Campus Universitario
Dpto. de Teoría de la Señal y Comunicaciones
Ctra. Madrid-Barcelona, Km. 36.6
28805 Alcalá de Henares (Madrid)
Telf: +34 91 885 88 99
Fax: +34 91 885 66 99

D. SANCHO SALCEDO SANZ, Profesor Titular de Universidad del Área de Conocimiento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá, y D. CARLOS LOZANO RODRIGUEZ, Científico Superior del Área de Dinámica de Fluidos del Instituto Nacional de Técnica Aeroespacial "Esteban Terradas"

CERTIFICAN

Que la tesis "**New strategies for the aerodynamic design optimization of aeronautical configurations through soft-computing techniques**", presentada por D. Esther Andrés Pérez y realizada en el Departamento de Teoría de la Señal y Comunicaciones bajo nuestra dirección, reúne méritos suficientes para optar al grado de Doctor, por lo que puede procederse a su depósito y lectura.

Alcalá de Henares, May 28, 2012.

Fdo.: Dr. D. Sancho Salcedo Sanz   Fdo.: Dr. D. Carlos Lozano Rodriguez

Universidad de Alcalá

Campus Universitario
Dpto. de Teoría de la Señal y Comunicaciones
Ctra. Madrid-Barcelona, Km. 36.6
28805 Alcalá de Henares (Madrid)
Telf: +34 91 885 88 99
Fax: +34 91 885 66 99

D. Esther Andrés Pérez ha realizado en el Departamento de Teoría de la Señal y Comunicaciones y bajo la dirección del Dr. D. Sancho Salcedo Sanz, la tesis doctoral titulada "**New strategies for the aerodynamic design optimization of aeronautical configurations through soft-computing techniques**", cumpliéndose todos los requisitos para la tramitación que conduce a su posterior lectura.

Alcalá de Henares, May 28, 2012.

EL DIRECTOR DEL DEPARTAMENTO

Fdo: Dr. D. Saturnino Maldonado Bascón.

Universidad de Alcalá

Escuela Politécnica Superior

Dpto. de Teoría de la Señal y Comunicaciones

Doctorado en Tecnologías de la Información y las Comunicaciones

Memoria de tesis doctoral

New strategies for the aerodynamic design optimization of aeronautical configurations through soft-computing techniques

Autor: D. Esther Andrés Pérez
Director: Dr. Sancho Salcedo Sanz
Codirector: Dr. Carlos Lozano Rodriguez

INTA

To my family

Abstract

This thesis deals with the improvement of the optimization process in the aerodynamic design of aeronautical configurations. Nowadays, this topic is of great importance in order to allow the European aeronautical industry to reduce their development and operational costs, decrease the time-to-market for new aircraft, improve the quality of their products and therefore maintain their competitiveness.

In particular, according to data collected in the report ”European Aeronautics: A vision for 2020” published by the European Commission, it is expected that, in the next fifteen years, air traffic over the world will double. This increase in air traffic must be considered in addition to its foreseen environmental impact, since, currently, the aviation contributes significantly to the emission of carbon dioxide to the atmosphere. In view of this situation, the ACARE (Advisory Council for Aerospace Research in Europe) has established several targets for 2020, as for example, 50% reduction in carbon dioxide emissions, fuel consumption, perceived noise and development time, as well as 80% reduction in emissions of nitrogen oxides.

The achievement of these challenges involves an unprecedented technological progress, not being able to fulfil the objectives through small changes in the traditional aircraft configurations and, therefore, making necessary to explore other unconventional settings and novel concepts that have not been considered so far. To this end, improved aerodynamic and multidisciplinary design phases will support the transition from the current aircraft configuration into the future aircraft.

Within this work, a study of the state-of-the-art of the aerodynamic optimization tools has been performed, and several contributions have been proposed at different levels:

- One of the main drawbacks for a fully industrial application of aerodynamic optimization tools is the huge requirement of computational resources. In aerodynamic design optimization problems, the flow fields are simulated using flow solvers based on Computational Fluid Dynamics (CFD) techniques. These high-fidelity numerical simulation codes have proved to be reliable and relatively cheap compared with experimental methods. But they are computationally expensive, highly memory demanding, and time consuming. For example, the simulation of a complete aircraft configuration, even if built on simplified models like the Reynolds-Averaged Navier-Stokes (RANS) equations, requires, in the steady flow case, approximately half a day for each simulation point using a high performance cluster of 24 processors. These drawbacks of analysis codes become more severe when they are utilized in the field of shape optimization since it requires rather more computations. For practical optimization problems, in which at least 100 design variables are to be considered, current methodological approaches applied in industry would need more than a year to obtain an

optimized aircraft (this is completely impractical for the aeronautical industry). For this reason, one proposed contribution of this work is focused on reducing the computational cost by the use of different techniques as surrogate modeling, control theory, as well as other more software-related techniques as code optimization and proper domain parallelization, all with the goal of decreasing the cost of the aerodynamic design process.

- Other contribution is related to the consideration of the design process as a global optimization problem, and, more specifically, the use of evolutionary algorithms (EAs) to perform a preliminary broad exploration of the design space, due to their ability to obtain global optima. Regarding this, EAs have been hybridized with metamodels (or surrogate models), in order to substitute expensive CFD simulations. In this thesis, an innovative approach for the global aerodynamic optimization of aeronautical configurations is proposed, consisting of an Evolutionary Programming algorithm hybridized with a Support Vector regression algorithm (SVMr) as a metamodel. Specific issues as precision, dataset training size, geometry parameterization sensitivity and techniques for design of experiments are discussed and the potential of the proposed approach to achieve innovative shapes that would not be achieved with traditional methods is assessed.

- Then, after a broad exploration of the design space, the optimization process is continued with local gradient-based optimization techniques for a finer improvement of the geometry. Here, an automated optimization framework is presented to address aerodynamic shape design problems. Key aspects of this framework include the use of the adjoint methodology to make the computational requirements independent of the number of design variables, and Computer Aided Design (CAD)-based shape parameterization, which uses the flexibility of Non-Uniform Rational B-Splines (NURBS) to handle complex configurations.

The mentioned approach is applied to the optimization of several test cases and the improvements of the proposed strategy and its ability to achieve efficient shapes will complete this study.

Resumen en Castellano

Esta tesis tiene como objetivo introducir mejoras en el proceso de optimización del diseño aerodinámico de configuraciones aeronáuticas. En la actualidad, este tema ha adquirido una gran importancia, con el propósito de permitir que la industria aeronáutica europea pueda reducir sus costes de desarrollo y operatividad, acortar el tiempo de lanzamiento al mercado de nuevos aviones, mejorar la calidad de sus productos y, por tanto, mantener su competitividad.

En particular, según los datos recogidos en el informe "Aeronáutica Europea: Una visión para el 2020", publicado por la Comisión Europea, se espera que en los próximos quince años, el tráfico aéreo en el mundo se duplique. En este aumento en el tráfico aéreo se debe considerar, además, su impacto ambiental, ya que, actualmente, la aviación contribuye significativamente a la emisión de dióxido de carbono a la atmósfera. En vista de esta situación, ACARE (Consejo Asesor para la Investigación Aeronáutica en Europa) ha establecido varios objetivos para el año 2020, como por ejemplo, el 50 % de reducción en las emisiones de dióxido de carbono, el consumo de combustible, el ruido percibido y el tiempo de desarrollo, así como el 80 % de reducción de las emisiones de óxido de nitrógeno.

La consecución de estos retos implica un progreso tecnológico sin precedentes, puesto que los ambiciosos objetivos propuestos no podrían alcanzarse mediante pequeños cambios en las configuraciones tradicionales de aeronaves, y, por tanto, resulta necesario explorar nuevos conceptos y formas no convencionales. Con este fin, resulta necesaria la mejora de la aerodinámica y la introducción de aspectos multidisciplinares en la fase de diseño, contribuciones que apoyarán el proceso de transición desde la configuración actual hasta el avión del futuro.

En este trabajo, se ha realizado un análisis del estado del arte de las herramientas de optimización aerodinámica que se utilizan actualmente, y se proponen varias contribuciones a diferentes niveles:

- Uno de los principales inconvenientes para una completa aplicación industrial de las herramientas de optimización aerodinámica es la fuerte demanda de recursos computacionales. En problemas de optimización aerodinámica, los campos de flujo son simulados utilizando resolvedores basados en técnicas de Dinámica de Fluidos Computacional (en inglés, CFD). Estos códigos de simulación numérica han demostrado ser métodos fiables y relativamente baratos en comparación con los métodos experimentales. Pero son muy costosos computacionalmente, exigentes en memoria, y requieren mucho tiempo de cálculo. Por ejemplo, la simulación de una configuración de avión completa, incluso si se realiza con modelos como las ecuaciones "Reynolds-Averaged Navier-Stokes" (RANS), requiere, en el caso estacionario, aproximadamente medio día si se utiliza un cluster de 24 procesadores. Este inconveniente se acentúa cuando los códigos de análisis se utilizan en un ciclo de diseño

aerodinámico, ya que se requiere una cantidad considerable de simulaciones. En la práctica, en problemas de optimización, al menos 100 variables de diseño deben ser consideradas, y, por tanto, si se utilizaran los enfoques tradicionales, se necesitaría más de un año para obtener un avión optimizado (lo cual resulta totalmente inviable). Por esta razón, una de las contribuciones de este trabajo se centra en la reducción del coste computacional mediante el uso de diferentes técnicas como los metamodelos, la teoría de control, así como otras técnicas más relacionadas con el software, como la optimización de código y la paralelización, todo ello con el objetivo de mejorar la eficiencia computacional del proceso de diseño aerodinámico.

- Otra contribución se centra en considerar el proceso de diseño como un problema de optimización global y utilizar algoritmos evolutivos para realizar una amplia exploración preliminar del espacio de diseño. En esta etapa, los algoritmos evolutivos se acoplan con metamodelos (o modelos de sustitución), a fin de sustituir las costosas simulaciones CFD. En esta tesis se propone un novedoso enfoque consistente en hibridizar un algoritmo de optimización global basado en programación evolutiva con un metamodelo basado en la técnica de las Máquinas de Vectores Soporte (SVM). Se analizan cuestiones específicas como la precisión, el tamaño del conjunto de datos de entrenamiento, la sensibilidad de la parametrización geometrica, y se evalúa el potencial del enfoque propuesto para lograr formas innovadoras que no podrían obtenerse utilizando los métodos tradicionales.

- Después de una amplia exploración del espacio de diseño, el proceso de optimización continua con la utilización de técnicas de optimización local basadas en gradientes, con el objetivo de obtener una mejora más fina de la geometría resultante. En esta fase, se presenta una herramienta de optimización automática para problemas de diseño aerodinámico. Los aspectos clave de esta herramienta incluyen el uso de la metodología adjunta, que posibilita la independecia del coste computacional con respecto del número de variables de diseño, y una parametrización de la geometría utilizando ”Non-Uniform Rational B-Splines” (NURBS), lo que proporciona gran flexibilidad para manejar configuraciones complejas.

El enfoque mencionado se ha aplicado a la optimización de varios casos de prueba, analizando las mejoras obtenidas de las contribuciones propuestas, así como la capacidad de la estrategia para obtener nuevas formas eficientes.

Acknowledgements

This chapter is dedicated to all those whose hard work and dedication have enabled and facilitated the development and completion of this thesis. In particular, the following people have contributed significantly to the research addressed in this work:

- Dr. Sancho Salcedo, whose excellent technical background, knowledge and research experience have allowed to overcome all the difficulties encountered during the realization of this thesis, and have contributed significantly to its international dissemination.
- Dr. Carlos Lozano and researchers Jorge Ponsín, Mario Martín, Aitor Caloto and Pablo Bitrián, whose daily support and collaboration helped me to understand the complexity of aerodynamics, and whose work has introduced an unquestionable added value to this thesis. I would not have written this thesis without you and the great working atmosphere we have. In particular, I would like to thank Mario for his support regarding the NURBS library and point inversion algorithms, and Carlos for his help with the mathematical formulation of the adjoint methodology.
- Fernando Monge and Antonio González, Head of the Fluid Dynamics Branch and Director of the Aerodynamics and Propulsion Department of INTA, respectively, whose leadership has allowed to create a motivated working-team where I have the honor to belong to. I would like to thank you sincerely for your effort dealing with the budget reductions we have suffered in the last three years.
- Researcher Markus Widhalm and Prof. Norbert Kroll from the German Aerospace Center (DLR), where I had the great opportunity to stay 4 months (from April to July 2008) working on improving the parallelization of the TAU code. Their kindness and technical support have been crucial for the outcomes of this thesis.

The research described in this thesis has been partially supported under the INTA activity Termofluidodinamica (INTA's code: IGB99001), by the Spanish Ministry of Education and Science (MEC) under the DOMINO Project (CIT-370200-2007-22) and the DOVRES PROJECT, by the Ministry of Science and Innovation (MICINN) under the project ECO2010-22065-03-02, and under the high performance activity for the TAU code at $C^2A^2S^2E$ at the Institute of Aerodynamics and Flow Technology in Braunschweig. I would also like to thank the Complutense and Polytechnic University of Madrid for the collaboration.

LIST OF ACRONYMS

Table 1: List of Acronyms

Acronym	Definition
AD	Automatic Differentiation
ANNs	Artificial Neural Networks
ASICs	Application Specific Circuits
CAD	Computer Aided Design
CCD	Central Composite Design
CEP	Classic Evolutionary Programming
CFD	Computational Fluid Dynamics
CST	Class Shape Transformation
DE	Differential Evolution
DOE	Design of Experiments
EAs	Evolutionary Algorithms
EP	Evolutionary Programming
FEP	Fast Evolutionary Programming
FFD	Free Form Deformation
FLO	Fine Local Optimization
FPGAs	Field Programmable Gate Arrays
GA	Genetic algorithm
GPUs	Graphics Processor Units
HDL	Hardware Description Language
HPC	High Performance Computing
HSFC	Hilbert Space-Filling Curve
IGES	International Graphics System
IFEP	Improved Fast Evolutionary Programming
KG	Kriging
LES	Large Eddy Simulation
LHS	Latin Hypercube Sampling
MAEAs	Metamodel-assisted evolutionary algorithms
MARS	Multivariate Adaptive Regression Splines
MDO	Multi-Disciplinary Optimization
NURBS	Non Uniform Rational B-Splines
PCI	Peripheral Component Interconnect
PR	Polinomial Regression
PDEs	Partial Differential Equations
PSO	Particle Swarm Optimization
RANS	Reynolds Averaged Navier Stokes
RBF	Radial Basis Functions
RCB	Recursive Coordinate Bisection
RIB	Recursive Inertial Bisection
RMSE	Root Mean Square Error
SA	Spalart-Allmaras
SAE	Spalart-Allmaras with Edwards modification
SBGO	Surrogate-Based Global Optimization
SVMs	Support Vector Machines
SVR	Support Vector Regression

Contents

List of Figures

List of Tables

Part I

Motivation, objectives and state-of-the-art

Chapter 1

Introduction

1.1 Motivation

The challenges of the aeronautical industry in the near future will require new computational tools for the design of the type of aircraft that will be demanded by the European industry, according to the guidelines stated at the ACARE 2020 and 2050 flight paths [aca20, aca50]. The industry agrees that these objectives make necessary the design of an innovative aircraft shape. Efficient and accurate shape design optimization tools, able to consider novel concepts through the use of global optimization strategies and flexible geometry parameterizations, are becoming a must for the aeronautical industry.

Aircraft have to acquire new shapes and sizes to achieve such targets and for this reason, the VII European Framework Programme has defined three specific key objectives for Aeronautics [com11] :

- The greening of air transport. This involves both the global issue of climate change and the local issues of noise and air quality. The objective here is to halve the aircraft CO_2 emissions and perceived noise.

- Improving cost efficiency. This comprises all the costs that arise in the entire air system design and operation. The development costs and time-to-market aim to be reduced by 50%.

- Pioneering the air transport of the future. An improvement in design capabilities and design will allow industry to obtain better and more optimized airplanes than before.

The application of optimization in the field of aerodynamic design is progressively increasing. Aerodynamic design optimization is gaining interest motivated by the demands of handling sophisticated geometries, tackling realistic flight conditions, and satisfying increasing design objectives. Since almost all modern aerodynamic design activities rely on numerical simulation computer codes, the use of aerodynamic design optimization tools was strongly encouraged by the increasing capabilities of modern computers and the continuously improving numerical schemes, simulation, and optimization algorithms.

To underline the importance of shape design optimization within the commercial air transport industry, for example with the objective of drag reduction [eps09] , consider the task of delivering a payload between two destinations. Based on the Breguet range equation, which applies to long-range missions of jet aircraft, the airline could increase the payload by 7.6 %, and, therefore, the benefit, if the drag is decreased by 1%, while maintaining the same fuel consumption. This example illustrates that a 1% delta in the total drag is a significant change.

This is the reason why Computational Fluid Dynamics (CFD)-driven aerodynamic shape design has gained increased interest in the last decade [moh01, vas02, eps05, eps04, pei04, vas06]. With the current maturity of the CFD codes, their contribution to aerodynamic shape design has introduced a significant value in the industry, although there are still some issues to solve, specially related to their high computational cost and the limited capabilities of the automatic aerodynamic design tools.

In addition, in order to achieve the mentioned objectives as the drastic reduction of the fuel consumption and the CO_2 emissions, innovative concepts have to be fully addressed within the automatic design optimization process, going beyond current small modifications in the traditional configurations. Moreover, for a shortening of the time-to-market in the case of these advanced configurations, novel capabilities have to be integrated into the design tools and the efficiency of the design process has to be increased, in order to be exploited by the industry. Therefore, this field constitutes an active research topic nowadays, and different contributions [bra05, buc05, car06, cas07, cat07, aso09, for09, bom10, mar10, kam11, nem11] are aiming to extend the aerodynamic design capabilities, considering also multidisciplinarity, and the efficiency of the process.

1.2 State of the art

This section presents a description of the state-of-the-art in the technological fields addressed in this thesis. The structure of this section has been chosen to properly cover the main steps of the aerodynamic shape optimization process, from geometry parameterization, coarse grain surrogate-based global optimization to fine grain local optimization using control theory, and also to point out the main problems for the full application of these methods in an industrial environment, as, for example, the high requirement of computational resources. Finally, some of the current software tools used by industry for analysis will be commented.

1.2.1 High speed CFD simulations

In computer science, Moore's law predicts that the speed achievable on a single chip doubles every 18 months, and this statement has held true for decades. But unfortunately, in a near future, the increasing transistor density will no longer deliver comparable improvements in performance. Until 2004, standard general purpose hardware was centered on single core CPUs, and a steady growth of CPU frequency. From 2004, increased processor performance is a result of the introduction of multiple cores in CPU chips with little growth in CPU frequency [aer08]. Quad core CPUs are today used in commodity PCs, eight and twelve core CPUs are soon expected to be shipped. The proliferation of multicore processors and multi-processors computational platforms means that the software developers must incorporate parallelism into their programming, in order to achieve increased application performance and scalability [int12] . The High Performance Computing (HPC) community has detected the situation and new research lines in computer science are being intensively explored. They include the efficient management of multi-core systems, but also specialized processors and hardware within heterogeneous computing architectures, in which conventional and specialized processors work cooperatively.

For CFD applications, the increasing demands for accuracy and simulation capabilities produce an exponential growth of the required computational resources. In particular, the high complexity of some of these processes frequently implies very long computation times. For example, the analysis of a complete aircraft configuration using a Reynolds-Averaged Navier-Stokes (RANS) modeling, can require more than a day, even using modern high parallel computational platforms. Furthermore, using CFD within a design optimization process, or increasing the target precision through the use of Large Eddy Simulation (LES) models, usually increase the computational requirements up to an unaffordable level. This situation calls for an efficient implementation of CFD codes and a proper handling of parallel platforms.

The performance capabilities of computational resources have increased rapidly over recent years. In particular, the introduction of highly parallel systems has brought with it massive increases in the number of processing units, where it is now common to have many tens of thousands of cores available for users codes. This development raises a number of significant challenges for the parallel performance of CFD applications [ber87, hen93, kar98, ter04]. Recently, new parallelization [gou09, alr05, mav07, mav02, sil05, and09] and optimization [gup06, pal07, nak11] techniques have been introduced in order to address these challenges at several different stages of the calculation.

With respect to domain decomposition, there are several mesh partitioning software packages available for a wide variety of problems as, for example, the well known sequential graph partitioner MeTiS [met97]. As the mesh sizes that are to be partitioned reach up to several million points, even the mesh generation has to be performed in parallel via parallel mesh generators. Thus, both in order to adjust to memory constraints and to avoid migration of data, the partitioning has to be done in parallel as well. To fit such large meshes in the memory of cluster nodes, the mesh has to be partitioned among a very large number of cores. Unfortunately the partitioning performance of parallel graph partitioning packages such as Par-MeTiS [par02] and PT-SCOTCH [pts06] decline with increasing number of cores used in the partitioning process. Hierarchical partitioning scheme utilizing Zoltan [zol08] has demonstrated better results. Further details can be found in 2.3.2

Furthermore, in addition to these improvements on the code execution time and parallel scalability, it is also necessary to look for novel simulation platforms based on heterogeneous architectures, in which conventional processors and specific hardware modules work together. Some alternatives in HPC for scientific applications are the acceleration using Graphics Processor Units (GPUs) or Field Programmable Gate Arrays (FPGAs). Modern graphics hardware outperforms the traditional desktop CPU in terms of computational processing power by several orders of magnitude with a very attractive cost/performance ratio [nvi07], and have evolved into high performance parallel architectures capable of executing fast computations in a wide variety of fields [kru03, bol03, bel08, mar08, har04], as will be described in 2.4.1. On the other hand, reconfigurable computing, as FPGAs, is intended to bridge the gap between hardware and software. The performance data confirm [che04, guo04, tod05] that reconfigurable systems can deliver between 10x and more than 100x improvement in computational efficiency (compared to traditional processor based machines) for many problems. This is achieved by tailoring hardware allocations to match the needs of applications. Dynamically reconfigurable supercomputers can potentially contribute to important value metrics, including time-to-solution, by reducing design cycle time and porting costs [bon02, luk04, com02]. In particular, the application of GPUs [bra08, jep09, kam10, aso11, aso12, sae12] and FPGAs [and08, fus08, sut11, san11] to accelerate CFD simulations has been an increasing field of research in the last years, showing promising results.

1.2.2 Geometry parameterization

The mathematical description of the aircraft components as wings, nacelles, fuselages and control surfaces is a key decision during the design process of an aircraft. There are many different ways to describe and manipulate geometries, ranging from point definitions connected with straight lines, through splines and polynomials, Hicks-Henne bump functions [hic78, ste03], Non-Uniform Rational B-Splines surfaces (NURBS) [pie97, mar11], class/shape function transformation (CST) [kul07], and free form deformation.

For example, Meaux et al[mea04] used NURBS to optimize complex 3D geometries. On the other hand, Jameson et al[jam88] used surface mesh points and a gradient smoothing algorithm to successfully improve the performance of aircraft wings. Although many works have been done on aerodynamic shape optimization using various representation techniques, only a few have studied the effect of shape parameterization on the design process [lep01, son04, mou07, sam08, cat07, sri10] focusing on several key characteristics. For example, flexibility is interpreted as the ability to represent a wide range of different shapes, but the large number of design variables and the design space complexity might prevent some optimization algorithms from locating the global optimal solutions and/or making them poorly performing. On the other hand, methods with fewer design variables may not be able to generate optimal shapes in all detail and the optimal objective function value cannot be achieved. Most of these comparison conclude that B-spline are capable of accurately representing a large family of airfoils with few control points. B-spline and CST approaches offer advantages such as reduced number of design variables and continuous gradients over the mesh point approach, providing a crucial advantage also over the bump function approach. An additional advantage of the B-spline approach is that it is used in most CAD packages to represent the geometry. Therefore, it provides the most natural way to integrate the CAD geometry into the design process.

In addition, the freeform deformation technique has been introduced successfully in many shape optimizations [sam04, ron05] as a reliable technique for generating smooth surfaces with a low number of lattice box points. The box points are used as design parameters directly which can move independently. On the other hand, allowing more locality, the use of NURBS has been suggested [lep00, pai04, ben05, mou07] as an efficient and flexible parameterization, able to represent complex configurations, giving the optimizer enough freedom to converge to a wide range of possible optimal designs, and at the same time, preventing the risk of numerical noise. Figures 1.1 and 1.2 show an example of FFD and NURBS parameterization and their corresponding deformation for an ONERA M6 wing.

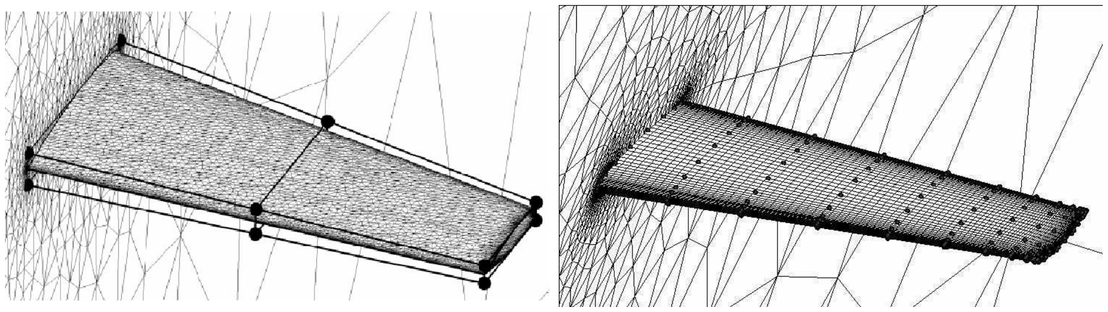


Figure 1.1: Inital FFD (left) and NURBS (right) parameterization of an ONERA M6 wing.

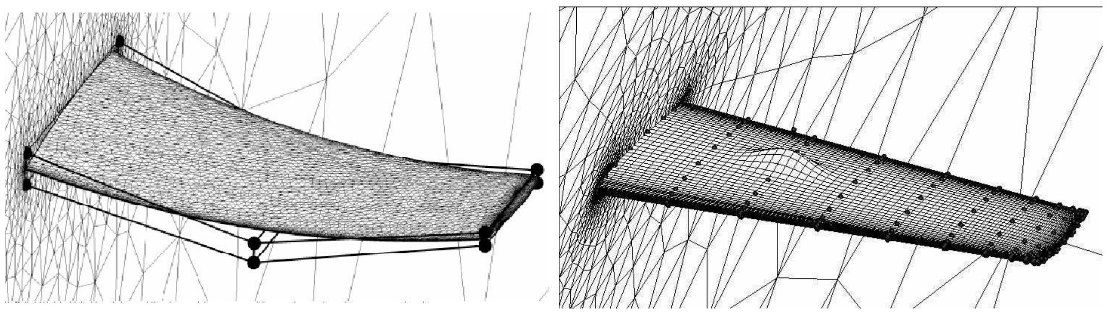


Figure 1.2: Deformed FFD (left) and NURBS (right) parameterization of an ONERA M6 wing.

Even if the computational cost of solving the sensitivity derivatives is independent of the number of design variables, the choice of the shape parameterization method is still extremely important in any design problem. This choice greatly affects both the performance and the accuracy of the shape optimization. Therefore, the initial parameterization of the geometry, i.e. the selection of the design variables, is a crucial step within the optimization process because it will determine the quality of the optimal solution to be found [and10].

As B-Spline parameterization, in particular NURBS, will be used for the fine grain local shape optimization in Chapter 4, the mathematical theory is briefly explained here.

NURBS parameterization

3D geometries can be parameterized with NURBS functions. The design is accomplished by directly moving the control points to modify the shape. The design variables are therefore the x, y and z locations of the control points. Equation 1.1 gives the NURBS surface as a function of the parameters η and ξ:

$$S(\xi,\eta) = \frac{\sum_i^I \sum_j^J U_{i,p}(\xi) V_{j,q}(\eta) w_{ij} C_{ij}}{\sum_i^I \sum_j^J U_{i,p}(\xi) V_{j,q}(\eta) w_{ij}} \tag{1.1}$$

where (ξ, η) are the parametric coordinates on the NURBS surface, C are the Cartesian coordinates of the control points, w are the weights of the control points and U and V are the basis functions that will be described in section 4.3.2.

Further details regarding NURBS and their use as parameterization for aerodynamic shape optimization with control theory will be described in Chapter 4.

1.2.3 Surrogate-based global aerodynamic shape optimization

Metamodel assisted evolutionary algorithms

The optimization methods, in general, can be classified into two main categories: deterministic and stochastic methods. Deterministic methods solve an optimization problem by generating a deterministic sequence of points converging to a global optimal solution. These methods converge quickly to the optimum, however they require the optimization problem having certain mathematical characteristics that may not exist in most computer simulation based global optimization problems. Therefore, global search methods are mostly based on stochastic optimization techniques, some of them population-based. The most commonly used population-based methods are the Evolutionary Algorithms (EAs; including Genetic Algorithms-GAs and Evolution Strategies-ES). However, there are also other alternatives, such as Particle Swarm Optimization (PSO) [ken95] or Bacterial Foraging Optimization (BFO) [mul02], Differential Evolution (DE) [sto97] etc...

Evolutionary algorithms (EAs) [duv04] are successful single- and multi-objective constrained optimization methods that can handle any kind of objective function and may accommodate any evaluation software as a black-box tool. Since, however, EAs ask for a large number of calls to the evaluation software in order to reach the optimal solution, they become very costly in applications with computationally demanding evaluation software. For this reason, EAs assisted by surrogate evaluation models (metamodels) have been devised [aso09, gad11]. The Metamodel–Assisted Evolutionary Algorithms (MAEAs) rely on inexpensive and, thus, approximate models of the problem-specific evaluation model. Figure 1.3 shows the flowchart of MAEAs.

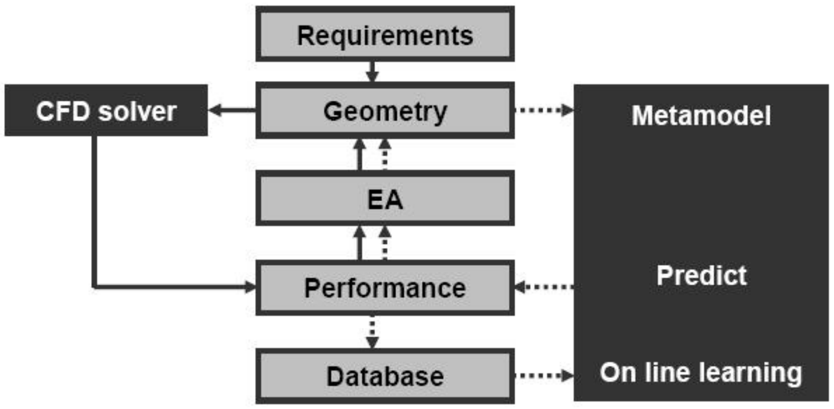


Figure 1.3: Flowchart of a Surrogate-based Global Optimization

Surrogate models are introduced as a cheap alternative that has a number of advantages, especially concerning the computational cost, memory and time budgets. A surrogate model replaces the simulation performed by computationally expensive codes in the sense that the search is directly coupled with a data base with a limited number of (previously, off-line, or on-line performed) simulations or snap shots. There are different kinds of surrogate modelling as for example Polynomial Regression (PR), Multivariate Adaptive Regression Splines (MARS), Gaussian Processes, Kriging (KG), Cokriging [zho10], Artificial Neural Networks (ANN) [wee05, mar10], Radial Basis Functions (RBF) [mor08] and Support Vector Machines (SVM) [and11], among others. A reference for recent advances in surrogate-based optimization techniques can be found in [for09], and a comparison of surrogate models for turbomachinery design in [pet07]. Also, surrogate modelling has been already applied for the design optimization of composite aircraft fuselage panels [van10]. In addition, the use of Kriging surrogate model in combination with evolutionary algorithms has been recently applied to the design of hypersonic vehicles [ahm10]. Furthermore, the use of Support Vector Regression algorithms (SVMr) [smo98, smo99] as metamodels has been applied to a large variety of regression problems, in many of them mixed with evolutionary computation algorithms [che11, sal11, jia12].

In MAEAs with off-line metamodels, the latter are trained beforehand, i.e. separately from the evolution which is exclusively based on them [bul99, jin02, buc05, won05]. The collection of the training data set requires the evaluation of a number of selected points in the search space, which is the computationally expensive task. The selection of training patterns is usually based upon systematic Design of Experiment techniques, such as factorial design, orthogonal arrays, etc...(see 1.2.3). Once a global metamodel has been trained, this acts as the exclusive low–cost evaluation tool during the EA–based search. Thus, the CPU cost of running an EA which relies only upon the trained metamodel is negligible, compared to the CPU cost for collecting the training samples and training the network. The optimal solution, according to the metamodel, must be exactly re–evaluated and, depending on the deviation, the model is re-trained or not. In MAEAs with on-line trained metamodels, the metamodel and the problem–specific model are used in an interleaving way during the evolution [gia02, bra05, jin05, emm06, szo09]. The training of metamodels is based on recorded (in a dynamically updated database) previously evaluated individuals. Then, the selected members of each generation are exactly re-evaluated in order to update the metamodel [kar06].

Apart from implementing metamodels which are capable of approximating the objective function value, MAEAs or SBGO methods in general, may also accommodate metamodels which provide information related not only to the objective function values predictions but also quantify the confidence of these predictions. This is the case of Gaussian Processes, including the widely used Kriging method [emm06, jon98]. The idea is simple: at a reasonable extra cost, a model such as Kriging provides also a measure of confidence for its prediction. It is reasonable that the confidence is expected to be higher if the training point density in the neighborhood of a newly proposed point is higher. Another important output of the metamodel is the variance of the output values and the average correlation between responses at neighboring points.

Current research on MAEAs focuses on the improvement of metamodels (by using artificial neural networks, Gaussian models, etc, or proposing metamodels variants [you08] based not only on the responses but also on the gradients) and different metamodels implementation schemes within the MAEA [aso09, lim10].

Model approximation

Approximation, or metamodeling, is the key in metamodel-based design optimization. The goal of approximation is to achieve a global metamodel as accurate as possible, at a reasonable cost. The most commonly used metamodeling techniques include Polynomials (linear, quadratic or higher), Splines, Multivariate Adaptive Regression Splines (MARS), Kriging, Radial Basis Functions (RBFs), Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and hybrid methods. In the following, only the techniques related with the work performed within this thesis are briefly described. For the rest of the methods, the related literature can be consulted [jin05].

- Neural Networks
  Neural networks have shown to be effective tools for function approximation. Both feed-forward multi-layer perceptrons and radial-basis-function networks have been widely used.

