymbol{x}) &\leq 0, \\ \boldsymbol{h}(\boldsymbol{x}) &= 0, \end{aligned} \tag{2.1}$$

where $\boldsymbol{x} = (x_1, x_2, ..., x_d)^T$ is the design vector, $x_1, x_2, ..., x_d$ are the design variables, $f : \mathbb{R}^d \to \mathbb{R}$ is the objective function, $\boldsymbol{g}$ is a vector of inequality constraints, $\boldsymbol{h}$ is a vector of equality constraints, and U^{ad} is the set of admissible designs.

The set U^{ad} is also called *design space*. Design variables in size and shape optimization are typically lengths and thicknesses of components and positions of control points. In topology optimization the design variables define the density of each of the cells.

Usually U^{ad} defines technical constraints in order that the design candidate makes sense (nonnegative thicknesses, nonintersecting parts in a CAD model, etc). The implicit constraint functions **g** and **h** usually depend on the performance of the design (state constraints). If a design satisfies the constraints **g** and **h**, we can denote that it belongs to the set of *feasible* designs $\mathbf{x} \in U^{fs}$.

Furthermore, we can "artificially" restrict U^{ad} or add constraints to confine the optimization, as many times the complete design space is too large, and the complete exploration of it would require unreasonable amounts of computation. For example, we might already have a reasonably good solution and thus be only interested in the exploration of the neighboring design space. Of course, this might as well be a bad decision if we happen to exclude the best solutions. In practice, design variables are either Boolean, discrete or continuous with a set or interval of admissible values.

2.4 Parameterization

In the scope of design optimization, we want to solve a computational model for a series of different geometries. Each of these is called a *design* and it is defined by the design variables. In shape optimization the variables are used, for example, to alter a parameterized CAD model in order to obtain different geometries. The way the parameterization is done in the model dictates the shapes we can represent with

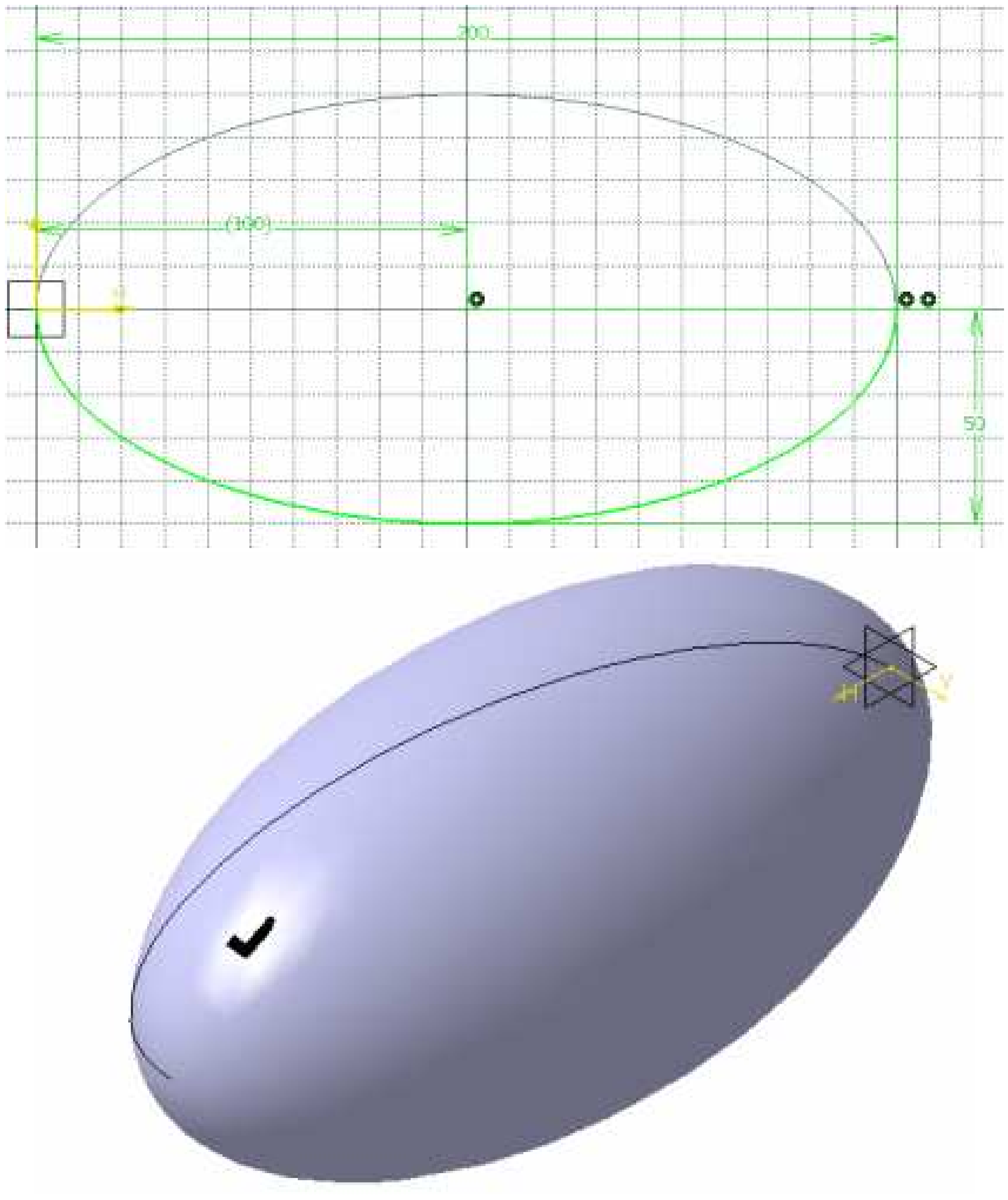

Figure 2.3: A simple parameterized CAD model of an ellipsoid in 2D and 3D views.

it. In figure 2.3 a simple parameterized Catia V5 model can be seen. The model has two parameters: length and height. The 3D model is formed when the 2D ellipse is rotated along its x-axis.

The design space parameterization has an important role in the performance of the design optimization process. As presented in [54], an ideal shape parameterization of an automated design optimization platform should:

1. Be able to generate a large variety of physically realistic shapes with as few design variables as possible.
2. Be robust meaning that a random perturbation of the design variables should still provide a realistic design.
3. Be generic to be applied to a large variety of shape optimization problems and able to be integrated or coupled with any existing CAD system.
4. Provide design variables that can easily be handled by an engineer in order to define design variable bounds.
5. Provide an easy optimization problem by minimizing the skewness and improving the conditioning of the design space.

2.5 Evaluating the Objective Function

2.5.1 The State Problem

The objective function $f(\mathbf{x})$ of the design optimization problem is a function constructed to describe the fitness of a design. It gives a numeric fitness value to a geometry. The objective function needs to be formulated in a way that the smaller it is, the better the design. The fitness value does not usually depend only on the design, but also on the state of the system. For example, in aerodynamics the state can be the lift and drag of our object, and in structural analysis the stresses and structural deformations inside our object. Usually the state problem is a partial differential equation (PDE).

Depending on analysis type (e.g. aerodynamic/structural) the PDE is solved inside or outside the object. Either way, the geometry of our object influences the computational domain Ω, and its boundary $\partial\Omega$. The reason why the computational area needs to be divided to the interior Ω and its boundary $\partial\Omega$ is because for the PDE to have a unique solution inside domain Ω, we need to define boundary conditions on the boundary $\partial\Omega$. This PDE with additional restraints is called a Boundary-Value Problem (BVP). The abstract BVP can be formulated as:

$$\begin{cases} \mathbf{A}(\mathbf{u}) = \mathbf{0} \text{ in } \Omega, \\ \mathbf{B}(\mathbf{u}) = \mathbf{0} \text{ on } \partial\Omega \end{cases} \tag{2.2}$$

where **A** and **B** are differential operators. **A** is the PDE that governs the solution in the domain Ω and components of **B** are the boundary conditions.

As a concrete example, let us formulate the Navier–Stokes BVP. Navier–Stokes equations (NS) describe the motion of viscous flows. Many interesting physical phenomena can be modeled by the Navier–Stokes equations (e.g. weather, water and air flows). Unfortunately, little is known of the mathematical properties of the Navier–Stokes equations. The existence and smoothness of a solution in three dimensions has not been proved. In fact, the solution of these problems is one of the Millennium Problems [25]. The mesh would need to contain an unreasonable amount of nodes for any problem in 3D that contains a moderate amount of turbulence. Usually some turbulence model is used to make the problem solvable (e.g. k-ϵ turbulence model). The incompressible steady-state Navier–Stokes BVP can be formulated as follows:

$$\begin{cases} \nabla \cdot \Big[-\rho \mathbf{I} + \eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \Big] - \rho(\mathbf{u} \cdot \nabla)\mathbf{u} + \mathbf{F} = \mathbf{0} & \text{in } \Omega, \\ \nabla \cdot \mathbf{u} = \mathbf{0} & \text{in } \Omega, \\ \mathbf{B}(\mathbf{u}) = \mathbf{0} & \text{on } \partial\Omega \end{cases} \tag{2.3}$$

where ρ and η are the density and dynamic viscosity of the fluid (respectively). **I** is the identity matrix, and **F** is the volume force affecting the fluid.

The domain Ω and the solution **u** of the BVP (the state problem) depend of course on the design **x**. Therefore their dependence is often emphasized by writing them as Ω_x and $\mathbf{u}_x$. The dependence of f on **x** is implicit, i.e.

$$x \rightarrow \Omega_x \rightarrow F(\mathbf{u}_x) =: f(\mathbf{x}) \tag{2.4}$$

where F is a functional that measures the fitness (cost) related to the state $\mathbf{u}_x$

For an engineering system, the geometry can be very complex. Furthermore, the boundary conditions can also be complicated. It is therefore generally impossible to solve the governing differential equation analytically [45]. In practice, most of the problems are solved using numerical methods. There are several methods to achieve that. The most popular and the one that we are using in this thesis is the Finite Element Method (FEM), but also worthy of a mention are the Finite Difference Method (FDM), the Boundary Element Method (BEM) and the Finite Volume Method (FVM). All of these methods give us an approximation of the analytic solution.

The reason why FEM is so popular is probably because it is computationally very effective, it has a solid theoretical foundation, and it can be applied virtually to any geometry. In consequence, to reduce computational effort adaptive schemes are applicable and popular. This is especially useful in problems where we have critical points of interest. It is also important to note that FEM uses the variational (weak) formulation of the BVP. For more information about FEM see [37].

2.5.2 Mesh generation

The finite element method requires the computational area to be split up to small elements. This way we can discretize the equations of the BVP. Mathematically speaking, we replace continuous differential equations with a system of simultaneous algebraic equations [67]. We call the discretized computational area a mesh (or grid). A mesh is usually generated with automatic mesh generation software (i.e. mesher). A mesher requires a surface representation of the geometry as its input. This input can be either an analytic boundary representation (B-rep) or a discrete representation consisting of sets of faces. In case of the latter, we lose some precision as curved surfaces will be approximated by a finite amount of faces (or lines in 2D).

The job of the mesh generator is then to produce a volume mesh that conforms to the boundary representation as closely as possible. The more elements the mesh has, the more accurately it can represent the original geometry, but it also requires more memory and computational time to generate and use such a mesh. The requirements for the mesh vary depending on the state problem. For example, structural analysis does not usually require a very fine mesh, whereas Computational Fluid Dynamics (CFD) does. Therefore a balance of accuracy and performance needs to be found for each particular problem.

Meshes can be divided into three categories: structured, unstructured, and hybrid meshes [28]. Structured meshes have a logical and repeating structure, and the elements always have the same number of neighbors. When speaking of structured meshes we are usually referring specifically to meshes with only rectangular or parallelepiped elements. Unstructured meshes, which are used with FEM, can have elements of any type. The most common element used is a tetrahedron (3D) or triangle (2D). Probably the most popular unstructured mesh generation methods are the Delaunay methods, which provide elements that are isotropic in nature. A mesh which is partly structured and partly unstructured is called a hybrid mesh. An unstructured 3D mesh generated with TetGen [66] can be seen in figure 2.4. For more about mesh generation see [67, 28].

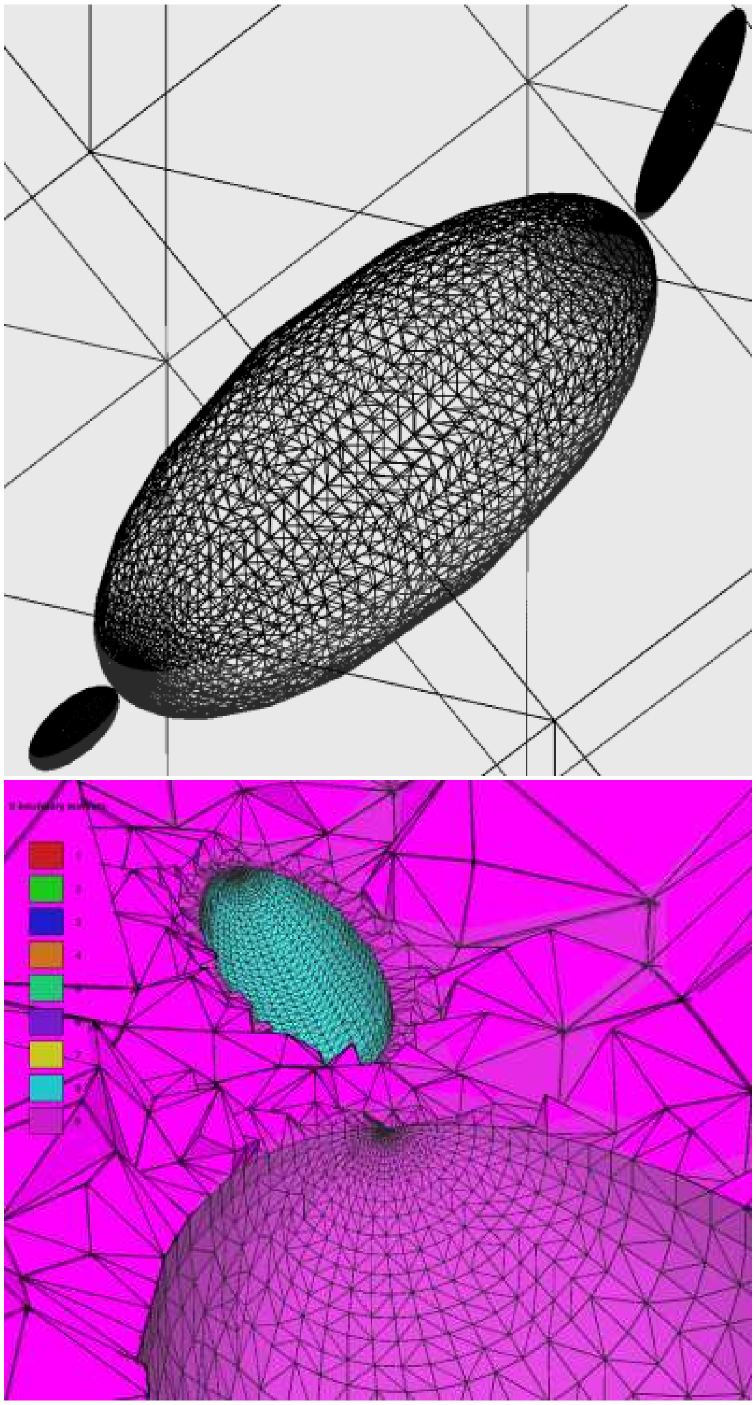

Figure 2.4: An unstructured 3D mesh with ellipsoidal cavities in boundary and cut plane views.

2.5.3 Solving the State Problem

When the mesh is ready and the mathematical model has been chosen, we have everything we need to solve the problem. We want to find a solver that is the most capable of doing this. Knowing which methods are best for each specific problem requires experience and theoretical knowledge. The computational model is fed to a solver, which is to solve the discretized system. Different solvers use different algorithms depending upon the physical phenomenon being simulated. The resulting algebraic equations can be solved either with a direct or an iterative method. Direct methods are well suited for small problems as they operate with the fully assembled equation matrix. Iterative methods are better suited for larger problems where the fully assembled equation matrix is too large to be handled at once.

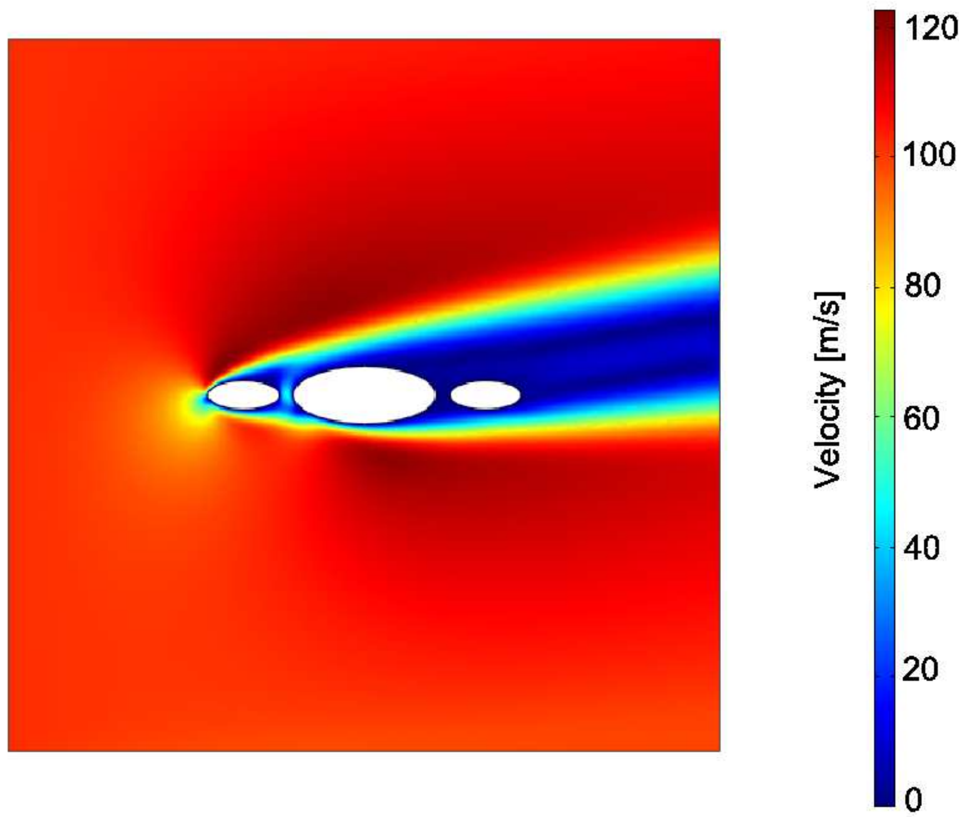

Figure 2.5: Solution of a 2D Navier–Stokes BVP.