  - Multilayer perceptrons [nor05]
    An MLP with one input layer, two hidden layers and one output neuron can be described by the following equation:

    $$y = \sum_{l=1}^{L} v_l f\left(\sum_{k=1}^{K} w_{kl}^{(2)} f\left(\sum_{i=1}^{n} w_{ik}^{(1)} x_i\right)\right) \tag{1.2}$$

    where n is the input number, K and L are the number of hidden nodes, and $f(\cdot)$ is called activation function, which is usually the logistic function:

    $$f(z) = \frac{1}{1+e^{-az}} \tag{1.3}$$

    where a is constant.

  - Radial Basis Functions Networks [orr96]
    The theory of radial basis function (RBF) networks can also be traced back to interpolation problems. An RBF network with one single output can be expressed by Equation 1.4.

    $$y(x) = \sum_{j=1}^{N} w_j \phi\left(\parallel x - x^{(j)} \parallel\right) \tag{1.4}$$

    where $\phi(\cdot)$ is a set of radial basis functions, $\parallel \cdot \parallel$ is usually a Euclidean norm, the given samples $x^{(j)}, j = 1, ..., N$ are the centers of the radial basis function, and w_j are unknown coefficients. However, this model is expensive to implement if the number of samples is large. Therefore, a generalized RBF network is more practical:

    $$y(x) = \sum_{j=1}^{L} w_j \phi\left(\parallel x - \mu^{(j)} \parallel\right) \tag{1.5}$$

    The main difference is that the number of hidden nodes (L) is ordinarily smaller than the number of samples (N), and the centers of the basis functions ($\mu^{(j)}$) are also unknown parameters that have to be learned. Usually, the output of a generalized RBF network can also be normalized:

$$y(x) = \frac{\sum_{j=1}^{L} w_j \phi\left(\| x - \mu^{(j)} \|\right)}{\sum_{j=1}^{L} \phi\left(\| x - \mu^{(j)} \|\right)} \tag{1.6}$$

- Support Vector Machines [cla05]
  The theory of support vector machines is mainly inspired from statistical learning theory [vap98]. Major advantages of the support vector machines over other machine learning models such as neural networks, are that there is no local minima during learning and the generalization error does not depend on the dimension of the space. Given l samples $(x_i, y_i), i = 1, ..., l$, the construction of a model is reduced to the minimization of the following regularized ϵ-insensitive loss function:

$$L = \| w \|^2 + C \cdot \frac{1}{l} \sum_{i=1}^{l} \max(| y_i - f(x_i) | - \epsilon) \tag{1.7}$$

  where ϵ is the tolerable error, C is a regularization constant and f is the function to be estimated:

$$f(x) = w \cdot x + b \quad w, x \in \mathbb{R}^n \quad b \in \mathbb{R} \tag{1.8}$$

  The minimization of Equation 1.7 is equivalent to the following constrained optimization problem:
  minimize

$$\frac{1}{2} \| w \|^2 + C \cdot \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) \tag{1.9}$$

  subject to

$$\begin{aligned} ((w \cdot x_i) + b) - y_i &\leq \epsilon + \xi_i \\ y_i - ((w \cdot x_i) + b) &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, i = 1, ..., l \end{aligned} \tag{1.10}$$

  Further details can be consulted in 3.2.3 .

Data sampling in Engineering Optimization (DOE)

Sampling, as the first step in design of experiments, is crucial in exploring the characteristics of the physical system or black-box computer analysis and simulation model, in an efficient way. In general, experimental design techniques can be classified into two categories: classical and space filling [gar06, you08].

- Classical methods
  These methods focus on planning experiments so that the random error in physical experiments has minimum influence in the approval or disapproval of a hypothesis. Widely used classic experimental designs include factorial or fractional factorial [myr95], central com-

posite design (CCD) [che95], Box-Behnken [myr95] and alphabetical optimal [mit74, giu97]. These classic methods tend to spread the sample points around boundaries of the design space and leave a few at the center of the design space. As computer experiments involve mostly systematic error rather than random error as in physical experiments, a good experimental design should tend to fill the design space rather than to concentrate on the boundary. Simpson et al. [jin01] stated that a consensus among researchers was that experimental designs for deterministic computer analysis should be space filling.

- Space filling methods
  Space filling designs spread experiment points evenly throughout the design space. Four types of space filling sampling methods are relatively more often used in the literature. These are orthogonal arrays [hed99], various Latin Hypercube Sampling (LHS) designs [ye00], Hammersley sequences [mec02], and uniform designs [fan00].

  A comparison of these methods can be found in [you08, gar06]. Following the conclusions in [you08], LHS methods seem to be a good choice because of their good properties for large scale problems, their capability to provide uniformity and flexibility on the size of the sample, and their properties to handle sampling where input variables have specified probability distribution. In addition, they are often applied also in uncertainty analysis.

Validation of Surrogate models

Metamodels are to be validated before being used as a “surrogate” of the computation-intensive processes [gar06]. Model validation has been a challenging task, and it shares common challenges with the verification and validation of other computational models [roa98, obe00]. In the following, the main approaches for accuracy validation of surrogate models are described [gar06]. Meckesheimer et al. [mec01, kal97] studied the cross-validation method. One starts with a dataset, $S\{X, Y\}$, consisting of N input-output data pairs (x, y), where y is the model response at the design sample point, x, and N is the total number of model runs. In p-fold cross-validation, the initial data set is split into p different subsets, that is, $S\{X, Y\} = S1\{X1, Y1\}, S2\{X2, Y2\}, \ldots, Sp\{Xp, Yp\}$. Then, the metamodel is fit p times, each time leaving out one of the subsets from training, and using the omitted subset to compute the error measure of interest. A variation of p-fold cross-validation is the leave-k-out approach, in which all possible

$$\begin{pmatrix} N \\ k \end{pmatrix}$$

subsets of size k are left out, and the metamodel is fit to each remaining set. Each time, the error measure of interest is computed at the omitted points. This approach is a computationally more expensive version of p-fold cross-validation.

Mitchell and Morris [mit92] described how the cross-validation error measure could be computed inexpensively for the special case of k = 1; this is called leave-one-out cross-validation. Based on the observations from the experimental study conducted to assess the leave-k-out cross-validation strategy [mec02], a value of k = 1 was recommended for providing a prediction error estimate for RBF and low order polynomial metamodels, but not for kriging metamodels. Choosing k as a function of the sample size used to construct the metamodel (that is, $k = 0.1N$ or

$k = \sqrt{N}$) was instead recommended for estimating the prediction error for kriging metamodels.

Lin [lin04] found through intensive testing that the leave-one-out cross-validation is an insufficient measurement for metamodel accuracy. The leave-one-out cross-validation is actually a measurement for degrees of insensitivity of a metamodel to lost information at its data points, while an insensitive metamodel is not necessarily accurate. With leave-one-out cross validation we are in danger of rejecting an accurate metamodel that is also sensitive to lost information at data points.

Given that cross validation is insufficient for assessing models, employing additional points is essential in metamodel validation. When additional points are used for validation, there are a number of different measures of model accuracy. The first two are the root mean square error (RMSE, see Equation 1.11) and the maximum absolute error (MAX, see Equation 1.12), defined below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \hat{y_i})^2}{m}} \tag{1.11}$$

$$MAX = \max \mid y_i - \hat{y_i} \mid, i = 1, ..., m \tag{1.12}$$

where m is the number of validation points; $\hat{y_i}$ is the predicted value for the observed value y_i. The lower the value of RMSE and/or MAX, the more accurate the metamodel. RMSE is used to gauge the overall accuracy of the model, while MAX is used to gauge the local accuracy of the model. An additional measure that can be also used is the R square value, defined by Equation 1.13.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y_i})^2}{\sum_{i=1}^{m}(y_i - \bar{y_i})^2} \tag{1.13}$$

where $\bar{y}$ is the mean of the observed values at the validation points. Variations of the three measures exist in the literature [jin01] .

Comparison of surrogate models

There are several papers that compare the performance of different approximation models [gar06, jin05]. However, no clear conclusions on the advantages and disadvantages of the different models have been drawn. This is reasonable not only because the performance may depend on the problem to be addressed, but also because more than one criterion needs to be considered. The most important factors are accuracy, computational complexity and transparency. Although, there is no conclusion about which model is definitely superior to the others, some insights have been gained through a number of studies [sim01, jin01] .

Firstly, it is recommended to implement a simple approximate model for a given problem, for example, a lower order polynomial model to see if the given samples can be fit reasonably. If a simple model is found to underfit the samples, a model with higher complexity should be considered, such as higher order polynomials or neural network models. However, if the design space is high-dimensional and the number of samples is limited, a neural network model is preferred. Secondly, if a neural network model, in particular a multi-layer perceptrons network is used, it is necessary to consider regulating the model complexity and try efficient training methods.

In general, the Kriging models are more accurate for nonlinear problems but difficult to obtain and use. Kriging is also flexible in either interpolating the sample points or filtering noisy data. On the contrary, a polynomial model is easy to construct, clear on parameter sensitivity, and cheap to work with but is less accurate than the Kriging model [jin01]. The RBF model can interpolate sample points and at the same time is easy to construct. Recently, a new model called Support Vector Regression (SVR) was used and tested [cla05] . SVR achieved high accuracy over all other metamodeling techniques including Kriging, polynomial, MARS, and RBF over a large number of test problems.

In [you08] a comparison between different surrogate models is carried out, and the conclusions for the behavior of ANNs and SVMs as metamodels are:

- Excellent for very high dimensional problems.
- Need for efficient training algorithms or limited applications in case of expensive fitness functions.
- Best suited for approximating functions in regression-type applications.
- Can model a combination of continuous and discrete numerical variables.
- Accuracy is based on the quality and quantity of the data used in modeling.

1.2.4 Local gradient-based aerodynamic shape optimization

In local gradient-based optimization techniques, the goal is to minimize a suitable cost or objective function with respect to a set of design variables using the gradient information for obtaining the search direction. Nowadays, the gradients can be computed both efficiently and accurately with adjoint methods, and in that way, only a single adjoint flow computation is required to evaluate sensitivities of a cost function with respect to any number of design variables. In the past, the finite differences approach was used for gradient computation, and it implied a complete flow computation for the deformation of each design variable, which made the time required for an industrial optimization process not suitable. There are two possible adjoint implementations, using the continuous and the discrete formulations. In the continuous adjoint approach [cas07], pioneered by Jameson [jam88], one first formulates the adjoint PDEs and boundary conditions, which are then discretized. On the other hand, in the discrete adjoint method [ell97], the discretized governing equations are used to derive the optimality conditions. The main advantage of this approach is that Automatic Differentiation (AD) techniques [gri00] can be used to generate adjoint codes with very little effort irrespective of the complexity of the flow solver code [gil03, nem11]. Yet another feature of discrete adjoints is that the discrete realizations of the turbulence models are algorithmically differentiable. On the contrary, the constant eddy viscosity or the so-called frozen turbulence assumption is a common practice in deriving the continuous adjoint equations, which may result in inaccurate sensitivities and robustness problems. Just recently, [zym09] presented, for the first time in the literature of continuous adjoint methods, the full differentiation of one- and two-equation turbulence models such as the Spalart-Allmaras (SA) and, later on, the k-w model with the wall-function technique [zym10]. It demonstrated and quantified the gain by using the full differentiation approach or, equivalently, the error from relying on the frozen turbulence viscosity assumption. Several researchers [zym09, pet10] have used both the continuous and discrete approaches in a wide variety of applications ranging from the design of two-dimensional airfoils to complex aircraft configurations.

Figure 1.4 shows the flowchart of the traditional local gradient-based optimization.

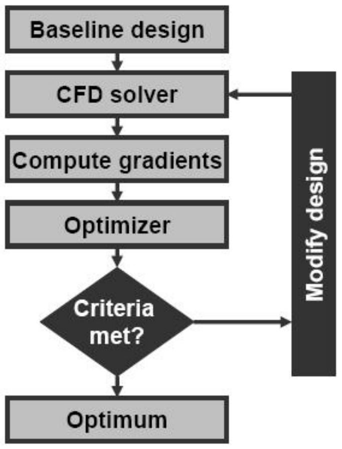


Figure 1.4: Flowchart of a Local Optimization

As this thesis will focus on the application of continuous adjoint method to efficiently compute gradients over the NURBS control points, as design variables, in the following, the mathematical background for computation of gradients via continuous adjoint [cas07, bre09] will be described.

Gradients via adjoint approach

Primal approach

Let the optimization problem be stated as

$$\min_{D} I(w, X, D) \tag{1.14}$$

subject to the constraint

$$R(w, X, D) = 0 \tag{1.15}$$

where I is a cost function such as lift or drag, D is a vector of design variables that control the shape of aircraft subject to aerodynamic design, $w(X, D)$ the vector of flow variables, $X(D)$ the computational mesh and $R(w, X, D)$ the residuals of the flow.

For a gradient based optimization strategy, the search for the minimum requires the total derivative of the cost function I with respect to the design variables D. This total derivative, also called the sensitivity, can be written as:

$$\frac{dI}{dD} = \frac{\partial I}{\partial X}\frac{\partial X}{\partial D} + \frac{\partial I}{\partial w}\frac{\partial w}{\partial D} \tag{1.16}$$

The first term of 1.16 expresses the direct effect of the geometry perturbation and the second term contains the effect of the flow alteration caused by the geometry perturbation on the cost function I. Solving the above equation can be done by applying finite differences which requires evaluations of the flow solver on n perturbed geometries, with n the number of design parameter. Alternatively, the adjoint approach allows a rapid evaluation of dI/dD for a large number of design variables D, without computing the flow solution on the perturbed geometry.

Dual approach

Instead of applying the chain rule to I, apply it to the Lagrangian:

$$L(w, X, D, \Lambda) = I(w, X, D) + \Lambda^T R(w, X, D) \tag{1.17}$$

where Λ are known as the adjoint variables. Since 1.15 holds for all D, $L = I$ for all Λ and all D. Hence,

$$\frac{dL}{dD} = \frac{dI}{dD} \quad \forall \Lambda, D \tag{1.18}$$

and so, applying the chain rule to L, the total derivative of I becomes:

$$\begin{aligned} \frac{dI}{dD} &= \left(\frac{\partial I}{\partial X}\frac{dX}{dD} + \frac{\partial I}{\partial w}\frac{dw}{dD} \right) + \Lambda^T \left(\frac{\partial R}{\partial X}\frac{dX}{dD} + \frac{\partial R}{\partial w}\frac{dw}{dD} \right) \\ &= \left(\frac{\partial I}{\partial X} + \Lambda^T \frac{\partial R}{\partial X} \right) \frac{dX}{dD} + \left(\frac{\partial I}{\partial w} + \Lambda^T \frac{\partial R}{\partial w} \right) \frac{dw}{dD} \end{aligned} \tag{1.19}$$

The unknown quantity dw/dD may be eliminated by choosing Λ such that

$$\left(\frac{\partial R}{\partial w} \right)^T \Lambda = -\left(\frac{\partial I}{\partial w} \right)^T \tag{1.20}$$

This is the flow adjoint equation, and must be solved only once to evaluate the gradient of a single I with respect to any number of design variables. The resulting Λ allows rapidly computing the total derivative using:

$$\frac{dI}{dD} = \left(\frac{\partial I}{\partial X} + \Lambda^T \frac{\partial R}{\partial X} \right) \frac{dX}{dD} \tag{1.21}$$

Continuous adjoint approach of 2D Euler equations

The steady compressible Euler equations on the domain Ω may be written for 2D flow:

$$\nabla \cdot F(w) = 0 \tag{1.22}$$

where $w = (\rho, \rho u, \rho v, \rho E)$ is the vector of conserved quantities, and the flux tensor F may be written:

$$F = \begin{bmatrix} \rho u & \rho v \\ \rho u^2 + p & \rho uv \\ \rho uv & \rho v^2 + p \\ \rho H u & \rho H v \end{bmatrix} \tag{1.23}$$

and ρ, u, v, E, p and H are the density, Cartesian components of velocity, total energy, pressure and enthalpy respectively, and the ideal gas relations are assumed. This equation is subject to slip boundary conditions on solid walls $\Gamma_w \subset \Gamma$

$$U \cdot n = 0 \quad \text{on } \Gamma_w \tag{1.24}$$

where $U = (u, v)$ and n is the surface normal vector. Furthermore we are interested in a cost function I given by

$$I(w) = \int_\Gamma g(w) d\Gamma \tag{1.25}$$

To derive the adjoint we multiply 1.22 by $\psi \in V$, where V is a Sobolev space containing the solution of 1.22, linearize about a given flow solution w_0, $w = w_0 + \phi$, and integrate by parts

$$\int_\Omega \psi^T \nabla \cdot (F'\phi) d\Omega = \int_\Gamma \psi^T n \cdot F'\phi d\Gamma - \int_\Omega \nabla\psi^T \cdot F'\phi d\Omega \tag{1.26}$$

where $F' = F'[w_0]$ is the derivative of F with respect to w evaluated at w_0. Therefore, the variational formulation of the adjoint problem is given by: find adjoint solution Ψ such that $\forall \phi \in V$

$$\int_\Gamma (n \cdot F'\phi)^T \psi d\Gamma - \int_\Omega (F'\phi)^T \cdot \nabla\psi d\Omega = I'[w_0]\phi \tag{1.27}$$

The continuous adjoint problem is therefore

$$\begin{aligned} -F'^T \nabla\psi = 0 \quad & \text{in } \Omega \\ (n \cdot F')^T \psi = g'^T \quad & \text{on } \Gamma \end{aligned}$$

The singularity of F' on slip walls leads to the well-known result that not all choices of g result in a well-posed adjoint problem.

Given the adjoint field ψ the derivative of I with respect to any design variable D may be written as in Equation 1.28

$$\frac{dI}{dD} = \int_\Gamma \frac{\partial g}{\partial D} d\Gamma + \int_\Omega \psi \frac{\partial}{\partial D} \nabla \cdot F d\Omega \tag{1.28}$$

which is notable for not containing any total derivatives of the flow solution dw/dD and, therefore, it makes the computation of the sensitivities essentially independent of the number of design variables.

Further details, as well as the formulation of adjoint sensitivities over the NURBS control points, can be read in 4.4.

1.2.5 Hierarchical optimization strategy for aerodynamic design

In order to exploit the advantages of both global and local search optimization methods, a hierarchical strategy can be employed. In a first stage, global optimization techniques, such as Evolutionary Algorithms (EAs), supported by surrogate models, search the whole design space for one or several viable global optimum designs. In a second stage, the optimization process is carried out with local gradient-based optimization techniques (i.e. adjoints) for a finer improvement of the geometry. Figure 1.5 shows the flowchart of a hierarchical global/local optimization.

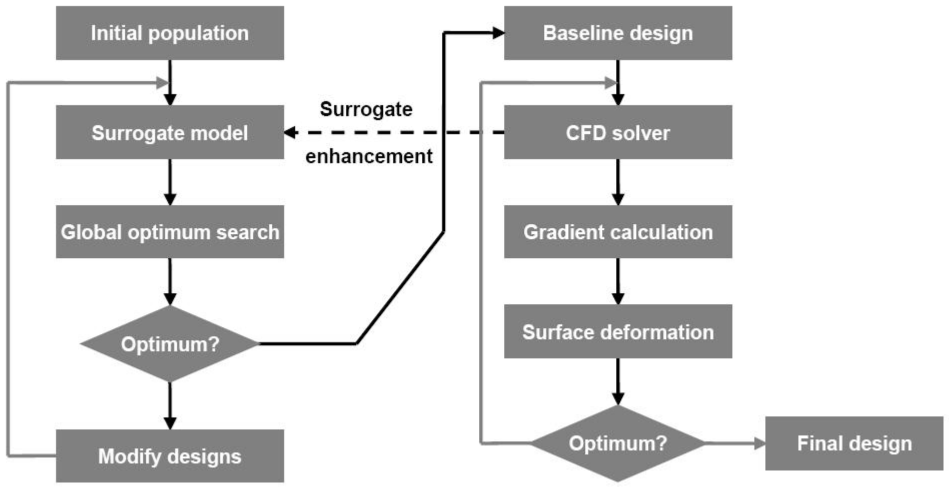


Figure 1.5: Flowchart of a Hierarchical Global/Local Optimization

The term “hierarchical” is used to denote the combined use of heterogeneous search methods (for instance, stochastic methods for the exhaustive search of the design space along with gradient-based methods for the refinement of promising solutions) [kam11]. Metamodel-assisted memetic algorithms [ong03, ong06, zho07, kon11] are also hybrid schemes that combine the use of global and local optimization methods [car06, bom10, kam11].

The first steps in the combination of this two stage global-local optimization strategy for a wing-body configuration, employing Response Surface (RS) and Kriging methods, in conjunction with EAs, have been carried out in [yim08]. Also, [iul09] offers an example of stochastic optimization, for global approach, and local refinement with gradients for the design of supersonic transport jet.

1.2.6 Software tool for aerodynamic analysis: The unstructured DLR TAU code

The DLR TAU [ger97, sch06] code is a modern software system for the prediction of viscous and inviscid flows around complex geometries from low subsonic to hypersonic flow regime, employing hybrid unstructured grids composed of tetrahedrons, pyramids, prisms and hexahedrons. The system, in the following just called TAU, is composed of a number of modules and libraries to allow easier development, maintenance and reuse of the code or parts of it. The main modules of

TAU can both be used as stand-alone tools with corresponding file I/O or within a Python [pyt08] scripting framework allowing for inter-module communication without file-I/O. In addition, to allow efficient simulation of complex configurations with several ten million grid points, TAU has been parallelized based on domain decomposition by using the message passing concept MPI.

The most significant modules of TAU are, firstly, the pre-processing which is used to prepare the metric of the grid. This edge-based data structure makes the solver independent of the element types in the primary grid and enables the use of a multi-grid technique based on the agglomeration approach.

Secondly, the flow solver is based on the compressible Navier-Stokes equations, employing Low Mach number pre- conditioning to extend its use into the incompressible flow regime. The spatial finite volume discretization is second-order accurate both for the viscous and inviscid terms, where the latter are computed based on central or a variety of upwind schemes. Steady solutions are obtained via time integration based either on explicit Runge-Kutta schemes or an implicit LU-SGS scheme, both with additional convergence acceleration by multi-grid. The turbulence models implemented include linear and non-linear eddy viscosity models (EVM) spanning from one- and two-equation EVM through Reynolds-stress models to hybrid RANS-LES models. The standard turbulence model used is the Spalart-Allmaras [spa92] model yielding highly satisfactory results for a wide range of applications while being numerically robust. The two equation models are k-ω based, with the Menter SST [men94] model being most popular. For time accurate computations, the dual time stepping approach of Jameson [jam91] is employed.

Finally modules are available for grid adaptation, which refines and de-refines the computational grid with different indicator functions, grid deformation to propagate the deformation of surface grid points to the surrounding volume grid and many more.

1.3 European efforts in improving the aerodynamic shape design

Table 1.1 shows a summary [gad11] and a short description of previous EC-funded projects focusing on improving the aerodynamic analysis and design capabilities and Figure 1.6 shows the roadmap of previous EU efforts regarding shape design.

With respect to the aerodynamic design process, the achievements of AEROSHAPE and INGENET constituted the baseline for methods. With respect to the design of novel configurations and concepts, there were important outputs and recommendations from HELIX, NEFA, VELA and NACRE projects, as well as the achievements of EUROLIFT, DESIREH and AWAHL for designing high lift devices. Regarding surrogate models, ALEF and ACFA 2020 produced interesting outcomes, the latter in the application of neural networks for active control. In addition, different strategies for the industrial design process were evaluated in VIVACE and CRESCENDO.

1.4 Objectives and main contributions of this work

This thesis focuses on the aerodynamic design of aeronautical shapes using an advanced strategy through efficient global and local search methods enhanced with high performance computing techniques. The objective is to enrich the current design capabilities by exploring extensively

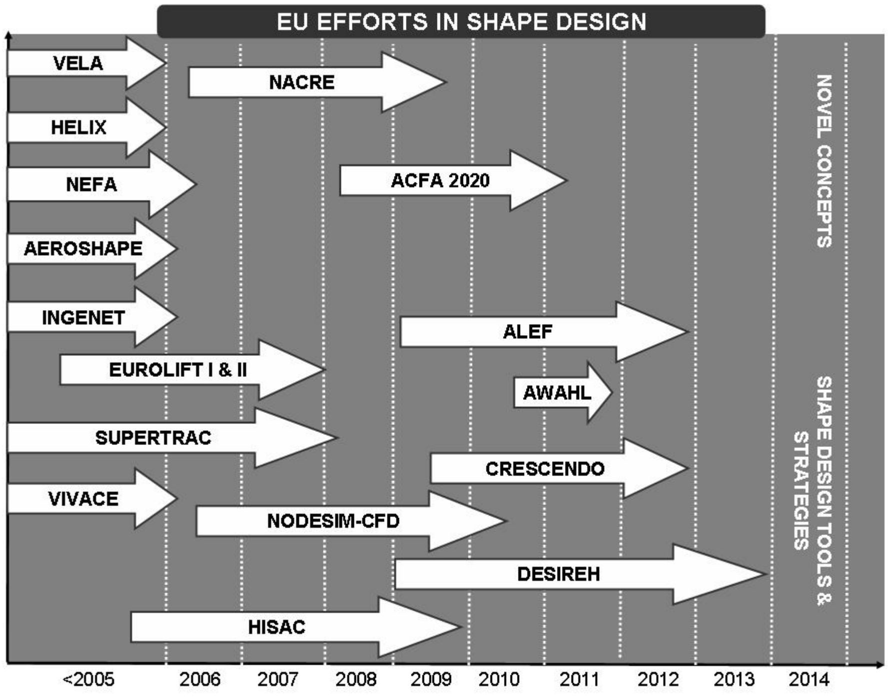


Figure 1.6: Roadmap of previous EU projects

the design space and providing new configurations, with improved aerodynamic efficiency.

The strategy for obtaining an efficient aerodynamic design process will involve three key stages:

- CFD acceleration: First, the improvement of the process efficiency is addressed through code optimization and parallelization using a balanced domain decomposition. At first, a profiling analysis of the most time-consuming processes of a RANS flow solver on a three-dimensional unstructured mesh is performed. Then, a study of the code scalability is considered and new partitioning algorithms are tested to show the most suitable algorithms for the selected applications.

- Coarse grain surrogate-based global optimization (SBGO): Second, the design space will be broadly analyzed to get the global optimal configuration. In this phase, the use of global optimization techniques will be considered due to their ability to work with noisy objective functions without assumptions on continuity and their high potential to find the global optimum of complex problems. In addition, the inherent parallelization of these algorithms will provide additional key advantages.

- Fine grain local optimization (FLO): An initial configuration will be also analyzed within a finer design phase. In this step, local optimizations will be performed in order to locally improve the design, and methods for saving of computational effort, as adjoints, will be applied.

Contributions to the state-of-the-art are:

- Novel approach for an efficient aerodynamic optimization of aeronautical wing profiles, consisting of an Evolutionary Programming algorithm hybridized with a Support Vector regression algorithm (SVMr) as a metamodel.
- Assessment of specific issues as precision, dataset training size and feasibility of the complete approach and the potential of global optimization methods (enhanced by metamodels) for aerodynamic shape design of aeronautical configurations.
- CFD acceleration through code optimization techniques and efficient domain decomposition algorithms.
- Application of a NURBS parameterization in the fine grain aerodynamic design coupled with control theory.

1.5 Structure of this thesis

The rest of this thesis is organized in three technical parts:

- First, in Chapter 2, the numerical simulation tool, in this case the CFD TAU code, key piece in the optimization process, is analyzed in order to find computational bottlenecks and several acceleration techniques are applied to reduce the time required for computing the solution for each design variable.
- Second, in Chapter 3, a coarse grain global optimization, surrogate-based, is performed. The design space is broadly analyzed to get a first approximation to the global optimal configuration.
- Third, in Chapter 4, a fine grain gradient-based local optimization is performed in order to locally improve the initial design.

The final assessment of results is done within each chapter by applying the proposed approach to the aerodynamic design optimization of different configurations. Finally, Chapter 5 will summarize the main outcomes of this research work.

Table 1.1: Related funded projects in the European Framework Programme

Acronym	Objective
AEROSHAPE	Multi-point aerodynamic shape optimization. Increase efficiency by introducing efficient techniques for aerodynamic shape optimization (adjoint formulation, evolutionary algorithms, surrogate-based optimization, . . .)
EPISTLE	European project for improvement of supersonic transport efficiency.
HELIX	Innovative aerodynamic high lift concepts. Explore new approaches to generate sufficient low speed aerodynamic performance
INGENET	Evolutionary Computing for Industrial Design. Promote interactions between Evolutionary Methods and Industrial Applications by evaluating and comparing different methodologies
NEFA	Assess V-empennage type on aircraft
VELA	Very Efficient Large Aircraft. Innovative transport aircraft concepts and associated development tools for Very Efficient Large Aircraft
EUROLIFT	Investigation of design optimization for high-lift configurations.
HISAC	Establish the technical feasibility of an environmentally compliant supersonic small size transport aircraft.
NACRE	New Aircraft Concepts Research. Develop solutions at a generic aircraft component level (cabin, wing, power plant system, fuselage), which will enable the results to be applicable for a range of new aircraft concepts
NODESIM-CFD	Research on uncertainties within the Computational Fluid Dynamics (CFD) simulation process
VIVACE	Development of methodologies in aeronautic engineering and aircraft and aero-engine design.
ACFA 2020	Innovative active control concepts for efficient 2020 aircraft configurations like the blended wing body (BWB) aircraft
ALEF	Aerodynamic loads estimation at extremes of the flight envelope. Enable the aeronautical industry to create complete aerodynamic data sets.
AWAHL	Advanced Wing And High-Lift Design. Wing/pylon/nacelle/HLD for advanced regional TF A/C configuration by multidisciplinary design
CRESCENDO	Collaborative and Robust Engineering using Simulation Capability Enabling Next Design Optimization.
DESIREH	Development of numerical design tools and experimental measurement techniques with the objective to improve the industrial process for the design of high lift devices
FFAST	Development of reduced order modeling strategies for unsteady aerodynamic and aeroelastic simulation
NOVEMOR	Novel Air Vehicles Configurations: From Fluttering Wings to Morphing Flight. Investigation on novel air vehicle configurations
SUPERTRAC	SUPERsonic TRAnsition Control. Explore the possibilities of viscous drag reduction on supersonic aircraft wings by delaying transition

Part II

Proposed contributions with numerical results

Chapter 2

Achieving High Speed CFD simulations

2.1 Introduction

Scientific computing with its core ingredients, modeling, simulation and optimization, is regarded by many as the third pillar of science, complementary to experiment and theory. In aeronautical engineering, the consistent use of mathematical and computational methods to simulate complex processes has become indispensable to save energy, reduce costs and pollution, and to increase safety. However, the high complexity of some of these processes frequently implies very long computation times. In particular, the analysis of a complete aircraft configuration including all relevant payload elements and flight components, even using a Reynolds-Averaged Navier-Stokes (RANS) modeling, at present still requires a huge computational effort, even using the modern high parallel computational platforms. Thus, reducing the time for aerodynamic analysis is one of the most important challenges of current research in CFD.

An efficient implementation for codes based on unstructured meshes is still a challenging task. In comparison to structured solvers [kro01], with their lower memory usage and block-structured data, their computational rates and scalability were for many years out of reach for any unstructured solvers. Through the use of effective edge-based structures, the memory requirements were reduced, and by using grid data-reordering techniques for a banded matrix, the efficiency was increased remarkably. Especially edge-based data structures enabled a homogeneous data-structure independent on the grid volume elements. Additionally, a cache optimization for PCs successfully increased the throughput. With all these improvements, unstructured solvers have matured enough to be applied for industrial applications [mav07, mav02, sch06, alr05].

Apart from these advantages, improvements in the computational speed can also be addressed on many levels. In this work, three different strategies are proposed, ranging from the basic code optimization to parallelization up to a new hardware architecture.

Code optimization is one of the first objectives in performance improvement, and it is often less considered with respect to other important goals such as stability, maintainability, and portability. This simple level of optimization is beneficial and should be always applied including an efficient implementation, reduction of operations, pre-computation of expensive variables and non-redundant interfaces. Code optimization is based on *execution profiling* while performing a time measurement of bottlenecks in the code. Applying optimization strategies over the most time consuming algorithms of the code can provide important reductions of the execution time [wyl07, gro00].

In recent years, much of the attention has been focused on methods for parallel computers to reduce the computation time by taking advantage of concurrent processing of data in different regions of the domain, and to increase the resolution of the model by using the larger memory available in parallel computers. To effectively exploit the current capabilities that parallel computers offer, with several hundreds or even thousands of processing units, it is important that the data has to be distributed over the processors in a balanced manner, so that each processor will complete its workload at approximately the same time to prevent idling of processors.

This distribution must be in the manner that the number of assigned elements to each processor is the same, and the number of adjacent elements assigned to different processors is minimized. The goal of the former condition is to balance the computations among the processors. The goal of the latter condition is to minimize the communications resulting from the placement of adjacent elements to different processors. Therefore, efficient partitioning algorithms for highly unstructured graphs are crucial for gaining fast solutions in a wide range of applications areas on parallel computers, and, particularly, in large-scale CFD simulations [mav07, mav02, zol08, sil05].

In addition, adaptive scientific computations require that periodic repartitioning, known as load balancing, occur dynamically to maintain load balance. A classic example is the simulation based on adaptive mesh refinement, in which the computational mesh changes between time steps. The difference is often small, but over time, the cumulative change in the mesh becomes significant. An application may therefore periodically re-balance, that is, move data among processors to improve the load balance. This process is known as dynamic load balancing or repartitioning and should be considered in modern applications.

Furthermore, a new alternative to boost the performance is to consider new heterogeneous architectures for high performance computing [put03, mah04, and08], combining conventional processors with specific hardware to accelerate the most time consuming functions.
Hardware acceleration gives best results, in terms of overall acceleration and value for money, when applied to problems in which:

- Condition A. A great amount of the computational effort is concentrated in a small portion of the whole code, and the computations have to be performed several times for a huge set of data.

- Condition B. The communication and I/O times are small with respect to the total computational cost [str06].

CFD simulations seem to gather all these desirable characteristics since:

- The numerical solution of the flow equations is based on a flux interchange between the discrete volumes that represent the physical domain. The numerical flux computation for each discrete volume requires several hundred floating-point operations, and each operation is repeated several million times in a real aeronautical simulation (Condition A).

- On the other hand, the numerical solution of the flow has data locality. This means that the computation to be performed at a certain location of the domain depends only on data that is located in a small neighborhood around it. Communication between the processors is required because the solutions on the subdomains depend on each other, but only on the subdomain boundaries. This is the reason why communication costs are small with respect to the overall cost (Condition B).

Therefore, CFD simulations are suitable to be executed, with expected performance acceleration, on a platform with a heterogeneous architecture [and08], where a specific hardware module is configured to perform the most time-consuming tasks, in order to decrease the overall computation time.

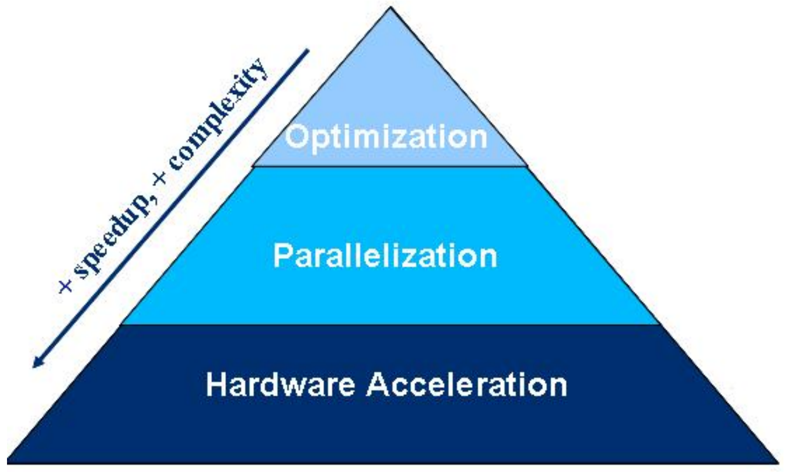


Figure 2.1: Scheme of the three-level code optimizations proposed for CFD

In this section, the performance of the DLR TAU code is analyzed and optimized. The improvement of the code efficiency is addressed through three key activities: Optimization, parallelization and hardware acceleration. At first, a profiling analysis of the most time-consuming processes of the Reynolds Averaged Navier Stokes flow solver on a three-dimensional unstructured mesh is performed. Then, a study of the code scalability with new partitioning algorithms are tested to show the most suitable partitioning algorithms for the selected applications. Finally, a short feasibility study on the application of FPGAs for the hardware acceleration of Computational Fluid Dynamics (CFD) simulations is presented.

2.2 Optimization: Analysis and remodeling of most time consuming algorithms

In the following, a performance analysis of the most time consuming functions in the flow solver will be outlined. This preliminary step, before considering any partitioning algorithm, is relevant to determine the flow solvers bottlenecks. It is important to be aware of computational routines which might hinder any effort in applying sophisticated grid partitioning. A flow solver is composed of very different algorithms for e.g. residual computation, boundary conditions treatment and pseudo-time integration. But the management of parameter settings, memory allocation and freeing and creating solution output can become additionally crucial if not well considered.

A flow solver profiling determines which algorithms are lacking in their efficiency. First, the flow solvers equations and discretization are presented in order to illustrate the various routines from the profiling later on.

2.2.1 Governing equations

The considered equations are the mass-weighted (Favre) averaged three-dimensional unsteady Navier-Stokes equations with a Spalart-Allmaras one equation turbulence model [spa92] with Edwards [edw96] modification (SAE). They may be written for an open physical domain $\Omega \subset \mathbb{R}^3$ enclosed by a smooth boundary $\partial\Omega$ under the time-independent coordinate $\mathbf{x}$ within a velocity field of

$$\mathbf{U}(\mathbf{x}) = \left[u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})\right]^T, \quad \mathbf{x} = \left[x, y, z\right]^T,$$

in integral and conservative form like

$$\int_\Omega \frac{\partial \mathbf{w}}{\partial t}\,\mathrm{d}|\Omega| + \int_{\partial\Omega} \left(\mathbf{F}_c \cdot \mathbf{n} - \mathbf{F}_v \cdot \mathbf{n}\right)\mathrm{d}|\partial\Omega| = \int_\Omega \mathbf{Q}\,\mathrm{d}|\Omega|, \tag{2.1}$$

where $\mathbf{n}$ denotes the face normal vector and $\mathbf{w}$ is the conservative state variables vector

$$\mathbf{w} = \left[\rho, \rho u, \rho v, \rho w, \rho E, \rho\tilde{\nu}\right]^T. \tag{2.2}$$

The flux density vectors of convective $\mathbf{F}_c$ and viscous $\mathbf{F}_v$ fluxes are given by:

$$\mathbf{F}_c \cdot \mathbf{n} = \begin{bmatrix} \rho V_n \\ \rho u V_n + p n_x \\ \rho v V_n + p n_y \\ \rho w V_n + p n_z \\ \rho H V_n \\ \rho\tilde{\nu} V_n \end{bmatrix}, \quad \mathbf{F}_v \cdot \mathbf{n} = \begin{bmatrix} 0 \\ \tau_{xj} \\ \tau_{yj} \\ \tau_{zj} \\ \Theta_j \\ \tau^t_j \end{bmatrix} \cdot \mathbf{n}, \tag{2.3}$$

including $V_n = \langle \mathbf{U}, \mathbf{n} \rangle$ the normal velocity over domain boundary $\partial\Omega$. Assuming a calorically perfect gas, the relation between the static pressure and temperature follows from the equation of state $p/\rho = \mathcal{R}T$, $\mathcal{R}$ is the specific gas constant, which closes the system. With the total enthalpy H and the relationship between static pressure and state variables by

$$H = E + p/\rho, \quad p = (\gamma - 1)\rho\left(E - \frac{|\mathbf{U}|^2}{2}\right), \tag{2.4}$$

the specific total energy E can be described. The viscous fluxes can be expressed by introducing the viscous shear-stress tensor $\tau(\mathbf{U}, \nabla\mathbf{U})$ together with the rate-of-strain tensor $\mathbf{S}$ which is

$$\begin{aligned} \tau_{ij} &= 2\mu_{eff}\mathbf{S}_{ij}, \\ \mathbf{S}_{ij}(\mathbf{U}, \nabla\mathbf{U}) &= \frac{1}{2}(\nabla\mathbf{U} + \nabla\mathbf{U}^T) - \frac{1}{3}\nabla\mathbf{U}I. \end{aligned} \tag{2.5}$$

The corresponding viscous energy flux Θ is then

$$\Theta = \left(\sum_{k=1}^{3} \mathbf{U} \tau_{jk} \right) + \kappa_{eff} \frac{\partial T}{\partial \mathbf{x}}, \quad j = 1, 2, 3, \tag{2.6}$$

μ_{eff} and κ_{eff} represent the effective viscosity and effective thermal conductivity given by their sums of laminar and turbulent parts

$$\mu_{eff} = \mu_l + \mu_t \quad \text{and} \quad \kappa_{eff} = c_p \left(\frac{\mu_l}{Pr} + \frac{\mu_t}{Pr_t} \right), \tag{2.7}$$

where Pr denotes the Prandtl number which for air is 0.92, Pr_t is the turbulent Prandtl number specified as relation between apparent turbulent shear stress and apparent turbulent heat flux and c_p is the specific heat capacity at constant pressure. μ_l, laminar viscosity, can be evaluated with a further relationship provided through Sutherland's law for molecular viscosity and is defined by

$$\mu_l = \mu_{l,ref} \left[\frac{T}{T_{ref}} \right]^{\frac{3}{2}} \left[\frac{T_{ref} + T_S}{T + T_S} \right], \tag{2.8}$$

with the Sutherland temperature $T_S = 110.4K$ for air. Modeling the turbulent viscosity $\mu_t = \mu_t(\rho\tilde{\nu})$ mainly involves a wall damping function based on empirical investigations with dependency on unknown turbulent quantity $\tilde{\nu}$. Turbulent diffusion indicated by τ_i^t being obtained based on Spalart-Allmaras one equation model and their contribution will read

$$\tau_i^t = \frac{(\mu_{eff} + \rho\tilde{\nu})}{\sigma} \frac{\partial \tilde{\nu}}{\partial x_i}, \tag{2.9}$$

with the constant coefficient $\sigma = 2/3$. Finally, the missing source term $\mathbf{Q}$ occurring on the right hand side of (2.1) can be further split into

$$\mathbf{Q}(\mathbf{x}) = \mathbf{Pr}(\mathbf{x}) - \mathbf{De}(\mathbf{x}) + \mathbf{Di}(\mathbf{x}), \tag{2.10}$$

where **Pr** corresponds to production, **De** is destruction and **Di** is the non-conservative diffusion term. These three terms are then deduced by the particular turbulence model used, in this case Spalart-Allmaras one equation model with Edwards modification.

2.2.2 Finite volume discretization

Applying an unstructured edge-based cell-vertex finite-volume discretization, (2.1) is solved using a spatial central numerical flux to compute the residual $\mathbf{R}$

$$\mathbf{R}(\mathbf{w}(\mathbf{x})) = \int_{\partial\Omega} (\mathbf{F}_c \cdot \mathbf{n} - \mathbf{F}_v \cdot \mathbf{n}) \, \mathrm{d}|\partial\Omega| - \int_{\Omega} \mathbf{Q} \, \mathrm{d}|\Omega|, \tag{2.11}$$

with mixed second- and fourth-order dissipation operators after the scheme of Jameson, Schmidt, Turkel (JST) [jam81] . The convective flux $\mathbf{F}_c$ over a domain boundary $\partial\Omega$ of a control volume

Ω, which are formed by a median dual grid approach, is written

$$\mathbf{F}_{c,JST} \cdot \mathbf{n}_{ij} = \frac{1}{2}(\mathbf{F}_c(\mathbf{w}_i) \cdot \mathbf{n}_{ij} + \mathbf{F}_c(\mathbf{w}_j) \cdot \mathbf{n}_{ij}) - \frac{1}{2}|\mathbf{A}_{ij}|[\mathbf{w}_i - \mathbf{w}_j], \tag{2.12}$$

where $|\mathbf{A}_{ij}|$ is, in case of scalar dissipation, the maximum absolute convective eigenvalue described as

$$|\mathbf{A}_{ij}| = |V_n| + c|\partial\Omega|, \;\; c = \sqrt{\gamma\frac{p}{\rho}}, \tag{2.13}$$

with c the speed of sound. The neighbors of point i are described by $N(i)$ and the face between the points i and j is ij with its adjacent normal vector $\mathbf{n}_{ij}$. Artificial dissipation is added with expression $[\mathbf{w}_i - \mathbf{w}_j] = \mathbf{D}_{ij}$ and is computed as

$$\mathbf{D}_{ij} = \phi_2(\mathbf{w}_j - \mathbf{w}_i) - \phi_4(L_j(\mathbf{w}) - L_i(\mathbf{w})) \tag{2.14}$$

$$L_i(\mathbf{w}) = \sum_{j \in N(i)} (\mathbf{w}_j - \mathbf{w}_i), \tag{2.15}$$

$$\phi_2 = \phi_2(p_{ij}, \lambda, \epsilon_2), \;\; \phi_4 = \phi_4(\epsilon_2, \epsilon_4, \lambda) \tag{2.16}$$

ϕ_2 acts as shock switch and ϕ_4 guarantees stability of the scheme in smooth regions. The coefficients are computed from pressure ratio p_{ij}, convective eigenvalues λ and constant coefficients $\epsilon_2 = 1/2$ and $\epsilon_4 = 1/64$. (2.14) is displayed simplified in contrast to the one used in TAU which further includes scaling factors for $\phi_{2,4}$ to increase robustness. (2.15) represents an undivided Laplacian from conservative variables and is evaluated over the immediate neighbors j of node i in a previous step before actual flux integration is performed.

Viscous flux $\mathbf{F}_v$ contributions are discretized in the same straightforward manner and read simplified as

$$\mathbf{F}_v \cdot \mathbf{n}_{ij} = \sum_{j \in N(i)} (\mathbf{F}_v \cdot \mathbf{n}_{ij})(\mathbf{w}_i, \mathbf{w}_j, \nabla\mathbf{w}_i, \nabla\mathbf{w}_j). \tag{2.17}$$

Required gradients from the flow variables $\nabla\,\mathbf{w}$ in (2.17) can be computed using the Green-Gauss method

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{w}^{GG} \approx \frac{1}{vol(\Omega_i)} \sum_{j \in N(i)} \frac{1}{2}(\mathbf{w}_i + \mathbf{w}_j)\mathbf{n}_{ij}\,|\partial\Omega|, \tag{2.18}$$

where $|\partial\Omega|$ is the area from domain boundary $\partial\Omega$ or with a *thin shear layer* (tsl) approximation:

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{w}^{TSL} \approx \sum_{j \in N(i)} \frac{\mathbf{w}_j - \mathbf{w}_i}{\mathbf{x}_j - \mathbf{x}_i}. \tag{2.19}$$

The last part to discretize concerns about turbulent source terms in (2.11). Again they are obtained similarly and can be written as

$$|\Omega_i|\mathbf{Q} = |\Omega_i|\mathbf{Q}(\mathbf{w}_i, \nabla\mathbf{w}_i), \tag{2.20}$$

where dependency from gradients occurs for $\mathbf{Di}(\mathbf{x}, t)$ representing diffusion and for computing the vorticity magnitude for production term $\mathbf{Pr}$, respectively. In fact, noteworthy is that turbulence models are strongly coupled to the mean flow equations, for further details see [lan11] . In principle, evaluating $\mathbf{R}$ is split into routines for convective and viscous part of the solution system. Furthermore, in case of turbulent flows, point sources $\mathbf{Q}$ and diffusion flux τ_i^t are added in a separate function. In addition, routines have to be taken into account for pre-computing convective eigenvalues, residual smoothing steps and helper functions for conversions between primitive and conservative variables.
Using a multi-step discretization in time, a large set of nonlinear equations is formed and marched to steady state in pseudo time using an agglomerated multi-grid algorithm [mav98] within each time step. Two pseudo time integration schemes are implemented in the TAU code. The explicit Runge-Kutta based time integration scheme [jam81] is applied like:

$$\Delta \mathbf{w}^n = -\frac{\Delta t}{\text{vol}(\Omega)}\mathbf{R} \tag{2.21}$$

$$\begin{aligned} \mathbf{w}^{(0)} &= \mathbf{w}^n \\ \mathbf{w}^{(a)} &= \mathbf{w}^{(0)} - \alpha_a \Delta t (\text{vol}(\Omega)\mathbf{R})^{a-1} \\ \mathbf{w}^{n+1} &= \mathbf{w}^{(a)} \end{aligned} \tag{2.22}$$

where n and $n+1$ are the actual and next integration points and a is related to the number of Runge-Kutta steps. The semi-implicit LUSGS scheme [jam87, dwi05] is applied in the manner:

$$\left(\frac{\text{vol}(\Omega)}{\Delta t} + \frac{\partial \mathbf{R}(\mathbf{w}^n)}{\mathbf{w}}\right)\Delta \mathbf{w}^n = -\mathbf{R}(\mathbf{w}^n), \tag{2.23}$$

and the term in parenthesis before $\Delta \mathbf{w}^n$ is called the implicit operator. Later on, the computation of the implicit operator will be referred as *implicit_ time_ integration*. Due to the iterative process the most often used algorithms are usually performed during the flow residual $\mathbf{R}$ evaluation and pseudo-time integration. Another feature of TAU is the storage of primitive variables instead of conservative variables. This may have advantages and/or disadvantages but will be taken into account with the keyword *additional_ state_ variables*, implied with (2.4).

2.2.3 TAU profiling and subroutines optimization

The computational grid used for this investigation is an Onera M6 [sch79] Navier-Stokes grid that has 450.634 points and 1.509.834 elements and contains a prismatic layer around the surface and tetrahedrons in the far field. The conditions are a free-stream Mach number of 0.8395 and an angle of attack of 3.06 degrees. For completeness, both implemented pseudo time integration schemes, an explicit Runge-Kutta and a semi-implicit LUSGS method, were profiled. The results for the profiling obtained with the sequential flow solver using an explicit Runge-Kutta and a semi-implicit LUSGS method are displayed in Table 2.1. The percentages shown in Table 2.1 are related to a simulation performed for 100 iterations using no multi-grid scheme and without performing any solution output. The abbreviations used are *tsl* for thin shear layer approximation, see (2.19) and *sae* for the Spalart-Allmaras one equation turbulence model with Edwards [edw96] modification. The most time consuming routine for both pseudo time

Table 2.1: Profiling results with a Runge-Kutta (left) and LUSGS (right) for the Onera M6

Function	% of time	Function	% of time
viscous_fluxes_tsl (2.17),(2.19)	39.2%	viscous_fluxes_tsl (2.17),(2.19)	21.6%
diffusion_fluxes_tsl (2.9)	10.0%	additional_state_variables (2.4)	10.4%
GreenGauss_gradients (2.18)	6.8%	convective_eingenvalues (2.13)	7.9%
scaling_factor_dissipation (2.16)	6.0%	GreenGauss_gradients (2.18)	7.4%
central_flux (2.12)	5.4%	scaling_factor_dissipation (2.16)	6.6%
convective_eingenvalues (2.13)	4.7%	diffusion_fluxes_tsl (2.9)	5.6%
turb_sae_sources (2.20)	3.5%	implicit_time_integration (2.23)	4.8%
additional_state_variables (2.4)	2.7%	convert_consvar_to_primvar	4.7%
scalar_dissipation (2.14)	2.0%	turb_sae_sources (2.20)	4.5%
convert_consvar_to_primvar	1.6%	central_flux (2.12)	2.9%
laplacian_consvar (2.15)	1.5%	linear_solver_lusgs	2.9%
RK_timestepsize (2.21)	1.5%	scalar_dissipation (2.14)	2.2%

integration schemes is the evaluation of the viscous fluxes, i.e. left table shows for the explicit Runge-Kutta scheme an overhead of 40% during a flow simulation. The second most expensive routines are diffusive fluxes (2.9) for Runge-Kutta and for the LUSGS scheme the evaluation of additional state variables (2.4), respectively. Every other routine is at or below 10 % time per execution and is not further considered for that investigation.

During that investigation it became practicable to perform improvements in two ways. At first to perform pre-computation of variables in routines locally which appeared often inside of expensive loops like differences for conservative variables $\mathbf{w}$ between two points, see nominator of (2.19). The second step considers the pre-computation of global constant variables which are independent of flow variables $\mathbf{w}$, such as point distance evaluations from the grid metrics, see denominator of (2.19). These variables are then computed once at the initialization step of the flow solver and kept stored in memory during a flow simulation. Specially for (2.19), the inverse grid point distance is stored and will be introduced into (2.19) as a multiplication finally. This approach becomes important whenever divisions or square root operations are involved. On the other hand more memory is needed and it becomes evident if computation time or memory is preferred but, finally, the total memory increase was about 3% of the total memory used from the code without optimization.

After optimizing the code, the obtained results are shown in Table 2.2 for either the Runge-Kutta and LUSGS time integration schemes. The modifications result in an improvement for the viscous flux, *viscous_fluxes_tsl* see (2.17) with gradient approach (2.19), and for the diffusion flux, *diffusion_fluxes_tsl* see (2.9) again using (2.19). The computation time was reduced more than half as without the proposed optimizations. In case of the LUSGS scheme, it can be easily seen that the first five main routines are approximately consuming the same amount of time now. This is preferable for a good distribution of workload during the evaluation of the residual $\mathbf{R}$. The first proposed optimization procedure was introduced in many other routines and it can be seen that the order has changed considerably for the functions.

Improving functions like gradient computation, indicated as *GreenGauss_gradient* or state vari-

Table 2.2: Profiling results after optimization using an explicit Runge-Kutta (left) or LUSGS (right) for the Onera M6 wing.

Function	% of time	Function	% of time
viscous_fluxes_tsl (2.17)	18.1%	additional_state_variables (2.4)	12.8%
GreenGauss_gradients (2.18)	10.6%	convective_eingenvalues (2.13)	9.7%
scaling_factor_dissipation (2.16)	9.3%	scaling_factor_dissipation (2.16)	9.3%
central_flux (2.12)	8.3%	GreenGauss_gradients (2.18)	9.2%
convective_eingenvalues (2.13)	7.2%	viscous_fluxes_tsl (2.17), (2.19)	8.0%
turb_sae_sources (2.20)	5.5%	implicit_time_integration (2.23)	5.7%
additional_state_variables (2.4)	4.1%	convert_consvar_to_primvar	5.6%
diffusion_fluxes_tsl (2.9),(2.19)	3.1%	central_flux (2.12)	3.6%
scalar_dissipation (2.14)	3.1%	linear_solver_lusgs	3.6%
convert_consvar_to_primvar	2.4%	scalar_dissipation (2.14)	2.7%
laplacian_consvar (2.15)	2.4%	turb_sae_sources (2.20)	5.6%
RK_timestepsize (2.21)	1.6%	diffusion_fluxes_tsl (2.9)	1.4%

able computation as *additional_state_variables* is much more difficult. These functions are already optimized due to their wide range of application areas. Usually, such an approach should be applied for each function introduced in any code but developers have to be aware to follow optimization guidelines carefully.

Finally, the impact of these improvements for a whole simulation is shown to present the computational gain. Table 2.3 and Table 2.4 show the *wall clock time* (WCT) on an i386 32 bit and a x86 64 bit Linux based machine for an Onera M6 wing simulation for the original and the optimized code using Runge-Kutta and LUSGS time integration schemes. The tables include the resulting global force coefficients lift (C_L) and drag (C_D) and the pitching moment coefficient (C_{MY}) to attest no physical change during a simulation.

Table 2.3: WCT for the Onera M6 test case to steady state for 32 and 64 bit Linux machines using an explicit Runge-Kutta method.

Architecture	Code version	C_L	C_D	C_{MY}	Time	Gain
32 bits	original	0.26933	0.015774	8.1587	7178 s.	-
	optimized	0.26933	0.015774	8.1587	5178 s.	27.9 %
64 bits	original	0.26932	0.015778	8.1585	4169 s.	-
	optimized	0.26932	0.015778	8.1585	3701 s.	11.2 %

Table 2.4: WCT for the Onera M6 Testcase to steady state for 32 and 64 bit Linux machine using a semi-implicit LUSGS method.

Architecture	Code version	C_L	C_D	C_{MY}	Time	Gain
32 bits	original	0.26927	0.015787	8.1569	5657 s.	-
	optimized	0.26927	0.015787	8.1569	5487 s.	3.0 %
64 bits	original	0.26925	0.015791	8.1562	4222 s.	-
	optimized	0.26925	0.015791	8.1562	4043 s.	4.2 %

The comparison was made with the same C-compiler version *gcc 4.2.3* using the optimization level *-O2*. One thousand iterations were performed for each simulation to ensure either well converged force coefficients like drag and lift and substantial time to measure the optimizations during the iterative process apart from additional time used for setup and IO.

To sum up, using the explicit Runge-Kutta time integration scheme on a 32 bits system, the WCT is decreased by 27.9 % and on the 64 bit system it decreased by about 10 %. These time reduction is mainly caused due to the improvements of the viscous and diffusion flux routines. In case of the semi-implicit LUSGS scheme, the WCT is decreased by 3% and 4.5% on a 32 and 64 bits machine respectively. In this situation, the improvements are smaller, compared to the previous case, because the functions *viscous_fluxes_tsl* and *diffusion_fluxes_tsl* were not as time consuming as in the Runge-Kutta time integration scheme. In addition, an important percentage of time, more than 10% is spent within the *additional_state_variables* function.

2.3 Parallelization: Solver scalability using different partitioning algorithms

The second point of activity for improving the application performance is an efficient parallelization. For parallel computing on large unstructured grids, domain decomposition is a powerful concept: Given a number P of computational cores, the whole computational grid is decomposed into P subgrids. Each of the P computational cores computes on one of the subgrids. Usually communication between these cores is required because the solutions on the subgrids depend on each other. In this section, a computational core will be also referred to as processor unit or simply processor. The solver operates in different ways on two sets of data:

- Operations that compute point variables directly from one or more point variables. For these operations, no data of other points is required to compute on one point.
- Operations that need data of neighboring points to compute the result for one point. For these operations connectivity information is required. The main kind of connectivity data used in the solver are the edges. Other connectivity data is given by the boundary faces where a near point is connected to a boundary point. More connectivity data is defined by the grid to grid connections of the different grid levels from the multigrid algorithm. As many as possible of these operations should be performed on one subdomain of the grid without communication.

In the following, the current partitioning status of the TAU-Code is first introduced, then advanced partitioning algorithms are explored and, finally, they are applied to complex industrial applications.

2.3.1 Current partitioning status of the DLR TAU Code

Parallelization in TAU is based on domain decomposition and the message passing concept using MPI. For parallel computations, the grids are partitioned in the requested number of domains at the beginning of the flow simulation. Up to now, a simple bisecting algorithm (referred here as geometric) is employed. The load balancing is performed on point weights which are adjusted for the needs of the solver, which is the most time consuming part of the simulation system. The edge cuts are computed directly according to the coordinates.

This geometric partitioning algorithm behaves in the following way: If two partitions have to be computed, first it is compared if the partitioning at x = const (const is the position on half the way between x-max and x-min) requires less cuts of edges than parallelization cut at y = const or z = const. The option that requires less edge cuts is selected and the partition is performed following this axis. If three domains have to be computed, the partitioning is performed by dividing first in 2 subdomains weighted with 1/3 and 2/3. The second subdomain is then divided again in 2 partitions with equal weights. All other numbers of subdomains are computed using the same algorithm recursively; e.g. 7 subdomains are obtained by dividing first in 2 domains weighted with 3/7 and 4/7. The first is then partitioned in 3; the second is two times divided in 2 partitions. The load balancing is performed on point weights based on the amount of edges which finish on each point. These weights try to represent the computation cost of each point in the flow solver. The main drawbacks of this algorithm are that the communication cost is not properly represented, and it is not possible to establish several point weights (multi-weighted or multi-constrained load balance) to deal with different aspects.

For testing purposes, the DLR F6 configuration is used. The mesh decomposition into different domains achieved with the TAU geometric partitioning algorithm is shown in Figures 2.2 and 2.3. Using this domain decomposition, the parallel scalability and speedup of the TAU flow solver for the F6 configuration, either for an Euler grid with about 1 mill. points (tetrahedrons only) and a Navier-Stokes grid with about 6 mill. points (prismatic boundary layer and tetrahedrons) can be observed in Figure 2.4.

The speedup values are computed using as the starting point the execution time for 8 processors. In these cases, it is considered a saturated speedup whenever its value from one computation (using n processors) to another computation (using 2n processors) is less than 1.5, which means that at least half of the new computational resources will be used making effective operations instead of waiting for communication. The speedup of the Euler test-case saturates when using more than 64 processors which implies that the acceptable minimum of points per domain should be more than around 16.000 points. The Navier-Stokes test-case becomes inefficient using more than 256 processors, so the number of points per processor should be higher than approximately 23.500 points.

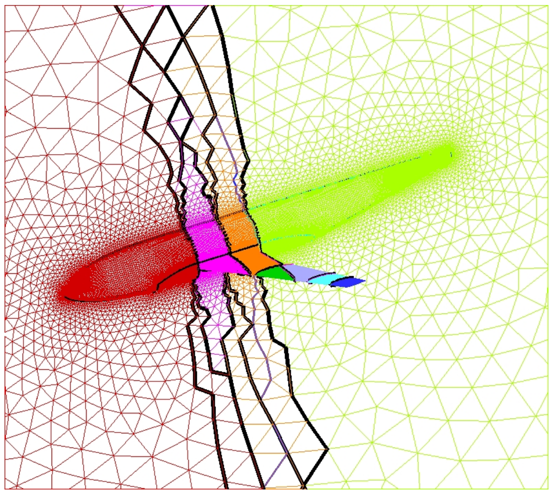


Figure 2.2: Volume view of the partitioned F6 grid into 8 domains.

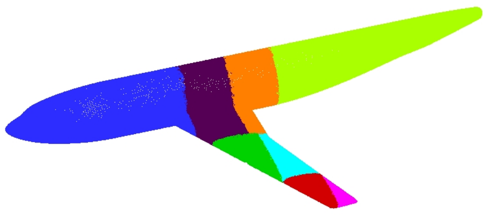


Figure 2.3: Surface view of the partitioned F6 grid into 8 domains.

2.3.2 Exploring different partitioning algorithms for the DLR TAU Code

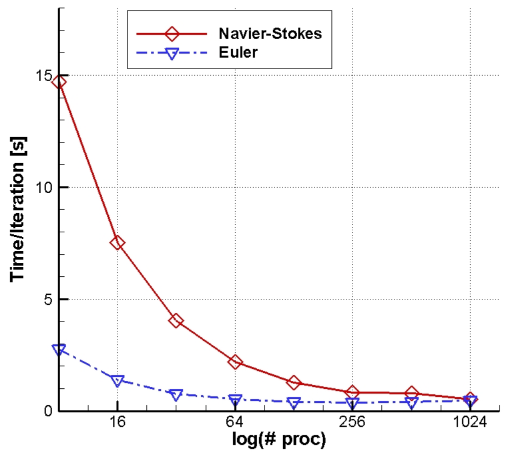
To extend the parallelization study, additional partitioning algorithms are included into the TAU code using the package ZOLTAN. This section briefly describes the main features of the additional partitioning algorithms to be analyzed. More information can be obtained in the ZOLTAN documentation [zol08] .

Recursive Coordinate Bisection (RCB)

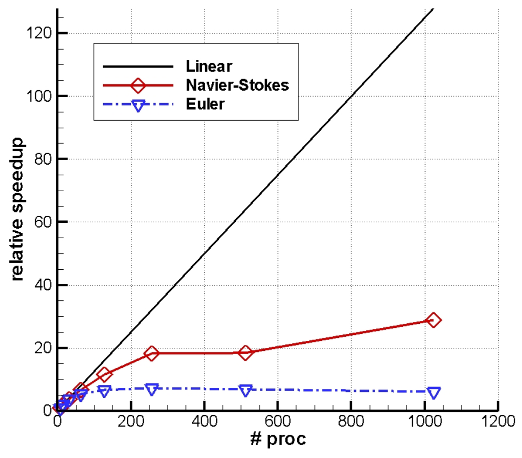
RCB was first proposed as a static load-balancing algorithm by Berger and Bokhari [ber87] , but it is attractive as a dynamic load-balancing algorithm because it implicitly produces incremental partitions. In RCB, the computational domain is first divided into two regions by a cutting plane orthogonal to one of the coordinate axes so that half the workload is in each of the sub-regions. The splitting direction is determined by computing in which coordinate direction the set of objects is most elongated, based upon the geometric locations of the objects. The sub-regions are then further divided by recursive application of the same splitting algorithm until the number of sub-regions equals the number of processors. By adjusting the partition sizes appropriately, any number of equally-sized sets can be created. If the parallel machine has processors with different speeds, sets with nonuniform sizes can also be generated.

Recursive Inertial Bisection (RIB)

RIB was proposed as a load-balancing algorithm by Williams [she94] and later studied by Taylor and Nour-Omid [tay95], but its origin is unclear. RIB is similar to RCB. It divides the domain based on the location of the objects being partitioned by use of cutting planes. In RIB, the computational domain is first divided into two regions using a bisection line orthogonal to the principal inertial axis, so that half the workload is in each of the sub-regions. The sub-regions are then further divided by recursive application of the same splitting algorithm until the number



(a) Parallel scability.



(b) Parallel speedup.

Figure 2.4: Parallel performance of the TAU code over the number of processors.

of sub-regions equals the number of processors. RIB in ZOLTAN does not support multiple weights, as in RCB.

Hilbert Space-Filling Curve (HSFC)

The Inverse Hilbert Space-Filling Curve functions map a point in one, two or three dimensions into the interval [0,1]. The Hilbert functions that map [0, 1] to normal spatial coordinates are also provided in the ZOLTAN library. (The one-dimensional inverse Hilbert curve is defined here as the identity function, $f(x) = x$ for all x). The HSFC partitioning algorithm seeks to divide [0,1] into P intervals each containing the same weight of objects associated to these intervals by their inverse Hilbert coordinates. N bins are created (where N > P) to partition [0,1]. The weights in each bin are summed across all processors. A greedy algorithm sums the bins (from left to right) placing a cut when the desired weight for current partition interval is achieved. This process is repeated as needed to improve partitioning.

Graph partitioning

Graph partitioning is a difficult, long-standing computational problem. It has applications to VLSI (Very Large Scale Integration) design, sparse matrix-vector multiplication, and parallelizing scientific algorithms. The general partitioning problem is described by a graph $G(V, E, W_V, W_E)$ where W_V and W_E are vertex and edge weights respectively. The output of partitioning G consists of subsets of vertices, $V_1, V_2, ...V_k$ where $V_i \bigcap V_j = \Phi$. The goal is to balance the sum of vertex weights for each V_i, and minimize the sum of edge weights whose incident vertices belong to different partitions. Graph partitioning can be used to successfully compute high quality partitions by first modeling the mesh by a graph, and then partitioning it into equal parts.

Figure 2.5 shows the volumetric and surface mesh decomposition into four domains achieved with the Geometric, RCB, RIB, HSFC and the Graph partitioning algorithms. From the top

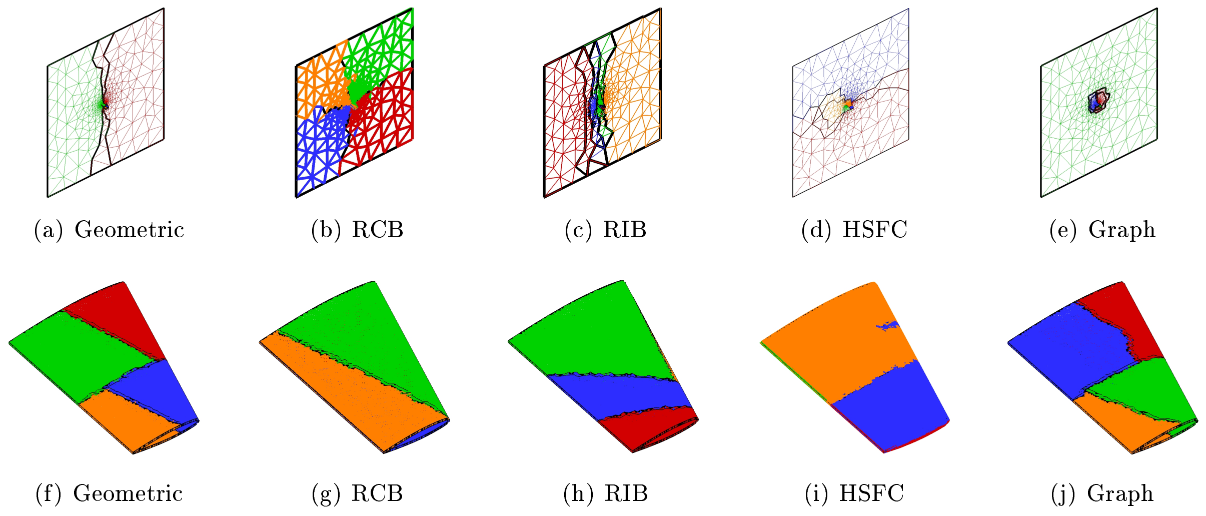


Figure 2.5: Volume view (top) and surface view (bottom) of the partitioned Onera M6 grid into 4 domains.

row in Figure 2.5 looking at the symmetry plane the extensions of the partitioned grid into the volume can be seen.