When a solution has been found, using our objective function we get a numeric fitness value for the solution. Usually it involves integrating over certain parts of the domain, with respect to some physical quantity obtained from the solution of the state problem. For example, we could integrate the pressure over the boundaries of a wing to get information about its lift.

An example solution of the Navier–Stokes BVP can be seen from figure 2.5. An inflow with an x-component of 100 m/s and y-component of 20 m/s is set on the left boundary of the domain. The ellipses have the no slip ($\mathbf{u} = \mathbf{0}$) boundary condition, and the rest of the boundaries are set as neutral. The parameters were set as $\mathbf{F} = \mathbf{0}$, $\eta = 1$ and $\rho = 1$. The value of dynamic viscosity $\eta = 1$ is very high, which causes the flow to have minimal turbulence. Therefore, the solution is easy to obtain. For example, the dynamic viscosity of air is around $\eta = 1.5 \cdot 10^{-5}$ (depending on pressure and temperature). The velocity field of the solution is plotted in the figure.

2.5.4 Errors in Finite Element Approximation

The mathematical model is always a simplification of the real world, and much consideration needs to be used when creating the mathematical model. The results are worthless if the model is created erroneously or does not correspond to the physical reality. This simplification always results in modeling error e_0. To be sure that we are getting results that are correct, a group of controlled test problems should be solved. These problems are such that we already know the answer, and we are only solving them to know if the mathematical model is valid [70].

The mathematical model is usually an infinite dimensional problem and to solve it, we need to approximate it with a finite dimensional problem. This introduces another error e_1 (discretization error). A third and final error e_2 (numerical error) is induced when the discrete problem is solved with iterative methods using floating point arithmetic [42]. Thus to insure that results correspond to what we want to know, care needs to be taken so that all of these errors remain small.

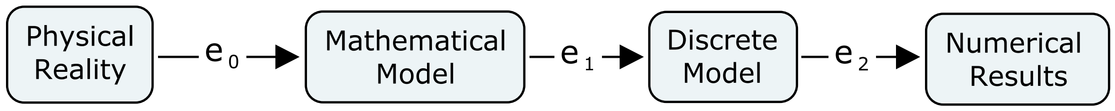

Figure 2.6: Errors involved in finite element approximation.

The control and identification of errors in finite element approximation is a broad and active research topic. For more on the subject see [1, 35, 50]

2.6 Optimization and Design of Experiments

To find increasingly better designs, we need some way of exploring the design space. The traditional approach is to test out different designs interactively. They could be defined by an experienced engineer for example. However, when a design optimization problem has tens or hundreds of design variables, the deduction of their impact on the state problem become impossible. Therefore, this trial-and-error type exploration of the design space is not sufficient. A systematic methodology of conducting experiments is called Design of Experiments (DOE) [2]. A set of designs are randomly or systematically generated using a DOE algorithm to try to cover the design space. However, usually an optimization algorithm is used instead in order to find the best designs more efficiently. The optimization algorithm can also be used in conjunction with the DOE sequence. For example, a DOE sequence can be used as a starting population of a genetic algorithm (GA). A DOE sequence can also be used to construct a response surface (RS) model.

The design optimization problem (2.1) is generally a d-dimensional nonlinear constrained optimization problem, and we can exploit the theory of nonlinear programming to find the best designs with as few function evaluations as possible. The complete process of evaluating the objective function is often computationally very expensive, and only a single objective function evaluation can take anywhere from minutes to hours. Therefore, the optimization algorithm should converge rapidly with a low number of function evaluations.

However, in worst case engineering problems the objective function is multimodal, high-dimensional and nondifferentiable [40], and the approximation methods used in its evaluation it cause it to be noisy [68]. These kind of difficult functions that cannot be analyzed with the usual mathematical tools are referred to as black-box functions. Finding an optimal solution to such a design optimization problem is difficult. For example, gradient-based methods have difficulties because of numerical noise caused by e_1 and e_2. The noise creates artificial local optima and causes gradient-based optimization algorithms to converge slowly or even fail to converge.

There are some methods to try to circumvent this problem, such as the aforementioned Response Surface Methodology (RSM) [31]. A response surface methodology can be loosely thought of as "d-dimensional curve fitting", and the result is a response surface model fitted to the available data. For detailed information about the response surface methodology see [49].

High-dimensionality causes the design space to be very large. We can restrict it with constraints, but doing so risks ruling out optimal solutions. Moreover, because of multimodality we can have multiple basins of attraction, and local optimization methods will risk getting stuck in local optima. To summarize, an optimization method of a design optimization problem should ideally have the following qualities:

1. Global properties[1]

2. Gradient free

3. Fast convergence

4. Minimal amount of function evaluations

5. Ability to cope with high-dimensional black-box functions

Unfortunately a perfect method that would fill all these requirements does not exist [23, 24, 39]. Evolutionary algorithms (EA) and Simulated Annealing (SA) are popular in design optimization because of their ability to deal with high-dimensional functions, and because they do not need any gradient information. Evolutionary algorithms [4] can cope with multimodal, noisy and high-dimensional objective functions, they are gradient free and with some probability they are able to find the global optimum. The downside of EAs is that generally they require many function evaluations and converge rather slowly [54]. Various types of EAs (e.g. memetic algorithms) have adaptations to get rid of the slow convergence. For examples see [12, 52].

Simulated annealing [41] is also gradient free and can cope with black-box functions. The behavior of SA depends on the cooling schedule. With a slow cooling schedule it should have a higher probability of finding the global optimum, but a slow convergence, and with a fast cooling schedule the opposite is true.

2.7 Multiobjective Design Optimization

In (2.1) definition of a *single* objective (design) optimization problem was presented. However, many real world optimization problems have multiple objectives that are conflicting. A simple example could be: buy a good and cheap car. Generally, the cheapest car is not the best, but it is not probably worth it to pay a fortune for the best car either. A compromise would have to be found. First we would of course need to define what is a “good” car. Then we can rule out some cars that are clearly too expensive for their quality. We would be left with cars that have the most attractive price/quality ratio. We call these cars *Pareto optimal* solutions if a better quality car always costs more, and a cheaper car is always of lower quality. In other words,

[1]i.e. ability to escape or avoid local optima and therefore have some probability of finding the global optimum. No method is guaranteed to find the global optimum of a general optimization problem [32].

improvement of one objective always leads to the deterioration of another objective. A trade-off needs to be made between quality and price to ultimately choose which car to buy. Therefore, the objective space of a Multiobjective Optimization (MOO) problem is often referred to as *trade space*. Let us define the multiobjective optimization problem, along with some other necessary MOO concepts:

Definition 2.2 *We call a multiobjective design optimization problem, the following optimization problem [47]:*

$$\begin{aligned} &\textit{Find } \boldsymbol{x} \in U^{ad} \subset \mathbb{R}^d, \\ &\textit{to minimize } \{(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_k(\boldsymbol{x})\}, \\ &\textit{subject to } \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}, \\ &\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \end{aligned} \tag{2.5}$$

where we have k (≥ 2) objective functions $f_i : \mathbb{R}^d \to \mathbb{R}$. We can denote the vector of objective functions by $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_k(\boldsymbol{x}))^T$.

Definition 2.3 *The feasible objective space (or trade space) Y of an MOO problem is the image of the feasible region U^{fs}. That is $Y = \boldsymbol{f}(U^{fs}) \subset \mathbb{R}^k$.*

Definition 2.4 *A design vector $\boldsymbol{x}^* \in U^{fs} \subset U^{ad}$ is Pareto optimal iff there does not exist another design vector $\boldsymbol{x}$ such that $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ for all $i = i, ..., k$ and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}^*)$ for at least one index j.*

An objective vector $\boldsymbol{y}^* \in Y$ is Pareto optimal iff there does not exist another objective vector $\boldsymbol{y} \in Y$ such that $y_i \leq y_i^*$ for all $i = 1, ..., k$ and $y_j < y_j^*$ for at least one index j.

Definition 2.5 *An objective vector $\boldsymbol{y}^* = (y_1^*, ..., y_k^*) \in \mathbb{R}^k$ is ideal, if it solves the problem:*

$$\begin{aligned} &\textit{Find } \boldsymbol{x} \in U^{ad} \subset \mathbb{R}^d, \\ &\textit{to minimize } f_i(\boldsymbol{x}) \\ &\textit{subject to } \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}, \\ &\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}, \\ &\textit{for } i = 1, ..., k \end{aligned}$$

note that $\boldsymbol{x}$ can be different for each i.

We can assume that at least some of the objectives are conflicting[2]. In consequence, it is not possible to find a solution to an MOO problem that would simultaneously be optimal for all objective functions f_i. Therefore, the ideal objective vector $\mathbf{y}^*$ is never a feasible solution[3], but it is useful as a point of reference. An illustration of the ideal objective vector and the Pareto front can be seen in figure 2.7.

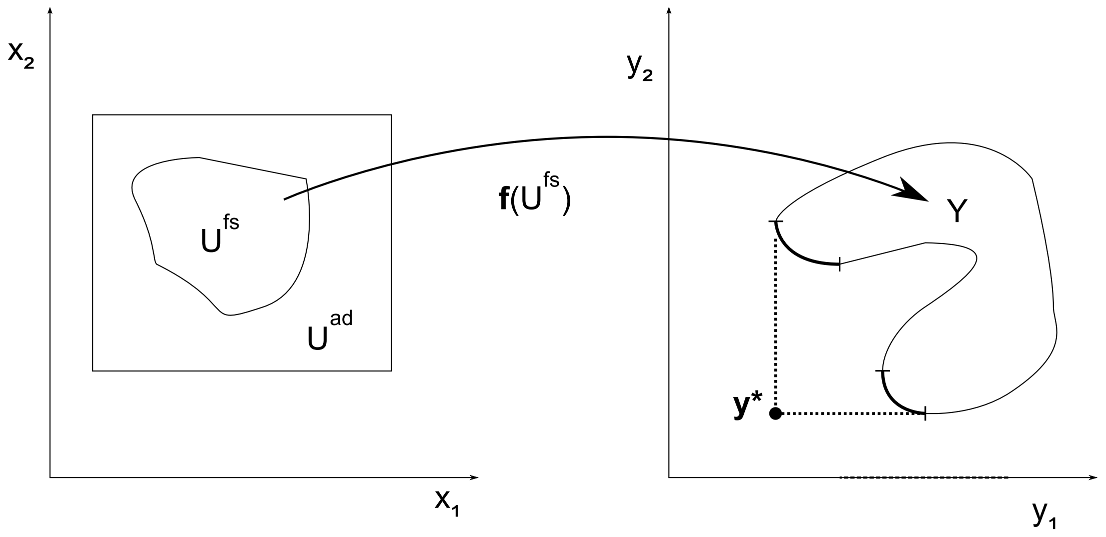

Figure 2.7: Ideal objective vector $\mathbf{y}^*$ and the Pareto front (bold).

The objective function $\mathbf{f}$ is vector valued, and ordering of vectors is ambiguous. For example, $(1, 1)^T$ can be said to be less than $(3, 3)^T$, but how to compare $(1, 3)^T$ and $(3, 1)^T$? Methodologies to obtain a single optimal solution always involve some sort of decision making. The most common approach is to use a *scalarizing function* to convert the MOO problem to a single objective optimization problem. Methods to solve an MOO problem can be divided in the following categories [47]:

- No-Preference Methods
- A Posteriori Methods
- A Priori Methods
- Interactive Methods

[2]if this is not the case, the problem returns to the single objective design optimization problem (2.1)

[3]i.e. $\nexists \mathbf{x} \in U^{fs}$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{y}^*$

Detailed description of methods falling in these categories goes beyond the context of this thesis. A comprehensive description and comparison of many MOO methods can be read from [47]. Let us describe the categories briefly with some examples.

2.7.1 No-Preference Methods

No-Preference methods return the MOO problem to a single objective optimization problem using some simple scalarizing function. The decision maker is then left with the option to either accept or reject the solution. As an example, the Method of the Global Criterion reformulates the MOO problem as follows [47]:

$$\begin{aligned} \text{Find } \mathbf{x} &\in U^{ad} \subset \mathbb{R}^d, \\ \text{minimize } &(\sum_{i=1}^{k} |f_i(\mathbf{x}) - y_i^*|^p)^{1/p} \\ \text{subject to } &\mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ &\mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned}$$

The idea is to find the Pareto optimal solution that has the minimal distance to the ideal objective vector $\mathbf{y}^*$ (using L_p-metrics). All objectives are equally important.

2.7.2 A Posteriori Methods

A Posteriori methods generate a set of Pareto optimal solutions. The decision maker then has to choose the preferred solution. The difficulty is that generating a set of Pareto optimal solutions is usually computationally expensive and sometimes difficult [47]. The most well-known A Posteriori method is the Weighting method. The problem is scalarized using a weighted sum of the objective functions:

$$\begin{aligned} \text{Find } \mathbf{x} &\in U^{ad} \subset \mathbb{R}^d, \\ \text{to minimize } &\sum_{i=1}^{k} w_i f_i(\mathbf{x}) \\ \text{subject to } &\mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ &\mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned}$$

where $w_i \geq 0$ for all $i = 1, ..., k$ and $\sum_{i=1}^{k} w_i = 1$.

The decision maker uses the weights w_i to prioritize the objectives. Objective functions should be scaled so that higher valued objective functions do not get unwanted advantage. The Weighting method can find all Pareto optimal solutions of a convex MOO problem [47]. However, the properties of objective functions are rarely known beforehand, and if the objective space is nonconvex, the Weighting method can only find a part of the Pareto optimal solutions. Therefore, an MOO problem should be considered to have *non-commensurable* objectives, unless it is known to be convex [27].

Recently Multiple Objective Genetic Algorithms (MOGAs) or more generally Multiple Objective Evolutionary Algorithms (MOEAs) have sparked a lot of interest [17]. They are claimed to robustly find a diverse Pareto optimal set of a nonconvex MOO problem [22]. For examples see [17, 27, 22].

2.7.3 A Priori Methods

With A Priori methods the decision maker has to make his or her preferences before the actual solution process. The difficulty is that the decision has to be made before additional information of the problem can be deducted from solutions of the state problem. As an example, let use formulate the Value Function Method:

$$\begin{aligned} \text{Find } \mathbf{x} &\in U^{ad} \subset \mathbb{R}^d, \\ \text{to maximize } & U(\mathbf{f}(\mathbf{x})) \\ \text{subject to } & \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}, \end{aligned}$$

where $U : \mathbb{R}^k \to \mathbb{R}$ is a value function that provides a complete ordering in the objective space.

The Value Function Method is an excellent method if the decision maker happens to know an explicit mathematical formulation for the value function, and if it truly represents the decision maker's preference [47]. However, usually this is not the case, and the formulation of the value function U is the difficulty of this method.

2.7.4 Interactive Methods

Interactive methods are designed to overcome the weaknesses of other MOO methods. A set of feasible solutions is generated, after which the decision maker is required to answer some questions. This cycle is repeated until an acceptable solution is found. Obviously, in a design optimization platform the interaction needs to be planned so that it is not needed constantly (e.g. a few times per day). Interactive methods are very versatile, and can be expected to produce satisfactory results. The formulations of interactive methods are rather long, but they can be read from [47].

2.8 Summary

To conclude this chapter, let us recapitulate the design optimization process. When a design optimization process is started, an initial guess or population of designs is generated using a DOE sequence (e.g. random generation). Next, the fitness of the designs is evaluated by the usual computer simulation cycle, which includes at least the following:

1. Surface generation
2. Mesh generation
3. Solving
4. Post processing and evaluation

Finally, the optimization algorithm generates a new design (or set of designs) that are to be evaluated. This process is repeated until a satisfactory design has been found. In Multidisciplinary Design Optimization the design optimization process is somewhat more complicated.

3 Multidisciplinary Design Optimization

3.1 Introduction

Real-world engineering systems are rather complicated, and to accurately simulate them in a computer environment, various different aspects should be taken into account. Advances in computer hardware and parallel computing methods have prepared the way for the construction of more complex optimization systems. Instead of optimizing only one subsystem of an engineering system, it is becoming possible to optimize an entire engineering system as a whole, including also a number of non-engineering disciplines (e.g. manufacturing, economics) [31].

The traditional way has been to analyze different disciplines sequentially one at a time with teams of people specialized in their own field of expertise. This approach has its problems. For example, if an aerodynamics teams finds an optimal shape in their perspective, this could leave the structural analysis team with a shape that is structurally unstable. Objectives of different disciplines are much of the time conflicting and all objectives should be considered simultaneously to achieve optimality.

More importantly, many physical phenomena in the real world are interacting. To accurately simulate them in a computer environment this interaction should be taken into account as well. In MDO the idea is to include all the necessary disciplines to the optimization simultaneously. An MDO method can exploit the interactions between disciplines and this is why a solution found by MDO is superior to the traditional method. The ideology of MDO seems intuitive, but the problem is in implementation. Simultaneous solving of disciplines significantly increases the complexity of the problem. That is not the only problem: the requirements for the tools and the organizational challenge of constructing an MDO system are also significantly higher.

3.2 Definition

A general definition of MDO was presented in [59] by J. Sobieszczanski-Sobieski as "MDO can be described as a methodology for the design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting subsystems". I would like to emphasize the word *interacting* in the definition. According to this definition, optimization of a system with multiple disciplines but no interaction among these disciplines is not MDO. Nevertheless, such systems are sometimes called multidisciplinary. Systems that share information in only one direction are mathematically equivalent to systems that have completely separated disciplines. They use multiple disciplines, but are not coupled and therefore depend implicitly only on design variables. The formal definition of MDO requires that disciplines are coupled. For simplicity, let us define the MDO problem with only 2 disciplines.