2.3.3 Runtime performance analysis for different partitioning algorithms.

A comparison between the different partitioning algorithms is performed using complex aircraft configurations to gather experience for industrial applications. First, a high-lift wing-body configuration has been selected. This configuration was investigated in the European project HiRett [kru08, rak05] , see Figure 2.6 . In addition, a high-lift wing-body-pylon-nacelle configuration with fully operable engines, an ALVAST [sch95, kio96] configuration, see Figure 2.7, has been also selected for this study. Both grids contain a structured prismatic boundary layer region for a viscous flow simulation. The computational grid around the HiRett configuration has about 13.6 mill. points with approximately 35.2 mill. volume elements. The grid around the ALVAST configuration contains about 13.2 mill. points with approximately 45.3 mill. volume elements. Both configurations, HiRett and ALVAST, have deployed slats and flaps and are validation test cases for take-off conditions at low speed.

The main amount of the execution time in a CFD computation is spent in the flow solver. In TAU, a pre-processing step is necessary before running any flow simulation. The pre-processor generates the dual mesh and metric information, like face normals from the primary grid. The data is then stored in an edge-based structure which is used for evaluating the fluxes independently of the different primary volumes used. However, the pre-processing time needs only once or twice the time of one flow solver iteration, so it is usually a negligible amount in comparison to the flow solver execution time in which this section is focusing on.

As the pre-processing step is necessary, it was also studied how the domain decomposition affects its execution time. Based on the performed tests, it can be stated that there is an improvement when using graph instead of geometric partitioning algorithms especially with a high number of processors.

Figures 2.8 and 2.9 show the differences between the volumetric mesh decomposition for

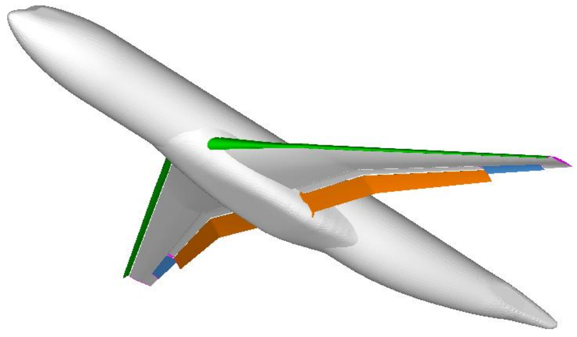


Figure 2.6: HiRett wing-body configuration with high-lift devices and ailerons.

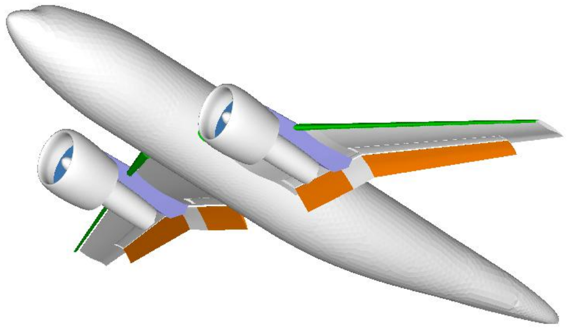


Figure 2.7: ALVAST high-lift configuration with fully operable turbines.

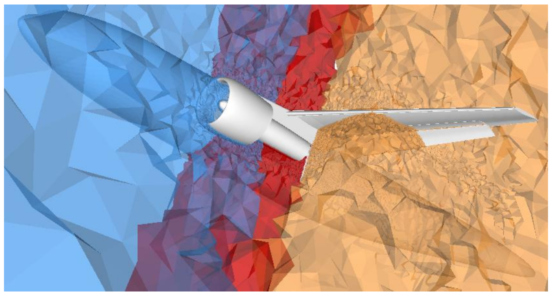


Figure 2.8: Volume view of the partitioned ALVAST grid into 8 domains using a Geometric partitioner.

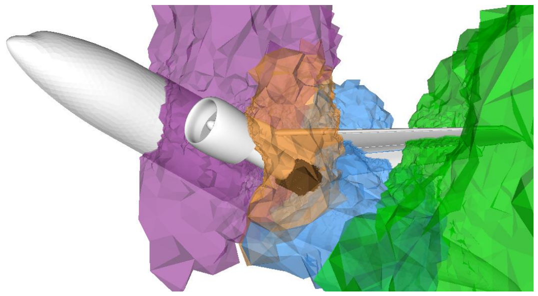


Figure 2.9: Volume view of the partitioned ALVAST grid into 8 domains using a Graph partitioner.

the ALVAST configuration using both geometric and graph partitioning. The surface mesh decomposition can be seen in Figure 2.10 and 2.11. The critical part in the ALVAST grid is the decomposition of the inflow and outflow of the engines. In order to keep the physical behavior of fan turbines effective, a mass flow coupling is needed which introduces an increased communication. Focusing on this subject, it can be observed for low domains that the geometric partitioner divides the inflow and outflow area into different domains while the graph partitioner keeps the inflow and the outflow in one domain.

Regarding the flow solver, the computations were performed with a Spalart-Allmaras one-equation turbulence model with Edwards modification (SAE) and the central spatial discretization scheme with scalar dissipation and a $4w$ multigrid cycle. The flow conditions of the considered configurations are displayed in Table 2.5 . While both configurations were created for

Table 2.5: Flow conditions for ALVAST and HiRett configurations.

	Mach number	Reynolds number	Angle of Attack
HiRett	0.2	25 mill.	19.0°
ALVAST	0.182	1.66 mill.	4.3°

take-off conditions, the purpose of each one was very different, as may be seen from the flow

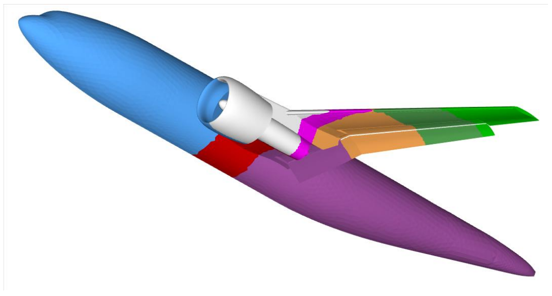


Figure 2.10: Surface view of the partitioned ALVAST grid into 8 domains using Geometric partitioner.

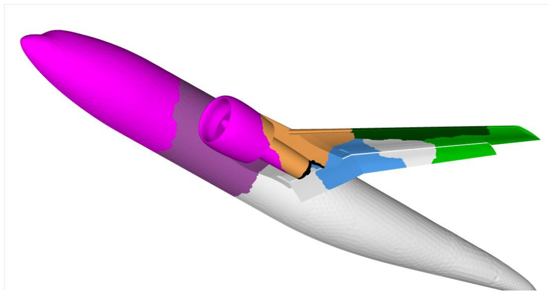
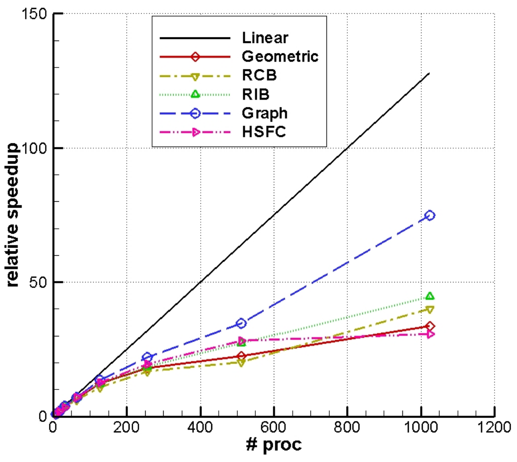


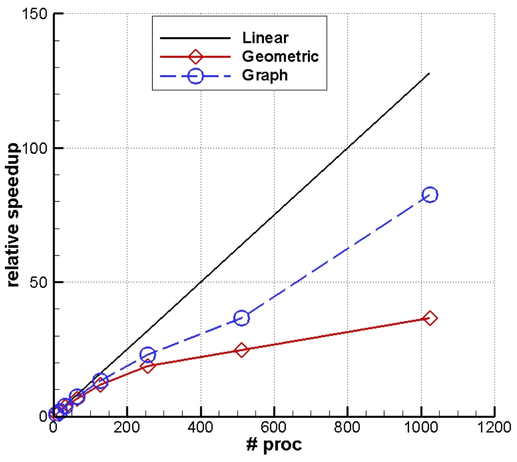
Figure 2.11: Surface view of the partitioned ALVAST grid into 8 domains using a Graph partitioner.

conditions. The HiRett model simulates an airplane shortly before reaching the maximum angle of attack in authentic flow conditions, while the ALVAST model was used in a wind tunnel experiment with a substantial reduced Reynolds number.

Since the solver seems to react in a different way to the many possible time and spatial integration schemes and unfortunately testing all these possibilities would incur a certain expense, the study will be concentrated on an explicit Runge-Kutta and a semi-implicit LUSGS scheme. The approach for speedup was done in the same manner as in section 2.3.1, where a decomposition on eight grid partitions was taken as the first point for linear speedup comparisons. The results in Figure 2.12 show that improvements in the solver speedup using a Runge-Kutta (left) and a LUSGS (right) time integration schemes can be obtained through graph partitioning algorithms for the HiRett test-case.



(a) Runge-Kutta



(b) LUSGS

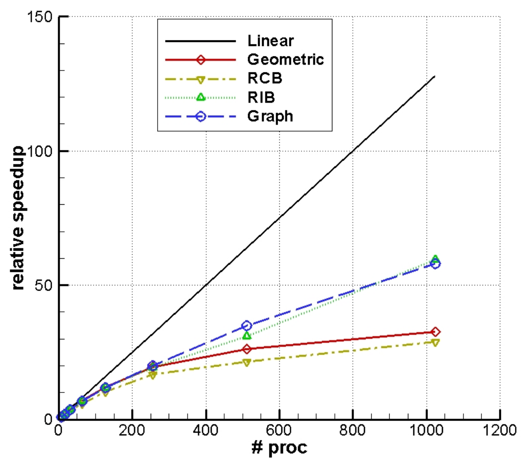
Figure 2.12: Speedup of the TAU Solver using different partitioning algorithms for the HiRett test-case

Table 2.6: Solver execution time per iteration using different partitioning algorithms for HiRett with Runge-Kutta and LUSGS.

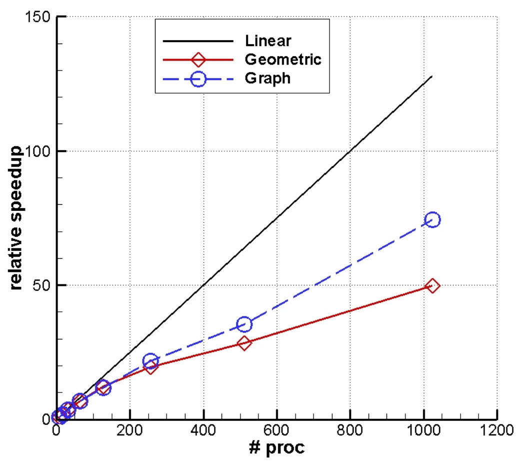
Partitioner	Time integration	8	16	32	64	128	256	512	1024
Geometric	RK	31.42	16.03	8.49	4.68	2.57	1.74	1.39	0.93
	LUSGS	30.15	15.09	8.22	4.58	2.54	1.6	1.22	0.81
RCB	RK	33.45	17.82	9.4	5.4	3.12	1.98	1.65	0.83
RIB	RK	32.11	17.64	8.46	4.74	2.64	1.73	1.18	0.72
HSFC	RK	34.01	17.13	9.8	4.87	2.69	1.74	1.2	0.85
Graph	RK	32.91	16.37	8.76	4.56	2.45	1.51	0.95	0.44
	LUSGS	31.89	15.95	8.4	4.41	2.39	1.39	0.87	0.39

Analyzing Table 2.6, a trend can be observed. Below 128 processors, the time per iteration for the geometric partitioner is smaller in comparison to the graph partitioner while it becomes, as expected, vice versa from 128 up to 1024 processors. It shows the importance of communications when using a high number of processors, and therefore, the need of efficient partitioning algorithms able to represent and minimize the communication between computational domains.

The performance results for the ALVAST case, Figure 2.13, show the achieved improvements in the solver speedup using a graph instead of geometric partitioner for Runge-Kutta (left) and LUSGS (right).



(a) Runge-Kutta



(b) LUSGS

Figure 2.13: Speedup of the TAU Solver using different partitioning algorithms for the ALVAST test-case.

Table 2.7: Solver execution time per iteration using different partitioning algorithms for ALVAST with Runge-Kutta and LUSGS.

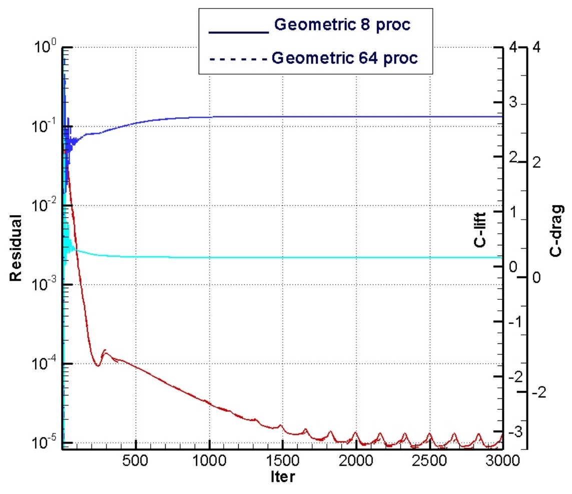
Partitioner	Time integration	8	16	32	64	128	256	512	1024
Geometric	RK	32	16.56	8.57	4.56	2.68	1.63	1.22	0.98
	LUSGS	31.59	16.16	8.45	4.58	2.59	1.62	1.11	0.63
RCB	RK	32.73	17.9	10.05	5.6	3.17	1.94	1.53	1.14
RIB	RK	32.08	16.61	8.74	4.71	2.7	1.66	1.04	0.54
Graph	RK	33.07	17.83	8.92	4.91	2.81	1.65	0.95	0.57
	LUSGS	32.67	17.51	8.77	4.77	2.78	1.51	0.92	0.44

In comparison to the HiRett case (Figure 2.12), the speedup of the ALVAST case (Figure 2.13) shows a visible lack of scalability due to the additional communication for the engine mass coupling. Concerning the time used in parallelization mode, there are two main parts which rely firstly on setting up the communication between two processors and the time used for transferring an amount of data. These two times may vary enormously between few to many partitions. Partitions with a very low number of points per partition spend a high amount of time setting up the complex socket communication. Engine mass coupling introduces in a simple way only one integral value or in a more complex way a partial mass flow over the inflow and outflow faces. This very small amount of data which have to be sent over domains is predominately driven by setting up the additional communication for a high number of processors.

The maximum allowable CFL number for the explicit three stages Runge-Kutta scheme with a $4w$ multigrid cycle remained unchanged throughout all simulations for both cases. Unlike the semi-explicit LUSGS scheme for the ALVAST computations for 512 and 1024 partitions. The CFL number had to be decreased by half of the CFL number as used for 8 domains to maintain a converging simulation. From the execution times obtained in Tables 2.6 and 2.7, it can be observed that even slight improvements per iteration could provide a significant enhancement in case of complete simulations over a huge number of processors.

Finally, the convergence history obtained for the HiRett case, using 8 and 64 processors is seen in Figure 2.14 on the facing page. Independently of the number of processors, the density residual and integral force coefficients should converge to the same values when using the same flow parameters for the simulation. The lift and drag coefficients are the same which is reassuring but the density residual exhibits a slightly non matching behavior between different processors. Flow phenomena, like separation or shocks are very sensitive with respect to slight but unforeseen algorithmic changes on partitioned boundaries.

To sum up, it has been observed that, as expected, graph partitioners show, even for a high number of processors, a speedup closer to the linear one, while the geometric partitioners saturate very soon at around 200 processors for HiRett and around 250 processors for ALVAST. All of the new partitioning algorithms used with TAU have some important features that can be adapted for complex applications. RCB, RIB and graph permit dynamic load balancing and graph permits also multi constrained partitioning to partition a graph in the presence of multiple balancing constraints. The idea is that each vertex has a vector of weights of size m associated



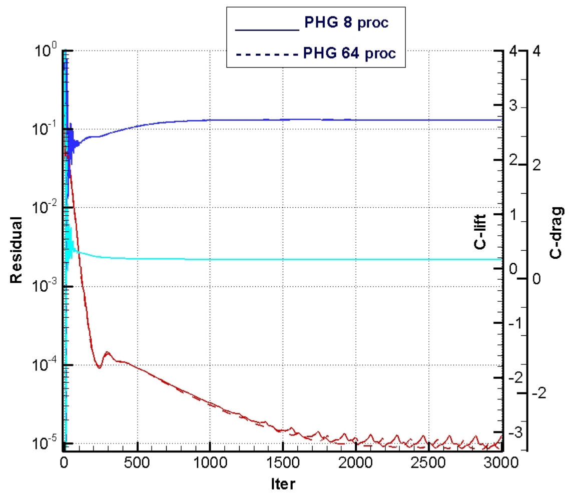


Figure 2.14: Solution comparison using Geometric (left) and Graph (right) for 8 and 64 processors.

with it, and the objective of the partitioning algorithm is to minimize the edge-cut subject to the constraints that each one of the m weights is equally distributed among the domains.
This feature is important when the architecture of the computation platform is heterogeneous, as occurs in hardware acceleration and will be subject of further investigations.

2.4 Acceleration: FPGAs and GPUs for CFD Simulations

The aeronautical industry requirements in the near future will not be accomplished by the usual evolution of current processors and architectures. Indeed, it is estimated that, in order to obtain "engineering-accuracy" predictions of surface pressures, heat transfer rates, and overall forces on asymmetric and unsteady turbulence configurations, the grid size will have to be increased to a minimum of 50-80 million nodes. This means that the simulation of a complete aircraft configuration will require several days to obtain solutions in a high performance cluster with hundreds of processors, so an exponential increment of the computational requirements is estimated.
Therefore, the improvements expected at optimization and parallelization levels are not enough, and it will be necessary to introduce new concepts in typical simulation platforms in order to satisfy these demands and to face more complex challenges.

As hardware acceleration is a very innovative task, in this chapter, some ongoing activities and feasibility studies in several projects will be introduced. Two alternatives to speed up the performance of PC clusters are quickly emerging due to recent technological advances.

The first one is based on Graphics Processing Units (GPU) [bra08, mar08, bol03] as computing resources that complement the processors in the PCs. These general-purpose devices have evolved to constitute powerful pipelined vector processors capable of delivering very high throughputs in optimal conditions at a relatively low cost.

The second alternative is based on reconfigurable hardware, in particular, Field Programmable

Gate Arrays (FPGA) as computing resources in accelerator boards connected to the PCs of the cluster. This technology uses a completely different approach to the problem. Instead of general-purpose resources, FPGAs provide application-specific data paths that are optimized to compute the critical parts of the CFD algorithms. The optimization includes the possibility of both, space and time parallelism so the implemented circuits are capable of delivering throughputs well above those provided by general-purpose processors [fus08] . The overall idea with FPGAs is to introduce a new level of parallelism in the system architecture. While the PCs in the cluster implement coarse grain parallelism resulting from partitioning the dataset, the specialized hardware in the accelerators implements fine grain parallelism at the operation level, accelerating the set of operations assigned to each local processor (parallelism in the hardware functional units that provide several results at the same time) [and08] .

One main idea is that all the computational nodes in the cluster are FPGA-powered, so load balance has to be taken into account. This means that the subdomain decomposition process has to assign to the FPGA-powered computational nodes an adequate workload in order to support efficient synchronization in the overall system. Due to the heterogeneous nature of the architectures existing in a cluster platform, including differences in performance between the general-purpose CPUs, special considerations about global synchronization have to be explored. The partitioning algorithm has to allow a multiconstraint domain decomposition with at least 2 weights per mesh node, one related to each level of paralellism. The new graph partitioning algorithm that has been linked to be used in TAU, permits also multiconstraint partitioning to partition a graph in the presence of multiple constraints. Each vertex can have a vector of weights of size m associated with it, and the objective of the partitioning algorithm is to minimize the edgecut subject to the constraints that each one of the m weights is equally distributed among the domains. It is assumed that at least two weights should be considered (for the fine and coarse level of parallelism) in the case of homogeneity between the type of processors and hardware accelerators.

Of course, porting a critical part of an algorithm to an FPGA implies a complete design cycle, but once completed, the design is ready to be loaded at any time in any FPGA. Since FPGAs are reconfigurable, different designs can be loaded at different times depending on the application needs, making FPGAs very interesting devices to act as coprocessors in computation intensive applications.

FPGAs have also some limitations. In particular, there is a limit on the amount of computations that can be performed in an FPGA at maximum throughput, depending on the available resources in the FPGA. When this limit is exceeded, resources must be shared among computations and performance reduces. Also, the communications interface between the PC memory and the FPGA is a major concern as the available bandwidth, for CPU-FPGA communications, can limit the effective global performance. From these considerations, it can be observed that FPGAs are best adapted to applications where most computational effort concentrates on a small portion of the code which is repeatedly executed for a large dataset. In addition, the I/O communication load has to be small when compared to the computational load, to avoid saturating the CPU-FPGA interface. CFD codes seem to fulfil these requirements and, therefore, appear as good candidates to benefit from FPGA-based accelerators.

In this section, a preliminary feasibility analysis is carried out for different scenarios, based on precision analysis and FPGA syntheses.

2.4.1 The architecture: A cluster of PCs with accelerator boards

This subsection is a short review of the state of the art of the specific technologies involved in the architecture. First, the families of high-performance FPGAs available in the market are presented. Second, the interface between the PC and the FPGA is shown and finally, the main characteristics of today's GPUs are briefly introduced.

Field Programmable Gate Arrays (FPGAs)

A field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing, hence the name "field-programmable". FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. They can be used to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications. With the addition of high-speed I/O and highly-efficient embedded processing and memory blocks, FPGAs are also gaining acceptance as relatively low-cost devices in production systems where high-performance, and not only flexibility, is required. Xilinx and Altera have been, in the last years, the FPGA market leaders and long-time industry rivals. Together, they control over 80 percent of the market, with Xilinx alone representing over 50 percent.

Xilinx has two main FPGA families: the high-performance Virtex series and the high-volume Spartan series. It also manufactures two Complex Programmable Logic Devices (CPLDs) lines, the CoolRunner and the 9500 series. Each model series has been released in multiple generations since its launch.

The latest Virtex-7 is said to consume 50 percent less power, cost 20 percent less, and have up to twice the logic capacity of previous generations of FPGAs

Table 2.8 summarizes the characteristics of the latest families of FPGAs from Xilinx.

Table 2.8: Characteristics of the latest Xilinx FPGAs.

Family	Logic cells ($x10^3$)	Memory (Mb)	Interface blocks for PCI-e
Virtex-5 LX	30.7-331.7	Up to 10,3	1
Virtex-5 LXT	19.9-331.7	Up to 11,6	1
Virtex-5 SXT	34.8-94.2	Up to 8,8	1
Virtex-6 LXT	74,5-758,7	Up to 22,7	2
Virtex-6 SXT	314,8-476,16	Up to 38,2	2
Virtex-7	864-1955	Up to 85	3x8

As the FPGAs are usually integrated into a bigger system is important to consider all the available technology for obtaining higher communication bandwidths [fus08] .

Table 2.9 shows a summary of the available interfaces nowadays.

Table 2.9: Summary of current high-speed communication interfaces.

Interface	Capacity (Gbps)
PCI Express (8 lane)	16.000
HyperTransport (1 GHz, 16-pair)	32.000
PCI Express (16 lane)	32.000
PCI Express (32 lane)	64.000
PCI Express 2.0 (32 lane)	128.000
HyperTransport (2.8 GHz, 32-pair)	179.200

Peripheral Component Interconnect(PCI) and the express version (PCI-e) are the most widely used I/O buses (peripheral buses). Used in computers of all sizes, provide a shared data path between the CPU and peripheral controllers, such as networks, graphics cards, or FPGAs boards. PCI and PCI-e are the most suitable interfaces in the architecture for CFD acceleration. The main reason for this, is the availability of FPGA development boards with these technologies in the market. However, it is necessary to comment here that bandwidth requirements are going to be a performance bottleneck with current technologies. Therefore, it is important to achieve a good balance between bandwidth requirements and performance and latest technologies as HyperTransport will have to be considered in the near future, and the development tools will have to be prepared for achieving high communication bandwidths.

Graphics Processor Units (GPU)

Nowadays, modern graphics hardware outperforms the traditional desktop CPU in terms of computational processing power by several orders of magnitude with a very attractive cost/performance ratio. Commodity graphics processing units (GPUs) have evolved into high performance parallel architectures, driven by multiple cores and very high memory bandwidths. With the increasing programmability and flexibility of graphics processing units, this hardware is capable of performing not only computations for image rendering applications, but also computations in a wide variety of fields: from sparse matrix multiplications techniques [kru03] to multigrid, conjugate gradient solvers for systems of partial differential equations [bol03], N-body simulations [bel08] and physical simulations such as fluid mechanics solvers [mar08, har04].

Although using consumer-level graphics hardware for general computation is not a new idea, the architecture of previous generations of graphic cards (those based on NVIDIA's 7900 model or Ati's X1900 and previous ones) were strongly oriented to graphic applications and were not flexible enough for some scientific applications as CFD simulation. However, the latest generations of graphic cards has overcome most of the drawbacks and, recently, this hardware has acquired interesting levels of both raw performance and programmability. In this context, a major step forward in graphic cards programming is the development of new programming languages and compilers for general purpose applications. Recently NVIDIA has released the Compute Unified Device Architecture (CUDA) software platform.

NVIDIA is the major manufacturer today in the market. Table 2.10 summarizes the characteristics of some of the most relevant GPUs. NVIDIA uses two clocks: One for the general speed at which the GPU operates (Core Clock) and other that dictates how fast the shaders operates (for example to perform texture render), independent of the GPU core. The bandwidth between the core and the main memory is usually referred as "Clock Memory".

Table 2.10: Main characteristics of GPU devices from NVIDIA source

GPU	Clock Core (MHz)	Clock Shader (Mhz)	Clock Memory (Mhz)	Memory Size (MB)
GeForce 8800 GTX	575	1,350	1,800	768
GeForce 8800 Ultra	612	1,500	2,160	768
GeForce 9800 GTX	675	1,688	2,200	754
GeForce 9800 GX2	600	1,500	2,000	2 × 754
Tesla D870	2 × 600	2 × 1,350	1,600	3,000
Tesla S870	4 × 600	4 × 1,350	1,600	6,000

2.4.2 Preliminary precision analysis

The main goal of this study, apart from establishing the technological feasibility of FPGA-based hardware accelerators for CFD applications, is to determine the precision requirements, in the case of a floating point data representation.

Precision analysis

FPGA performance depends on the latencies of their functional units, and it is related with the word-lengths assigned to their operands in the quantization design phase. Therefore, it is necessary to make an analysis of the precision obtained with different data formats to find out a minimum word-length that fullfils the precision requirements in the specific CFD applications.

Here, it is considered only the floating point data representation due to its large range representation capability. The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is the most widely-used standard for floating-point computation, and is followed by many hardware and software implementations. Most of the computer languages allow or require that some or all arithmetic are carried out using IEEE 754 formats and operations. The current version is IEEE 754-2008, published in August 2008; it includes nearly all of the original IEEE 754-1985 (published in 1985) and the IEEE Standard for Radix-Independent Floating-Point Arithmetic (IEEE 854-1987). The standard defines arithmetic formats, interchange formats, rounding algorithms, operations, exception handling, etc.

Most of the major FPGA manufactures offer nowadays Intellectual Property (IP) cores for floating point operations (with a custom number of exponent and mantissa).

As floating point representation is coming into the FPGA technology (due to the larger FPGA devices and the availability of IP cores for the most common operations in the market), an analysis about how the selection of a floating point format (single 32 bits or double 64 bits) affects the integral coefficients of lift (C_L) and drag (C_D) has been performed within this work. Further analysis about a custom (determined number of bits of the exponent and mantissa) floating point format will be performed in the near future.

The tests performed and the flow conditions are shown in the Table 2.11.

Table 2.11: Tests performed for single versus double floating point comparison ("-" indicates Euler computations)

	Mach number	Reynolds number	Angle of Attack
NACA0012	0.8	-	1.25^o
ONERA M6	0.84	-	3.06^o
RAE2822	0.75	6.2×10^6	2.8^o
F6	0.749	3×10^6	1^o

The results of the convergence histories of the density residual, lift coefficient and drag coefficient for all the test cases are displayed in Figures 2.15 and 2.16.

Tables 2.12, 2.13, 2.14 and 2.15 show the differences in the solution (fifth decimal digit in drag and lift coefficients) for the NACA0012, ONERAM6, RAE2822 and DLR-F6 configurations, respectively.

Table 2.12: Results for the NACA0012 Euler flow precision study.

	Residual	Cl	Cd
single (32 bits)	3.97131316277e-06	0.343801110983	0.0244647357613
double (64 bits)	9.7957943602e-13	0.343800099497	0.0244645519529

Table 2.13: Results for the ONERAM6 Euler flow precision study.

	Residual	Cl	Cd
single (32 bits)	2.65640096586e-07	0.288318365812	0.0123302051798
double (64 bits)	1.21071858268e-08	0.288318825812	0.0123301392507

Table 2.14: Results for the RAE2822 viscous flow precision study.

	Residual	Cl	Cd
single (32 bits)	1.65293609816e-06	0.749991750526	0.0252934138158
double (64 bits)	1.65323609682e-06	0.750008681358	0.0252944455683

Figure 2.16 shows the convergence history and the global force coefficients for the RAE2822 and the DLR-F6 [bro99, bro01] configuration. The DLR-F6, see Figure 2.2, is a simplified wing-fuselage geometry which has been used in the past for validation of CFD codes at the second [laf05] and third [vas07] AIAA sponsored Drag Prediction Workshops. The computational grid around the F6 Navier-Stokes grid has 5.8 mill. points and 16.1 mill. elements. The flow conditions are a free-stream Mach number of 0.749 and a fixed angle of attack at 1 degree with a Reynolds number of 3×10^6. Figure 2.16 does not show any significant differences in the solution, only the fifth decimal digit (in drag and lift coefficients) is affected by the numerical error

due to the operand representation. This comparison has shown that single or double precision computation has to be further considered which is mainly driven by the accuracy requested from CFD engineers, usually less than one drag or lift count and will have mainly an important impact for any hardware-software platform like FPGAs. For bigger and complex configurations usually the double floating point format is required to permit the convergence of the solver, but this requirement comes from the geometric data of the grid representation. Future work will address the possibility of a mixed (single, double) precision or a custom floating point precision.

Table 2.15: Results for the DLR-F6 viscous flow precision study.

	Residual	Cl	Cd
single (32 bits)	-5.6701	0.62095	0.032841
double (64 bits)	-5.671	0.62098	0.032844

Table 2.15 shows the results of lift and drag coefficients for single and double precision calculation in the DLR-F6 viscous case.

FPGA synthesis results

The data word-length affects obviously to the precision in the solution obtained, but also to the area requirements in terms of hardware implementation into an FPGA.

Table 2.16 summarizes the reported synthesis values of the floating-point designs.

Table 2.16: Design synthesis results for maximum throughput using a floating point format.

Word-length	Bit slices	DSP cores	Max Frequency
24/8	69,772	0	263.8
24/8	56,960	42	249.3
24/8	62,860	64	244.1
54/10	185,452	0	208.2

The advantages of using shorter word-lengths are clearly shown. Not only higher processing speeds are obtained, but reduced I/O rates are required for such maximum speeds.

Future developments should continue studying the possibilities of different types of data representation for hardware acceleration in CFD applications, because both precision and performance depend on it. Some analysis have shown [and08, fus08] that the speedup could be even greater than two orders of magnitude, but specific implementations have to be performed to obtain the real advantages of this new programming paradigm.

2.5 Conclusions

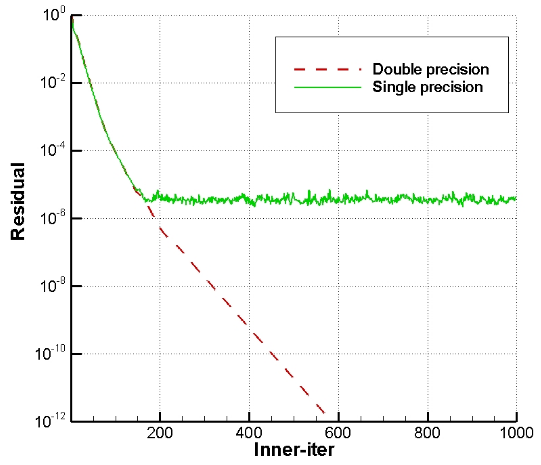
The improvement of the code efficiency has been addressed through optimization by applying different tuning strategies to reduce the execution time of the unstructured DLR TAU solver.

Significant computational gains (around 11% for RK and 4% for LUSGS for a 64 bits machine) have been achieved and it makes clear the necessity of code profiling and optimization involving multidisciplinary knowledge to reduce the execution time and memory consumption. Regarding value precision, a study of single 32 bits and double 64 bits precision has shown an important improvement (specially in memory consumption) using a reduced precision if there are not significant differences in the solution.

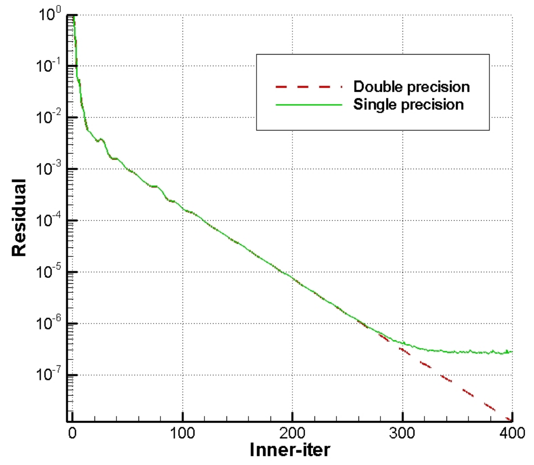
Additionally, the analysis of different algorithms for parallelization to make a more efficient grid partitioning has been performed together with a feasibility study of the effective performance of the hardware acceleration. Several partitioning algorithms have been included in the TAU code, and highly parallel simulations using up to 1024 processors have been performed to test the efficiency and scalability of the selected new algorithms for industrial configurations. The conclusion here is that the graph partitioner algorithm maintains the speedup much closer to the linear one for a high number of processors in all the tests performed.

Moreover, the current state-of-the-art of the development of mixed hardware-software computational platforms for simulation has been presented focusing in precision and area estimations which shows a promising technology when the data representation format required is limited. Word-lengths must be determined by error analysis and precision requirements and they will affect the final speedup achieved. Future efforts in hardware acceleration will be performed to achieve a complete implementation with optimal balance between area and performance and a mixed hardware-software cluster architecture to test the effective acceleration into a simulation platform.