Definition 3.1 *The 2-discipline MDO problem [71]:*

$$\text{Find } \boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_c)^T \in U^{ad} \subset \mathbb{R}^d, \tag{3.1a}$$

$$\text{to minimize } \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} f_1(\boldsymbol{x}_1, \boldsymbol{x}_c, \boldsymbol{z}_1) \\ f_2(\boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_2) \\ f_c(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_1, \boldsymbol{z}_2) \end{bmatrix}, \tag{3.1b}$$

$$\text{subject to } \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \boldsymbol{g}^1(\boldsymbol{x}_1, \boldsymbol{x}_c, \boldsymbol{z}_1) \\ \boldsymbol{g}_2(\boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_2) \\ \boldsymbol{g}_c(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_1, \boldsymbol{z}_2) \end{bmatrix} \leq \boldsymbol{0}, \tag{3.1c}$$

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{x}_1, \boldsymbol{x}_c, \boldsymbol{z}_1) \\ \boldsymbol{h}_2(\boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_2) \\ \boldsymbol{h}_c(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_1, \boldsymbol{z}_2) \end{bmatrix} = \boldsymbol{0}, \tag{3.1d}$$

$$\boldsymbol{z}_1 = w_1(\boldsymbol{x}_1, \boldsymbol{x}_c, \boldsymbol{z}_2^c) \tag{3.1e}$$

$$\boldsymbol{z}_2 = w_2(\boldsymbol{x}_2, \boldsymbol{x}_c, \boldsymbol{z}_1^c) \tag{3.1f}$$

where $\boldsymbol{x}$ is the design vector, which is divided into discipline specific local design variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and the common design vector $\boldsymbol{x}_c$. The objective function vector $\boldsymbol{f}(\boldsymbol{x})$ is also divided into local $(\boldsymbol{f}_1, \boldsymbol{f}_2)$ and common $(\boldsymbol{f}_c)$ objective functions. $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are the inequality and equality constraint vectors, and their components are the local constraint vectors $(\boldsymbol{g}_1, \boldsymbol{g}_2)$ and $(\boldsymbol{h}_1, \boldsymbol{h}_2)$, and the common constraint vectors $\boldsymbol{g}_c$ and $\boldsymbol{h}_c$. w_1 and w_2 are the analyzers of disciplines 1 and 2 respectively, and $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ are state variable vectors. $\boldsymbol{z}_1^c$ and $\boldsymbol{z}_2^c$ are coupling variable vectors.

What sets an MDO problem apart from an ordinary design optimization problem is that in an MDO problem two or more analyzers are coupled. Therefore, they need the results of each other as their input. In other words, the objective and/or constraint functions of an MDO problem are not only functions of design variables, but they also depend on state variables. The state variables $\mathbf{z}_1$ and $\mathbf{z}_2$ are divided into coupling variables $(\mathbf{z}_1^c, \mathbf{z}_2^c)$ and noncoupling variables $(\mathbf{z}_1^{nc}, \mathbf{z}_2^{nc})$. Coupling variables of a discipline are input of the other discipline (e.g. $\mathbf{z}_2^c$ is an input variable vector of the first discipline). Note that if no coupling between analyzers exists, the MDO problem (3.1) returns to the ordinary design optimization problem (2.1).

3.3 MDO Methods

To solve an MDO problem we not only have to find the optimal value for $f(\mathbf{x})$ and satisfy the constraints (3.1c) and (3.1d), but we also need to satisfy (3.1e) and (3.1f). Because $\mathbf{z}_1$ and $\mathbf{z}_2$ are *interdependent*, they need to be solved iteratively, beginning with some initial given values that do not need to satisfy (3.1e) and (3.1f). Therefore, we have the following situation:

$$\mathbf{z}_1 = w_1(\mathbf{x}_1, \mathbf{x}_c, \mathbf{s}_2) \tag{3.2a}$$

$$\mathbf{z}_2 = w_2(\mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1) \tag{3.2b}$$

where $(\mathbf{s}_1, \mathbf{s}_2)$ are the initial "guesses" of coupling variable vectors $(\mathbf{z}_1^c, \mathbf{z}_2^c)$. At this stage, if the disciplines satisfy the constraints (3.1c) and (3.1d), the disciplines are said to have reached *single discipline feasibility* [20]. Furthermore, if all the disciplines of an MDO problem have reached single discipline feasibility, the situation is called *individual discipline feasibility*. The goal of the MDO method is to ultimately achieve *multidisciplinary feasibility*, which is attained when:

1. Each discipline has achieved individual discipline feasibility.
2. (3.1e) and (3.1f) are satisfied (i.e. the inputs correspond to the outputs of the other discipline).

In this situation the states of the coupled disciplines are in *equilibrium*. This entire process that aims to achieve multidisciplinary feasibility is called *multidisciplinary analysis* (MDA).

MDO methods can be divided into two categories: single-level and multilevel methods. Single-level methods generally have a single optimizer and directly use the inherent nonhierarchical structure of an MDO problem, whereas multilevel methods modify the nonhierarchical structure to a hierarchical structure. Each level of the hierarchical structure has an optimizer. The following seven MDO methods were presented in [3]:

- Single-level MDO methods

    - All-at-once (AAO) [20]
    - Individual-discipline-feasible (IDF) [20]
    - Multiple-discipline-feasible (MDF) [20]
    - Multidisciplinary optimization based on independent subspaces (MDOIS) [53]

- Multilevel MDO methods
    - Concurrent subspace optimization (CSSO) [63]
    - Bi-level integrated system synthesis (BLISS) [60]
    - Collaborative optimization (CO) [10]

In [71] the aforementioned seven MDO methods were benchmarked with two mathematical test cases, and the multilevel methods were proven overall inferior. However, the benchmarks were done sequentially with a single computer, even though disciplines of a multilevel method can be run parallel. Further details of the multilevel methods can be read from [71, 63, 60, 10]. Moreover, a comparison of MDF, IDF and AAO can be read from [38]

Note that the classification of *constraints* and the *objective function* to discipline specific components is only necessary with the multilevel methods and the MDOIS method, and is therefore redundant with AAO, IDF and MDF. Let us briefly go through the AAO, IDF and MDF methods.

3.3.1 Multiple-Discipline-Feasible

MDF is probably the simplest MDO method. It uses the basic formulation (3.1) of the MDO problem. During each optimization iteration, a system analysis is performed in order to achieve multidisciplinary feasibility. This is usually done by using the fixed point iteration method to achieve equilibrium for equations (3.1e) and (3.1f). This process converges to a mathematical optimum, and MDF is therefore used as a standard solution when MDO methods are compared [3].

MDF maintains multidisciplinary feasibility throughout the optimization process. This usually results in a large number of function calls, particularly if the couplings are complex. However, MDF is practical due to its simple formulation.

3.3.2 Individual-Discipline-Feasible

IDF uses an alternative formulation of the MDO problem. The coupling vectors $(\mathbf{z}_2^c, \mathbf{z}_1^c)$ of (3.1e) and (3.1f) are replaced by surrogate vectors, and the coupling relationship is maintained by setting new equality constraints [3]:

$$
\begin{aligned}
\text{Find } & \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_2^c, && (3.3a)\\
\text{to minimize } & f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c, \mathbf{z}_1, \mathbf{z}_2), && (3.3b)\\
\text{subject to } & \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}, && (3.3c)\\
& \mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0} && (3.3d)\\
& \mathbf{s}_1^c - \mathbf{z}_1^c = \mathbf{0} && (3.3e)\\
& \mathbf{s}_2^c - \mathbf{z}_2^c = \mathbf{0} && (3.3f)\\
& \mathbf{z}_1 = w_1(\mathbf{x}_1, \mathbf{x}_c, \mathbf{s}_2^c) && (3.3g)\\
& \mathbf{z}_2 = w_2(\mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c) && (3.3h)
\end{aligned}
$$

Multidisciplinary feasibility is not required during every optimization iteration. This drastically reduces the amount of needed function calls. Furthermore, analyzers of different disciplines do not need to wait for each other and they can be run simultaneously.

Individual discipline feasibility is achieved by doing simulations using the surrogate vectors $\mathbf{s}_1^c$ and $\mathbf{s}_2^c$. Optimization is being done on the surrogate vectors $(\mathbf{s}_1^c, \mathbf{s}_2^c)$ as well, and the equality constraints (3.3e) and (3.3f) should eventually force the optimization process to achieve multidisciplinary feasibility.

3.3.3 All-At-Once

Also called the Simultaneous Analysis and Design (SAND) method, the AAO method requires neither individual nor multidisciplinary feasibility during an optimization iteration. The whole system analysis is eliminated by converting the system analysis equations into equality constraints, and treating state variables as design variables. Therefore, the responsibility of multidisciplinary feasibility is passed on to the optimizer. The AAO method uses a following formulation [71]:

$$\text{Find } \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_1^{nc}, \mathbf{s}_2^c, \mathbf{s}_2^{nc} \tag{3.4a}$$

$$\text{to minimize } f(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_1^{nc}, \mathbf{s}_2^c, \mathbf{s}_2^{nc}), \tag{3.4b}$$

$$\text{subject to } \mathbf{g}(\mathbf{x}, \mathbf{s}) = \begin{bmatrix} \mathbf{g}^1(\mathbf{x}_1, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_1^{nc}) \\ \mathbf{g}_2(\mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_2^c, \mathbf{s}_2^{nc}) \\ \mathbf{g}_c(\mathbf{x}_1, \mathbf{x}_2, \mathbf{s}_1^c, \mathbf{s}_1^{nc}, \mathbf{s}_2^c, \mathbf{s}_2^{nc}) \end{bmatrix} \leq \mathbf{0}, \tag{3.4c}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{s}) = \begin{bmatrix} \mathbf{h}_1(\mathbf{x}_1, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_1^{nc}) \\ \mathbf{h}_2(\mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_2^c, \mathbf{s}_2^{nc}) \\ \mathbf{h}_c(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c, \mathbf{s}_1^{nc}, \mathbf{s}_2^c, \mathbf{s}_2^{nc}) \end{bmatrix} = \mathbf{0}, \tag{3.4d}$$

$$w_1(\mathbf{x}_1, \mathbf{x}_c, \mathbf{s}_2^c) - \mathbf{s}^1 = \mathbf{0} \tag{3.4e}$$

$$w_2(\mathbf{x}_2, \mathbf{x}_c, \mathbf{s}_1^c) - \mathbf{s}^2 = \mathbf{0} \tag{3.4f}$$

The downside is that this reformulated optimization problem is significantly more challenging than the basic MDO problem. In other words, although system analysis is not needed with AAO, the optimization problem itself becomes more difficult. Moreover, multidisciplinary feasibility is attained only at the absolute end of the optimization process, thus designs attained during the optimization process are not feasible [38].

3.4 Practical Implementation

The definitions can be generalized to more than two disciplines. In that case, more complex couplings are possible. Even with only two disciplines, the need for function evaluations is vastly greater in an MDO problem due to the increase in problem size (e.g. AAO, IDF) or due to system analysis (e.g. MDF). Therefore, it is common to use adaptive optimization algorithms that use low fidelity meshes in the early stages of the optimization and gradually increase the fidelity when nearing an optimum. It is also more worthwhile to spend time to obtain sensitivity information from an MDO problem to enable the use of gradient based algorithms.

An approach presented in [31] argues why it is worthwhile to use time and effort to create a response surface model, and then solve the optimization problem using the response surface model. Sensitivity analysis and function evaluations can then be made using the response surface model, and time consuming simulations are not needed. However, the construction of an accurate response surface model might be an equally challenging task as solving the actual MDO problem.

In practice, the complex nature of MDO problems usually causes their objectives to be conflicting, and thus MDO problems are usually multiobjective. The formulation of a single objective MDO problem is still useful, as most of the multiobjective methods ultimately return the MOO problem back to a single objective optimization problem [47]. We can have a total of four different types of design optimization problems. Their key properties are presented in table 3.1.

	Single Discipline
Single Objective	Single or multiple *nonconflicting* objectives Single or multiple *noninteracting* disciplines Has an unambiguous solution (not necessarily unique)
Multiobjective	Multiple *conflicting* objectives Single or multiple *noninteracting* disciplines Pareto front of optimal solutions

	Multidisciplinary
Single Objective	Single or multiple *nonconflicting* objectives Multiple *interacting* disciplines Has an unambiguous solution (not necessarily unique)
Multiobjective	Multiple *conflicting* objectives Multiple *interacting* disciplines Pareto front of optimal solutions

Table 3.1: Characteristics of different types of design optimization problems.

4 CAD-CAE Interoperability

4.1 Multidisciplinary Requirements

The MDO problem was presented as a coupled problem, but organizational difficulties of having multiple disciplines in an optimization platform are present even if the disciplines are not coupled. In practice, different disciplines operate on different meshes, and there is no guarantee that mesh nodes exist in identical locations. Therefore, to obtain multidisciplinary analysis information on a single point in Ω, interpolation routines are needed. Furthermore, different analyzers and CAD system have their strengths and weaknesses, and the use of various CAD systems and/or analyzers in a single organization is not uncommon. Data transfer among CAD systems and analysis softwares (i.e. downstream applications) is problematic.

File standards have been created, but they are deficient for the needs of MDO. In order for the interpolation routine to function with at least some accuracy, it is crucial that the underlying B-rep is identical among downstream applications. Additionally, the requirements for the geometry vary according to discipline. For example, a complete CAD model of a car is not necessary (and in fact is too complicated) for aerodynamic simulations. Only the exterior is needed. However, in crash test simulations the interior is essential, whereas some small details of the exterior that are significant for aerodynamics might have a nonexistent impact on the crash test simulations (e.g. small chamfers).

Traditionally, different CAD models have been used for the analysis of each discipline. It is hard, if not impossible, to do multidisciplinary analysis if the underlying CAD models are different. However, if the CAD model is constructed correctly, the CAD model can be stripped down by suppressing some of its features to specifically suit a discipline. Using such a CAD model it is possible to use a single CAD model for all the disciplines.

4.2 CAD System Geometry Representation

Traditionally data between CAD and CAE applications has been transferred via the neutral IGES, STL or STEP file formats, or through proprietary formats [9]. This approach is referred to as the CAD-centric approach [43] (illustrated in figure 4.1). The

IGES file format stores geometry in form of separate curves and surfaces, whereas STL is a discrete representation of the geometry consisting of only faces, neither of these two file formats store topological information. The STEP file format supports topology as well as geometry, and is therefore a preferable format over IGES and STL.

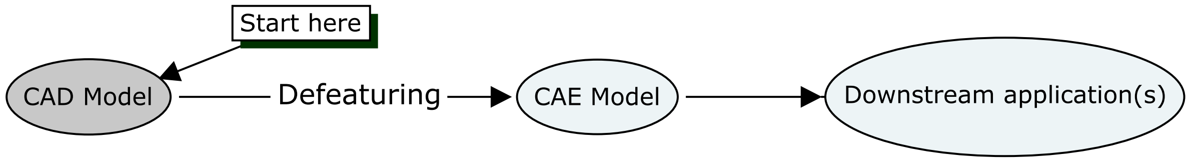

Figure 4.1: The CAD-centric approach to CAD-CAE integration.

CAD systems that support solid modeling have a properly closed solid representation of the geometry with a defined topology. As such, it should be a sufficient representation for the needs of computational analysis. However, closed solid objects of CAD systems are actually only "almost" closed. Nodes bounding an edge or edges bounding a face do not necessarily sit on each other [33]. They are only required to be inside a certain tolerance in order for them to be interpreted as being connected. Moreover, this tolerance is different for different CAD systems, and it is generally much larger than the precision of double precision floating point arithmetic.

When CAD geometry is saved to a file and imported to another application, the solid objects that used to be interpreted as being closed often have gaps because of the tolerances in CAD systems geometry representation. To create a proper volume mesh, the mesher requires a closed boundary representation. To achieve that, the gaps in the geometry need to be repaired. Therefore, it is necessary to deduce the topology of the geometry to identify the gaps, and then patch them up. This process is described in [5, 69]. The patching process usually introduces "sliver" surfaces that are problematic for meshers [33]. Because of these "sliver" surfaces, the mesh needs to be interactively repaired so that it is of sufficiently high quality for the analysis. This is clearly out of the question if we are planning on doing automated design optimization with hundreds of simulations.

For historical reasons, modern commercial CAD systems create a dual model, consisting of a procedural shape model and a secondary boundary representation model [56]. The procedural model is a set of instructions on how to construct the CAD model from a set of primitives using operations and features such as extrusions, revolutions, holes, lofting, chamfers, fillets, etc., whereas the B-rep model is a snapshot of the procedural model. It only records the resulting geometry, topology and tolerances, while details of the construction process are lost.

The B-rep model is easier to read and interpret as it is an explicit representation of the geometry, but the downside is that it is very hard to change a B-rep model after it has been created. To keep the B-rep model up to date, the old B-rep model is discarded and a new one is created every time the procedural model is changed. The B-rep model is used mostly for visualization and geometric computations. Usually this "dumb" B-rep model is the one used when exporting the CAD model for the use of computational analysis. This is unfortunate because in design optimization new variations of the geometry are needed constantly. There have been various attempts to overcome the problems created by CAD system tolerances and "dumb" B-rep models.

One approach is to initially eliminate CAD from the analysis by creating the geometry in the CAE application, and export the geometry to the CAD system when the analysis is complete. This approach is referred to as the CAE-centric approach [43] (illustrated in figure 4.2). The problem with this approach is that usually the geometry creation tools of CAE application are vastly inferior to the ones of a CAD system. In order to construct a fully functional CAD model based on a CAE model, one needs to create it from scratch or use semi-automatic (detail addition) conversion tools. Even if this could be done automatically, it involves updating and maintaining two separate models.

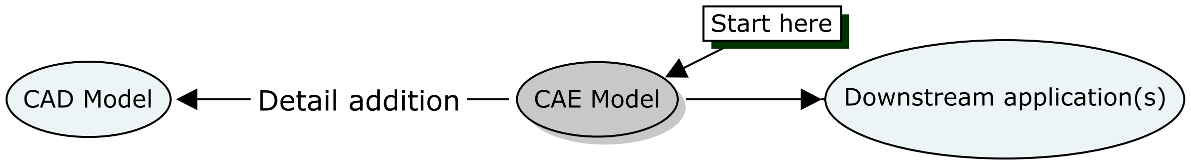

Figure 4.2: The CAE-centric approach.

An extension to the STEP file format standard introduced in [56] proposes the addition of complementary generative model information to the STEP file format based on the theory by Leyton [44]. The article claims that the STEP file format could be retrofitted to interpret the low level procedural shape representation presented by Leyton as higher level construction operators, and could therefore be used for a formal *generative* shape representation. However, this technology is described only at a conceptual level and only a proof of concept demonstration is presented. Moreover, before it could be considered as a viable solution, the majority of CAD systems would need to adopt the file format, and additionally they should also strictly conform to the standard, as historically this has been a problem with the STEP standard [56].

4.3 Direct Coupling to CAD Systems

Some analysis suites couple directly to the CAD system [33]. First, this allows the surface discretization procedure to communicate directly with the CAD system and verify that the resulting surface discretization is a close match to the CAD model (i.e. does not contain unwanted gaps and follows the geometry accurately). Second, it makes available the use of the high-level CAD geometry representation, and therefore modifying features and parameters is possible. These two factors facilitate the modification of the CAD model and eliminate the need of interactive repairing, and thus enable the construction of integrated CAD-CAE design optimization platforms.

The coupling can be done in two ways: originating from the CAD system or from the downstream application. Unfortunately, analysis initiated by the CAD system is simply too limiting for the needs of MDO. Modules would have to exist for all of the desired downstream applications. Moreover, we want the control of the analysis process to be maintained by the optimization platform. Usually this is not possible when the analysis is initiated from a CAD system.

Modules that couple from the downstream application to the CAD system such as the Comsol CAD Import Module [18], are better suited for design optimization, but from an MDO perspective, it is still not an ideal solution. Multiple downstream applications would need to have a CAD import module for the used CAD system (or in worst case, multiple CAD systems). Furthermore, the modules most likely would not interpret the geometry in an identical manner, and thus would create a problem of inconsistency. To address this problem, a neutral solution would be needed, that would translate geometry identically regardless of used CAD systems and analysis softwares.

Several existing projects are developing or have developed CAD neutral Application Programming Interfaces (API). These are the Common Geometry Module (CGM) [65], Simulation Environment for Engineering Design (SEED) [58], Computational Analysis PRogramming Interface (CAPRI) [13], GeomSim [30] and OMG CAD Services [15]. The common goal for these projects is to serve as an intermediate between CAD systems and downstream applications. Regardless of CAD system, the neutral API should provide the CAD topology and geometry to all its clients in an identical manner. Furthermore, direct coupling allows the modification of parameters and defeaturing of the CAD model. In [6] CAD neutral APIs were shown to be the most flexible approach to access CAD geometry for analysis.

However, although direct coupling allows access to the CAD system, the low-level geometry construction of CAD systems differs. Direct construction of geometry through an API is not always possible. Only the CAD systems that are based on geometry kernels (and allow the use of the kernel) can perform direct construction (Parasolid, ACIS, OpenCASCADE, Granite) [33]. For design optimization purposes, direct construction is not generally required. It is sufficient if one can change shapes and suppress features. Usually the API connects to the Master Model [36] of the CAD system to offer a CAD vendor neutral approach.

The Master Model is basically a tree of features used in the construction of a solid. It allows the suppression of features and modification of exposed parameters of the CAD model. After the Master Model has been modified, the CAD model can be regenerated to reflect the changes. By using the Master Model, a family of geometries can be generated from a single CAD model. Therefore, the discipline specific geometry requirements of MDO problems can be met. The Master Model architecture is supported by all major CAD systems. GeomSim on the other hand uses a combined approach by offering access to the Master Model and additionally to the geometry kernel whenever available. Unfortunately, GeomSim does not currently support Catia.

4.4 CAPRI

The CAPRI [13] CAD neutral API is used in the optimization platform of our research group. Instead of the usual CAD-centric or CAE-centric approaches to CAD-CAE integration, CAPRI uses a geometry centric approach. CAPRI's approach to CAD-CAE integration is illustrated in figure 4.3. For another geometry centric approach see [43].

CARPI has its own geometry definition, which combines geometry and topology. CAPRI makes accessible the actual geometry (instead of a discretized version) to all its clients. The geometry that resides in the CAD system is still considered as the "correct" geometry, CAPRI merely offers a *common* interface to the geometry. The geometry representation is therefore consistent among all downstream applications.

In practice, the geometry representation of CAPRI has to be converted to a format that the mesher(s) can read. It is up to the user of CAPRI if he or she wants to write an interpreter to convert the CAPRI's B-rep to the mesher's B-rep. However, most meshers can use discretized geometries as their input (such as STL). For this

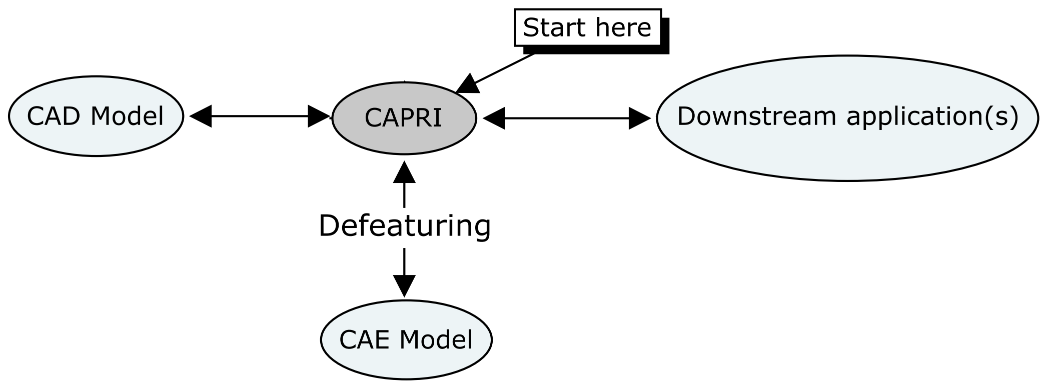

Figure 4.3: CAPRI's (geometry centric) approach to CAD-CAE integration.

purpose CAPRI includes a surface triangulator that can create a discretization of the geometry. CAD systems can often save their geometries in such a format as well. Moreover, analysis suites that utilize direct coupling can simultaneously provide the actual geometry and its discretized version. CAPRI provides the same, but the difference is that CARPI's representation is *identical* to all downstream applications, regardless of CAD system. The discretization of CAPRI is also said to be high quality and "watertight" [33].

CAPRI is intended to be used from within a (multidisciplinary) analysis suite, and for this purpose it supports the C and FORTRAN programming languages. An example CAPRI application written in C can be seen from appendix A. Note that when utilizing CAPRI in a program, the used CAD system always needs to be specified. Currently CAPRI supports the following CAD systems:

- Catia V5
- OpenCASCADE
- Parasolid
- Pro/ENGINEER
- SolidWorks
- UGS UG NX

As an additional feature, CAPRI offers a client/server architecture. CAPRI server is located on a computer server along with the CAD system. Computers that do simulations have the CAPRI client component and connect to the server when they need to make geometry queries. This approach has several benefits when dealing with large computational analysis platforms. First, it allows the centralization of the CAD systems so that cumbersome CAD installations are not needed on every workstation, and thus CAD licenses are required for the servers only. Second, if the simulations are done in a distributed environment, the geometry definition can have a central repository. For an optimization example using CAPRI see [29].

4.5 Summary

Direct coupling is a decent approach to solve the CAD-CAE interoperability issue, and can be recommended for analysis where only one solver is needed. Assuming that the direct coupling is implemented well (e.g. no geometry exchange issues). However, if an MDO platform is being built with multiple solvers, it is critical that the B-rep can be provided to all solvers identically because of the interpolation necessary for the interdisciplinary mappings. In such a case a neutral API such as CAPRI is recommended. The downside to CAPRI can be said to be the fact that it needs an additional license, but that is usually the case with other coupling solutions as well. CAPRI can also be said to be a bit hard to use, as it requires programming. However, it can also be seen as an advantage because it allows exact control of the geometry exchange. To conclude the chapter and to compare different geometry exchange method a summary of the pros and cons of different geometry is presented below:

- File based explicit boundary representation (STL, IGES, STEP)

  Pros:

  + Highly standardized
  + No license required

  Cons:

  – Can contain gaps and overlaps (i.e. “dirty” geometry)
  – Hard/Impossible to parameterize (needs to be recreated every iteration)

- Generative file based representation (proposed new STEP)

  Pros:

  + Easier to modify than explicit file formats
  + No license required

  Cons:

  – Not ready for use

  - – Has to be adopted by major CAD systems
  - – Probably has still unknown issues

- Coupling from CAD to solver

  Pros:

  - + Access to high level CAD features and procedural representation of the geometry

  Cons:

  - – Control of the analysis process usually needs to be maintained by the CAD system
  - – Relies on modules that likely do not exist for all needed solvers

- Comsol CAD Import bidirectional interface (Coupling from solver to CAD)

  Pros:

  - + Able to verify the topology from the CAD system
  - + Access to the Master Model

  Cons:

  - – Only works with SolidWorks (with other CAD systems operates like traditional file transfer)
  - – Relies on a module that likely does not exist in other solvers
  - – Even if a module would exist in each solver there is no guarantee that they would interpret geometry from the CAD system in an identical manner (inconsistency)

- CAPRI

  Pros:

  - + Access to the Master Model
  - + Able to verify the topology from the CAD system
  - + Modules exist for all major CAD systems
  - + Able to provide a common B-rep for all downstream applications (consistent)
  - + No geometry translation

  Cons:

  - – License required for all used CAD systems and CAPRI itself

5 Design Optimization Platform

5.1 Logical Structure and Data flow

Our research group has implemented a functioning design optimization platform. The components of the optimization platform are:

- The CAD system and CAPRI
- Mesher
- Solver
- ModeFRONTIER

The CAD geometry is read from the Catia V5 [14] CAD system with the CAPRI API. CAPRI is then used to defeature and prepare the geometry for analysis, and ultimately to create a discretization of the geometry. The discretization is then passed to the mesher. The volume mesh is created with Tetgen [66]. Comsol Multiphysics [19] or Elmer [26] is used to solve the BVP. ModeFRONTIER [48] controls the optimization process.

The optimization system is launched from the Graphical User Interface (GUI) of ModeFRONTIER. ModeFRONTIER allows the creation of a flow chart that is used to control the optimization. The optimization problem is defined in the flow chart using ModeFRONTIER components such as: input file, output file, external script, variable, target and constraint. The flow of the optimization is controlled with this flow chart by logical expressions. An example flow chart can be seen in figure 5.1. Basically, it works like batch processing with a GUI. However, ModeFRONTIER offers some additional features that are very valuable, like plotting the history of solutions real-time in various different ways, and analyzing the optimization results.

The components can be distributed over several workstations/servers. Currently ordinary workstations are used to run all the software, but the plan has been to move the solver and mesher to a server with more computational power in the near future. Currently ModeFRONTIER and the solver are running on their exclusive workstations, and the rest (CAPRI, CAD, Mesher) are running on a third workstation. There were plans to have the CAD system run on its own exclusive

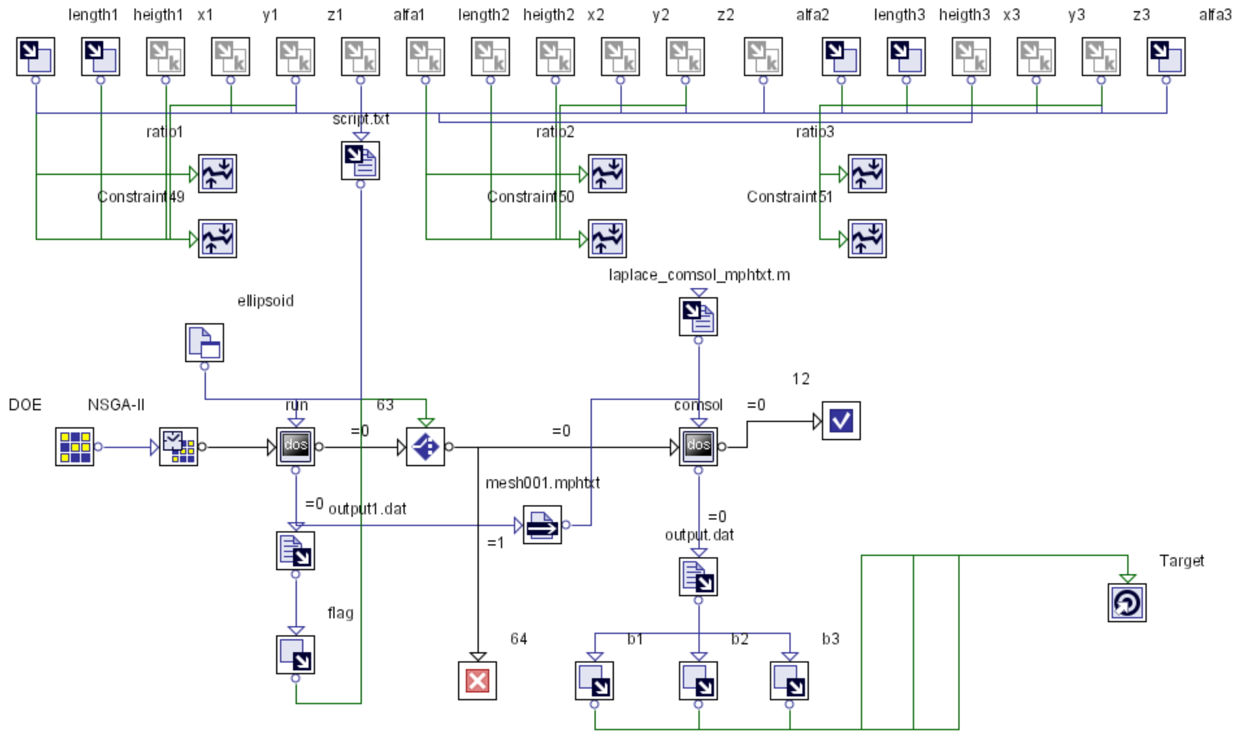

Figure 5.1: ModeFRONTIER Flow Chart.

workstation, which should be easily achieved by using the client/server feature of CAPRI.

The distribution of the components on different workstations is practical, if different people work on different parts of the system. For example, not all of the workstations need to have all of the licenses, and files do not need to be moved back and forth. The distribution does not help speed up the actual analysis process in any way as it is still run sequentially. Parallel or distributed computing can be useful in other circumstances. For example, with genetic algorithms (GA) we can divide the population to several subpopulations and solve them simultaneously [11]. Also the plan has been to exploit parallel computing in the future with multidisciplinary problems.

The data flow of the system is illustrated in figure 5.2. The figure shows an idealistic version, in which CAPRI has been separated to CAPRI client and server, even though currently this is not the case in our system. We have CAPRI running on only one workstation and so no client/server methodology is needed.

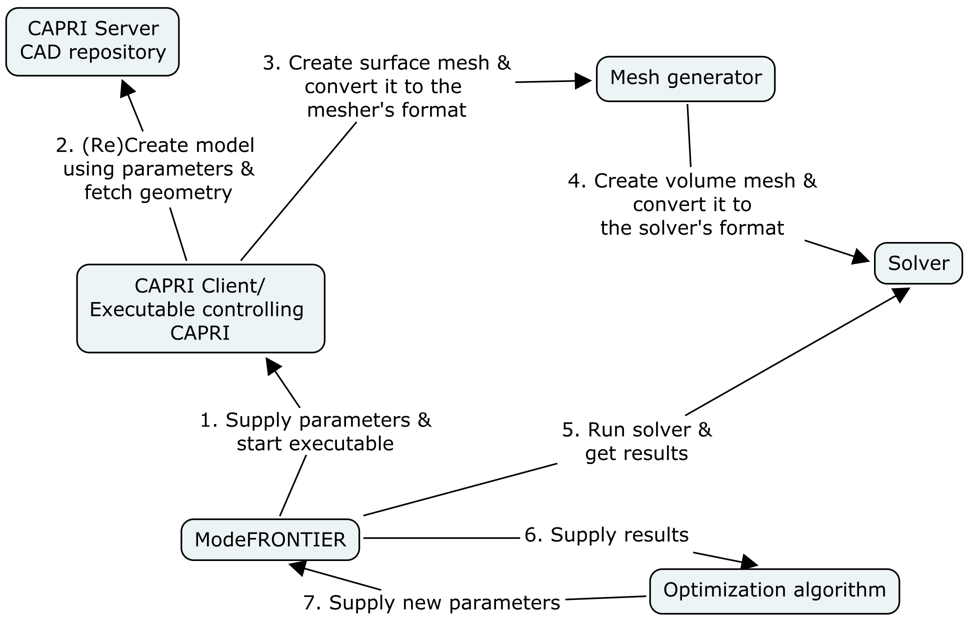
CAPRI Server
CAD repository
3. Create surface mesh & convert it to the mesher's format
Mesh generator
2. (Re)Create model using parameters & fetch geometry
4. Create volume mesh & convert it to the solver's format
Solver
CAPRI Client/ Executable controlling CAPRI
1. Supply parameters & start executable
5. Run solver & get results
ModeFRONTIER
6. Supply results
7. Supply new parameters
Optimization algorithm

Figure 5.2: Data flow of the system.

5.2 Detailed Process Description

The optimization process is iterative and during each iteration we are roughly:

1. (Re)constructing a CAD model with Catia V5 via CAPRI using parameters given by the optimizer
2. Constructing a surface mesh with CAPRI
3. Building a volume mesh from the surface mesh
4. Solving the problem with the volume mesh
5. Using an optimization algorithm to determine new parameters for the CAD model to try to find a better solution

5.2.1 Reconstruction of the CAD model

In this phase CAPRI is instructed to open the CAD model and to create a new model suitable for analysis. Usually this involves at least the creation of a bounding box, cutting the model out of the bounding box and setting boundary conditions. Because this process can be different for each optimization problem, the program code usually needs to be modified. Setting boundary conditions could be done as early as in the CAD system, and this way we could reduce the amount of manual modification of the code.

This can be done by first setting boundary conditions to the CAD model in Catia V5 as properties. Then they could be read, parsed and interpreted as boundary conditions in CAPRI. This is something that was considered for a while, but the idea was abandoned when it was realized that the other manual modifications need to be made regardless (e.g. creation of a bounding box or suppression of features, etc). These modifications will add new or change existing boundaries of the model, so setting boundary conditions using Catia V5 is not always possible.

5.2.2 Construction of the surface and volume meshes

In this phase the model generated by the previous phase is opened and boundary condition numbers are assigned to its boundaries. TetGen is then used to create the volume mesh. TetGen accepts a variety of parameters, so a balance of accuracy and performance for the mesh can usually be found by tweaking these input parameters. Although, the mesher is very easy to change if another mesher is needed.