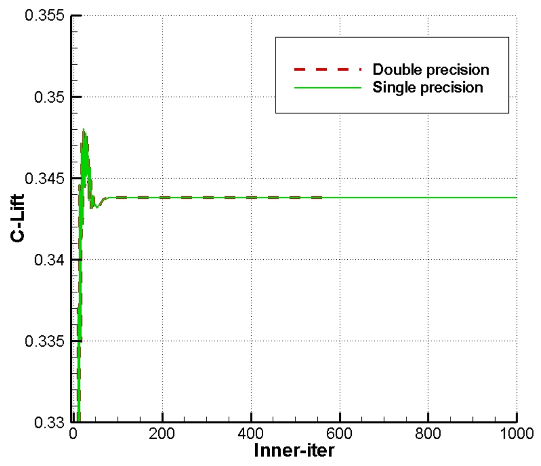
This chapter has focused on the acceleration of the CFD codes, which are a key part of the aerodynamic shape design process, and therefore, improvements on their efficiency are also improvements on the process efficiency. However, even with the proposed contributions, the full applicability of aerodynamic shape optimization tools in the industry is far from being in a mature stage. In order to reach such maturity, further contributions are needed, not only for decreasing the computational requirements, but also to allow a global optimization independent from initial shapes. Chapter 3 focuses in such global aerodynamic optimization phase, proposing the use of a hybrid Evolutionary Programming (EP) and Support Vector Machines (SVMs) with the aim of both allowing a broad design exploration and reducing the computational cost.



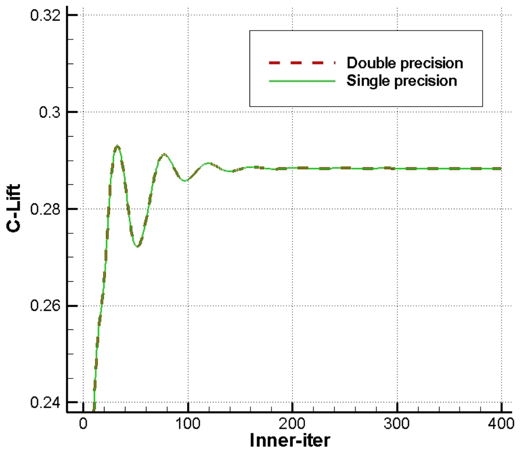
(a) Residual



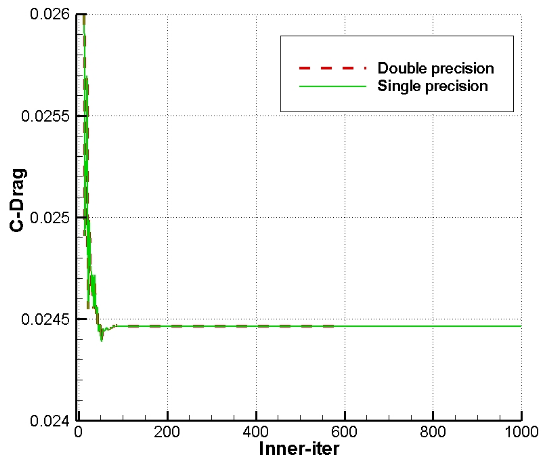
(b) Residual



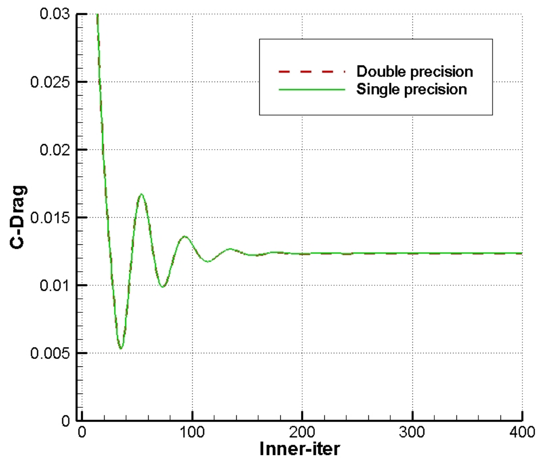
(c) C_L



(d) C_L

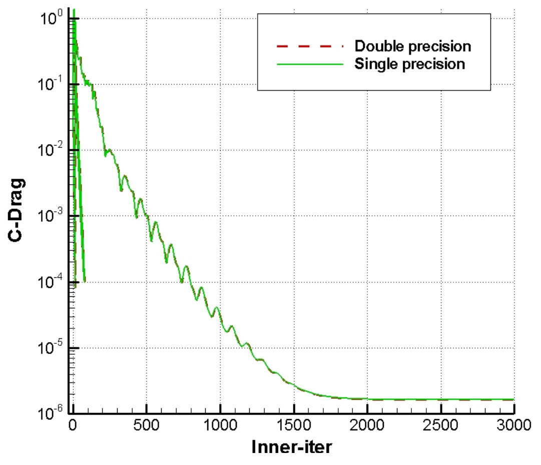


(e) C_D

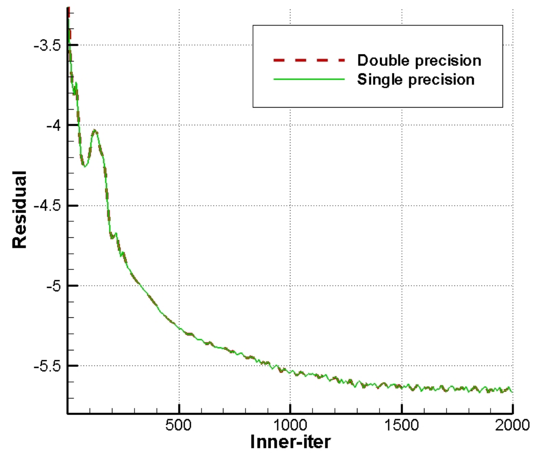


(f) C_D

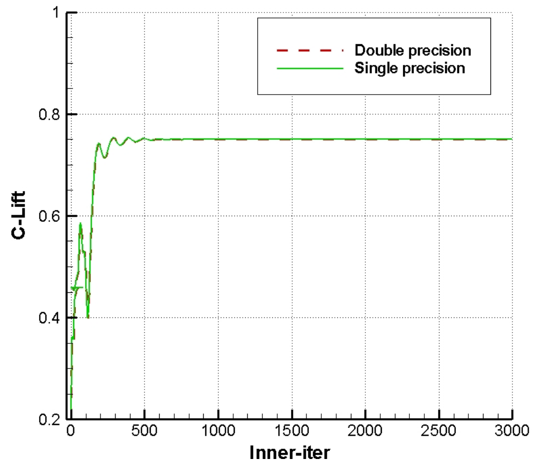
Figure 2.15: NACA0012 (left) and ONERA M6 (right) Euler flow precision studies. Convergence history of the density residual (a) and (b), lift coefficient (c) and (d), and drag coefficient (e) and (f) with respect to the time iterations of the CFD solver.



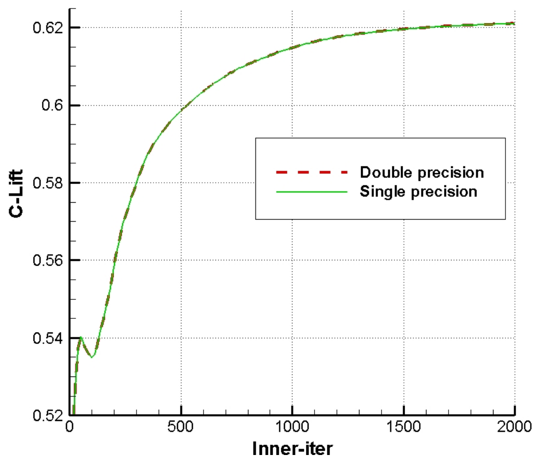
(a) Residual



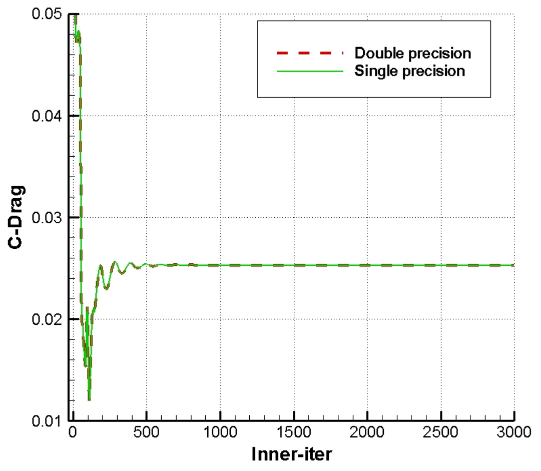
(b) Residual



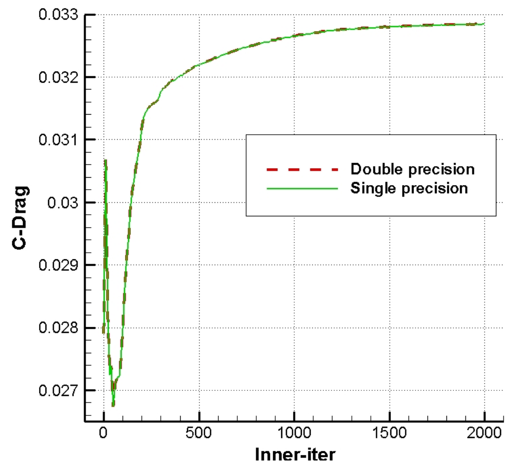
(c) C_L



(d) C_L



(e) C_D



(f) C_D

Figure 2.16: RAE2822 (left) and DLR-F6 (right) viscous flow precision studies. Convergence history of the density residual (a) and (b), lift coefficient (c) and (d), and drag coefficient (e) and (f) with respect to the time iterations of the CFD solver.

Chapter 3

Coarse grain surrogate-based global aerodynamic optimization

3.1 Introduction

The aerodynamic design problem can be solved using either deterministic or non deterministic methods. Deterministic approaches often require the gradient information of the objective function. These gradient-based methods have been broadly used but they need a continuous evaluation function and have a weak performance in a noisy environment. In addition, they are strongly dependent on the initial configuration and could get trapped into a local minimum. On the other hand, non-deterministic methods such as evolutionary algorithms (EA) have the ability to work with noisy objective functions, without assumptions on continuity [lia10]. They also have a high potential to find the global optimum of complex problems involving a large amount of design parameters. However, they require a vast number of evaluations to obtain the optimum solution, even for a small number of design variables.

In the case of aerodynamic design, each evaluation of an individual in the EA requires a complete CFD analysis which makes the method unfeasible, in terms of computational cost, even when applying the contributions proposed in the previous chapter. To overcome this problem there are different approaches in the literature, such as the use of powerful processing machines, such as Graphic Processing Units [kam10] or, more frequently, the use of surrogate models or metamodels [zho10, jin05]. A metamodel is an inexpensive and approximate model of a costly evaluation method. Regarding the aerodynamic design using EAs, the metamodel technique could be used to calculate the fitness of the candidate solutions by replacing the time demanding CFD tools, as previously shown in the literature [gia06, lia08]. For this purpose, regressors based on neural computation could be used as metamodels, once they have been trained based on previous evaluations.

There are well-documented examples of the applicability of soft-computing approaches (Neural networks and Evolutionary-based techniques) in a broad range of prediction and optimization problems including some parts of aerodynamic or multidisciplinary optimization processes. The majority of published studies uses some type of evolutionary algorithms hybridized with neural networks as metamodels: In [gia06] a complete study of different types of neural networks working as metamodels in an aerodynamic shape design problem is carried out. The study includes multilayer perceptrons and Radial Basis Functions networks. In [lia08] a grid-based hierarchical evolutionary algorithm hybridized with a Radial Basis function network is proposed also in different parts of aerodynamic design problems. There are more recent works discussing different aspect of hybridizing evolutionary algorithms and neural networks as metamodels for airfoil design [ste03, san08, aso09, bom10, coh12]. Other perspectives of the problem are also discussed in the literature, such as in [jah11], where an approach based on evolutionary algorithms directly hybridized with an unstructured CFD solver and a neural network (multi-layer perceptron) as metamodel for the first step of the approach is proposed. Other authors have tested the performance of alternative evolutionary approaches such as Particle Swarm Optimization [khu09, pra09]. Furthermore, other methods applied to aerodynamic shape design, such as fuzzy logic approaches [hos11], multiobjective algorithms [kam08], works involving cokriging techniques [zho10], and papers that describe computation frameworks developed to enhance the design process [kim09].

This chapter focuses on the first phase of the aerodynamic design process, i.e., obtaining an approximation to the best candidate from a broad design space (dataset of different geometries, including unconventional ones). The main contribution of this work is to study the performance of an Evolutionary Programming (EP) approach hybridized with a Support Vector regression algorithm (SVMr) as metamodel in a problem of optimal airfoil design. To our knowledge, this important regression technique has not been extensively applied to aerodynamic design, and may have important advantages over previously mentioned metamodels, such as neural networks. It will be showed that the proposed approach is able to obtain preliminary airfoil designs which can be used, at a later stage, as input for a finer design process using methods which require more computational resources, such as CFD.

This chapter is structured as follows: next section describes the proposed hybrid EP - SVMr approach, giving details on the EP and SVMr algorithms. Then, the experimental part of the work is explained, where different results on the SVMr performance as a metamodel are displayed. Finally, remarks on the feasibility of the proposed approach in case of industrial configurations are outlined.

3.2 Proposed approach

The process of the proposed approach for aerodynamic shape optimization is shown in Figure 3.1. The objective is the shortening of the design cycle through a combined approach, where the first stage makes use of evolutionary algorithms together with metamodels to estimate the aerodynamic data. In this phase, the inputs to the process are the target design point (flow conditions), the objective function and the constraints. The output is an approximation to the global optimal solution. As evolutionary algorithm, an Evolutionary Programming approach [xin99] is proposed, which will evolve different airfoil geometries, in terms of an objective function, given by the metamodel. The use of a Support Vector Regression (SVMr) is proposed to this end. In this section, the EP algorithm used to tackle the optimal evolution of airfoil geometries is explained, together with a detailed description of the EP encoding and the SVMr, which will be applied to obtain the aerodynamic coefficients associated to each geometry, in a fast and accurate way.

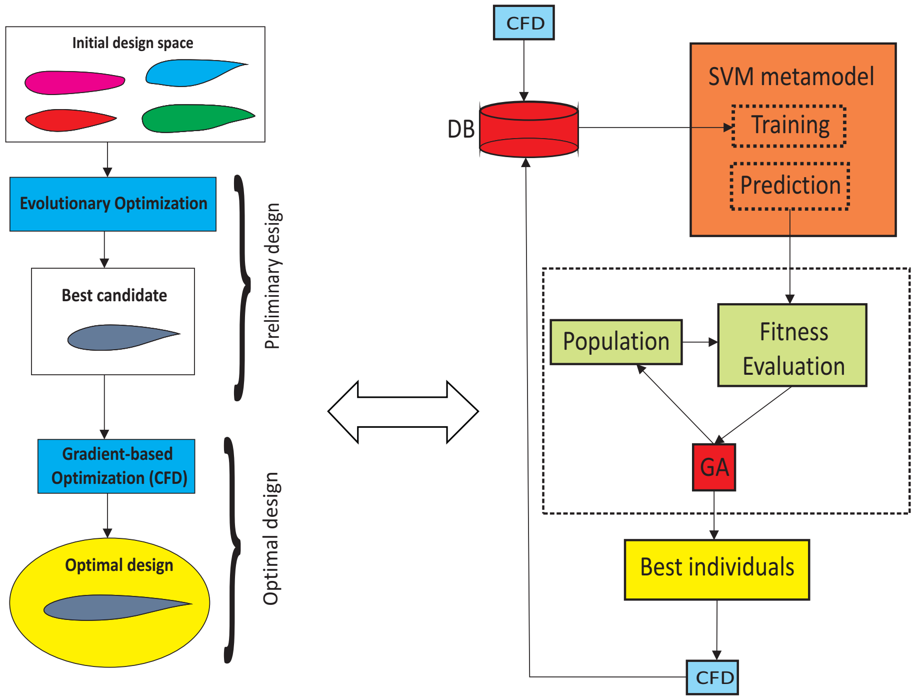


Figure 3.1: Proposed two-step design cycle: first step using evolutionary algorithms together with metamodels and final step for fine optimization.

3.2.1 EP encoding: airfoil parameterization

Sobieczky parameterization [li98] is used, which can represent a wide variety of airfoils with a reasonable number of parameters. This parameterization employs mathematical expressions for the proper representation of generic airfoil geometry (shape functions). This is accomplished by the use of polynomial functions for the airfoil thickness (y_t) and camber (y_c) lines:

$$y_t = a_1\sqrt(x) + a_2x + a_3x^2 + a_4x^3 + a_5x^4 \quad (3.1)$$
$$y_c = b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + b_6x^6 \quad (3.2)$$

The upper- and lower-side y-coordinates at a given chord location are given by:

$$y_u = y_t + y_c \quad (3.3)$$
$$y_l = y_t - y_c \quad (3.4)$$

The geometric parameters defining the airfoil are: position of the leading edge control point, position and airfoil maximum thickness, trailing edge thickness line angle, trailing edge thickness, leading edge camber line angle, camber at maximum thickness, position and maximum camber, camber at maximum thickness, trailing edge camber line angle and trailing edge camber. Figure 3.2 shows the parameters used for airfoil definition. Note that the mean curvature line (colored in red) has been multiplied by 10 only for representation purpose.

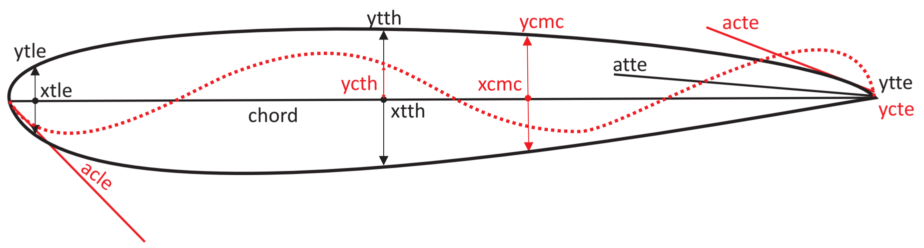


Figure 3.2: Sobieczky's parameterization for airfoil definition.

It is possible to link a_n and b_n coefficients in Equations (3.1) and (3.2) to the geometric variables described in Table 3.1 as it is shown from Equation (3.5) to (3.14). Using this parameterization, an airfoil shape is defined by basic geometric parameters, instead of the coefficient of shape functions directly. This provides more knowledge about the flow around the airfoil and therefore, about the aerodynamic performance.

Table 3.1: Variables for airfoil geometry parameterization and their values' range.

Short name	Variable for geometry parameterization	Min	Max
xtle	X Position for leading edge control point	0.015	0.015
ytle	Y Position for leading edge control point	0.036	0.036
xtth	X position for maximum thickness	0.300	0.450
ytth	Maximum thickness	0.100	0.170
atte	Trailing edge thickness line angle	-10.000	-4.500
ytte	Trailing edge thickness	0.006	0.006
acle	Leading edge camber line angle	-7.500	5.000
ycth	Camber at maximum thickness	-0.008	0.005
xcmc	X position for maximum camber	0.700	0.800
ycmc	Maximum camber	-0.010	0.020
acte	Trailing edge camber line angle	-15	0
ycte	Trailing edge camber	0	0

To obtain the a_n coefficients related to thickness distribution Equations (3.5) to (3.9) are used:

$$
\begin{aligned}
0 &= \frac{a_1}{2\sqrt(xtth)} + a_2 + 2a_3xtth + 3a_4(xtth)^2 + 4a_5(xtth)^3 && (3.5)\\
ytth &= a_1\sqrt(xtth) + a_2xtth + a_3(xtth)^2 + a_4(xtth)^3 + a_5(xtth)^4 && (3.6)\\
atte &= \frac{a_1}{2} + a_2 + 2a_3 + 3a_4 + 4a_5 && (3.7)\\
ytte &= a_1 + a_2 + a_3 + a_4 + a_5 && (3.8)\\
ytle &= a_1\sqrt(xtle) + a_2xtle + a_3(xtle)^2 + a_4(xtle)^3 + a_5(xtle)^4 && (3.9)
\end{aligned}
$$

Equations (3.10) to (3.14) are used to compute the b_n coefficients for camber:

$$
\begin{aligned}
acle &= b_1 && (3.10)\\
ycth &= b_1xtth + b_2(xtth)^2 + b_3(xtth)^3 + b_4(xtth)^4 + b_5(xtth)^5 + b_6(xtth)^6 && (3.11)\\
0 &= b_1 + 2b_2xcmc + 3b_3(xcmc)^2 + 4b_4(xcmc)^3 + 5b_5(xcmc)^4 + 6b_6(xcmc)^5 && (3.12)\\
acte &= b_1 + 2b_2 + 3b_3 + 4b_4 + 5b_5 + 6b_6 && (3.13)\\
ycte &= b_1 + b_2 + b_3 + b_4 + b_5 + b_6 && (3.14)
\end{aligned}
$$

The geometries used for training and validation, are generated from variation of these design variables, within the considered ranges displayed in Table 3.1. As it can be observed in the table, the leading edge control point, the trailing edge thickness and the trailing edge camber are maintained constant in order to compare the results with previous work [san08]. Therefore, two or three values in the range of each of the remaining eight geometric variables are used to generate a database of 5000 geometries. Figure 3.3 (left) shows the airfoils defined by the minimum, maximum and averaged value in each of the geometric variables. A huge set of airfoils are included in the database in order to exploit the potential of the evolutionary optimization methods to broadly explore the design space and find the global optimum. Unconventional airfoils are also included in the database to be considered in the optimization process. Figure 3.3 (right) shows examples of ten of the airfoils included in the data base and their geometric variables are displayed in Table 3.2.

3.2.2 Evolutionary Programming algorithm

Evolutionary algorithms [xin99, fog94], are robust problems' solving techniques based on natural evolution processes. They are population-based techniques which codify a set of possible solutions to the problem, and evolve it through the application of the so called *evolutionary operators* [gol89]. Among EAs, Evolutionary Programming (EP) approaches are usually applied to continuous optimization problems. This algorithm is characterized by only using mutation and selection operators (no crossover is applied). Several versions of the algorithm have been proposed in the literature: The Classical Evolutionary Programming algorithm (CEP) was first described in the work by Bäck and Schwefel in [bac93], and analyzed later by Yao et al. in

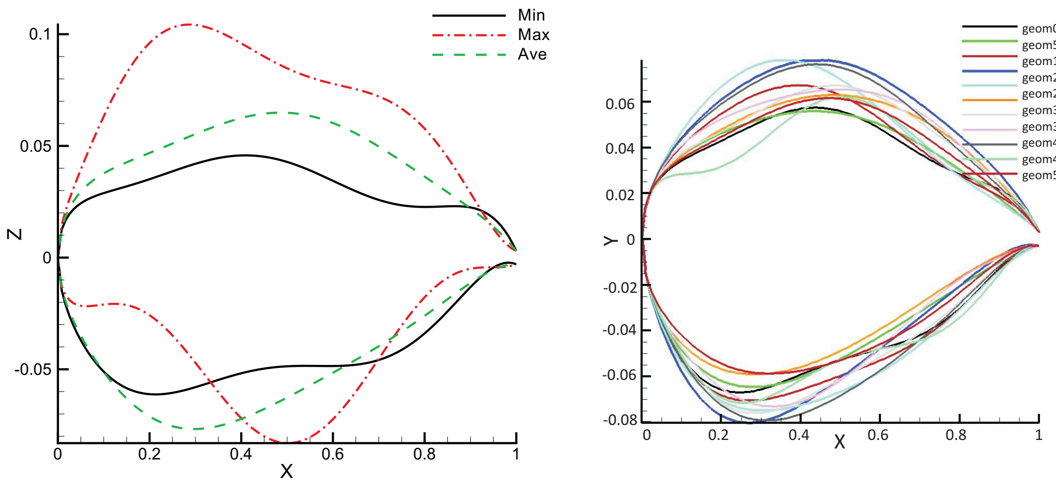


Figure 3.3: Examples of airfoils in the training database; (left) Maximum and minimum configurations in the data base; (right) Example of ten airfoils in the database.

Table 3.2: Geometric variables of different airfoils in the training database.

Geometry	xtth	ytth	atte	acle	ycth	xcmc	ycmc	acte
0	0.337	0.117	-8.625	-4.375	-0.004	-0.725	-0.002	-11.250
500	0.337	0.117	-5.875	-4.375	-0.004	-0.750	-0.005	-3.750
1000	0.337	0.135	-7.250	-4.375	-0.001	-0.725	-0.002	-7.5
1500	0.337	0.152	-8.625	-4.375	-0.001	0.750	0.012	-11.250
2000	0.337	0.152	-5.875	-4.375	-0.001	0.725	-0.002	-3.750
2500	0.375	0.117	-7.250	-4.375	-0.001	0.750	0.012	-7.5
3000	0.375	0.135	-8.625	-1.250	-0.004	0.725	0.005	-11.250
3500	0.375	0.135	-5.875	-1.250	0.004	0.750	0.012	-3.750
4000	0.375	0.152	-7.250	-1.250	0.001	0.725	0.005	-7.500
4500	0.412	0.117	-8.625	-1.250	0.001	0.775	-0.002	-11.250
5000	0.412	0.117	-5.875	-1.250	0.001	0.725	0.005	-7.5

[xin99]. It is used to optimize a given function $f(\mathbf{x})$, i.e. obtaining $\mathbf{x}_o$ such that $f(\mathbf{x}_o) < f(\mathbf{x})$, with $\mathbf{x} \in [lim\_inf, lim\_sup]$. The CEP algorithm performs as follows:

1. Generate an initial population of μ individuals (solutions). Let t be a counter for the number of generations, set it to $t = 1$. Each individual is taken as a pair of real-valued vectors $(\mathbf{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \cdots, \mu\}$, where $\mathbf{x}_i$'s are objective variables, and $\boldsymbol{\sigma}_i$'s are standard deviations for Gaussian mutations.

2. Evaluate the fitness value for each individual $(\mathbf{x}_i,\ \boldsymbol{\sigma}_i)$ (using the problem's objective func-

tion).

3. Each parent $(\mathbf{x}_i, \boldsymbol{\sigma}_i)$, $\{i = 1, \cdots, \mu\}$ then creates a single offspring $(\mathbf{x}'_i, \boldsymbol{\sigma}'_i)$ as follows:

$$\mathbf{x'}_i = \mathbf{x}_i + \boldsymbol{\sigma}_i \cdot \mathbf{N_1(0,1)} \tag{3.15}$$

$$\boldsymbol{\sigma}'_i = \boldsymbol{\sigma}_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot \mathbf{N(0,1)}) \tag{3.16}$$

   where $N(0,1)$ is an unidimensional normal distribution with mean zero and standard deviation one. The parameters τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively [xin99], where n is the length of the individuals.

4. If $x_i(j) > lim\_sup$ then $x_i(j) = lim\_sup$ and if $x_i(j) < lim\_inf$ then $x_i(j) = lim\_inf$.

5. Calculate the fitness values associated with each offspring $(\mathbf{x}'_i, \boldsymbol{\sigma'}_i)$, $\forall i \in \{1, \cdots, \mu\}$.

6. Conduct pairwise comparison over the union of parents and offspring: for each individual, p opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is better than the opponent's, it receives a "win".

7. Select the μ individuals out of the union of parents and offspring that have the most "wins" to be parents of the next generation.

8. Stop if the halting criterion is satisfied, and if not, set $t = t + 1$ and go to Step 3.

A second version of the algorithm is the so called Fast Evolutionary Programming (FEP). The FEP was described and compared with the CEP in [xin99]. The FEP is similar to the CEP algorithm, but it performs a mutation following a Cauchy probability density function, instead of a Gaussian based mutation. The one-dimensional Cauchy density function centered at the origin is defined by:

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2} \tag{3.17}$$

where $t > 0$ is a scale parameter, see [xin99]. Using this probability density function, the FEP algorithm is obtained by substituting step 3 of the CEP, by the following equation:

$$\mathbf{x'}_i = \mathbf{x}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\delta} \tag{3.18}$$

where $\boldsymbol{\delta}$ is a Cauchy random variable vector with the scale parameter set to $t = 1$.

Finally, in [xin99] the *improved FEP* (IFEP) is also proposed, where the best result obtained between the Gaussian mutation and the Cauchy mutation is selected to complete the process.

Note that, in the design problem to be considered in this case, each vector $\mathbf{x}$ is composed by a given parameterization of an airfoil geometry, i.e., $\mathbf{x} = [xtle, ytle, xtth, ytth, atte, \ldots, ycte]$. On the other hand, the objective function f to be optimized is given by the airfoil performance, that in this case will be modeled using a Support Vector Machine approach (that acts as a meta-model), as it is described in the next subsection.

3.2.3 Objective function approximation with a SVMr algorithm

One of the most important statistic models in the field of prediction are the Support Vector Regression algorithms (SVMr) [smo98, smo99]. The SVMrs are appealing algorithms for a large variety of regression problems, in many of them mixed with evolutionary computation algorithms [che11, sal11, jia12]. Although there are several versions of SVMr, in this case the classic model presented in [smo98] will be described.

The ϵ-SVMr method for regression consists of training a model of the form $y(\mathbf{x}) = f(\mathbf{x}) + b = \mathbf{w}^T\phi(\mathbf{x}) + b$, given a set of training vectors $C = \{(\mathbf{x_i}, y_i), i = 1, \ldots, l\}$, to minimize a general risk function of the form

$$R[f] = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} L\left(y_i, f(\mathbf{x})\right) \tag{3.19}$$

where $\mathbf{w}$ controls the smoothness of the model, $\phi(\mathbf{x})$ is a function of projection of the input space to the feature space, b is a parameter of bias, $\mathbf{x_i}$ is a feature vector of the input space with dimension N, y_i is the output value to be estimated and $L\left(y_i, f(\mathbf{x})\right)$ is the loss function selected. In this case, it is used the L1-SVMr (L1 support vector regression), characterized by an ϵ-insensitive loss function [smo99]

$$L\left(y_i, f(\mathbf{x})\right) = |y_i - f(\mathbf{x_i})|_\epsilon \tag{3.20}$$

In order to train this model, it is necessary to solve the following optimization problem [smo99]:

$$\min\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)\right) \tag{3.21}$$

subject to

$$y_i - \mathbf{w}^T\phi(\mathbf{x_i}) - b \leq \epsilon + \xi_i, \quad i = 1, \ldots, l \tag{3.22}$$

$$-y_i + \mathbf{w}^T\phi(\mathbf{x_i}) + b \leq \epsilon + \xi_i^*, \quad i = 1, \ldots, l \tag{3.23}$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, l \tag{3.24}$$

The dual form of this optimization problem is usually obtained through the minimization of the Lagrange function, constructed from the objective function and the problem constraints. In this case, the dual form of the optimization problem is the following:

$$\max\left(-\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x_i}, \mathbf{x_j}) - \right.$$

$$\left. -\epsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*)\right) \tag{3.25}$$

subject to

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \tag{3.26}$$

$$\alpha_i, \alpha_i^* \in [0, C] \tag{3.27}$$

In addition to these constraints, the Karush-Kuhn-Tucker conditions must be fulfilled, and also the bias variable, b, must be obtained [smo99] . In the dual formulation of the problem the function $K(\mathbf{x_i}, \mathbf{x_j})$ is the kernel matrix, which is formed by the evaluation of a kernel function, equivalent to the dot product $\langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle$. A usual selection for this kernel function is a Gaussian function, as follows:

$$K(\mathbf{x_i}, \mathbf{x_j}) = exp(-\gamma \cdot \|\mathbf{x_i} - \mathbf{x_j}\|^2). \tag{3.28}$$

The final form of function $f(\mathbf{x})$ depends on the Langrange multipliers α_i, α_i^*, as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(\mathbf{x_i}, \mathbf{x}) \tag{3.29}$$

In this way it is possible to obtain a SVMr model by means of the training of a quadratic problem for given hyper-parameters C, ϵ and γ. However, obtaining these parameters is not a simple procedure, being necessary the implementation of search algorithms to obtain the optimal ones or the estimation of them [ort09].

The selection of hyper-parameters for SVMs is a key point in the training process of these models when applied to regression problems. Unfortunately, an exact method to obtain the optimal set of SVM hyper-parameters is unknown, and search algorithms are usually applied to obtain the best possible set of hyper-parameters. In general, these search algorithms are implemented as grid searches, which are time-consuming, so the computational cost of the SVM training process increases considerably. The search algorithms used to obtain SVM hyper-parameters can be divided in three groups. The first group of algorithms for SVM hyper-parameters is based on *grid search* [aka98], where the search space of parameters is divided into groups of possible parameters to be tested, usually in a uniform fashion. The second group of search algorithms is formed by local search type approaches, such as *pattern search* proposed in [mom02]. Finally, the third group is based on metaheuristic, or global optimization algorithms, such as evolutionary computation [wan05]. All these search algorithms have similar problems: first, the selection of the initial ranges of parameters, which limit the search space. In most cases, these initial ranges are selected by experience of the researcher, or using large ranges of parameters. The former case is highly dependent on the regression problem studied, and it is a very difficult and specific task, not useful in the majority of occasions. In the latter case, the usage of large parameter ranges implies the increasing of the search space, and thus the training time of the SVM.

In [ort09], a novel effort to improve the SVM training time through the reduction of the search space is presented. The objective is to reduce the training time necessary to find the final parameters of the SVM model, while maintaining the performance of the models. This search space reductions are generated by bounding the SVM hyper-parameters, mainly parameter C. This parameter is bounded by taking to account its relation with parameters γ and ϵ, through an approximation of the SVM model. Parameter γ is bounded using characteristics of the Gaussian kernel function used, and finally parameter ϵ bound is constructed basing on some previous results in the literature [kwo03]. All these reductions are applied to a grid search algorithm, in order to find the parameters which obtain the best SVM performance, reducing the computation time of the full search space case.

Given the amount of aerodynamic data available to train the SVMr-based model, it was necessary to split them into several groups, each corresponding to a different value of the angle of attack. Subsequently, each of these datasets is employed to train a different model, which will be associated to its corresponding angle. In case of values different from those 24 angles existing in the original data, a linear interpolation of the two nearest models has been carried out. Figure 3.4 shows the architecture of the proposed SVMr model. Note that the problem is tackled with a network of SVMr banks, defining a single SVMr for each angle of attack, both in the train and test periods.

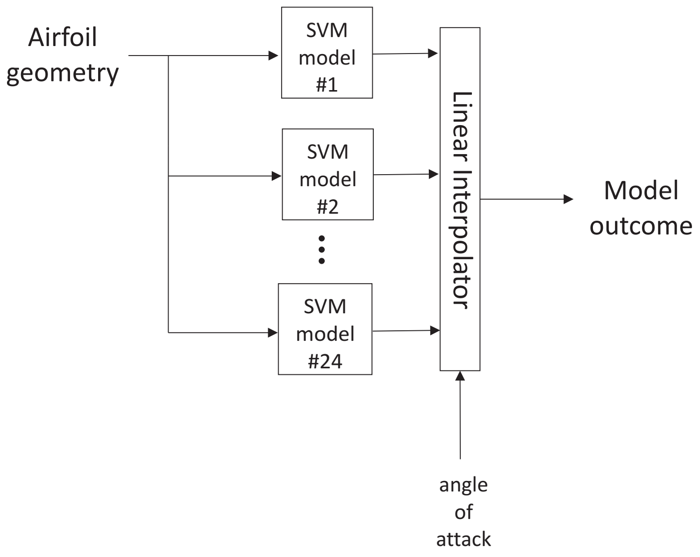


Figure 3.4: SVMr banks architecture applied in this work.

3.3 Experiments and results

3.3.1 Experiments on the metamodel obtention (SVMr)

In this section, only subsonic computations are considered for both prediction and airfoil design optimization. The panel code XFOIL [dre89] was used to generate the necessary data. In addition, the first steps in the prediction of aerodynamic coefficients of transonic configurations have been performed using the RANS TAU code and preliminary results will be mentioned. The data base is comprised of 5000 geometries and their related aerodynamic coefficients (drag, lift and momentum coefficients). In particular, 4000 geometries and their related data were used for training, and the other 1000 geometries were used as validation tests to measure the precision. The chosen Mach and Reynolds numbers for the dataset samples are 0.3 and 10^7 respectively. The angle of attack is ranging from 4^o to 14^o and free transition is used.

Once the network has been trained, a set of 1000 geometries is used for validation purposes. The airfoil geometries used in the validation were obtained in a random way from the initial data set. Figure 3.5 shows a comparison of lift (C_l) curve (left) and drag polar (right) predicted and the ones calculated with XFOIL for two of the validation geometries. Note that there is a very good agreement between the predicted and calculated curves. Indeed, these values could be predicted by the neural network with a small mean error as it is showed in the next section.

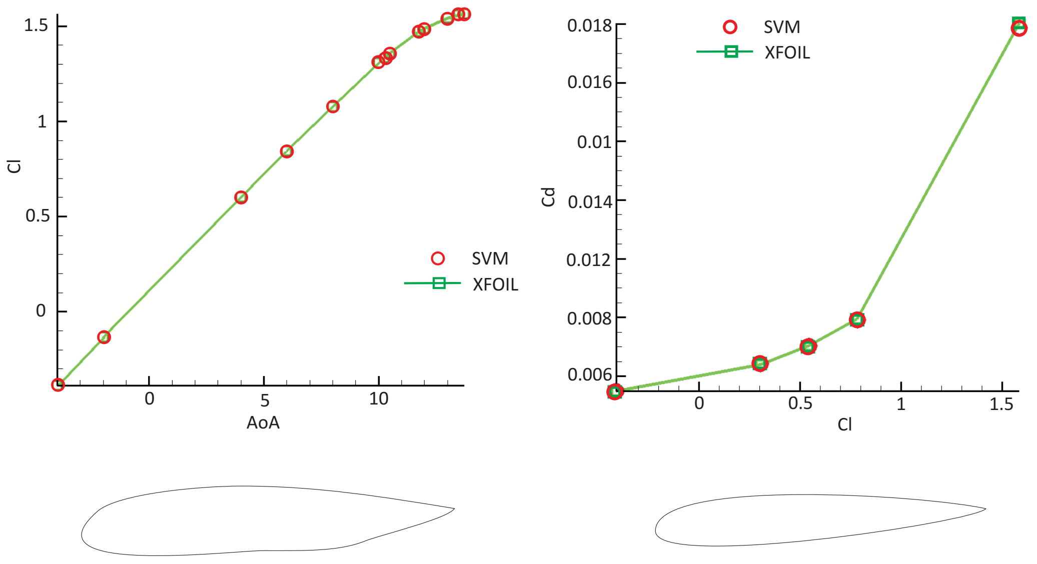


Figure 3.5: Coefficients prediction for the displayed airfoils; (left) Lift (C_l) curve prediction; (right) Drag (C_d) polar prediction.

The error of the network in the prediction of the aerodynamic coefficients (drag, lift and momentum) was analyzed with all the geometries selected for validation. Table 3.3 shows the Root Mean Square Error (RMSE) and the Maximum Error for each coefficient. At a first glance, the error seems to be high because the maximum error for the lift coefficient reaches a value higher than 200 lift counts, but the RMSE provides a more reasonable value. It is clear that all the errors could be reduced by increasing the training dataset size, but precision should be balanced together with time constraints. The relation between the training dataset size and precision is briefly analyzed later.

Table 3.3: Maximum error and RMSE for lift, drag and momentum coefficients in subsonic configurations.

	RMSE	Maximum Error
Cl	0.00475	0.20473
Cd	0.00055	0.03380
Cm	0.00099	0.00756

Figure 3.6 (left) shows the histogram of the error in predicting the drag coefficient, and reflects how most of the samples have an error close to zero, and only isolated samples have a significant error. Figure 3.6 (right) shows the RMSE for the prediction of drag, lift and momentum coefficients when using different angles of attack. It can be observed that the maximum RMSE occurs for lift coefficient prediction in high angles of attack (10-14 degrees), close to the stall, but the values are still lower than $9 \cdot 10^{-3}$ which could be considered reasonable for a fast prediction method.

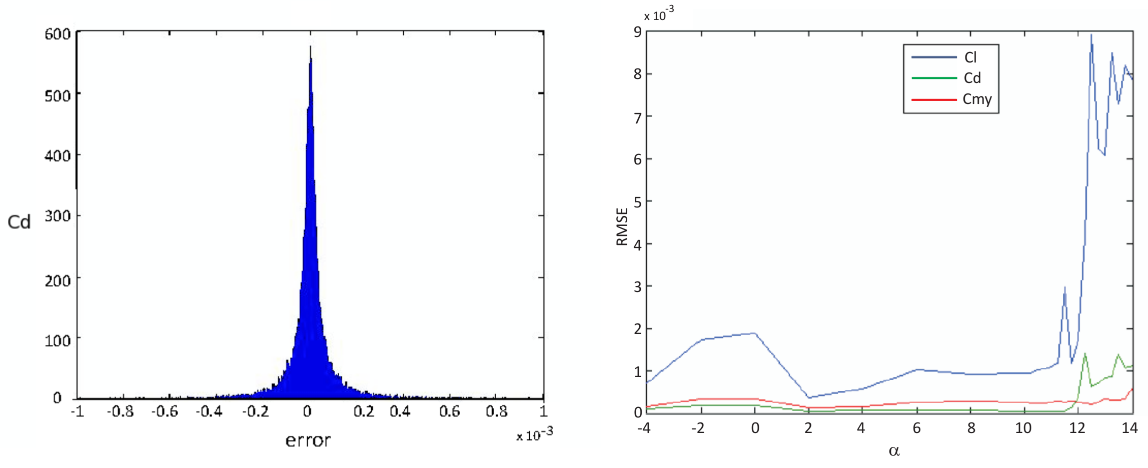


Figure 3.6: Coefficients prediction characteristics; (left) Histogram of the error in predicting the drag coefficient; (right) RMSE for the prediction of drag, lift and momentum coefficients when using different angles of attack.

Regarding the precision of the proposed approach, several experiments have been carried out in order to analyze how the obtained results depend on the training data size. As it could be expected, the achieved accuracy is reduced to some extent as the number of patterns employed to train the network decreases. To illustrate this effect, Figure 3.7 represents the performance provided by the metamodel (SVMr) respect to the training data size. Specifically, the ratio between the RMSE produced at each point and the best RMSE value (achieved with the maximum number of patterns, 4000) is plotted. It is interesting to note how each of the considered aerodynamic coefficients behaves on a different way when the number of training samples decreases. While C_l maximum error only increases about 15 times as training data sets are shortened, C_{my} seems to be much more sensitive to reductions in the amount of patterns, since RMSE is increased by a factor of 230 in this case.

3.3.2 Inverse airfoil design given subsonic flow conditions

In this section, the proposed approach given in Figure 3.1 is applied to the preliminary optimization phase of airfoils. The flow conditions are set to Mach=0.3, AoA=2°, $Re = 10^6$ and the network is asked for the optimal airfoil in such design point. The objective function has been set as the efficiency (Cl/Cd) to be maximized. This optimization problem could be considered as an inverse design, in the sense that the evolutionary approach returns an approximation to the optimal airfoil for certain flow conditions. The precision of this approximation will be related

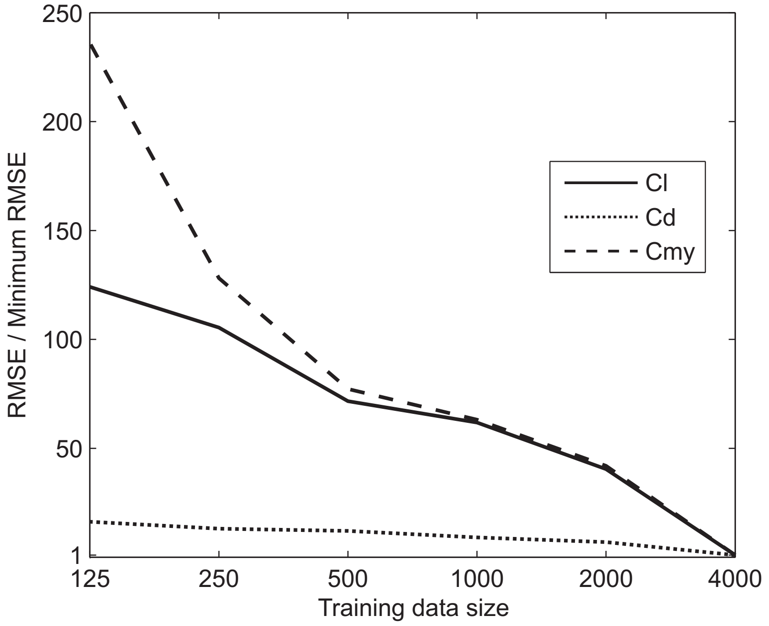


Figure 3.7: SVMr performance in terms of the training data size.

to the precision of the SVMr used as the metamodel to estimate the objective function of the evolutionary algorithm (Cl/Cd). Comparing the airfoil provided by the SVMr model to those airfoils employed along the training phase and their Cl/Cd XFOIL values (the best of them are drawn in Figure 3.8), only 0.15% of these were able to outperform the proposed profile. This point agrees with the distribution function of the training profiles and its corresponding cumulative distribution, which are shown in Figure 3.9. Both plots include a red arrow, indicating the value $Cl/Cd = 88.72$, which corresponds to the XFOIL computation of the airfoil obtained through the evolutionary method (the SVMr predicted value was $Cl/Cd = 96.60$). As can be seen, this value is placed at the end of the distribution tail, whereas there exist a 99.85% of training patterns providing lower performance, when considering XFOIL computations.

A further flow analysis, around the optimal geometry, returned by SVMr, provides the results in Table 3.4. Figure 3.8 shows the best 5 geometries for the considered target (including the one returned from the SVM model) and their XFOIL Cl/Cd value. These geometries have the same value for all the design parameters, except for the trailing edge thickness line angle, leading edge camber line angle and position for maximum camber which differ slightly. Therefore, the returned airfoil, whose geometric parameters are displayed in Table 3.5, could be a good starting point, close to the optimal solution, for detailed design.

Table 3.4: Aerodynamic results for the returned airfoil after the optimization process.

Cl	Cd	Cm	L/D	Trans. x/c upper part	Trans. x/c lower part
0.495	0.0056	-0.0748	88.7	0.2889	0.4941

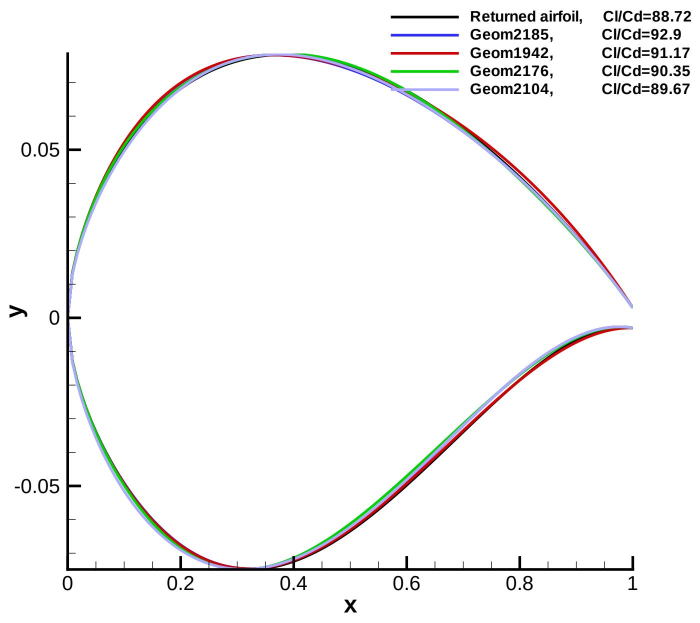


Figure 3.8: Best 5 geometries obtained in the optimization process (including the one returned from the SVMr model) and their XFOIL Cl/Cd value.

Table 3.5: Geometric variables for the returned airfoil after the optimization process.

xtth	ytth	atte	acle	ycth	xcmc	ycmc	acte
0.347	0.152	-6.166	1.384	0.001	0.770	0.011	-7.446

This approach allows extensively exploring the design space, without any dependence on an initial solution and expensive CFD computations, but as it uses a metamodel to estimate the aerodynamic coefficients, the return geometry is also an approximation to the optimal geometry, and therefore, a further detailed design using gradient-based methods should be applied (see next chapter). For completeness purpose, the flow conditions are set to Mach=0.3, $Re = 10^6$, as in the previous case, but now the angle of attack is varying from 0^o to 10^o, and the network is asked for the optimal airfoil in each of the design points. The objective function was again the efficiency (Cl/Cd) to be maximized. Figure 3.10 shows the returned airfoils for each of the angles of attack and their aerodynamic properties are displayed in Table 3.6. It can be observed that the returned airfoil for angles of attack 4^o and 6^o and for 8^o and 10^o are quite similar, and due to the small differences with airfoils from previous angles of attack were not visible in this figure. Table 3.7 shows the geometric variables for these returned airfoils.

As it was mentioned previously, speed and broad exploration of the design space are usually more important than precision in a preliminary design phase. Therefore, it is necessary to evaluate the computational cost of the proposed approach, in order to support its applicability

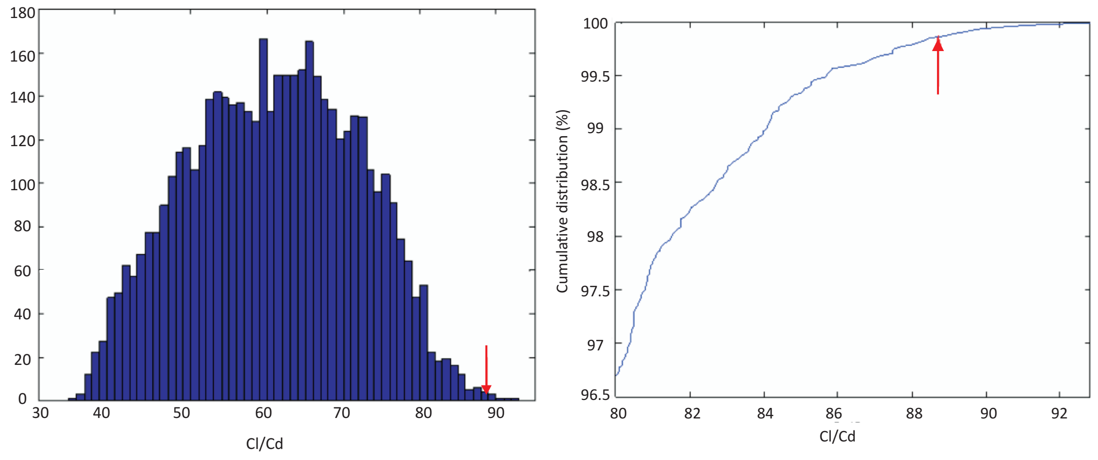


Figure 3.9: Distribution function of the training profiles and its corresponding cumulative distribution; (left) Distribution function; (right) Cumulative function.

Table 3.6: Aerodynamic results of several returned airfoils for different angles of attack.

AoA	Cl	Cd	Cm	L/D	Trans. x/c upper part	Trans. x/c lower part
0º	0.209	0.006	-0.069	34.83	0.1813	0.3517
2º	0.495	0.005	-0.074	88.7	0.2889	0.4941
4º	0.642	0.007	-0.061	86.58	0.0255	0.5200
6º	0.881	0.008	-0.061	103.02	0.0175	0.5554
8º	1.106	0.010	-0.070	103.19	0.0124	0.5865
10º	1.305	0.013	-0.056	99.71	0.0077	0.7262

in such a design step. For the SVM network used as a metamodel to estimate the objective function, an initial training phase is necessary. It is important to remark that this phase is only performed at the beginning of the process and, once the network is trained, it can be used for optimization without any additional cost. Table 3.8 shows the computational time for each phase of the proposed approach. Note that once the network is trained, it is only 32 seconds to return the optimized geometry for a given condition.

3.3.3 A short note on predicting coefficients in transonic configurations

In order to provide a complete overview of the prediction capabilities of the network, the analysis could not be limited to subsonic cases, and therefore, transonic configurations were also preliminary considered. For that purpose, a training data set of around 100 samples was obtained using the DLR TAU code for the NACA0012 profile with Mach ranging from 0.5 to 0.8, angle of attack ranging from 0 to 12 degrees and Reynolds number of $6 \cdot 10^6$. Given the short number of samples considered, a 5-fold cross validation procedure was applied over this set to ensure a meaningful

Table 3.7: Geometric variables of several returned airfoils for different angles of attack.

AoA	xtth	ytth	atte	acle	ycth	xcmc	ycmc	acte
0°	0.338	0.118	-8.625	-1.477	-0.001	0.724	0.008	-9.266
2°	0.347	0.152	- 6.166	1.384	0.001	0.770	0.011	-7.446
4°	0.412	0.129	-5.874	1.875	-0.003	0.724	0.012	-3.936
6°	0.412	0.129	-5.874	1.874	-0.003	0.725	0.012	-3.936
8°	0.412	0.150	-8.576	-4.374	-0.004	0.765	0.010	-9.978
10°	0.412	0.150	-8.576	-4.374	-0.004	0.765	0.010	-9.978

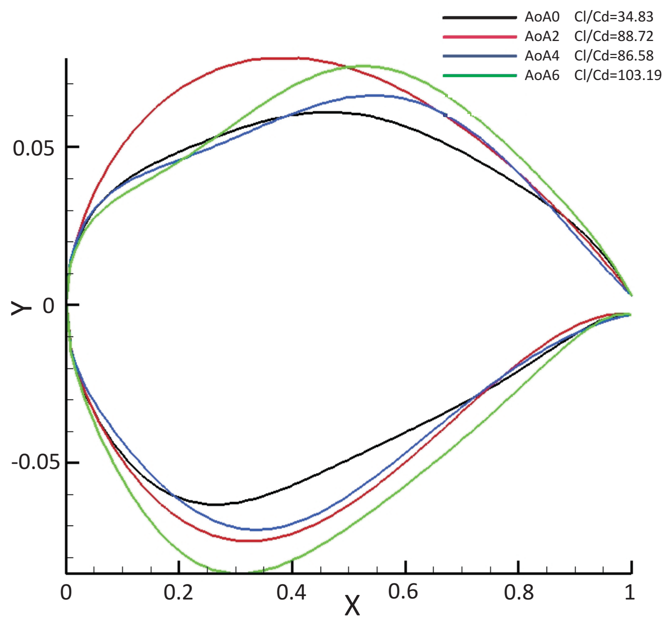


Figure 3.10: Several returned airfoils for different angles of attack.

error value. This method consists in splitting the available data in 5 sets, after having randomly shuffled them. Each of these sets is employed once to test the performance of the prediction model that is obtained from the other 4 sets. Then, the final error is obtained by averaging the error values related to each of the 5 models.

Table 3.9 shows the RMSE and the Maximum Error for each aerodynamic coefficient prediction. The error measures are one order of magnitude higher compared to the previous case. This is mainly due to the training data set size and these errors could be lowered by increasing the size or redistributing the initial samples. An RMSE of 16 lift counts or 37 drag counts could be reasonable or not, depending on the particular application and its requirement. If computational time constraints are much higher than accuracy constraints, as actually occurs in the preliminary

Table 3.8: Computational time of each phase.

Phase	Time
Initial training of the SVM network	11300 s.
Evaluation of the objective function(Cl/Cd)	1.42 ms.
Optimization	32 s.

Table 3.9: Maximum error and RMSE for lift, drag and momentum coefficients in transonic configurations.

	RMSE	Maximum Error
CL	0.0166	0.0562
CD	0.0037	0.0093
CM	0.0056	0.0137

design, then the prediction could be reasonable and accepted. The maximum error is lower than the previous case because the geometry is kept constant.

In addition, as the objective of this phase is to provide an initial shape for a finer design phase, to be detailed in the next chapter, this level of accuracy can be considered acceptable.

3.4 Conclusions

This chapter has focused on the preliminary design step of the aerodynamic shape optimization process. The main contribution is the application of an hybrid EP+SVMr approach showing promising results. The evolutionary algorithm and the Support Vector Regression algorithm used in the process have been detailed. The good performance of the complete approach has been shown in different experiments for subsonic and transonic configurations. Future work will address the design of 3D transonic configurations, through the use of the proposed combined approach where the savings in computational time would be relevant. Furthermore, future activities will also consider other disciplines in addition to aerodynamics, in order to exploit the potential of metamodel assisted evolutionary techniques to perform multidisciplinary optimizations (MDO).

Next chapter focuses on the fine design phase, which will evolve an initial configuration, provided from the application of the proposed approach for global optimization, into the final optimized geometry.

Chapter 4

Fine grain CAD based aerodynamic design

4.1 Introduction

With the improving capabilities of computational fluids dynamics for the prediction of aerodynamic performance, CFD tools are now being increasingly used for aerodynamic design optimization in the aerospace industry. In this chapter, an automated optimization framework is presented to address inviscid aerodynamic fine grain design problems. Key aspects of this framework include the use of the continuous adjoint methodology to make the computational requirements independent of the number of design variables, and Computer Aided Design (CAD)-based NURBS shape parameterization, which uses the flexibility of Non-Uniform Rational B-Splines (NURBS) to handle complex configurations.

Within a gradient-based optimization approach, the use of the adjoint methodology [jam03, ell95, jam98, mav06, and99, nad07] has been introduced during the last decade and it has demonstrated to be an efficient method to compute the gradients of an objective function with a cost which is essentially independent of the size of the parameter space of the geometry.

In addition to gradient based methodologies, there are many methods for various optimal design applications. Gradient-based methods are very efficient if a well formed objective function is employed, but require the determination of the derivatives of the objective function with respect to shape variations. A comparison of common and recently introduced global optimization methods can be found in [you08].

The adjoint approach yields the flow sensitivities for a set of design variables at a cost which is independent of the size of the design space. The most obvious strategy is to directly employ the surface grid points as design variables. These methods are referred as CAD-free [stu11], since the optimization is done using the computational mesh and the original geometry parameterization is no longer involved. One of the main problems of using the grid points directly as design variables is the appearance of surface bumps during the optimization process, which requires a smoothing algorithm [sch08]. The presence of bumps usually produces a geometry which is optimized for a specific flight condition, but which behaves very poorly for off-conditions.

In order to solve these problems, a higher-level parameterization should be considered, such as Hicks-Henne functions [hic78], global shape functions or analytic descriptions of foil sections [kul06, kul07] associated to a set of predefined parameters, or as the PARSEC method [sob98], which uses eleven geometric parameters to manipulate the shape of an airfoil. The computational cost to calculate the derivatives of the objective function using the finite difference method make parameterizations with few design variables attractive, but these parameterizations are usually restricted for a particular design problem and 3D geometries may not be properly represented. The use of NURBS has been suggested in [lep00, ben05, mou07] to represent irregular and complex geometric forms, which is not limited to aerodynamic. Moreover, CAD-systems predominantly use NURBS as a general and flexible parameterization scheme for representing surfaces.

The main advantages of the approach considered for shape optimization in this work are:

- The use of the adjoint methodology based on boundary integrals for fast gradient computation at a cost which is essentially independent of the number of design variables.
- The use of Computer Aided Design (CAD)-based parameterization via NURBS representation of the geometry. This is of particular interest, as it enables the fast incorporation of the optimization procedure into existing design chains within the aerospace industry.

The proposed approach combines the adjoint methodology for calculating the gradients of the cost function with a NURBS representation of the geometry. Independently, both have been the focus of recent research for optimal design, but the application of both techniques has not been deeply considered so far. The essential piece that links both methodologies is the so-called "point inversion problem" (the computation of the parametric coordinates , which are calculated as the projection of a surface grid points onto the NURBS).

The organization of this chapter is as follows: First, the proposed CAD-based aerodynamic shape optimization process is described to introduce all the steps involved. Then, the NURBS parameterization and the approach employed for the point inversion problem are explained. The continuous adjoint formulation, as well as the procedure to obtain the gradients with respect to the position of the control points of the NURBS, are described in section 4. Finally, the optimization framework is applied to several 2D and 3D configurations and the numerical results are explained.

4.2 CAD-based aerodynamic shape optimization process

As has been mentioned above, the proposed approach is a combination of the continuous adjoint methodology for gradient computation with a NURBS-based parameterization for the geometry. The CAD-based shape optimization process thus comprises the following steps, displayed in Figure 4.1. In the following, each of the steps involved in the process will be briefly described:

4.2.1 CAD geometry and parameterization

The original geometry is an input to the process, and it can be obtained directly from CAD applications such as, for example, CATIA (in IGES or STEP formats). In shape design, the best representation of the geometry remains an open problem. In the current study, geometries are modeled with NURBS, which offer great flexibility in the representation of surfaces. The

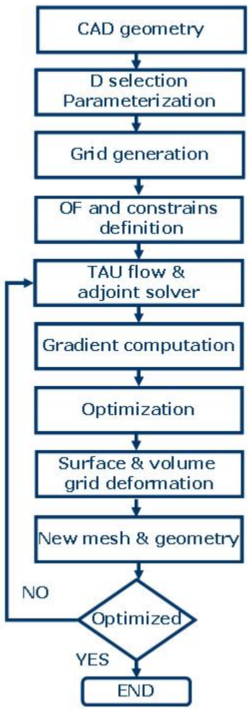


Figure 4.1: CAD-based optimization process. D stands for design variable, while OF stands for objective function.

design variables are the coordinates (x,y,z) and weights of the NURBS control points, so for each control point there are four design variables (this is also the case in 2D, where obviously, design variables corresponding to spanwise deformations are not considered in the optimization process). The main advantage of using NURBS is that they provide a global parameterization with a smooth surface and a better control of the curvature while still maintaining the locality in the deformation. In addition, the optimized surface at the end of the process has the correct format to be fed directly to CAD and grid generation applications. Having the possibility to handle different NURBS patches for each part of an aircraft, as for example the wing and fuselage, this parameterization is able to represent almost any kind of surface of industrial interest. While this feature allows different levels of optimization, working with several NURBS patches requires nevertheless a correct treatment of the intersections, particularly with regard to the continuity between patches.

In practice, the NURBS description provided directly by CAD tools is not usually appropriate for an optimization process and has to be modified, for example, by changing the order of the NURBS or by setting a more uniform distribution of the control points, in order to improve the chances for a successful optimization.

4.2.2 Objective functions and constraints

The most common objective functions such as, for instance, those aiming at drag minimization at constant lift, have been implemented in the optimization framework. An analysis of the efficiency of each objective function has been performed and the results will be presented later. During the optimization, the following constraints can be applied:

- Fixed control points in all directions.
- Fixed control points in one direction.
- Fixed control points weights.

For wing sections, it is a common practice to fix the control points located at both the trailing and leading edges, thus keeping the chord length constant. Volumetric and other geometric constraints can also be applied, but the objective function of such problems is strongly non-linear, so that the optimization most likely converges towards a local minimum that is similar to the original configuration.

4.2.3 TAU Flow and Adjoint solver

The simulation of the flow and the surface cost function sensitivities are calculated using the TAU solver. The continuous adjoint approach yields sensitivities which depend only on surface data [cas07]. TAU flow and adjoint solver modules are executed to obtain local surface sensitivity values for the selected cost function over the boundary grid points. As it will be shown later, the computation of the flow and adjoint solutions consumes most of the required computational time in the whole process.

4.2.4 Gradient computation over the NURBS control points

A perturbation δD of a given design variable D gives rise to a perturbation δx of the geometry X as well as a perturbation δw of the flow variables w. The gradient of a cost function I with respect to D can then be obtained, within the continuous adjoint framework [cas07], as the boundary integral displayed in Equation 4.1

$$\delta I = \int_S (\delta x \cdot n) G(w, \Lambda) ds, \tag{4.1}$$

where the local surface sensitivity G is a computable function of the flow w and adjoint Λ variables and their derivatives on the surface $S, \delta x = (\partial x / \partial D)\delta D$, are the geometric sensitivities, which can be analytically calculated from the NURBS equations, n is the surface normal and ds is the surface measure.

Figure 4.2 (right) shows a plot of the local sensitivity G for the drag coefficient for the inviscid transonic flow around a RAE2822 airfoil (left). Notice that the gradient in equation 4.1 does not depend on the tangent part of the geometric sensitivities, whose contribution to the integral vanishes identically in the continuous limit. However, it remains to be seen whether this is also the case at the numerical level, especially at or near geometric singularities such as the trailing edge of wings or airfoils.

4.2.5 Surface and volume deformation

A simple steepest descent optimization algorithm has been implemented into the framework for testing purposes. The design variables are the $\{x, y, z\}$ coordinates and weight w of the NURBS

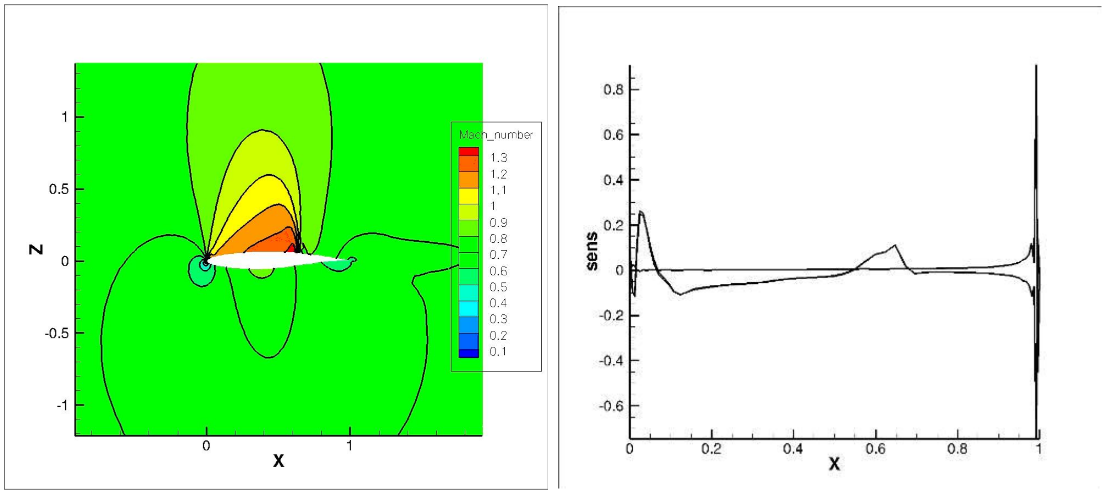


Figure 4.2: Mach number (left) and C-drag surface sensitivity (right) along an RAE2822 airfoil at flow conditions M = 0.729 and $\alpha = 2.31^o$.

control points. In practice, however, the weight is rarely used. Variations of the knots distribution and of the basis functions are not considered. For each design variable, the desired surface deformation is achieved by the perturbation of the NURBS control points and the regeneration of the surface grid using the parametric coordinates calculated from the point inversion algorithm. Figure 4.3 (left) shows the surface deformation for a RAE2822 profile. In this process, a point on the previous (or initial) surface is assumed to move to the new surface while remaining at fixed values of the NURBS parametric coordinates. This guarantees that the surface grid points always remain on the NURBS surface. Furthermore, as the same parametric values are used throughout the optimization process, the point inversion step needs only to be done once and its contribution to the computational cost is independent of the number of optimization steps of the process.

Once the new surface grid is obtained, the volume grid is deformed with an advancing front algorithm [dlr94] using the TAU deformation module, as can be observed in Figure 4.3 (right).

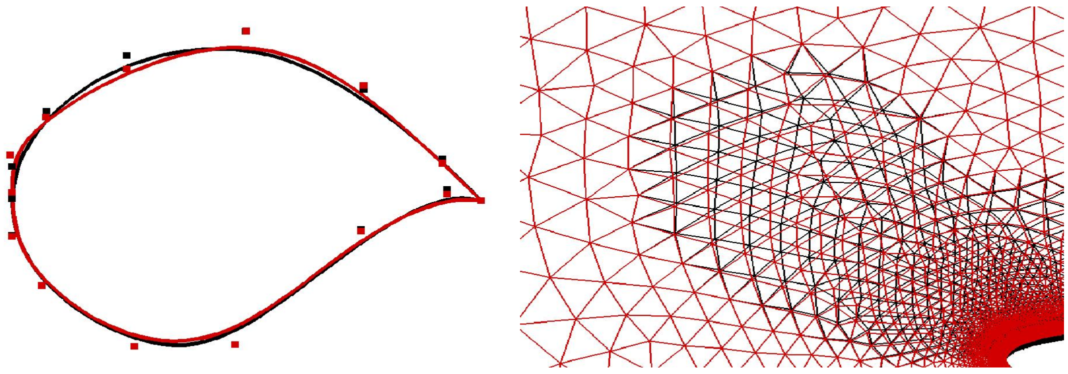


Figure 4.3: Surface (left) and volume (right) grid deformation for a RAE2822 profile.

4.3 Shape parameterization

4.3.1 NURBS for the optimization process

The geometry parameterization is crucial in an automatic aerodynamic design optimization problem. NURBS have demonstrated to be able to accurately represent a large family of geometries; furthermore, they are also the standard for geometry definition and representation in IGES (International Graphics Exchange Specification) [ige06]. In addition, from a practical point of view, using the same CAD format significantly reduces the integration effort necessary to carry out multidisciplinary design of complex configurations. In general, NURBS provided directly by the CAD applications are designed to accurately describe the geometry, but they are usually not the most appropriate as the initial parameterization in an optimization process. Numerical testing suggests that optimizations tend to work better with a low number of control points and high order NURBS [mag11]. There is then a trade-off between the number and distribution of the control points, the accuracy of the geometric representation and the range of possible shapes the optimization can reach. In all the tests presented in this work, the maximum error between the NURBS parameterized shape and the reference geometry is lower than 0.5%, although this error can be obviously reduced by increasing the number of control points.