First, CAPRI is called to open the CAD model and boundary conditions are assigned to it. Then CAPRI is instructed to create discretization of the geometry. The discretization is then is stored in memory in TetGen format (piecewise linear complex) with boundary conditions as additional information. Now we have a surface mesh presentation of the CAD model and we use it to create the volume mesh with TetGen. The completed mesh with its boundary conditions is then stored in memory. Finally, depending on the solver, we usually need to convert the mesh to its respective file format. In this case we are going to convert the mesh to either COMSOL's or Elmer's native file format.

5.2.3 Solving the State Problem

This phase is totally dependent on the solver we want to use. Generally, we want to set up the problem and import the mesh with its boundary conditions. In order to solve the problem automatically, we need to have some way to run the solver without user interaction. So far two solvers have been used in our optimization platform: Elmer and Comsol. Practically any solver can be used, providing that it can be run without user interaction.

Comsol is an interactive multiphysics mesher-solver-analyzer. Comsol has a script interface (Comsol Script) which is used to run the solver without user interaction. Using the script file the mesh is imported along with boundary conditions, and then a problem setup is loaded from a template file. The script file then commands Comsol to solve the problem, and the information needed for the objective function evaluation are written to a file. A sample script file can be seen from appendix B.

Elmer is also a mesher-solver-analyzer whose solver component was found to be very capable. However, Elmer did not provide such readily available tools for integration. Therefore, Comsol was used in the actual design optimization test case. Elmer is command-line driven and thus it is easy to have it run without user interaction.

5.2.4 Optimization and analyzing

In this phase the fitness of the design is analyzed based on the results given by the solver. As usually in optimization, we want to have some rules to determine when we want to stop the optimization. For example, we can have either maximum number of iterations or some tolerance for the solution, or both. Also, we need some way of choosing a new set of parameters for the parameterized CAD Model. The amount of parameters, objective functions and the nature of the objective function will dictate the optimization methods we can use. ModeFRONTIER provides some ready-to-use optimization algorithms, but naturally additional algorithms can be imported as well.

If the tolerance or maximum numbers of iterations has not been reached, the optimization algorithm gives a new set of parameters, which will be sent to CAPRI. ModeFRONTIER then calls CAPRI to remodel the CAD model with the new parameters. This way the process is started from the beginning, and will be continued until the tolerance or maximum iterations is reached. The performance of the ongoing optimization can be monitored from ModeFRONTIER.

6 Test Runs and Results

6.1 Test Case Description and Definition

The domain of the test case is a solid box of dimensions $2000 \times 2000 \times 2000$ with three ellipsoidal cavities. The three ellipsoids are parameterized by six parameters each (x, y, z, length, height, angle). The parameterization is illustrated in figure 6.2. The original plan was that to do CFD with the platform at some point, and that is why a geometry and parameterization was used that could be used with CFD analysis as well. The domain is illustrated in figure 6.1.

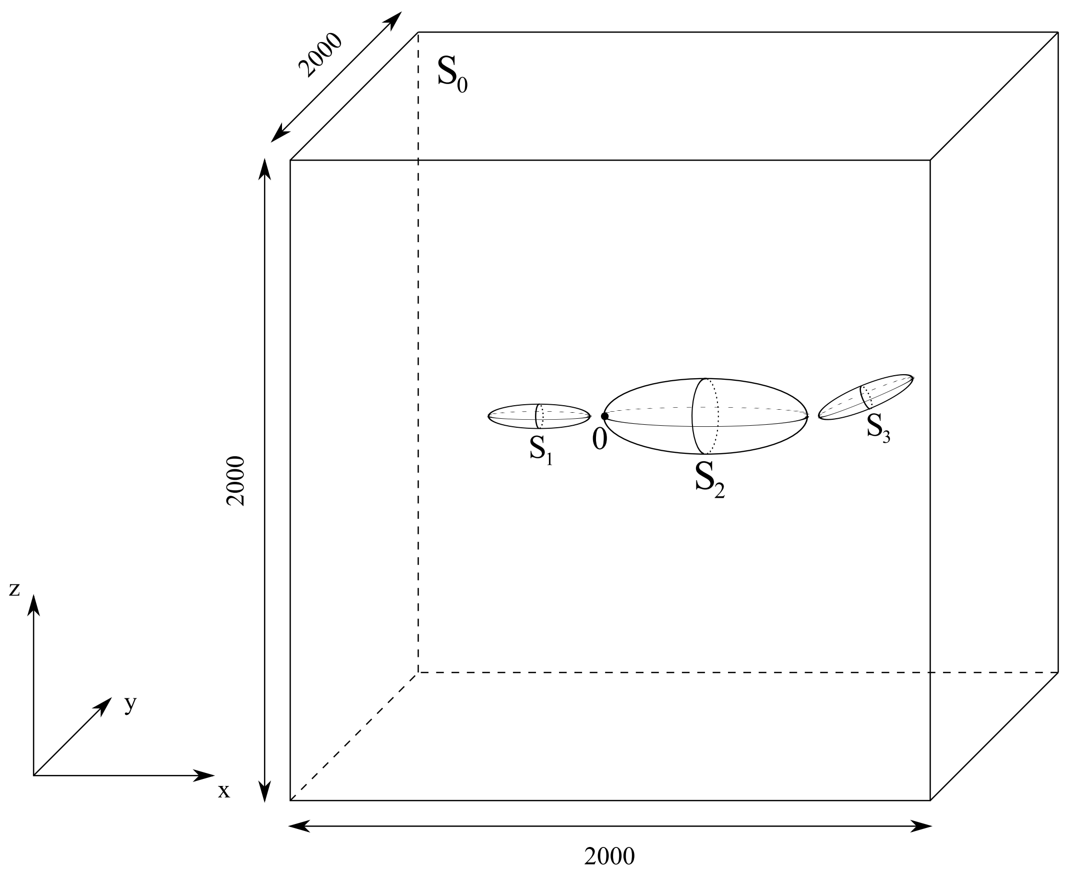

Figure 6.1: The domain Ω and its boundaries S_0, S_1, S_2 and S_3.

However, CFD analysis in three dimensions is quite challenging, so with the hardware that was available, a less computationally expensive problem was chosen to start out with. Therefore, the Laplace's equation was chosen, which the used workstations can solve with a direct solver in less than a minute.

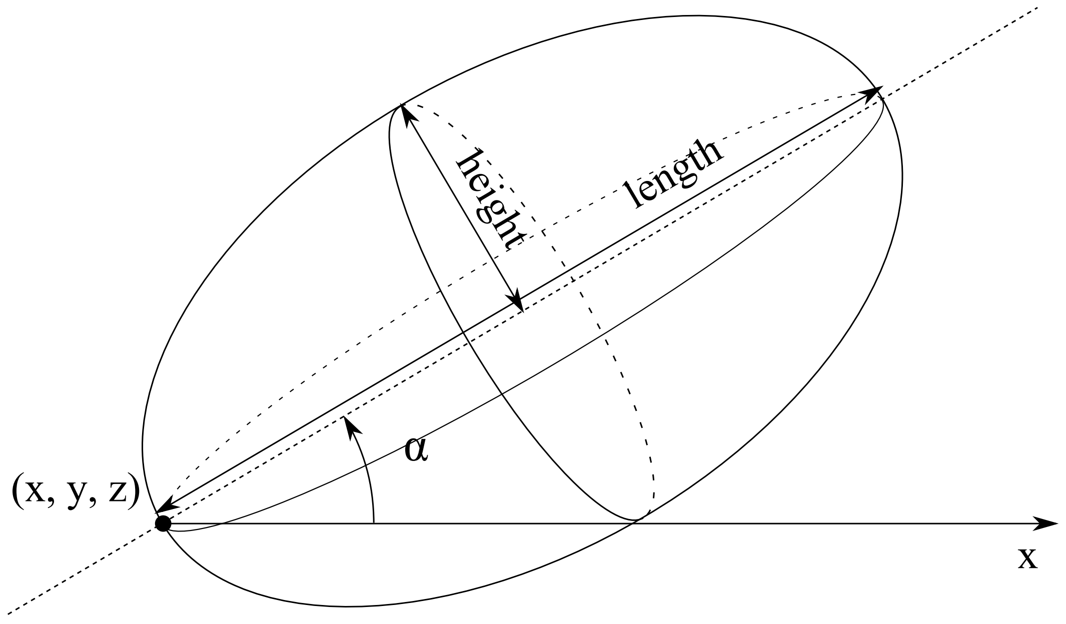

Figure 6.2: Parameterization of a single ellipsoid.

Let S_0 be the boundary of the solid box and S_1, S_2 and S_3 the boundaries of the three ellipsoids respectively. We formulate the state problem (or BVP) as follows:

$$\begin{aligned} &\text{Find } \mathbf{u} \text{ such that,} \\ &\Delta\mathbf{u}(x,y,z) = \mathbf{0} \text{ in } \Omega, && (6.1a) \\ &\mathbf{u}(x,y,z) = x \text{ on } S_0, && (6.1b) \\ &\nabla\mathbf{u}(x,y,z)\cdot\mathbf{n} = \mathbf{0} \text{ on } S_1, S_2, S_3 && (6.1c) \end{aligned}$$

where **n** is the outward normal (at (x,y,z)) and Δ is the Laplace operator ($\Delta = \nabla\cdot\nabla = \nabla^2$). The equation (6.1a) is called the Laplace's equation and the boundary conditions (6.1b) and (6.1c) are called the Dirichlet and Neumann boundary conditions respectively.

One physical interpretation of (6.1) could be: fix the temperature on the boundary and wait until the temperature of the interior does not change anymore. The temperature distribution in the interior will then be given by the solution of the BVP. The boundaries S_1, S_2 and S_3 can be seen as thermally insulated obstructions in the interior.

The solution of (6.1) also gives us the flow potential of a nonviscid, irrotational, incompressible flow. In this case (6.1b) would define the direction and magnitude of the flow, and (6.1c) specifies that the flow cannot enter the ellipsoids nor fluid is allowed to emerge from the ellipsoids. The velocity field of the flow can be calculated from $v = \nabla \mathbf{u}$.

Now that we have defined the state problem, we can define the design problem. For simplicity, let us restrict the amount of design variables to only 5 of the total 18 parameters, namely length($=: l_1$) and height($=: h_1$) of the first ellipsoid, and length($=: l_3$), height($=: h_3$) and angle($=: \alpha_3$) of the third ellipsoid. The second ellipsoid will therefore remain constant, and also the positions of the ellipsoids will not be changed. For simplicity, the bounds of the variables were set as follows:

$$\begin{aligned} l_1 &\in [28, 32] \\ h_1 &\in [4, 6] \\ l_3 &\in [48, 52] \\ h_3 &\in [4, 6] \\ \alpha_3 &\in [0.4, 0.6] \end{aligned}$$

and therefore the design space is:

$$U^{ad} = [28, 32] \times [4, 6] \times [48, 52] \times [4, 6] \times [0.4, 0.6] \subset \mathbb{R}^5 \tag{6.2}$$

Naturally, to test the optimization platform, we want to have a design problem to which we already know an optimal solution. First, the state problem (6.1) was solved with the default values of:

$$(l_1, h_1, l_3, h_3, \alpha_3)^T = (30, 5, 50, 5, 0.5)^T =: \mathbf{x}^*$$

and then from the numerical solution the boundary integrals of the expression $(1 - |\nabla \mathbf{u}|)^2$ over S_1, S_2 and S_3 were calculated. Let us define this as the operator $b_i(\mathbf{x})$:

$$\int_{S_i} (1 - |\nabla \mathbf{u}|)^2 dS_i =: b_i(\mathbf{x}) \tag{6.3}$$

The following results were obtained:

$$\begin{aligned} b_1(\mathbf{x}^*) &= 12.6019 \\ b_2(\mathbf{x}^*) &= 537.186 \\ b_3(\mathbf{x}^*) &= 36.867 \end{aligned}$$

We want the design $\mathbf{x}^*$ to be optimal, so let us devise a reconstruction problem by formulating the objective function as follows:

$$f(\mathbf{x}) = (b_1(\mathbf{x}^*) - b_1(\mathbf{x}))^2 + (b_2(\mathbf{x}^*) - b_2(\mathbf{x}))^2 + (b_3(\mathbf{x}^*) - b_3(\mathbf{x}))^2 \tag{6.4}$$

No additional constraints are set on the optimization, so the test case design optimization problem is simplified to:

$$\begin{aligned} \text{Find } \mathbf{x} = (l_1, h_1, l_3, h_3, \alpha_3) \in U^{ad} \\ \text{to minimize } f(\mathbf{x}), \end{aligned} \tag{6.5}$$

The objective function was defined $f(\mathbf{x})$ using boundary integrals of the expression $(1 - |\nabla \mathbf{u}|)^2$. What does this expression mean intuitively? We can visualize it by realizing that if we would not have any obstructions inside the box, $\nabla \mathbf{u}$ would be $(1, 0, 0)$ with all $\mathbf{u} \in \Omega$. We can think this as a flow toward the positive x-axis. More importantly, if there would be no obstructions, $|\nabla \mathbf{u}|$ would be identically 1. Therefore, the expression $(1 - |\nabla \mathbf{u}|)^2$ would be identically 0.

Now if we think about adding an obstruction inside the box, the boundary condition (6.1c) forces the gradients on the boundary to be *orthogonal* to the outward normal $\mathbf{n}$. We can expect that on locations of the boundary where the angle between the x-axis and $\mathbf{n}$ is small, the value of $|\nabla \mathbf{u}|$ will largely differ from 1. A boundary plot seen in figure 6.3 seems to conform to this idea. Therefore, $(1 - |\nabla \mathbf{u}|)^2$ was chosen to be a heuristic to measure the disturbance an object creates in the flow potential. All that really matters is that this quantity is dependent on the shapes of the ellipsoids so that the shape optimization process has some sense.

Now we are ready to do some test runs using this problem. We are of course hoping that the optimization platform would find the optimal solution $\mathbf{x}^*$. However, there is no guarantee that this is a unique solution. First, let us go through the practical implementation of the test case.

6.2 Practical Implementation

To create the geometry a CAD model of a single ellipsoid was created. This model was illustrated earlier in figure 2.3. Then the following steps are executed (using CAPRI) to create the geometry for analysis:

1. Create a Catia V5 model with a solid box of dimensions $2000 \times 2000 \times 2000$

2. Open a connection to the ellipsoid model and its Master Model

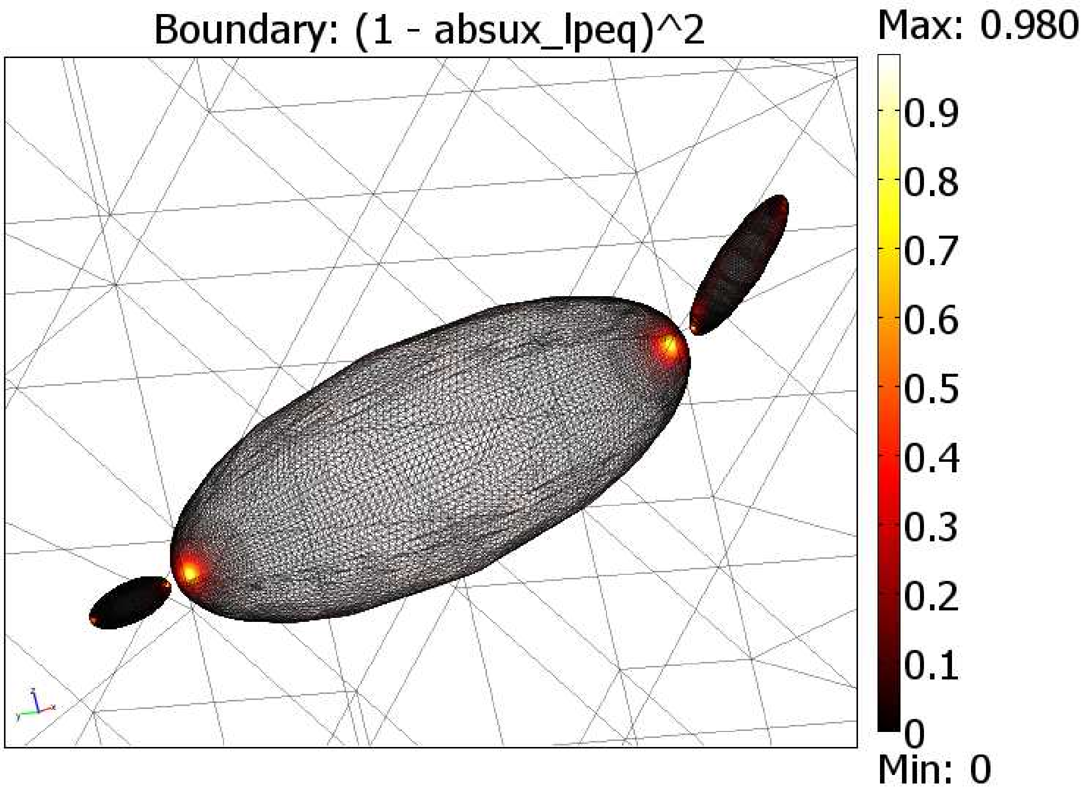

Figure 6.3: Boundary plot of the expression $(1 - |\nabla \mathbf{u}|)^2$.

3. Create an ellipsoidal cavity inside the box model by following these steps:
   (a) Change the values of length and height parameters of the ellipsoid model via the Master Model
   (b) Regenerate the ellipsoid model
   (c) Create a displacement/rotational matrix using parameters $(x_i, y_i, z_i, \alpha_i)$[1]
   (d) Apply the matrix on the ellipsoid)
   (e) Subtract the (current) ellipsoid from the box model (Boolean difference)
4. Repeat step 3 to create two more cavities
5. Save the box model geometry (box with three holes) to a Catia V5 file

Next, the procedure that was described in the previous chapter is followed to solve the state problem (6.1) with the current geometry. TetGen was given an addi-

[1]In this test case (x_i, y_i, z_i) are constants

tional quality parameter $q2.2$ (minimum radius-edge ratio 2.2). A typical mesh had around 70 000 elements and the solution of the state problem usually took around 50 seconds with the UMFPACK direct solver. From the solution the boundary integrals $b_1(\mathbf{x}), b_2(\mathbf{x}), b_3(\mathbf{x})$ were then calculated using a Comsol script file. Finally, the fitness of the design was evaluated in ModeFRONTIER by calculating (6.4). A sample solution can be seen in figure 6.4.

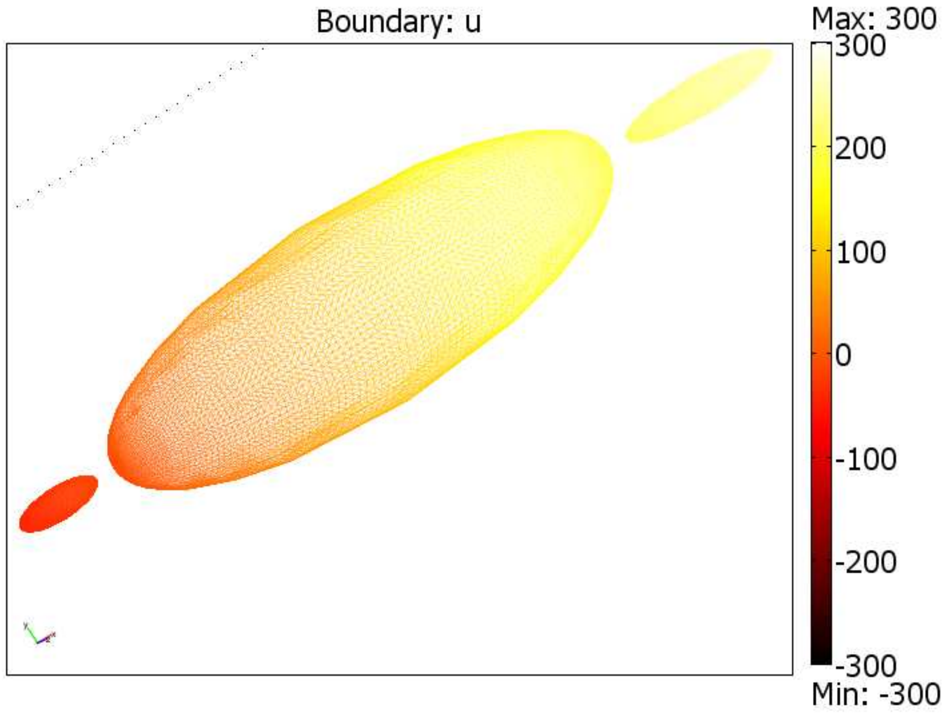

Figure 6.4: A solution visualized.

6.3 Optimization Runs

A total of three optimization runs were done with a different optimization algorithm in each run. The algorithms used were the Nelder–Mead (or simplex) method [51], a combined simplex-simulated annealing algorithm [16] and the Non-dominated Sorting Genetic Algorithm II (NSGA-II)[22]. As there was no information about the numerical accuracy of the solution, the runs were done with a fixed number of iterations. For the third run the ellipsoid CAD model was slightly modified, so unfortunately it is not directly comparable to the first two.

There is apparently some bug in TetGen, Comsol or the conversion code from TetGen to Comsol. In some runs the geometry was missing a single tetrahedron. Comsol could still read the geometry, but it was corrupted. Fortunately this was not very common (only 5-10 designs per run). The computations were done on a PC with a 2.8 GHz Pentium 4 processor.

6.3.1 Optimization Run 1: The Nelder–Mead method

Nelder-Nead was the only fully deterministic algorithm used and it converged with about 50 function evaluations (see figure 6.6). Interestingly, the Nelder–Mead method converged very close to the predefined optimal design $\mathbf{x}^*$ (see figure 6.5). The initial simplex was generated randomly so it might be just a coincidence, or more likely because of the fact that $\mathbf{x}^*$ is at the geometric center of the design space.

Also an interesting observation was made from this test run. Once the algorithm had done about 150 iterations, the changes the algorithm made to the design variables were minimal. At this point many of the designs started to give exactly the same objective function value. It therefore seems that the floating point precision of Catia V5 was reached as 4 out of the 5 design variables were implemented via the CAD system (lengths and heights).

Optimization Run Summary	
Number of Designs	159
Number of Feasible Designs	154
Number of Error Designs	5
Total Run Time	$2h\ 21min$

	Design id	l_1	h_1	l_3	h_3	α_3	$f(\mathbf{x})$
Best Design	106	30.079	5.099	48.998	4.979	0.4989	0.3276
Worst Design	3	28.024	5.927	51.759	5.894	0.5874	414.5

Table 6.1: Optimization Run 1 Summary.

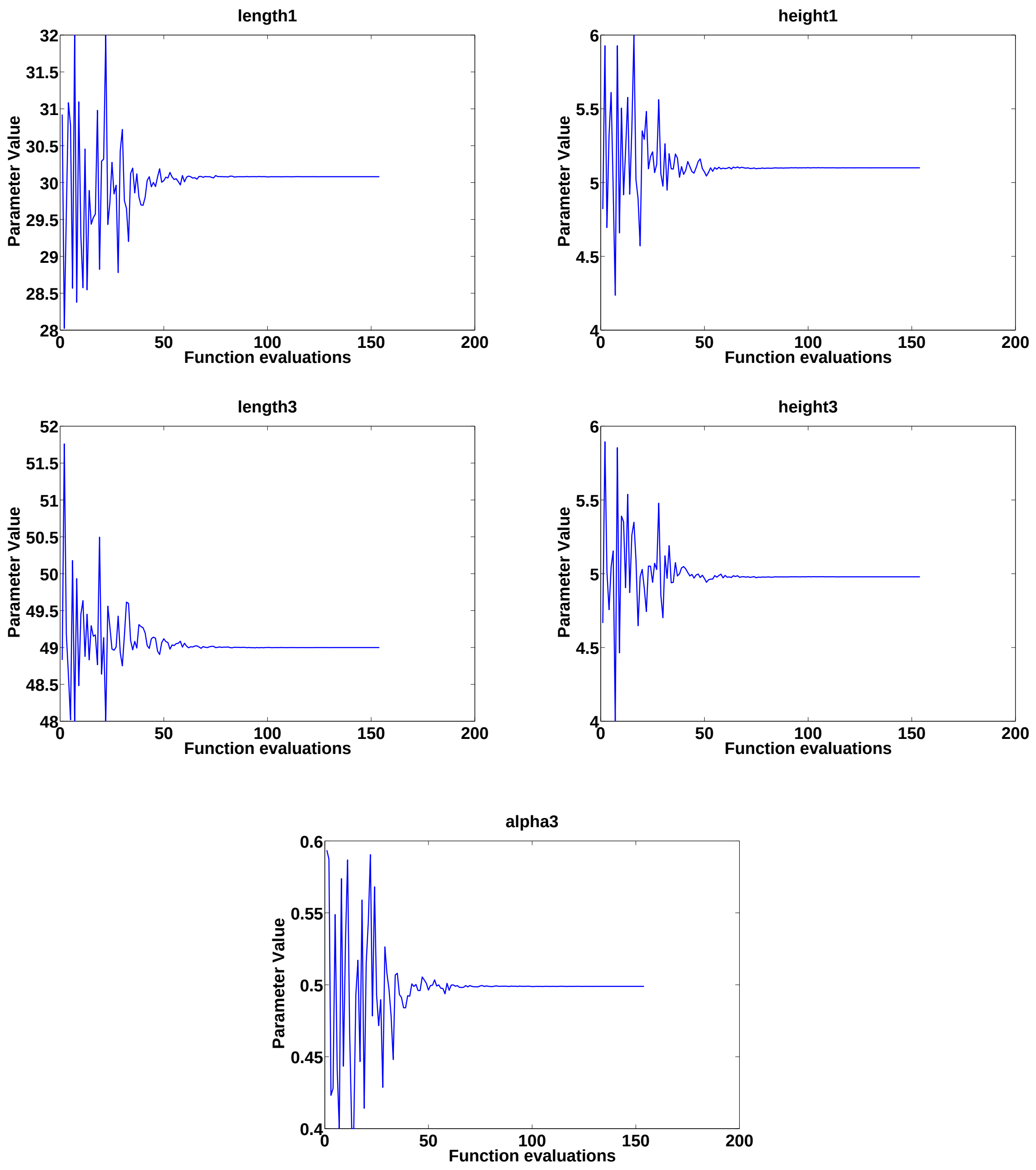

Figure 6.5: Optimization Run 1: Design variable convergence history.

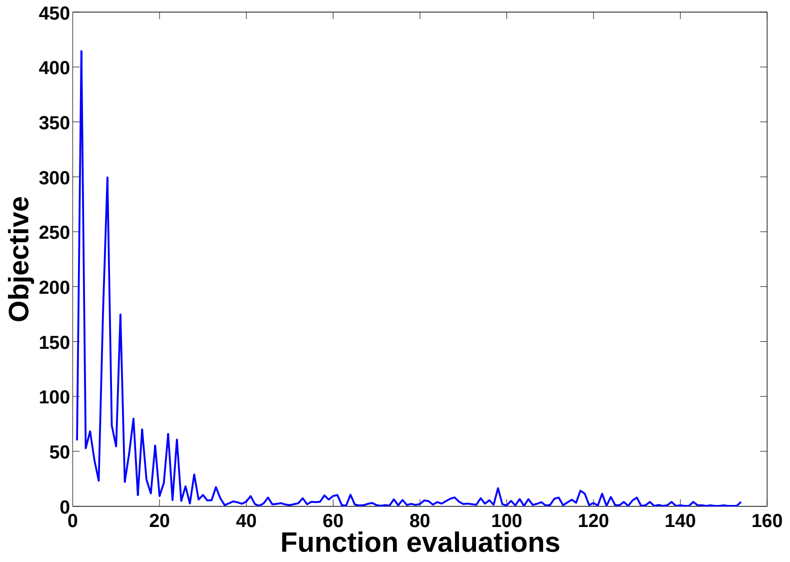

Figure 6.6: Optimization Run 1: Objective function convergence history.

6.3.2 Optimization Run 2: Simplex-Simulated Annealing

This algorithm combines the Nelder–Mead algorithm with a simulated annealing type random variation. This gives the algorithm a chance to escape local optima. The algorithm works by *subtracting* a logarithmically distributed (positive) random variable from the *objective function value* associated with each of the vertexes of the simplex. Additionally, a similar random variable is *added* to the objective function value at every new replacement point. In both cases the perturbation is proportional to the temperature t of the system [16]:

$$\begin{aligned} (f_{perturbed})_k &= f_k - t * ln(rnd) \qquad k = 1, ..., d+1 \\ (f_{perturbed})_{new} &= f_{new} + t * ln(rnd) \end{aligned} \tag{6.6}$$

The same initial simplex was used as in the previous run, and the objective function convergence history (in figure 6.7) seems quite similar to the previous run with only a bit more variation (or noise) due to the random perturbation. However, the design variable convergence (in figure 6.8) appears a lot more random than in the first run. Therefore, this algorithm does seem to make a better job exploring the design space, and if we had a highly multimodal objective function, this algorithm probably would be a decent choice.

Algorithm Parameters	
Cooling Coefficient	$p = 1.0$
Initial Temperature	$t_0 = 10$
Maximum Number of Designs	$n = 300$
Annealing Schedule	$t = t_0(1 - i/n)^p$

Optimization Run Summary	
Number of Designs	302
Number of Feasible Designs	292
Number of Error Designs	10
Total Run Time	$4h\ 27min$

	Design id	l_1	h_1	l_3	h_3	α_3	$f(\mathbf{x})$
Best Design	281	28.693	4.784	51.777	5.641	0.4511	0.3779
Worst Design	1	28.024	5.927	51.759	5.894	0.5874	435.8

Table 6.2: Optimization Run 2 Summary.

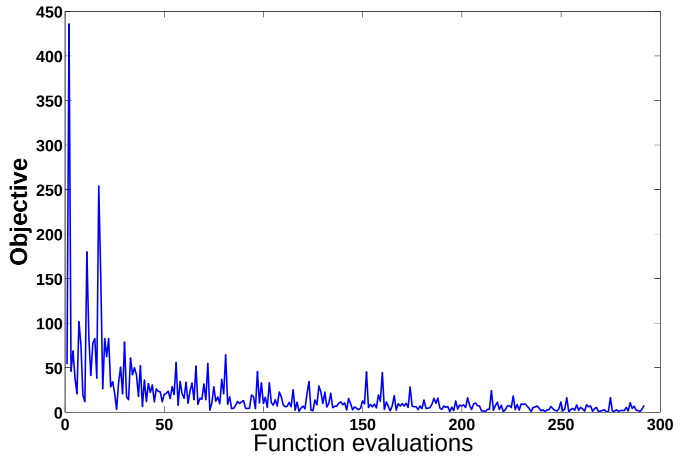

Figure 6.7: Optimization Run 2: Objective function convergence history.

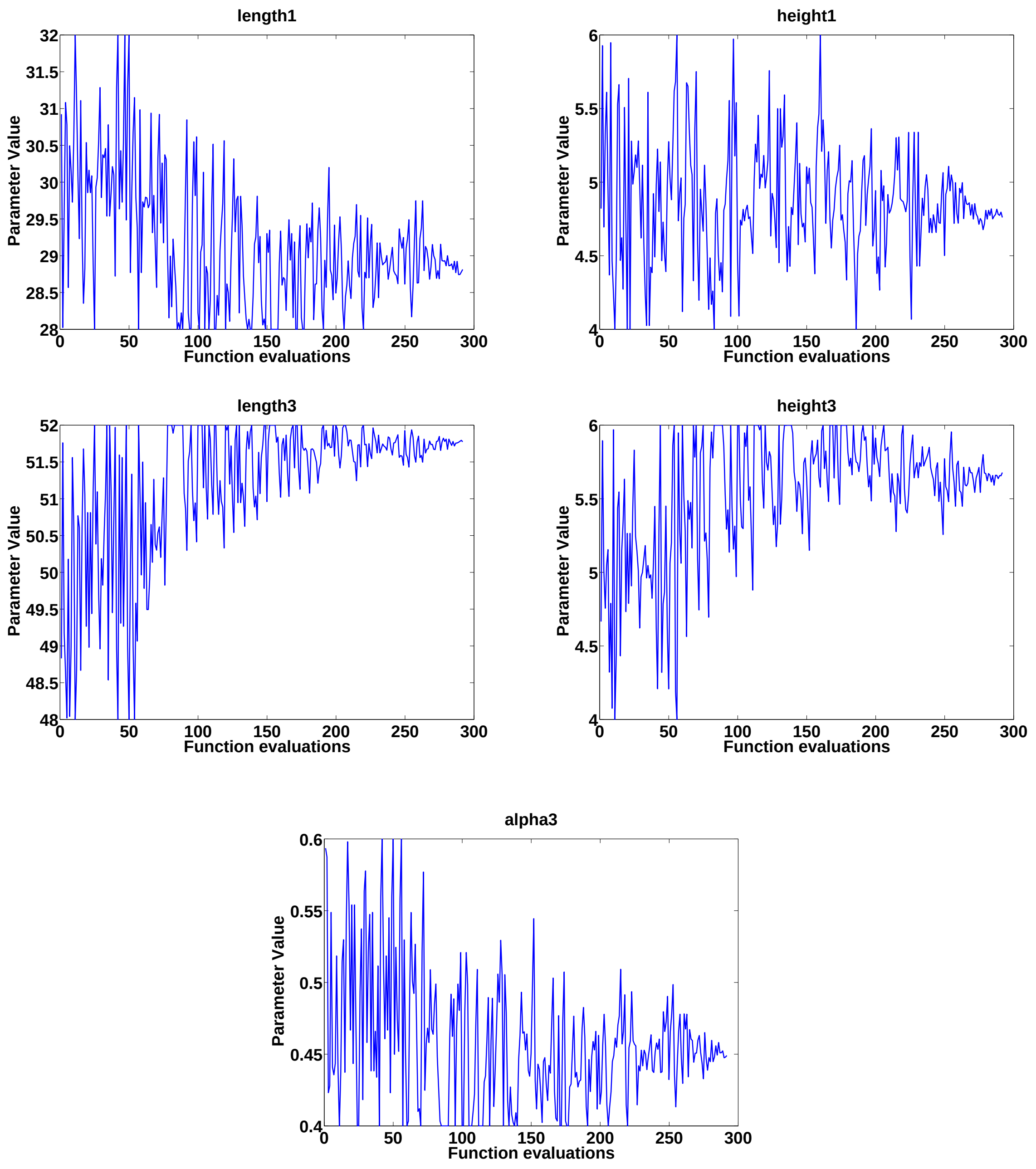

Figure 6.8: Optimization Run 2: Design variable convergence history.

6.3.3 Optimization Run 3: The NSGA-II Genetic Algorithm

This test run was by far the most extensive that was done totaling almost 12 hours of computations. NSGA-II is an elitistic multiobjective optimization algorithm, and therefore it is not really designed for an optimization problem of this type. Nevertheless, it was tested because the plan is to solve more difficult optimization problems in the future.

In retrospect, a larger population with fewer generations should have been used. As can be seen from figure 6.10, the randomly generated population of 10 individuals degenerated quickly into a very homogeneous population. This is probably attributed to the elitism of NSGA-II. On the other hand the algorithm also converged surprisingly fast with only about 5 generations (see figure 6.9), and after that the majority of newly generated designs were already existing, and thus only 324 designs out of the 1000 were unique. Therefore, 100 generations was excessive with only 10 individuals and a mutation probability of 0.05.

Before the population degenerated, the algorithm seemed to explore the design space very well. In [22] the algorithm is also claimed to accurately find a *diverse* Pareto optimal set of a multiobjective optimization problem. Unfortunately, the test problem is single objective so this claim could not be tested. Despite this, the NSGA-II algorithm seems like a relatively fast genetic algorithm for multimodal multiobjective problems. For more about the NSGA-II see [22].

Algorithm Parameters	
Number of Generations	100
Individuals per Generation	10
Crossover Probability	0.9
Mutation Probability	0.05
Crossover Distribution Index	20
Mutation Distribution Index	20

Optimization Run Summary	
Number of Designs	324
Number of Feasible Designs	314
Number of Error Designs	10
Total Run Time	$11h\ 45min$

	Design id	l_1	h_1	l_3	h_3	α_3	$f(\mathbf{x})$
Best Design	**224**	30.759	4.963	50.213	5.954	0.4413	0.3660
Worst Design	1	28.024	5.927	51.759	5.894	0.5874	359.1

Table 6.3: Optimization Run 3 Summary.