NURBS ensure good smoothness properties [mou07], reducing the risk of numerical noise, while the parameterization is still local, which means that when a control point is perturbed, only a portion of the surface is modified, leaving the rest intact, and gives enough freedom for the optimizer to converge to an optimal design [and10].

4.3.2 Brief mathematical background of Non Uniform B-Splines (NURBS)

From a mathematical point of view, a surface $S(\xi, \eta)$ can be described with a NURBS function defined by equation 4.2

$$S(\xi,\eta) = \frac{\sum_i^I \sum_j^J U_{i,p}(\xi) V_{j,q}(\eta) w_{ij} C_{ij}}{\sum_i^I \sum_j^J U_{i,p}(\xi) V_{j,q}(\eta) w_{ij}} \tag{4.2}$$

where (ξ, η) are the parametric coordinates on the NURBS surface, C are the Cartesian coordinates of the control points, w are the weights of the control points and U and V are the basis functions which are given by expression 4.3:

$$U_{i,k=1}(\xi) = \begin{cases} 1 & \text{if } u_i \leqslant \xi < u_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

$$U_{i,k}(\xi) = \frac{(\xi - u_i) U_{i,k-1}(\xi)}{u_{i+k-1} - u_i} + \frac{(u_{i+k} - \xi) U_{i+1,k-1}(\xi)}{u_{i+k} - u_{i+1}} \tag{4.3}$$

(and an analogue expression for $V_{j,m}$) in which i corresponds to the i-th control point and k corresponds to the degree of the basis function. The parameters u_i are the knots, which are conventionally assembled into a knot vector $\overline{U}$ as Equation 4.4.

$$\overline{U} = \{\underbrace{0,...,0}_{p+1}, u_{p+1}, ..., u_i, ..., u_I, \underbrace{1,...,1}_{p+1}\} \tag{4.4}$$

The basis functions are zero everywhere except for an interval delimited by the order of the NURBS, defining the area of influence of each control point. A more extended reference for NURBS can be found in [pie97].

4.3.3 Point inversion algorithm

Starting from a given CAD geometry, a NURBS representation and a discrete surface grid are independently derived. To proceed with the optimization algorithm, the surface grid points have to be projected onto the original CAD geometry which is stored as a set of NURBS. In practice, this requires the assignment of the corresponding parametric coordinates (ξ, η) to each point of the surface grid. In this way, during the optimization process, using the parametric coordinates (ξ, η) it is easy to map the vertices of the computational grid onto the deformed NURBS surface and obtain the new spatial coordinates. The process $\mathbb{R}^3\{x, y, z\} \to \mathbb{R}^2\{\xi, \eta\}$ to assign space coordinates to parametric coordinates of the NURBS is usually referred to as the point inversion problem. The parametric values are not provided by the grid generator application and therefore the point inversion is necessary. Furthermore, the NURBS provided by CAD applications are usually not suitable for an optimization process, as was observed in 4.3.1, and therefore a new NURBS, based on the original one, is required to represent the geometry. This new NURBS is usually obtained by reducing the number of control points and increasing the order of the NURBS.

Various iterative point inversion algorithms are available [hu05, lia03, sel06, red09], all of which suffer from the same problem: they perform well only if an appropriate interval or initial value is provided. To make the problem even harder, the vertices of the grid do not always exactly lie on the NURBS surface, and the presence of discontinuities, intersections and kinks is also frequent. The approach followed seeks to provide an initial estimate of the solution to the point inversion problem and then improve the accuracy by using an iterative Newton-Raphson method. Two algorithms were developed for the initial estimation. The source code of these algorithms can be requested at DOMINO Project [dom11].

First method, based on the normals

In this approach, the initial estimate for the point inversion algorithm is obtained by projecting the surface grid points onto a simpler, second-order NURBS, along the local vertex normal vector N. At each surface grid point P, the normal projection equations are calculated using Equation 4.5

$$\sum_{i=I-1}^{I} \sum_{j=J-1}^{J} U_{i,2}(\xi) V_{j,2}(\eta) G_{i,j} = \phi(r \cdot N + P) \tag{4.5}$$

which have to be solved for r and (ξ, η), where $U_{i,2}$ and $V_{j,2}$ are the basis functions for the reduced-order NURBS (which has order 2), $G_{i,j}$ are the weighted control point coordinates,

$$G_{i,j} = C_{i,j} w_{i,j} \tag{4.6}$$

and ϕ is the normalization factor in the NURBS equation

$$\begin{aligned}\phi = \sum_i^I \sum_j^J U_{i,2}(\xi)V_{j,2}(\eta)w_{i,j} = u_i v_j w_{i,j} + u_{i+1}v_{j+1}w_{i+1,j+1} - \\ u_{i+1}v_j w_{i+1,j} - u_i v_{j+1} w_{i,j+1}\end{aligned} \tag{4.7}$$

The most common situation occurs when all control points have the same weight, in which case ϕ is a constant. Expanding Equation 4.5 yields the following second order system of equations

$$\bar{a}\xi\eta + \bar{b}\xi + \bar{c}\eta + \bar{d}r + \bar{e} = 0 \tag{4.8}$$

where

$$\begin{aligned}\bar{a} &= G_{i-1,j-1} + G_{i,j} - G_{i-1,j} - G_{1,j-1} \\ \bar{b} &= v_j G_{i-1,j} + v_{j+1}G_{i,j-1} - v_{j+1}G_{i-1,j-1} - v_j Gi,j \\ \bar{c} &= u_{i+1}G_{i-1,j} + u_i G_{i,j-1} - u_{i+1}G_{i-1,j-1} - u_i G_{i,j} \\ \bar{d} &= -\phi(u_{i+1} - u_i)(v_{i+1} - v_i)N \\ \bar{e} &= u_{i+1}v_{j+1}G_{i-1,j-1} + u_i v_j G_{i,j} - u_{i+1}v_j G_{i-1,j} - \\ &\quad u_i v_{j+1} G_{1,j-1} - \phi(u_{i+1} - u_i)(v_{i+1} - v_i)P\end{aligned} \tag{4.9}$$

Since this is a second order NURBS, each knot sequence is paired with one control point. Each combination of $\{i, j\}$ represents a knot interval; therefore there is one system of equations for each combination of $\{i, j\}$. A second order NURBS can be considered as a composition of linear polypanels for each $\{i, j\}$. Solving this system for all possible combinations of pairs $\{i, j\}$ results in several candidate parametric values $\xi_{i,j}$ and $\eta_{i,j}$.

Usually only one valid solution is obtained from the above equations, or two if the NURBS is a closed surface, except if the solution is close to an intersection of the linear polypanels, where more solutions could appear. For a solution to be valid it must fulfil the condition of belonging to the knot interval. Among all possible 'valid' solutions the one closest to the vertex P is more likely to be the correct solution, all of which amounts to the following conditions:

$$\{\xi^*_{i,j}, \eta^*_{i,j}\} = \min\{\| S(\xi_{i,j}, \eta_{i,j}) - P \|^2\} \tag{4.10}$$

with

$$\begin{aligned}U_i - \sigma^*_u \le \xi^*_{i,j} < U_{i+1} + \sigma^*_u \\ V_i - \sigma^*_v \le \eta^*_{i,j} < V_{i+1} + \sigma^*_v\end{aligned} \tag{4.11}$$

where σ^*_u and σ^*_v are relaxation factors, which virtually extend the coverage of the polypanels, e.g.,

$$\begin{aligned} \sigma_u^* &= \gamma(U_{i+1} - U_i) \\ \sigma_v^* &= \gamma(V_{i+1} - V_i) \end{aligned} \tag{4.12}$$

Possible values of γ could be 0.01, 0.1, 0.25, 0.5, and even higher. A value of zero is possible, but not recommended. High relaxation values will provide more valid solutions and, depending on the complexity of the inversion problem, such values could be actually necessary. As assumed above, the parametric values obtained from a second order NURBS are close to the real values, or at least close enough to be valid initial values to be used in iterative methods. Additionally, the above approach provides several alternative candidate values in case the iterative method fails to converge with one particular solution.

Second method, based on minimum distance

The previous method, based on the normals, is considered very robust, because the vertex normal provides additional information to obtain the inversion. However, situations arise in which the normal vectors can not be properly calculated, e.g. along edges or kinks, or they are simply not available. One can then use a second approach based on distance minimization. The equation to be solved is

$$\min[\| S(\xi,\eta) - P \|^2] = \min \sum [(S(\xi,\eta) - P)^2] \tag{4.13}$$

The left hand side of Equation 4.13 denotes the square of the Euclidean norm, while the right hand side denotes the arithmetic addition of the square of the vector components. As in the previous case, a valid initial solution can be obtained by specializing to a second order NURBS surface, for which Equation 4.13 takes the form:

$$\min \left[\sum \left(\frac{\bar{a}\xi\eta + \bar{b}\xi + \bar{c}\eta + \bar{e}}{\phi} - P \right)^2 \right] \tag{4.14}$$

where the notation is the same as in Equations 4.7 and 4.8. Equation 4.14 is solved by setting to zero the (ξ, η) derivatives of the distance function, which yields

$$\begin{aligned} F_1 &= \frac{\partial[\sum (S(\xi,\eta) - P)^2]}{\partial \xi} = 2\sum [(\bar{a}\eta + \bar{b})(\bar{a}\xi\eta + \bar{b}\xi + \bar{c}\eta + \bar{e} - P)] = 0 \\ F_2 &= \frac{\partial[\sum (S(\xi,\eta) - P)^2]}{\partial \eta} = 2\sum [(\bar{a}\xi + \bar{c})(\bar{a}\xi\eta + \bar{b}\xi + \bar{c}\eta + \bar{e} - P)] = 0 \end{aligned} \tag{4.15}$$

This leads to a third order equation. The roots can be calculated very efficiently using a Newton-Raphson method.

$$\begin{aligned} \{\xi,\eta\}_{n+1} &= \{\xi,\eta\}_n - [f']^{-1} \cdot f \\ \{\xi,\eta\}_0 &= \{u_i, v_j\} \end{aligned} \tag{4.16}$$

where

$$f = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{4.17}$$

and

$$f' = \begin{bmatrix} \frac{\partial F_1}{\partial \xi} & \frac{\partial F_1}{\partial \eta} \\ \frac{\partial F_2}{\partial \xi} & \frac{\partial F_2}{\partial \eta} \end{bmatrix} \tag{4.18}$$

Just as with the method of the normals explained before, the above equations are to be solved for each combination of pairs $\{i, j\}$, discarding those solutions that do not comply with the condition in Equation 4.10. It is also advisable to use the relaxation factor defined in Equation 4.11.

4.4 Fast computation of the gradients

4.4.1 Continuous adjoint methodology

The computation of the gradients or sensitivity derivatives of the cost function, such as drag or aerodynamic efficiency, is carried out using the continuous adjoint formulation introduced in section 1.2.4. The total derivative of a cost function I with respect to a design variable D can be written as:

$$\frac{dI}{dD} = \frac{\partial I}{\partial D} + \frac{\partial I}{\partial w}\frac{\partial w}{\partial D} \tag{4.19}$$

where w denotes the vector of flow variables. Upon deformation of the surface, the cost function varies due to a variation of the geometry and the flow solution. In aerodynamic optimization problems, suitable cost function examples include aerodynamic coefficients such as drag and lift, which are directly calculated from the pressure distribution over the aerodynamic surface S.

$$I = \int_S f(w) dS \tag{4.20}$$

It is assumed that the cost function f is differentiable. Even though this assumption may not be valid in the presence of shock waves or other discontinuities, numerical dissipation is generally enough to mitigate this effect and provide an approximate solution.

In the cases considered in this work, f does not depend on the time variable or the time derivatives of the flow variables w. Hence, Equation 4.19 can be written as:

$$\frac{dI}{dD} = \int_{\delta S} f(w) ds + \int_S \frac{\partial f}{\partial w}\frac{\partial w}{\partial D} ds \tag{4.21}$$

The Equation 4.21 includes a term related to the geometric variation of the surface and a second term related to the flow variation. Cost functions such as drag or lift are composed of terms that involve non-geometric and geometric quantities such as the surface normal n. Hence, the first term can be expanded as follows [and99, cas07]

$$\int_{\delta S} f(w)ds = \int_S \frac{\partial f}{\partial w}\left(\frac{\partial X}{\partial D}\cdot\nabla w\right)ds + \int_S \frac{\partial f}{\partial n}\frac{\partial n}{\partial D}ds + \int_S f\frac{\partial ds}{\partial D} \tag{4.22}$$

where $\frac{\partial X}{\partial D}$ is the geometric sensitivity of the surface points, $\frac{\partial n}{\partial D}$ is the sensitivity of the normal vector, and $\frac{\partial ds}{\partial D}$ is the sensitivity of the measure.

The term $\int_S f\frac{\partial ds}{\partial D}$ is usually referred to as the curvature term, and has the form:

$$\int_S f\frac{\partial ds}{\partial D} = \int_S fKds \tag{4.23}$$

with

$$K = \begin{cases} \partial_{tg}(\frac{\partial x}{\partial D})_{tg} - k(\frac{\partial x}{\partial D})_n & \text{if 2D} \\ \nabla_{tg}\cdot(\frac{\partial x}{\partial D})_{tg} - 2H_m(\frac{\partial x}{\partial D})_n & \text{if 3D} \end{cases}$$

in which k and H_m are the curvature of the profile and mean curvature of the surface, respectively.

To account for the flow sensitivities, one introduces the adjoint state Λ, subject to the adjoint equations

$$\left(\frac{\partial F}{\partial w}\right)^T\cdot\nabla\Lambda = 0 \tag{4.24}$$

where F is the inviscid flux vector, together with the appropriate adjoint boundary conditions on S so as to allow the computation of the flow sensitivities [jam98, and99, cas07]. Optimization is possible in this case with respect to any functional that depends on the distribution of the pressure on the aerodynamic surface. For drag and lift optimization problems the functional is defined by the following expression:

$$I = \int_S C_p(n\cdot d)ds \tag{4.25}$$

where

$$d = \begin{cases} (\cos\alpha\cos\beta, 0, \sin\alpha\cos\beta) & \text{for drag coefficient} \\ (-\sin\alpha, 0, \cos\alpha) & \text{for lift coefficient} \end{cases}$$

where C_p is the pressure coefficient, d is the force direction vector, α is the angle of attack and β is the sideslip angle. By defining

$$\Psi = \rho\lambda_1 + \rho v_1\lambda_2 + \rho v_2\lambda_3 + \rho v_3\lambda_4 + \rho H\lambda_5 \tag{4.26}$$

where ρ is the density, v_1, v_2, v_3 are the Cartesian components of the fluid velocity, H is the enthalpy, λ_i are the adjoint variables, and by separating the normal and tangent parts of the deformation, compact expressions result for the gradients of the drag and lift coefficients from Equation 4.21:

$$\delta I = \int_S (d\cdot\nabla C_p + (\nabla\cdot v)\Psi + (v\cdot t)\partial_{tg}\Psi)\delta x_n ds \tag{4.27}$$

where n and t are the normal and tangent vectors to the surface element respectively, and v is the velocity vector of the fluid. The equation above only considers deformations through the normal direction, which is accurate in most cases with the exception of kinks and geometries with strong curvatures, such as the trailing edge of a wing profile [loz11] .

In those situations where the tangential component is required, the extended formulation can be employed:

$$\delta I = \int_S (\vec{d}\cdot\vec{n})\delta x\cdot\nabla C_p ds + \int_S (\vec{d}\cdot\vec{n})C_p\delta ds + \int_S (\vec{d}\cdot\delta\vec{n})C_p ds + \int_S (((\delta x\cdot\nabla)v)\vec{n} + v\cdot\delta\vec{n})\Psi ds \quad (4.28)$$

The TAU code provides the flow and adjoint solutions and the surface sensitivity. The computation of the gradients as per equation 4.27 requires knowledge of the grid sensitivities $\frac{\partial X}{\partial D}$ for the surface grid points.

The geometric derivatives corresponding to a displacement of the position of the control point through a Cartesian direction e_k are exactly the basis coefficients:

$$\frac{\partial x}{\partial C_{ij}^k} = \frac{U_{i,p}V_{j,q}\vec{e}_k}{\phi} \quad (4.29)$$

while the geometric derivative related to the weight of the control point is calculated with the following expression:

$$\frac{\partial x}{\partial w_{ij}} = \frac{U_{i,p}V_{j,q}(C_{ij}-x)}{\phi} \quad (4.30)$$

where x and C_{ij} are the spatial coordinates of the surface vertex and control point respectively.

4.4.2 Validation using finite differences

For validation purposes, the gradients have also been calculated using finite differences. The finite-difference gradients should be similar to those computed with the adjoint methodology. However, insufficient grid resolution and inaccurate normal calculation may degrade the solution leading to a failure in the optimization process. Figure 4.4 shows the gradients computed for the lift and drag coefficients for the inviscid flow around a NACA0012 airfoil with $M = 0.8$ and $\alpha = 1.25^o$. The geometry is parameterized by a NURBS curve with 18 control points depicted in Figure 4.7. As can be observed in Figure 4.4, the adjoint gradients are fairly accurate, although there are important discrepancies in the gradients of the design variable which corresponds to a vertical displacement of the control point located at the trailing edge of the airfoil. Design variables 1-18, 19-36, 37-54 and 55-72 correspond, respectively, to changes in x, y, z coordinates and weight w of each of the 18 control points depicted in Figure 4.7.

Figure 4.5 (left) shows the lift gradients for design variables 37-54 (which correspond to vertical displacements of the control points). Figure 4.5 (right) indicates that the derivatives are fairly accurate except for design variable 46, which corresponds to a vertical translation of the control point at the trailing edge.

These discrepancies indicate that the assumption underlying Equation 4.1 is failing, that is to say, that the contribution of the tangent part of the geometric sensitivities to the cost function

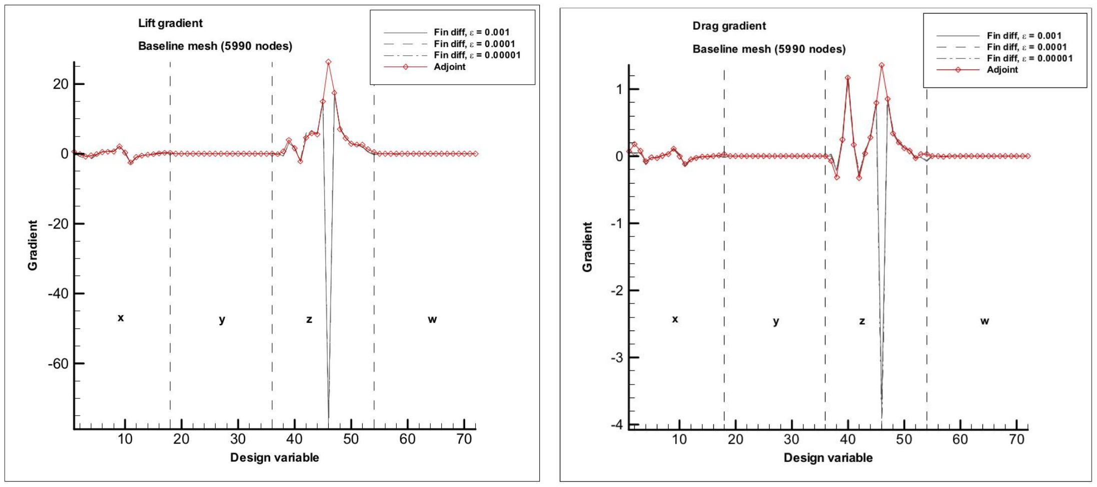


Figure 4.4: Adjoint vs. finite-difference gradients for lift (left) and drag (right) coefficients for a NACA0012 at $M = 0.8$ and $\alpha = 1.25^o$.

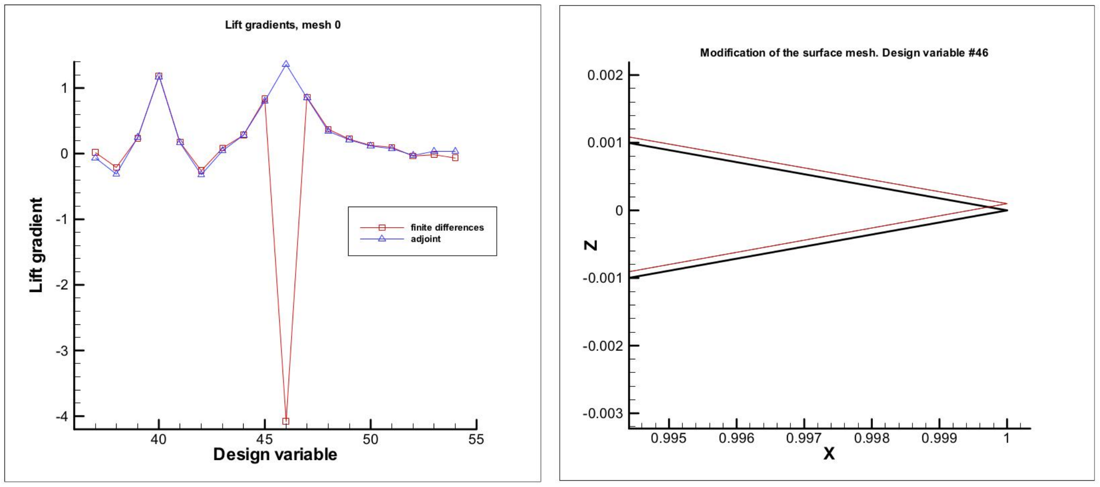


Figure 4.5: Close-up view of lift gradients (left) and surface deformation for design variable 46 (right) in a NACA0012 at $M = 0.8$ and $\alpha = 1.25^o$.

derivative does not vanish at the numerical level [loz09, loz12]. This issue can be tackled by incorporating those tangent contributions, but it will not be of concern here as the control point located at the trailing edge will be fixed throughout the optimization process. If that were not the case (if, say, one would wish to optimize the relative position of the different elements of a multi-element airfoil), it would be necessary to take this problem into consideration.

For that reason, formulations 4.27 and 4.28 have been compared with finite differences for a NACA0012 airfoil at $M = 0.8$ and $\alpha = 1.25^o$ as shown in Figure 4.6. The gradients obtained were almost exactly the same, with the exception of the trailing edge, where major discrepancies are encountered, and where the extended formulations is necessary to provide an acceptable accuracy.

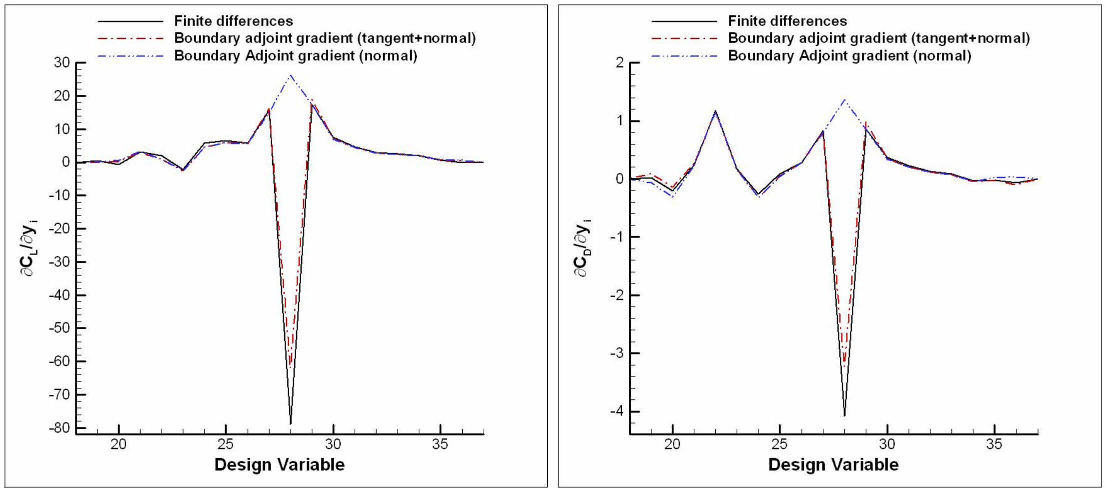


Figure 4.6: Comparison of sensitivities between finite differences and two boundary formulations for a NACA0012 at $M = 0.8$ and $\alpha = 1.25^o$.

4.5 Application to fine grain aerodynamic shape optimization

In this section, the methodology introduced in the present work is applied to the inviscid aerodynamic optimization of 2D NACA0012 and RAE2822 airfoils and a 3D ONERA M6 wing configurations.

4.5.1 Optimization of a 2D NACA0012 profile

The first design case is drag minimization for a single airfoil. The initial geometry is an NACA0012 airfoil [mar09] at Mach number $M = 0.8$ and angle of attack $\alpha = 1.25^o$. The geometry is described by a NURBS curve with 18 control points, which is depicted in Figure 4.7.

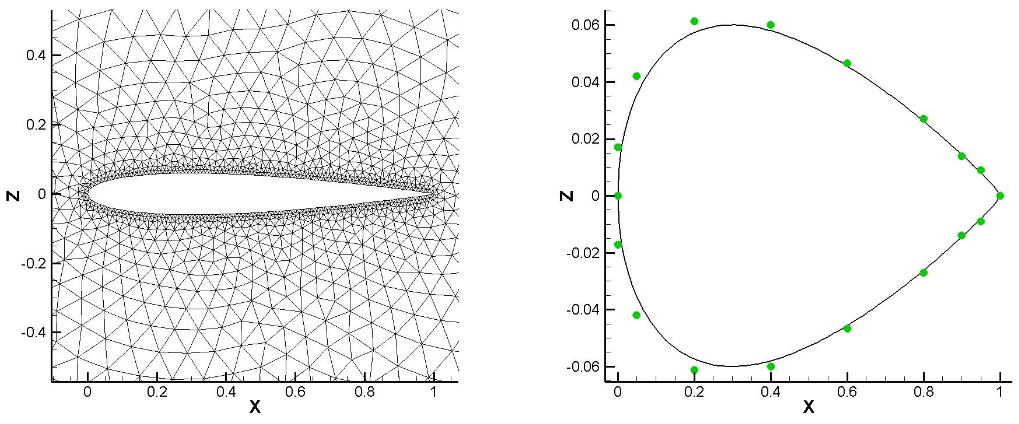


Figure 4.7: Computational grid around NACA0012 airfoil (left) and corresponding NURBS parameterization (right) described with 18 control points.

The design optimization problem selected is the minimization of the drag coefficient holding the lift coefficient constant with a 10% tolerance. Hence, the cost function is defined as

$$\min\{C_d + 10(C_L - C_L^T)^4\} \tag{4.31}$$

where C_d and C_L are the drag and lift coefficients, respectively, and C_L^T is the target lift. The optimization algorithm used is a simple descent method in which small steps are taken in the negative gradient direction [ger97], as it can be observed in Equation 4.32. During the whole optimization process the maximum deformation is held constant at approximately 1% of the camber.

$$\delta = -\{\dot{c_d} + 4 \cdot 10 \cdot \dot{c_L}(C_L - C_L^T)^3\} \tag{4.32}$$

where $\dot{c_d}$ and $\dot{c_L}$ are the normalized values of drag and lift coefficients.

During the process, the following constraints are applied:

- The control points situated on the leading and trailing edge are held fixed in order to maintain the chord length and angle of attack.
- All control point weights are kept fixed.

Figure 4.8 shows the evolution of the force coefficients and aerodynamic efficiency throughout the optimization process. All the coefficients are normalized to their initial values.

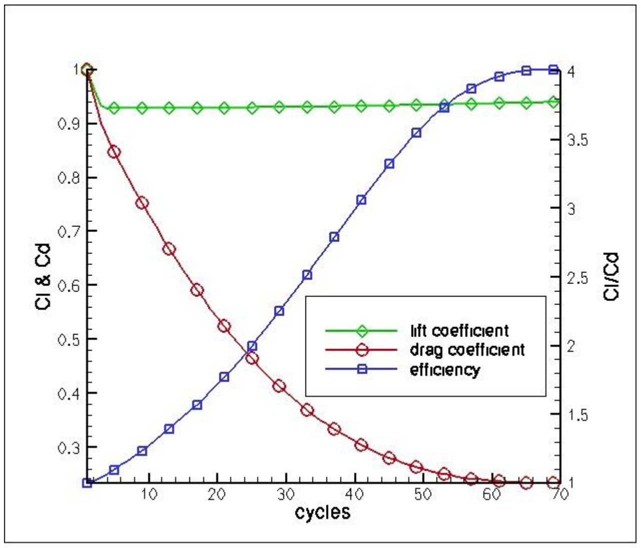


Figure 4.8: Evolution of the aerodynamic coefficients within the optimization process for a NACA0012 airfoil at M=0.8, α=1.25°

After 70 optimization cycles the drag improvement was of 70%, which was obtained with a lift decrement of 5%, well within the specified range. It can be observed in Figures 4.9 and 4.10 that the optimization process has largely reduced the strong shock on the suction side of the airfoil.

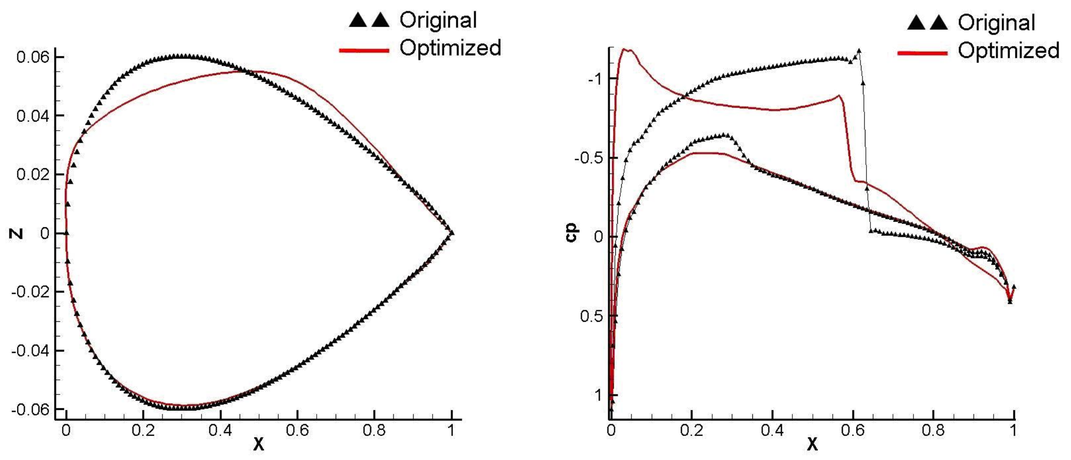


Figure 4.9: Change of airfoil shape (left) and pressure distribution Cp (right) for the initial and optimized NACA0012 airfoil at $M = 0.8$ and $\alpha = 1.25^{o}$

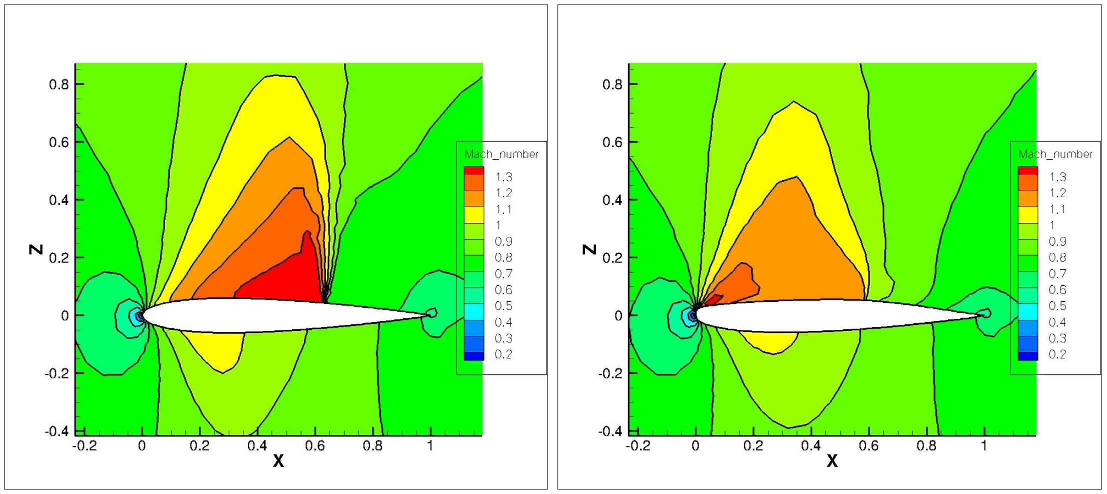


Figure 4.10: Distribution of the Mach number for the initial (left) and optimized (right) NACA0012 airfoil at $M = 0.8$ and $\alpha = 1.25^{o}$

The weak shock that appears near the leading edge, after the optimization, seems to be a consequence of the formulation used to compute the sensitivities. This fact will be further studied, and the tangential component of the derivatives will be included in the formulation to improve the solution.

4.5.2 Optimization of a 2D RAE2822 profile

The methodology is also applied to a RAE2822 airfoil [coo79] at $M = 0.729$ and $\alpha = 2.31^{o}$. The geometry is described by a NURBS curve with 14 control points depicted in Figure 4.11. The computational grid has 10.600 points and 10.500 elements, and the NURBS is composed of 14 control points. For this case, it has also been checked the extent to which the difference between the initial geometry and the one obtained from the NURBS parameterization which is used for optimization, affects the aerodynamic coefficients and the differences found were not significant. However, as was previously mentioned, if necessary, the error could be even further reduced by increasing the number of control points, which would make the geometry to be closer to the original one, along with its aerodynamic properties.

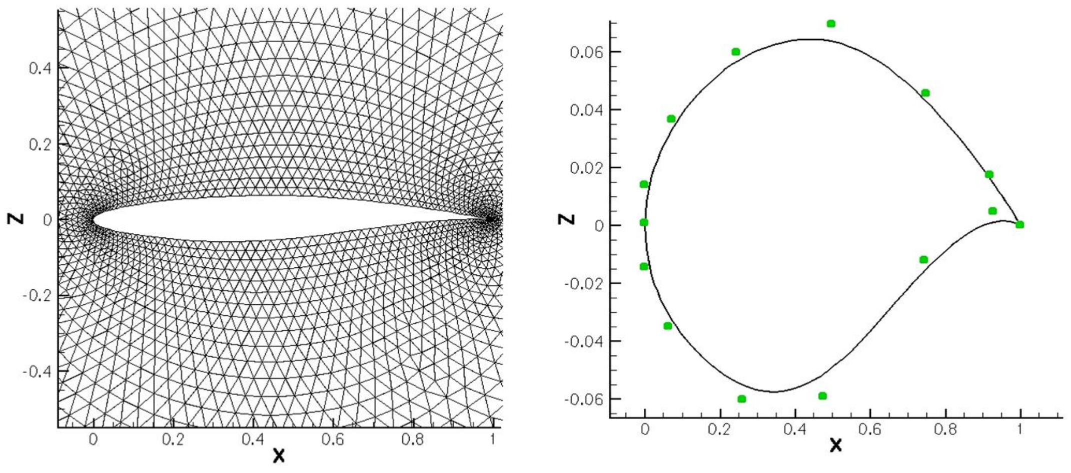


Figure 4.11: Computational grid around RAE2822 airfoil (left) and NURBS parameterization with 14 control points (right)

The objective function selected is the same as in the previous case: minimization of the drag coefficient maintaining the lift coefficient within a 10% range of its initial value.