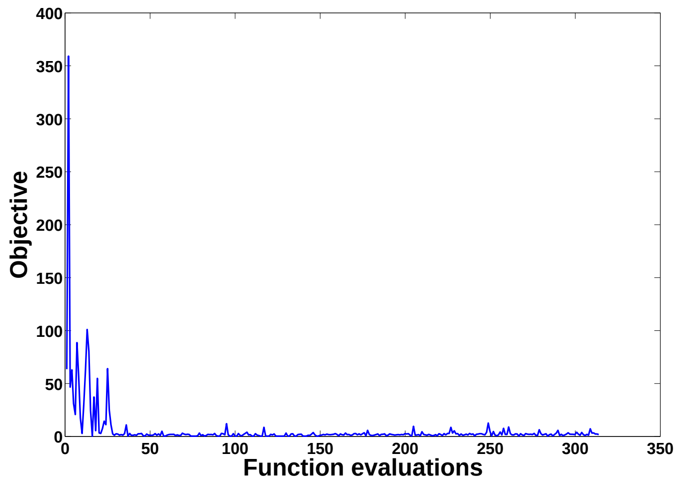

Figure 6.9: Optimization Run 3: Objective function convergence history.

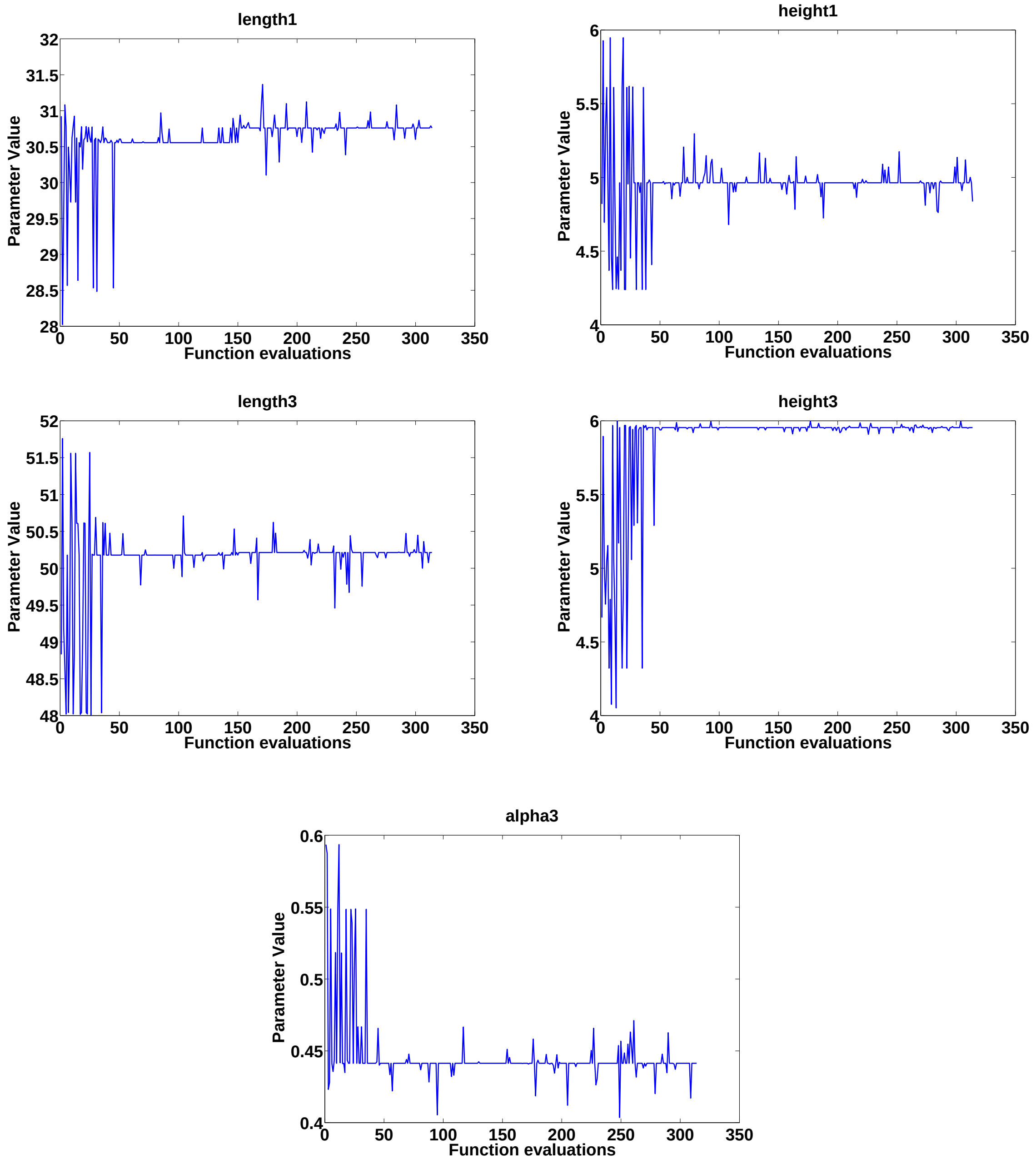

Figure 6.10: Optimization Run 3: Design variable convergence history.

6.4 Remarks and Conclusions from the Optimization Runs

As can be seen from the optimization results, the predefined optimal solution $\mathbf{x}^*$ is not unique. Judging from the three optimization runs, the objective function (6.4) seems to be linear. Therefore, we can guess to find a linear dependence between the design vector $\mathbf{x}$ and the objective function f. We can also expect that $b_1(\mathbf{x})$ depends only on (l_1, h_1) and $b_3(\mathbf{x})$ depends only on (l_3, h_3, α_3). Figures 6.11 and 6.12 strongly support this hypothesis.

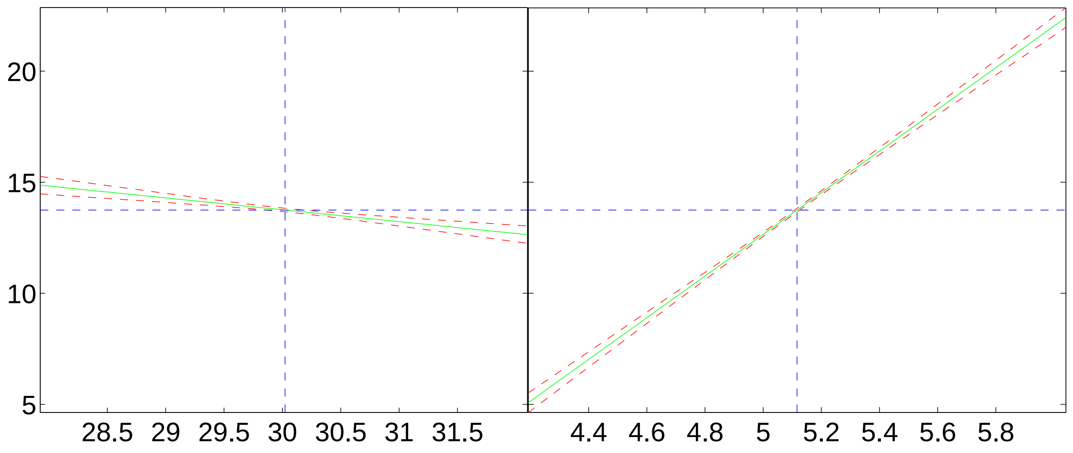

Figure 6.11: Optimization Run 1: Multivariable linear fitting of l_1 and h_1 versus $b_1(\mathbf{x})$.

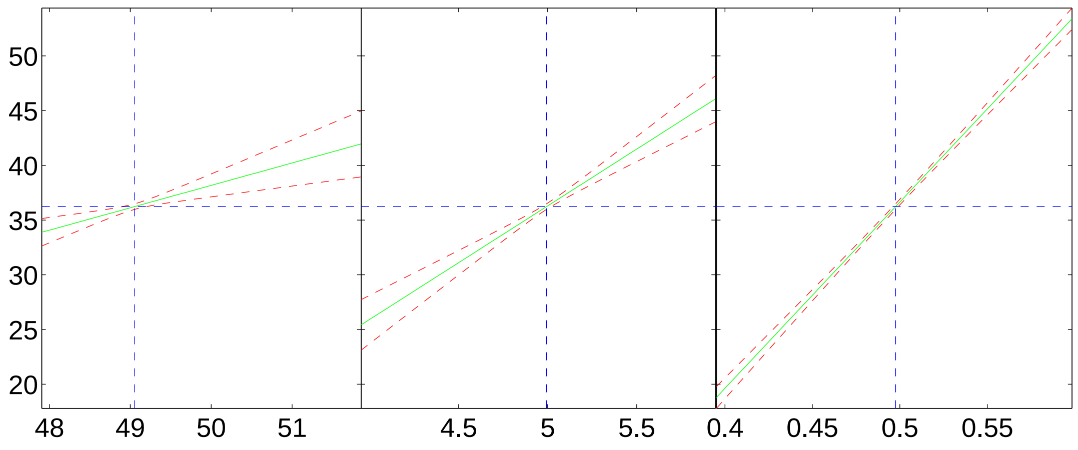

Figure 6.12: Optimization Run 1: Multivariable linear fitting of l_3, h_3 and α_3 versus $b_3(\mathbf{x})$.

The set of optimal designs of (6.1) should therefore be a line in U^{ad}. Statistical proof of this claim will not be presented as this is not really the main concern of this thesis, but a response surface model (see figure 6.13) fitted to the data of the second optimization run illustrates the impact of h_1 and l_1 on $(b_1(\mathbf{x}^*) - b_1(\mathbf{x}))^2$.

From tables 6.1, 6.2 and 6.3 is evident that in each case the best solution is close to around 0.35. This generally seems to be the best solution that can be acquired with the used mesh fidelity. The use of an algorithm such as the NSGA-II is a bit of overkill for such a simple problem, but what matters is that the optimization platform works as intended and is capable of finding optimal solution(s) of a design optimization problem.

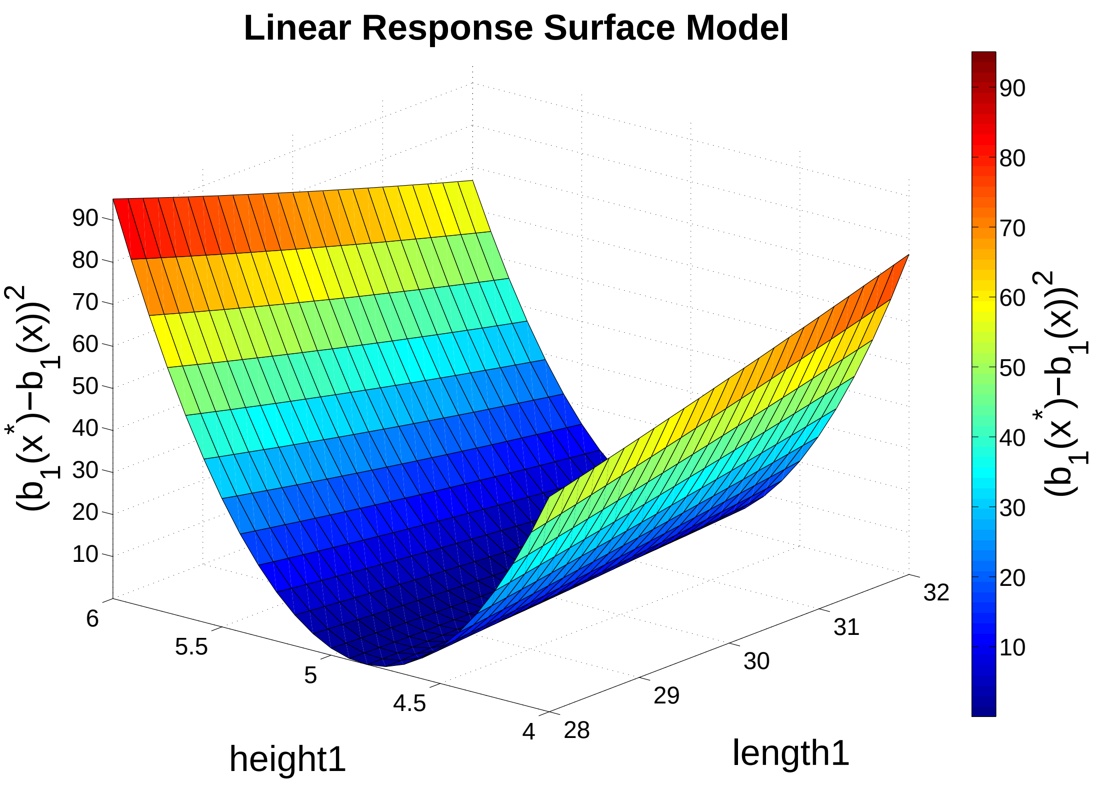

Figure 6.13: Optimization Run 2: Response Surface Model of l_1 and h_1 versus $(b_1(\mathbf{x}^*) - b_1(\mathbf{x}))^2$.

7 Expanding the Platform to an MDO Platform

How could we expand the optimization platform to an MDO platform? Let us take a brief speculative glance on how to expand the platform, and what problems possibly could arise from generalizing the presented optimization platform to encompass multiple disciplines.

The type of an MDO method used obviously affects the way we need to construct the MDO platform. For instance, with the AAO formulation minimal changes to the optimization platform are needed, whereas with multilevel methods a system analysis with multiple optimizers is needed. It is also reasonable to assume that an MDO problem has conflicting objectives and therefore multiobjective optimization methods are needed. Let us assume that we would have an MDO problem as shown in figure 7.1.

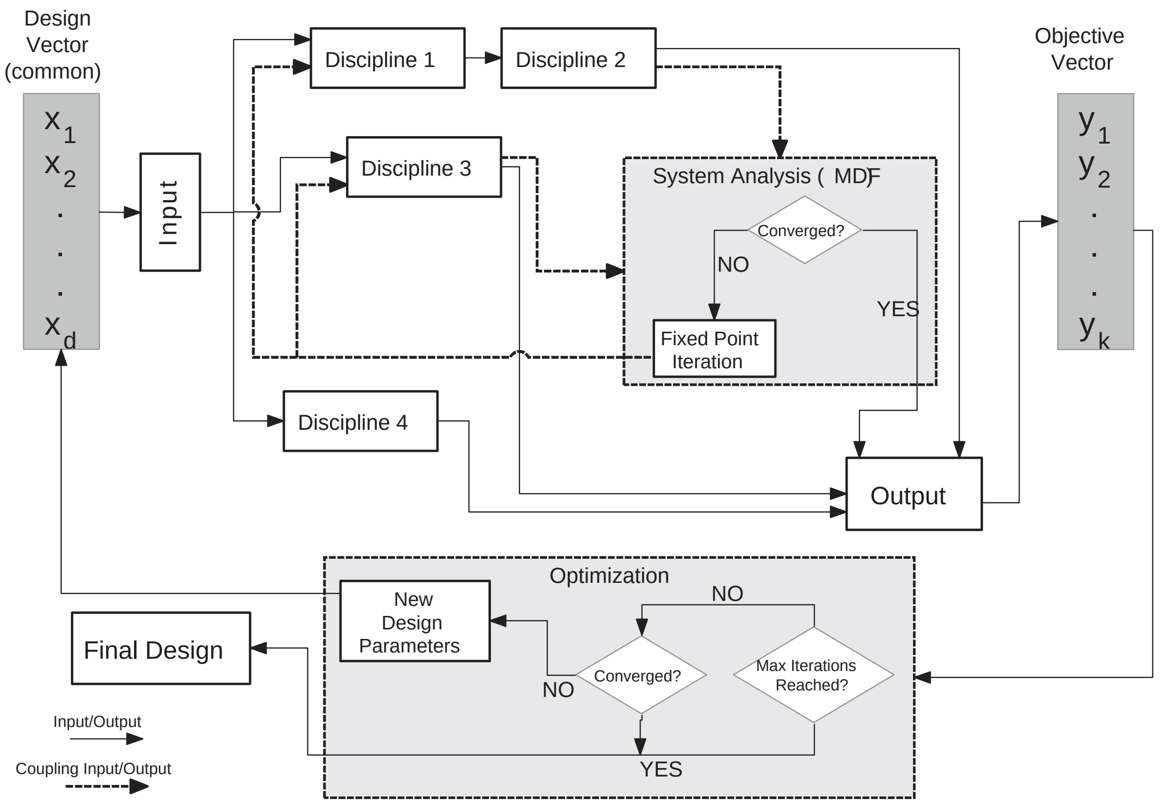

Figure 7.1: An example MDO problem with the MDF method.

The MDO problem has four disciplines from which disciplines one, two and three are coupled. With the MDF method, multidisciplinary feasibility is required during each design optimization iteration. The simplest way to achieve feasibility is to use the fixed point iteration method. The fixed point iteration method is applied every design optimization iteration until equilibrium is achieved (within an acceptable tolerance). If we assume that on average with this MDO problem multidisciplinary feasibility would be achieved when $i = N$, we would need to do approximately $3 \cdot N \cdot M + M$ simulations, where M is the number of design optimization iterations.

How to implement such an MDO platform in practice with the tools that were presented earlier? A sketch plan can be seen from figure 7.2. Only one CAPRI server is depicted in the figure. However, there is no limitation on the amount of CAPRI servers. It might be necessary to have more CAPRI server/CAD system combinations if the CAPRI server becomes a bottleneck. Another added difficulty is that running multiple solvers parallel requires the optimization process to be divided into separate threads. Moreover, each solver probably needs parallel or distributed computing to finish its respective computation in a reasonable time.

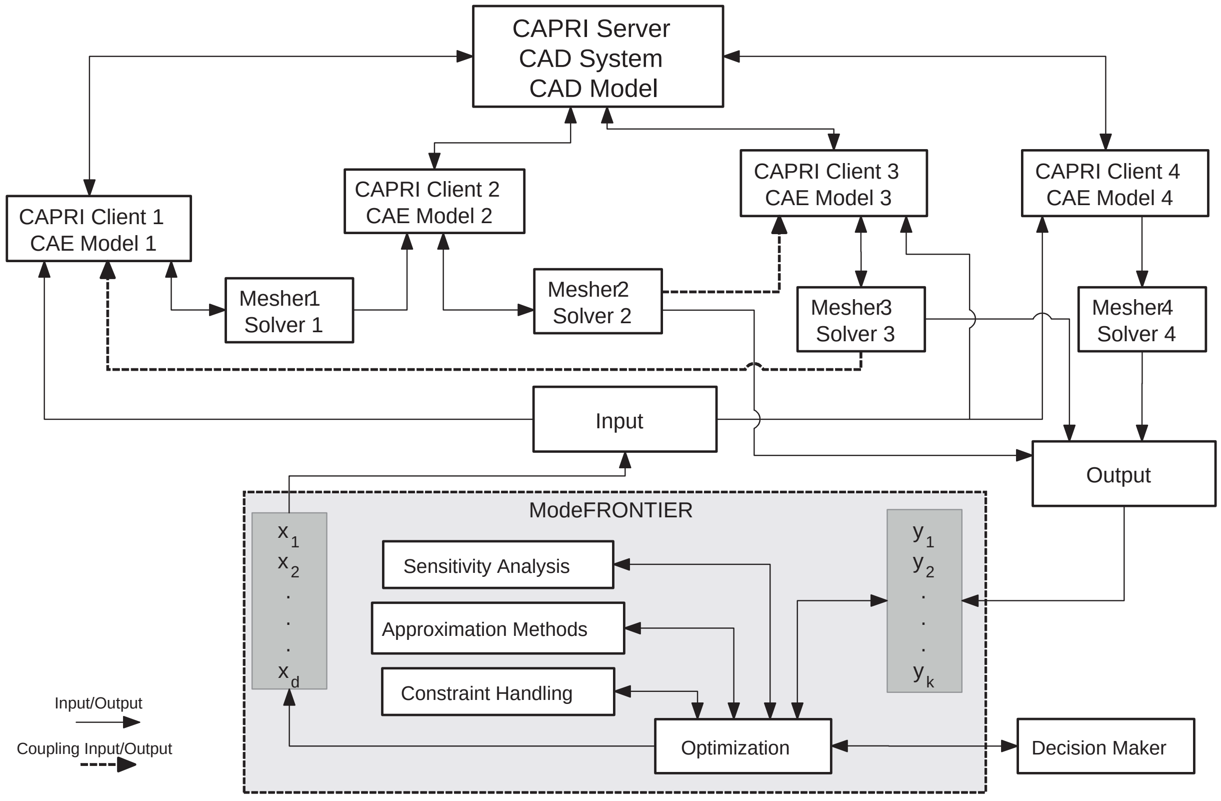

Figure 7.2: A sketch of an example MDO platform.

The amount of simulations can rise high with the MDF method, but the good thing about using the MDF method is that multidisciplinary feasibility is maintained after every optimization iteration. Therefore, conclusions can be made from the optimization results even if the process has not yet converged. This also means that information can be gathered from multiple disciplines in the middle of optimization, and via the CAPRI clients interpolation can be done to approximate values on locations that do not contain a mesh node in the respective discipline.

The organizational difficulty of the example platform is already formidable. Various experts from different fields (e.g. programming, optimization, analysis experts) are no doubt needed to operate and design such a platform. Research and planning needs to be made to evaluate different approaches one can have to solve the organizational challenges of such a platform. For more information on integrating CAD with an MDO platform see [21, 46].

8 Concluding Thoughts and Future Work

A general framework for an MDO platform was presented, and challenges in the practical implementation of MDO were studied. In design optimization the objective function is usually difficult and little is known of its properties beforehand. Moreover, multiobjective optimization methods become more relevant than in single discipline design optimization due to the more complex nature of MDO problems.

The MDO problem was formulated and methods to solve it were studied briefly. A good starting point for an MDO platform is the MDF method due to its simplicity. The IDF method is generally more efficient, and can also be adopted rather easily. Other MDO methods can be considered when the MDO platform has been tested with one of these two methods.

The CAD-CAE interoperability issue was presented. In an MDO environment the easy parameterization and defeaturing of the geometry is essential. Moreover, the geometry boundary representation should be identical to all the used meshers. This way gaining information on a single point in Ω from several disciplines simultaneously is possible with at least some accuracy. A unified geometry access API such as CAPRI was shown to be the most flexible approach to alleviate the interoperability issue, but also direct coupling to CAD can be considered in some circumstances.

The CAD-CAE interoperability issue can be considered largely solved by CAPRI, and probably by other neutral APIs in the future as well. However, the use of CAPRI is still a bit troublesome. It is by no means attributed to design flaws of CAPRI, but due to the fact that CAE applications have no standardized information exchange protocol. Therefore, including several CAE applications to a single MDO platform will always be troublesome if this does not change.

Optimization runs showed that the optimization platform of our research group can solve single discipline design optimization problems. The next step would be to solve test cases with multiple disciplines. Nokia has promised to provide an industrial test case with a simplified CAD model and problem definition. There was talk that the test case will have two or three disciplines, but no agreement was made whether they are to be coupled or not. Regardless, it will be a chance to test the platform with a significantly more challenging problem. The disciplines of the test case will most likely be electromagnetic and thermal analysis, and perhaps structural analysis as well.

Practically any solver and mesher can be used with the approach that was used in the optimization platform of our research group. Additionally, CAPRI supports all of the major CAD systems and therefore a wide variety of CAD systems can be used. Probably some time in the future, off-the-shelf MDO platforms will emerge from major optimization software companies. It remains to be seen if they prove to be as flexible as the approach that was presented in this thesis.

References

[1] M. Ainsworth, J. T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley, 2000.

[2] J. Antony, *Design of Experiments for Engineers and Scientists*, Butterworth-Heinemann, 2003.

[3] J. S. Arora, *Optimization of Structural and Mechanical Systems*, World Scientific, 2007.

[4] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, Springer, 2006.

[5] G. Barequet and M. Sharir, *Filling Gaps in the Boundary of a Polyhedron*, Computer-Aided Geometric Design, 12(2), 207-229, 1995.

[6] M. W. Beall, J. Walsh, M. S. Shephard, *Accessing CAD Geometry for Mesh Generation*, 12th International Meshing Roundtable, 2003.

[7] M. P. Bendsøe, O. Sigmund, *Topology Optimization*, Springer, 2003.

[8] F. Bianconi, *Bridging the gap between CAD and CAE using STL files*, International Journal of CAD/CAM, 2(1), 55-67, 2002.

[9] F. Bianconi, P. Conti, L. Di Angelo, *Interoperability among CAD/CAM/CAE systems: a review of current research trends*, Proceedings of the Geometric Modeling and Imaging - New Trends, 82-89, 2006.

[10] R. D. Braun, *Collaborative optimization: an architecture for large-scale distributed design*, Stanford University, PhD Thesis, 1996.

[11] E. Cantú-Paz, *A Survey of Parallel Genetic Algorithms*, Techincal Report 97003, Illinois Genetics Algorithms Laboratory, University of Illinois, 1998.

[12] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, M. Sumner, *A Fast Adaptive Memetic Algorithm for Off-line and On-line Control Design of PMSM Drives*, IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Memetic Algorithms, 37(1), 28-41, 2007.

[13] CAPRI CAE Gateway, CADNexus, website at http://www.cadnexus.com/

[14] CATIA V5, Dassault Systèmes, website at http://www.3ds.com/products/catia/

[15] R. Claus, M. Kazakov, *OMG CAD Services V1.0 standard: an approach to CAD/CAx integration*, International Journal of Product Lifecycle Management, 2(2), 157-172, 2007.

[16] M. F. Cardoso, R. L. Salcedo, S. Feyo de Azevedo, *The simplex-simulated annealing approach to continuous non-linear optimization*, Journal of Computers & Chemical Engineering, 20(9), 1065-1080, 1996.

[17] C. A. Coello Coello, D. A. van Veldhuizen, G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2002.

[18] COMSOL CAD Import Module, The COMSOL group, website at http://www.comsol.com/products/cad/

[19] COMSOL Multiphysics, The COMSOL group, website at http://www.comsol.com/

[20] E. J. Cramer, J. E. Dennis Jr, P. D. Frank, R. M. Lewis, G. R. Shubin, *Problem Formulation for Multidisciplinary Optimization*, AIAA Symposium on Multidisciplinary Design Optimization, 1993.

[21] C. Crawford, R. Haimes, *Synthesizing an MDO Architecture in CAD*, AIAA Paper 2004-0281, 2004.

[22] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, *A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6(2), 182-197, 2002.

[23] S. Droste, T. Jansen, and I. Wegener, *Optimization with randomized search heuristics: the (A)NFL theorem, realistic scenarios, and difficult functions*, Theoretical Computer Science, 287(1), 131-144, 2002.

[24] T. English, *No More Lunch: Analysis of Sequential Search*, Proceedings of the 2004 IEEE Congress on Evolutionary Computation, 227-234, 2004

[25] C. Fefferman, *Existence and smoothness of the Navier–Stokes equations*, 2000, available at www.claymath.org/prizeproblems/navierstokes.htm

[26] Elmer - Finite Element Software for Multiphysical Problems, CSC-Scientific Computing Ltd., website at http://www.csc.fi/elmer/

[27] C. M. Fonseca, P. J. Fleming, *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*, Proceedings of the Fifth International Conference on Genetic Algorithms, 416-423, 1993.

[28] P. J. Frey, P-L. George *Mesh Generation, Application to Finite Elements*, Hermes Science Publishing , 2000.

[29] D. M. Fudge, D. W. Zingg, R. Haimes, *A CAD-Free and a CAD-Based Geometry Control System for Aerodynamic Shape Optimization*, 2005.

[30] GeomSim, Simmetrix, website at http://www.simmetrix.com/

[31] A. A. Giunta, *Aircraft Multidisciplinary Design Optimization Using Design of Experiments Theory and Response Surface Modeling Methods*, Virginia Polytechnic Institute and State University, PhD Thesis, 1997.

[32] A. O. Griewank, *Generalized Descent for Global Optimization*, Journal of Optimization Theory and Applications, 34(1), 11-39, 1981.

[33] R. Haimes, C. Crawford, *Unified Geometry Access for Analysis and Design*, Technical Report, 12th International Meshing Roundtable, 2003.

[34] J. Haslinger, R. A. E. Mäkinen, *Introduction to Shape Optimization, Theory, Approximation, and Computation*, SIAM, 2003.

[35] I. Hlaváček, J. Chleboun, I. Babuška, *Uncertain Input Data Problems and the Worst Case Scenario*, Elsevier, 2004.

[36] C. M. Hoffman, R. Joan-Arinyo, *CAD and the Product Master Model*, Journal of Computer Aided Design, 30, 905-918, 1998.

[37] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, 2000.

[38] K. F. Hulme and C. L. Bloebaum, *A Comparison of Solution Strategies for Simulation-Based Multidisciplinary Design Optimization*, Proceedings Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2143-2153, 1998.

[39] Igel C., and Toussaint M. *A No-Free-Lunch Theorem for Non-Uniform Distributions of Target Functions*, Journal of Mathematical Modelling and Algorithms 3, 313-322, 2004

[40] P. Kere, J. Lento, *Design optimization of laminated composite structures using distributed grid resources*, Journal of Composite Structures, 71(3-4), 435-438, 2005.

[41] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *Optimization by Simulated Annealing*, Science, 220, 671-680, 1983.

[42] M. Křížek, P. Neittaanmäki, *Mathematical and Numerical Modelling in Electrical Engineering*, Kluwer Academic Publishers, 1996.

[43] S. H. Lee, *A CAD-CAE Integration Approach Using Feature-Based Multi-Resolution and Multi-Abstraction Modelling Techniques*, Journal of Computer Aided Design, 37(9), 941-955, 2004.

[44] M. Leyton, *A Generative Theory of Shape*, Springer, 2001.

[45] G. R. Liu, S. S. Quek, *The Finite Element Method: a Practical Course* , Butterworth Heinemann, 2003.

[46] T. Manning, P. Gage, J. Nguyen, R. Haimes, *ComGeom2: A Geometry Tool for Multidisciplinary Analysis and Data Sharing*, Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2004.

[47] K. M. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer Academic Publishers, 1999.

[48] modeFRONTIER, Esteco, website at http://www.esteco.com/

[49] R. H. Myers, D. C. Montgomery, *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*, John Wiley & Sons, 1995.

[50] P. Neittaanmäki, S. Repin, *Reliable Methods for Computer Simulation: Error Control and a Posteriori Estimates*, Elsevier, 2004.

[51] J. A. Nelder, R. Mead, *A Simplex Method for Function Minimisation*, The Computer Journal, 1965.

[52] F. Neri, J. Toivanen, R. Mäkinen, *An Adaptive Evolutionary Algorithm with Intelligent Mutation Local Searchers for Designing Multidrug Therapies for HIV*, Applied Intelligence, Special Issue on Computational Intelligence in Medicine and Biology, Springer, 27(3), 219-235, 2007.

[53] G. J. Park, *Analytic methods in design practice*, Springer, 2007.

[54] S. Pierret, *Multi-objective and Multi-Disciplinary Optimization of Three-dimensional Turbomachinery Blades*, Proceedings of the 6th World Congresses of Structural and Multidisciplinary Optimization, 2005.

[55] A. D. Polyanin, *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, CRC Press, 2001.

[56] M. J. Pratt, *Extension of ISO 10303, the STEP Standard, for the Exchange of Procedural Shape Models*, IEEE International Conference on Shape Modeling and Applications, 2004.

[57] S. D. Rajan, D.T. Nguyen, *Design Optimization of discrete structural systems using MPI-enabled genetic algorithm*, 2004.

[58] M. S. Shephard, M. W. Beall, R. M. O'Bara, B. E. Webster, *Toward simulation-based design*, Journal of Finite Elements in Analysis and Design, 40(12), 1575-1598, 2004.

[59] J. Sobieszczanski-Sobieski, *Multidisciplinary Design Optimization: An Emerging New Engineering Discipline*, NASA, 1993.

[60] J. Sobieszczanski-Sobieski, J. Agte, R. Sandusky, *Bi-level integrated system synthesis*, Proceedings of AIAA/USAF/NASA/ISSMO symposium on multidisciplinary analysis and optimization, AIAA Paper AIAA-98-4916, 1998.

[61] J. Sobieszczanski-Sobieski, R. T. Haftka, *Multidisciplinary aerospace design: survey of recent developments*, Journal of Structural and Multidisciplinary Optimization, 14(1), 1-23, 1997.

[62] P. Tang, K. Chang, *Integration of topology and shape optimization for design of structural components*, Journal of Structural and Multidisciplinary Optimization, 22(1), 65-82, 2001.

[63] R. V. Tappeta, S. Nagendra, J. E. Renaud, K. Badhrinath, *Concurrent Sub-Space Optimization(CSSO) MDO Algorithms in iSIGHT: validation and testing*, GE Research & Development Center, 1998.

[64] G. Tassey, *Interoperability Cost Analysis of the U.S. Automotive Supply Chain*, Technical report, National Institute of Standards and Technology, 1999.

[65] T. J. Tautges, *Common Geometry Module: A Generic Extensible Geometry Interface*, Proceedings of the 9th International Meshing Roundtable, Sandia National Laboratories, 2000.

[66] TetGen - A Quality Tetrahedral Mesh Generator, website at http://tetgen.berlios.de/

[67] J. Thompson, B. Soni, N. Weatherill, *Handbook of Grid Generation*, CRC Press, 1999.

[68] V. Toropov, *Multipoint approximation method in optimization problems with expensive function values*, Proceedings of the 4th International Symposium on Systems Analysis and Simulation, 207-212, 1992.

[69] A. E. Uva, G. Monno, and B. Hamann, *A New Method for the Repair of CAD Data with Discontinuities*, In II Italian-Spanish Seminar on Design and Feasibility of Industrial Products, 59-73, 1998.

[70] M. M. Woolfson, G. J. Pert, *An Introduction to Computer Simulation*, Oxford University Press, 1999.

[71] S. I. Yi, J. K. Shin, G. J. Park, *Comparison of MDO methods with mathematical examples*, Journal of Structural and Multidisciplinary Optimization, 35(5), 391-402, 2007.

A An Example CAPRI Program

```
#include <stdio.h>
#include "capri.h"

/* Hello -- An example CAPRI program
* Initializes CAPRI, loads a CAD model, prints the number of
* volumes in the model and finally closes CAPRI.
*/

int main(int argc, char *argv[])
{
int status;
if (argc != 2) {
printf(" Usage: hello Modeller Model\n");
return 0;
}

/* start CAPRI */
status = gi_uStart();
printf(" gi_uStart status = \%d\n", status);
if (status != 0) exit(1);

/* load the specified model */
status = gi_uLoadModel(NULL, argv[1], argv[2]);
printf(" Part \%s: gi_uLoadModel status = \%d\n", argv[2], status);

/* print the number of volumes */
printf(" Number of volumes = \%d\n", gi_uNumVolumes());

/* stop CAPRI */
gi_uStop(0);
return 0;
}
```

B The Comsol Script

```
flclear fem

% COMSOL version
clear vrsn
vrsn.name = 'COMSOL 3.3';
vrsn.ext = 'a';
vrsn.major = 0;
vrsn.build = 511;
vrsn.rcs = '$Name:  $';
vrsn.date = '$Date: 2007/02/02 19:05:58 $';
fem.version = vrsn;

% Import mesh
marr = meshimport('mesh001.mphtxt');
fem.mesh = marr{1};

% Application mode 1
clear appl
appl.mode.class = 'Laplace';
appl.assignsuffix = '_lpeq';
clear bnd
bnd.r = {0,'x'};
bnd.type = {'neu','dir'};
bnd.ind = [2,2,2,2,2,2,1,1,1];
appl.bnd = bnd;
fem.appl{1} = appl;
fem.frame = {'ref'};
fem.border = 1;
clear units;
units.basesystem = 'SI';
fem.units = units;
```

```
% Multiphysics
fem=multiphysics(fem);
% Extend mesh
fem.xmesh=meshextend(fem);
% Solve problem
fem.sol=femstatic(fem, ...
                  ’solcomp’,{’u’}, ...
                  ’outcomp’,{’u’});

% Save current fem structure for restart purposes
fem0=fem;

% Integrate over ellipsoid 1
press1=postint(fem,’(1 - absux_lpeq)^2’, ...
           ’unit’,’’, ...
           ’dl’,[8], ...
           ’edim’,2);
% Integrate over ellipsoid 2
press2=postint(fem,’(1 - absux_lpeq)^2’, ...
           ’unit’,’’, ...
           ’dl’,[9], ...
           ’edim’,2);
% Integrate over ellipsoid 3
press3=postint(fem,’(1 - absux_lpeq)^2’, ...
           ’unit’,’’, ...
           ’dl’,[7], ...
           ’edim’,2);

% Calculate residuals and print them to file output.dat

fid = fopen(’output.dat’,’wt’);
    fprintf(fid,[’b1 = ’,num2str(press1),’\n’]);
    fprintf(fid,[’b2 = ’,num2str(press2),’\n’]);
    fprintf(fid,[’b3 = ’,num2str(press3),’\n’]);
fclose(fid);
```