During the optimization, the following constraints are applied:

- The control point situated at the trailing edge is fixed in all directions.
- All control points are fixed in X and Y directions, only Z (vertical) displacements are allowed, therefore the chord is kept constant.
- Control points weights are held fixed.

Figure 4.12 shows the evolution of the force coefficients and the aerodynamic efficiency (lift /drag ratio) throughout the optimization process. All the coefficients are normalized to their initial values.

After 35 optimization cycles the drag improvement was of 60%, which was obtained with a lift decrement of 1.5%, well within the specified range.

As can be observed in Figures 4.13 and 4.14, the shock has almost disappeared, as it was expected for an inviscid case, after the optimization process; however a new weak shock has appeared near the leading edge.

4.5.3 Optimization of a 3D ONERA M6 wing

The methodology is also applied to optimize a three dimensional ONERA M6 wing at inviscid flow conditions of $M = 0.8395$ and $\alpha = 3.06^{o}$, described by a NURBS surface (for both the upper

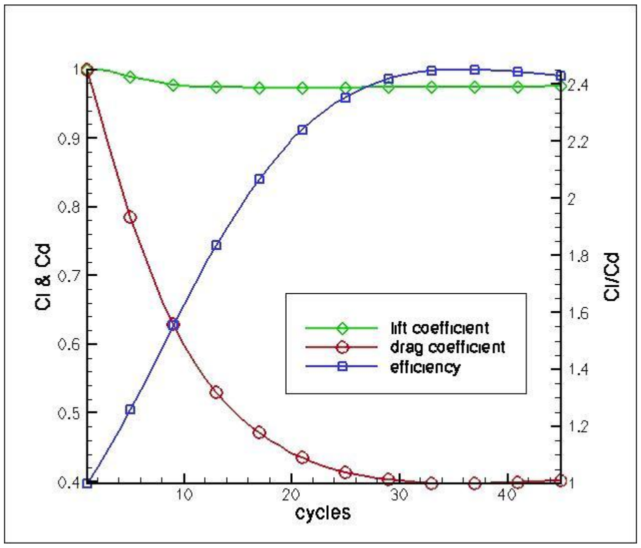


Figure 4.12: Evolution of the aerodynamic coefficients within the optimization process for the RAE 2822 airfoil at $M = 0.729$ and $\alpha = 2.31^o$.

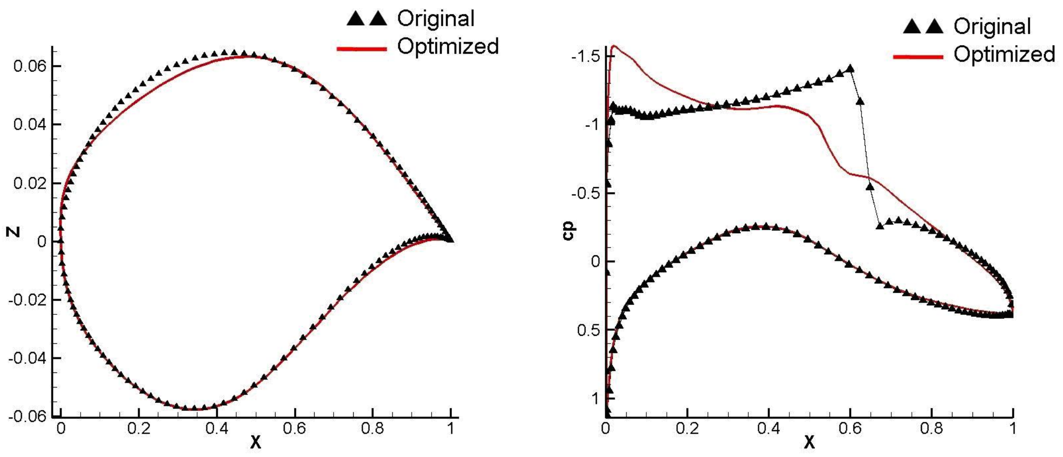


Figure 4.13: Change of airfoil shape (left) and pressure distribution Cp (right) for the initial and optimized RAE2822 airfoil at $M = 0.729$ and $\alpha = 2.31^o$.

and lower wing surfaces) with 25×18 control points depicted in Figure 4.15. The computational grid has 40.000 points and 205.000 elements.

For the optimization process, the control points located at the trailing edge of the wing are held fixed in all directions.

In this case, after 35 optimization cycles the drag coefficient has been reduced by 15%, while the lift coefficient is only reduced by 1%. Figure 4.18 shows a comparison between the original and optimized section shapes and corresponding C_p distributions at various locations along the wing, while Figure 4.17 shows the comparison of the surface Mach number distribution between the original and the redesigned configurations. From these figures it can be seen that the lambda

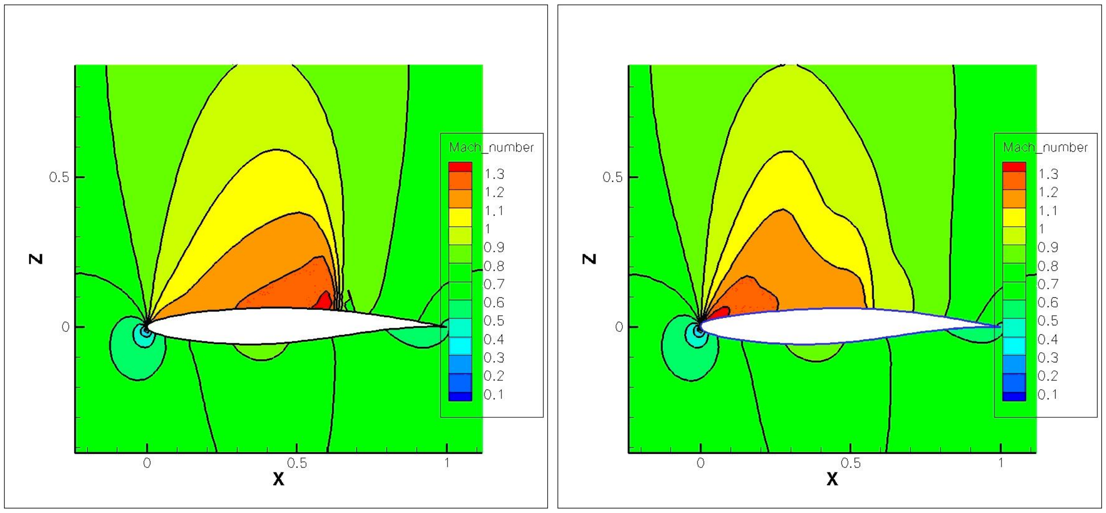


Figure 4.14: Distribution of the Mach number for the initial (left) and optimized (right) RAE2822 airfoil at $M = 0.729$ and $\alpha = 2.31^o$.

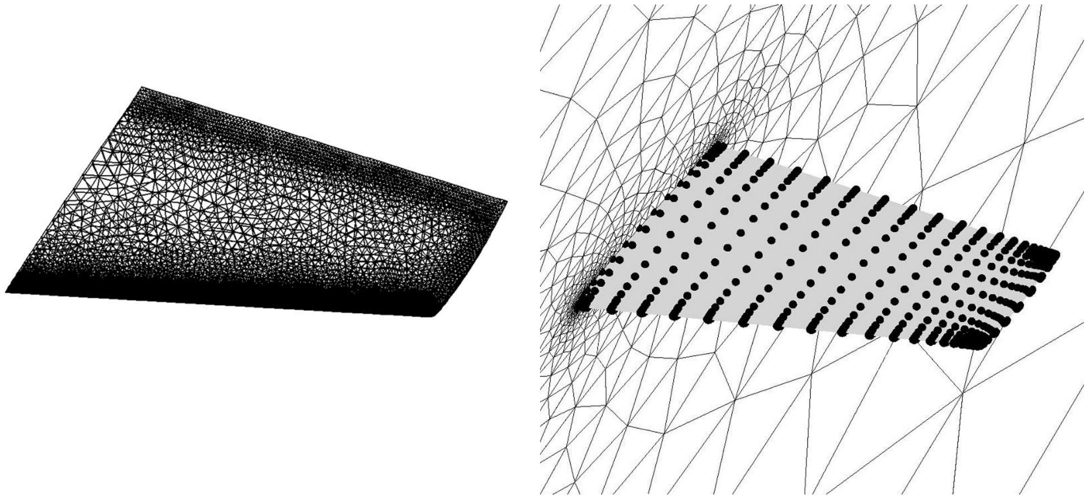


Figure 4.15: Surface grid around Onera M6 wing (left) and control points (right)

shock across the upper surface has been considerably weakened after the optimization process.

4.5.4 Considerations regarding execution time

In order to measure the efficiency of the proposed methodology in terms of computational time, a profiling has been performed for one optimization cycle of the ONERA M6 wing configuration. The results are displayed in Table 4.1

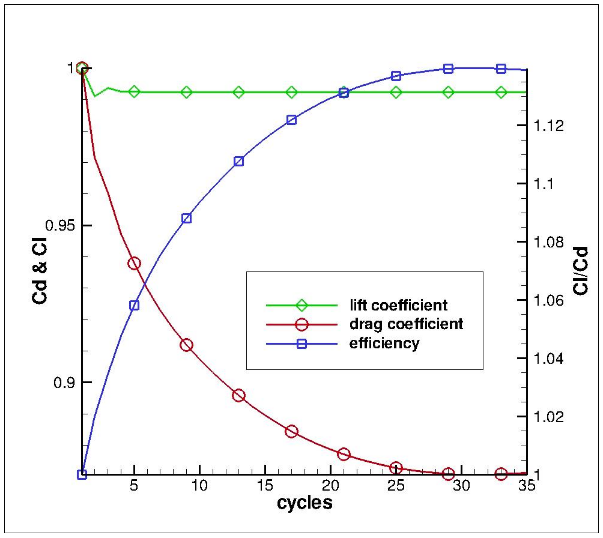


Figure 4.16: Evolution of the aerodynamic coefficients for the ONERA M6 optimization.

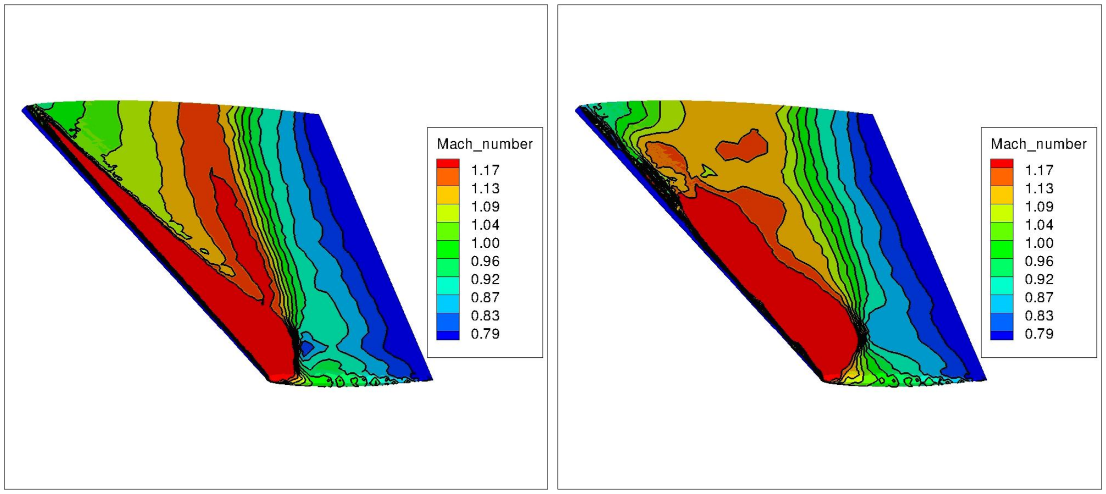


Figure 4.17: Mach number distribution for the initial (left) and optimized (right) ONERA M6.

Table 4.1: ONERA M6 drag minimization: execution-time breakdown for one optimization cycle.

Step	Time (s.)	%
Point inversion (only once for the whole process)	142	-
Preprocessing	1	0.18 %
Flow Solver	314	58.97 %
Adjoint Solver for drag coefficient	110	20.66 %
Adjoint Solver for lift coefficient	106	19.90 %
Gradient computation	0.5	0.1 %
Surface deformation	0.01	<0.01 %
Volume deformation	1	0.18 %
TOTAL per cycle	532.51	-

As can be concluded from the above table, the most time-consuming steps are the flow and adjoint solver executions, which together amount to 99% of the computational time per cycle. The computational requirements of the rest of the process are almost negligible in comparison to it.

In the above computations, the flow solver residual has been reduced by 11 orders of magnitude, while for the adjoint solver it has been reduced by 3 orders of magnitude, which has been shown to be enough for obtaining good quality gradients [loz09]. The total execution time for 50 optimization cycles in the 2D NACA0012 profile was 3000 seconds (50 min.), while the optimization of the 3D ONERA M6 wing took 26000 seconds (7h), using a single processor on a Linux x86 computer.

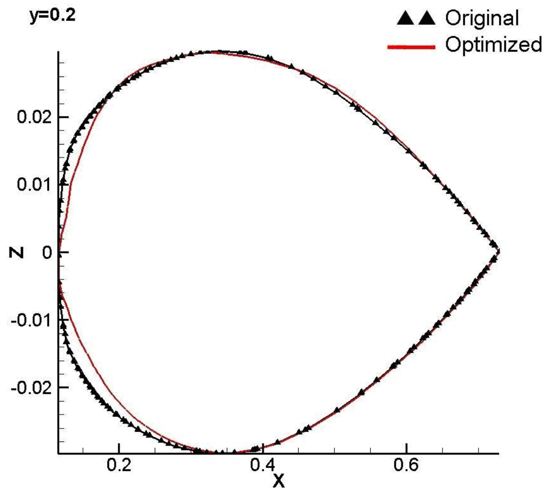
4.6 Conclusions

This chapter has presented an efficient approach for fine grain CAD-based shape aerodynamic design under inviscid flow conditions. It is important to stress here that the aim of the work is to develop a shape optimization method which can be incorporated into the design chain of the aerospace industry. With that premise in mind, the proposed optimization methodology offers two key capabilities:

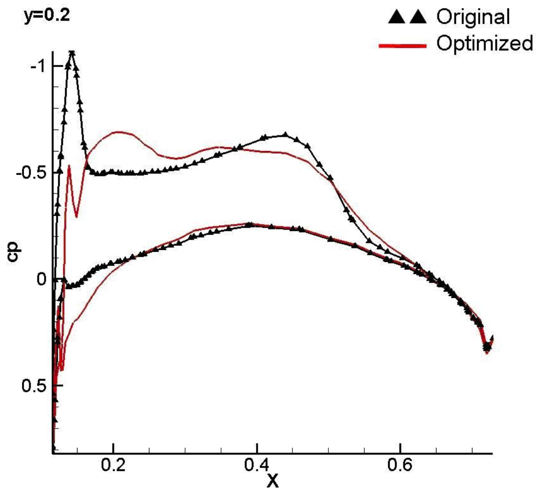
- Fast computation of the gradients through the use of the adjoint methodology based on boundary integrals. This allows independency from the number of design variables, and, therefore, the possibility to use fine grain geometry parameterization, involving a high number of design variables, without a prohibitive computational cost.

- The use of NURBS parameterization, that allows a local shape optimization for finer improvement of the aerodynamics efficiency. It also allows the representation of complex geometries, and enables the fast incorporation of the optimization procedure into existing design chains within the aerospace industry.

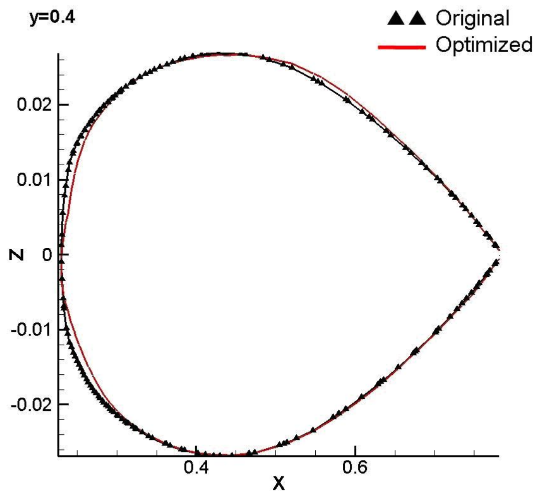
This approach has been implemented and is fully integrated within the TAU code [dlr94], being able of performing optimizations by considerably reducing the drag for airfoils and wings in transonic flows.



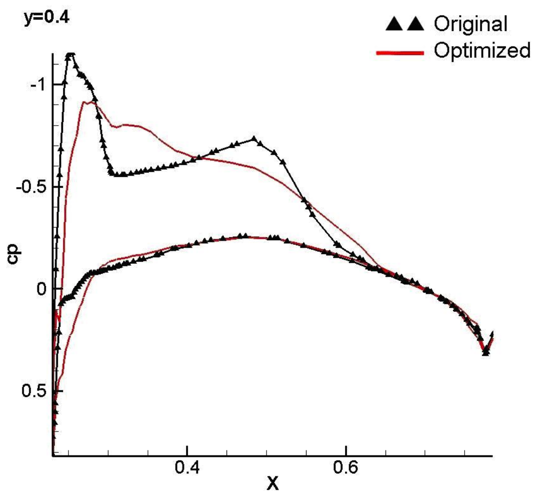
(a) Residual



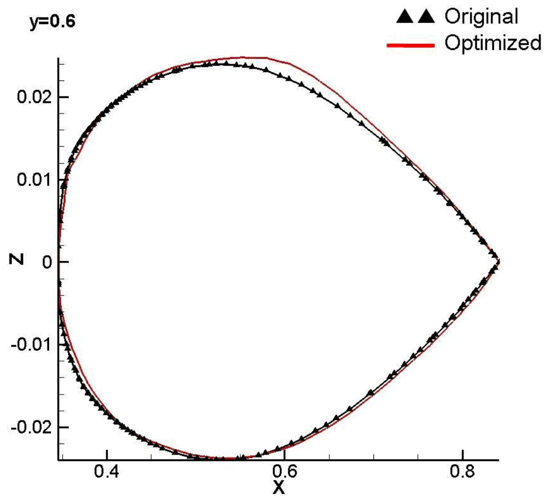
(b) Residual



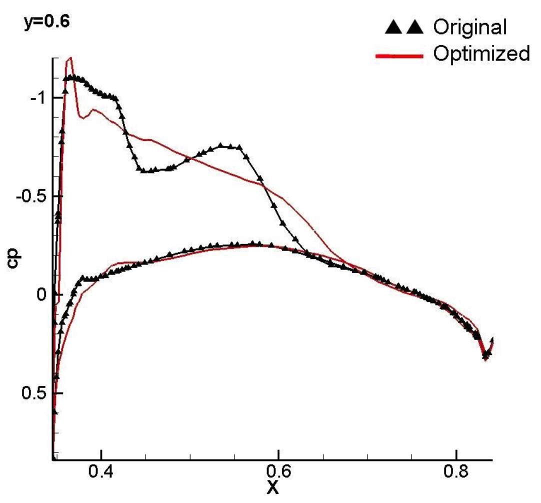
(c) C_L



(d) C_L



(e) C_D



(f) C_D

Figure 4.18: Comparison of section shapes (left) and pressure distributions Cp (right).

Part III

Final remarks and future research activities

Final remarks

This thesis deals with the improvement of the optimization process in the aerodynamic design of aeronautical configurations, proposing several contributions at three different levels: first, code optimization and efficient parallelization of the most time-consuming step in the design process, the CFD analysis tool; second, coarse grain surrogate-based global optimization, through the use of a novel EP-SVMr approach, and, finally, fine grain local optimization, involving the application of a framework with key ingredients as adjoint methods, for fast gradient computation, and NURBS for a flexible parameterization of complex geometries.

From the results of the research activity developed within this work, several conclusions can be extracted and those of main significance are summarized in this chapter.

- An hybrid EP+SVMr approach for coarse grain surrogate-based global shape optimization has been proposed and tested for the aerodynamic design of airfoils, showing promising results. This approach strongly benefits from the key features of global optimization methods, allowing a broad exploration of the design space, i.e. including also unconventional airfoil shapes within the optimization process. In addition, the efficiency of the process has been improved through the use of SVMr as metamodels to substitute the expensive CFD code, and the capability of these models to make accurate predictions of the aerodynamic coefficients has been demostrated.

- A SVMr network has been built and trained as metamodel for lift (C_L) curves and polar prediction of subsonic and transonic airfoils. The inputs of the network were the geometry, defined by a Sobieczky parameterization composed of 12 parameters, the flow conditions and the aerodynamic coefficients of the training samples, evaluated with a CFD tool.

- The sensitivity of the training data size in the case of the proposed SVMr metamodel has been assessed and results have shown that the RMSE measurements are of around few lift and drag counts for coefficients prediction, which is acceptable for an initial extensive exploration of the design space.

- The SVMr network has been coupled to an Evolutionary Programming algorithm, and the complete approach has been applied to the inverse design of subsonic airfoils. The obtained results demonstrated the feasibility of this methodology to obtain good approximations close to the global optimal shapes, with a very reduced computational time of 32 seconds for the whole optimization process.

- Regarding fine grain CAD-based aerodynamic shape optimization, an efficient approach has been applied to the design of airfoils and wings in transonic flow conditions, with the aim of drag reduction. The approach couples a fast computation of the gradients, through the adjoint methodology based on boundary integrals, with the use of NURBS geometry parameterization.

- The framework has been validated in the optimization of two inviscid configurations, a 2D NAC0012 airfoil and a 3D ONERA M6 wing. The approach was able to reduce the drag considerably, 60% for the NACA0012 and 15% for the ONERA M6, mainly by eliminating the shock wave on the upper part of the configurations.

- Regarding code optimization, an improvement of the code efficiency has been addressed through optimization by applying different tuning strategies to reduce the execution time of the unstructured DLR TAU solver. Significant computational gains (around 11% for RK and 4% for LUSGS for a 64 bits machine) have been achieved and it makes clear the necessity of code profiling and optimization involving multidisciplinary knowledge to reduce the execution time and memory consumption.

- With respect to parallel scalability, different algorithms for domain decomposition to make a more efficient grid partitioning have been implemented and tested in TAU for high parallel simulations, using up to 1024 processors, showing that the graph and hypergraph partitioner algorithms maintain the speedup much closer to the linear one for a high number of processors in all the tests performed. Moreover, the current state-of-the-art of the development of mixed hardware-software computational platforms, using GPUs or FPGAs, for simulation has been presented focusing in precision and area estimations which shows a promising technology when the data representation format required is limited.

The results achieved in this research work have been presented at several international events and accepted for scientific publications. In particular, during the last months, 3 papers have been accepted for publication in relevant international journals, and during the realization of this thesis, 8 papers have been presented in international congresses. A complete list of the papers related to the research work performed in this thesis can be seen in IV.

Future research lines

Despite the different results obtained from this research work, there are still several directions in which subsequent studies could progress. Some of the detected areas to be addressed more deeply in near future include:

- The first natural extension of the research regarding global optimization is the validation of the proposed approach for the shape optimization of 3D configurations, where the computation of the aerodynamic data for the training samples will be more time consuming, and efficient techniques for on-line training should be considered to build the metamodel.
- Another extension of this work is the application of different levels of parameterization, from few to many design variables, to study the behavior of each optimization step, coarse and fine grain.
- This thesis focuses on the optimization of aeronautical shapes, considering only aerodynamics, but not other disciplines such as structures or acoustics. However, nowadays, multidisciplinary design optimization (MDO) is a very active field of research. The proposed global shape optimization approach could be extended to deal with other disciplines. In particular the SVMr network, used as metamodel for substituting CFD, would have to be trained to replace Computational Structure Mechanics (CSM) and Computational Aeroacoustics (CAA) codes, but the strategy and the Evolutionary Programming technique to be applied would remain the same.
- With respect to geometry parameterization, the management of intersections between different NURBS patches, needed for representation of complex geometries, still requires further research work. In addition, the sensitivity of the NURBS parameterization within the optimization process should be more deeply studied. The capability to automatically obtain optimal NURBS (with a proper number and distribution of control points) from the original CAD definition (for instance an IGES or STEP file) is still missing and it is of high interest for the industry. Moreover, a tool for extracting the NURBS definition from a given computational mesh would allow to recover the geometry at any time of the optimization process.
- Regarding gradient-based optimization using adjoints, there are still some efforts required for industrialization of the whole process. In particular, the continuous adjoint solver lacks robustness for 3D complex viscous configurations.

Part IV

Appendix

List of publications

This section is a compilation of the various scientific publications produced as a result of this research work, apart from those other studies conducted during the training process.

Papers related to the research work performed in this PhD.

Papers in international journals

1. E. Andrés, S. Salcedo-Sanz, F. Monge and Á. M. Pérez-Bellido, "Efficient Aerodynamic Design through Evolutionary Programming and Support Vector Regression Algorithms," *Journal of Expert Systems with Applications*, In press 2012, (JCR 2010: 1.924), DOI 10.1016/j.eswa.2012.02.197.

2. C. Lozano, E. Andrés, M. Martín and P. Bitrián, "Domain versus boundary computation of flow sensitivities with the continuous adjoint method for aerodynamic shape optimization problems," *International Journal for Numerical Methods in Fluids*, 2011 (JCR 2010: 1.060), DOI 10.1002/fld.2743.

3. M. Martín, E. Andrés, C. Lozano, and M. Widhalm, "Non-uniform rational B-splines-based aerodynamic shape design optimization with the DLR TAU code," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 0954410011421704*, Published on 7th December 2011, (JCR 2010: 0.420).

4. E. Andrés, M. Widhalm and A. Caloto, "Optimization, Parallelization, and FPGA Acceleration for the Unstructured DLR TAU Code," *Submitted to AIAA Journal*, (JCR 2010: 1.174)

Papers in international congresses

1. E. Andrés, S. Salcedo-Sanz, M. Widhalm, "Aerodynamic coefficients prediction for transonic configurations using Support Vector Machines within a global shape optimization process," *Abstract accepted for the ECCOMAS Congress 2012*, Wien (Austria), September 2012.

2. E. Andrés, S. Salcedo-Sanz, F. Monge and Á. M. Pérez-Bellido, "Metamodel-assisted aerodynamic design using evolutionary optimization," *International Conference on Evolutionary and Deterministic Methods for Design, Optimisation and Control (EUROGEN)*, Capua (Italy), 2011

3. E. Andrés, P. Bitrián, C. Lozano, M. Martin and M. Widhalm, "Preliminary comparisons between two CAD-based aerodynamic shape optimization approaches using adjoint methods for fast gradient computation," *International Congress of Engineering Optimization (ENGOPT)*, Lisbon (Portugal), 2010

4. M. Martin, E. Andrés, M. Widhalm, P. Bitrián and C. Lozano, "CAD based shape optimisation for inviscid configurations with the DLR TAU code," *International Congress of the Aeronautical Sciences (ICAS)*, Nice (France), 2010.

5. E. Andrés, M. Widhalm and A. Caloto, "Achieving High Speed CFD Simulations: Optimization, Parallelization, and FPGA Acceleration for the Unstructured DLR TAU Code," *47th AIAA Aerospace Sciences Congress*, Orlando (USA), 2009.

6. E. Andrés, C. Carreras, G. Caffarena, M. Molina, O. Nieto and F. Palacios, "A Methodology for CFD Acceleration Through Reconfigurable Hardware," *46th AIAA Aerospace Sciences Congress*, Reno (USA), 2008.

7. M. Martin, E. Andrés and C. Carreras, "Solving the Euler Equations for Unstructured Grids on Graphical Processor Units using CUDA," *49th International Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA)*, Norway, 2008.

Other results achieved during the training process

Papers in international congresses

1. J. Ponsín, A. Caloto, E. Andrés, P. Bitrián and C. Lozano, "Implementation of an error estimation and grid adaptation module into the DLR TAU code," *International Congress of the Aeronautical Sciences (ICAS)*, 2010.

2. G. Botella, E. Ros, M. Rodríguez, A. García, E. Andrés, M. Molina, E. Castillo and L. Parrilla, "FPGA based Architecture for Robust Optical Flow Computation," *IV Southern Programmable Logic Conference (SPL)*, Bariloche (Argentina), 2008.

3. E.Andrés, M. Molina, G. Botella, A. del Barrio and J.M. Mendias, "Area optimization of combined integer and floating point circuits," *IV Southern Programmable Logic Conference (SPL)*, Bariloche (Argentina), 2008.

4. A. Barrio, M. Molina, J. M. Mendias, E. Andres, G. Botella, R. Hermida, and F. Tirado, "Applying branch prediction techniques to implement adders," *XXIII Conference on Design of Circuits and Integrated Systems (DCIS)*, Grenoble (France), 2008.

5. A. Barrio, M. Molina, J. M. Mendias, E. Andres, and R. Hermida, "Restricted chaining and fragmentation techniques in power aware high level synthesis," *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD)*, 2008.

6. A. Barrio, M. Molina, J. M. Mendias, E. Andres, R. Hermida and F. Tirado, "Applying speculation techniques to implement functional units," *26th International Conference on*

Computer Design (ICCD), Proceedings, pp.74-80, Lake Tahoe (USA), 2008. IEEE International Conference on Computer design.

7. P. Valle, D. Atienza, I. Magan, J. Flores, E. Andrés, J. M. Mendias, L. Benini, G. De Micheli, "A Complete Multi-Processor System-on-Chip FPGA -Based Emulation Framework," *Proceedings of 14th Annual IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Nice (France), 2006.

8. P. Valle, D. Atienza, I. Magan, J. Flores, E. Andrés, J. M. Mendias, L. Benini, G. De Micheli, "Architectural Exploration of MPSoC Designs Based on an FPGA Emulation Framework," *Proceedings of XXI Conference on Design of Circuits and Integrated Systems (DCIS), p. 12-18*, Barcelona (Spain), 2006.

Part V

Bibliography

[aca20] ACARE. «European Aeronautics: A vision for 2020.» *Tech. rep.*, Advisory Council for Aeronautics Research in Europe, 2001. URL `http://www.acare4europe.com/docs/Vision2020.pdf`.

[aca50] —. «Aeronautics and Air Transport: Beyond vision 2020 (towards 2050).» *Tech. rep.*, Advisory Council for Aeronautics Research in Europe, 2010. URL `http://www.acare4europe.org/docs/Towards2050.pdf`.

[aer08] Aerofast, Consortium. «Aerofast EU Proposal: Collaborative framework for an enhancement in CFD design capabilities.» In *proposal to the VII European Framework Programme*. 2008.

[ahm10] Ahmed, M., and N. Qin. «Metamodels for Aerothermodynamic Design Optimisation.» In *48th AIAA Aerospace Sciences Meeting. AIAA 2010-1318*. 2010.

[aka98] Akay, M. F. «Support vector machines combined with feature selection for breast cancer diagnosis.» *Expert Systems with Applications*, 36, (2009), 3240–3247.

[alr05] Alrutz, T. «Investigation of the parallel performance of the unstructured DLR-TAU-Code on distributed computing systems.» In *Proceedings of Parallel CFD 2005*. Elsevier, Washington, DC (USA), 2005, 509 – 516.

[and99] Anderson, W. K., and V. Venkatakrishnan. «Aerodynamic Design Optimization on Unstructured Grids with Continuous Adjoint Formulation.» *Computers and Fluids*, 28(4-5), (1999), 443–480.

[and08] Andres, E., C. Carreras, M. Molina, G. Caffarena, O. Nieto, and F. Palacios. «A Methodology for CFD Acceleration Through Reconfigurable Hardware.» In *46th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA, Reno, Nevada, 2008.

[and10] Andrés, E., P. Bitrián, C. Lozano, M. Martin, and M. Widhalm. «Preliminary comparisons between two CAD-based aerodynamic shape optimization approaches using adjoint methods for fast gradient computation.» In *International Congress of Engineering Optimization (ENGOPT)*. 2010.

[and11] Andrés, E., F. Monge, A. Pérez, and S. Salcedo. «Metamodel-Assisted Aerodynamic Design Using Evolutionary Optimisation.» In *Evolutionary And Deterministic Methods For Design, Optimisation And Control (EUROGEN)*. 2011.

[and09] Andrés, E., M. Widhalm, and A. Caloto. «Achieving High Speed CFD Simulations: Optimization, Parallelization, and FPGA Acceleration for the Unstructured DLR TAU Code.» In *47th AIAA Aerospace Sciences Congress*. 2009.

[aso09] Asouti, V., I. Kampolis, and K. Giannakoglou. «A Grid-Enabled Asynchronous Metamodel-Assisted Evolutionary Algorithm for Aerodynamic Optimisation.» *Genetic Programming and Evolvable Machines, Parallel and Distributed Evolutionary Algorithms, Part One*, 10(4), (2009), 373–389.

[aso11] Asouti, V.G, E.A. Kontoleontos, X.S. Trompoukis, and K.C. Giannakoglou. «Shape Optimization using the One-Shot Adjoint Technique on Graphics Processing Units.» In *7th GRACM International Congress on Computational Mechanics*. 2011.

[aso12] Asouti, V.G, X.S. Trompoukis, I.C. Kampolis, and K.C. Giannakoglou. «Unsteady CFD Computations Using Vertex-Centered Finite Volumes for Unstructured Grids on Graphics Processing Units.» *International Journal for Numerical Methods in Fluids*, 67(2), (2012), 232–246.

[bac93] Back, T., and H. P. Schwefel. «An overview of evolutionary algorithms for parameter optimization.» *Evolutionary Computation*, 1, (1993), 1–23.

[bel08] Belleman, R., J. Bedorf, and S. Zwart. «High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA.» *International Journal in Astronomy and Astrophysics. Elsevier*, 13(2), (2008), 103–112.

[ben05] Bentamy, A., F. Guibault, and J. Trepanier. «Aerodynamic Optimization of a Realistic Aircraft Wing.» In *AIAA paper 2005-332*. 2005.

[ber87] Berger, J., and H. Bokhari. «A partitioning strategy for nonuniform problems on multiprocessors.» In *IEEE Transactions on Computers*. 1987, volume C-36(5), 570–580.

[bol03] Bolz, I., E. G. Farmer, and P. Schroeder. «Sparse matrix solvers on the GPU: conjugate gradients and multigrid.» In *A Quarterly Journal in Modern Foreign Literatures, Vol. 22, Issue 3*. 2003, pp. 917-924.

[bom10] Bompard, M., J. Peter, and J. Desideri. «Surrogate models based on function and derivative values for aerodynamic global optimisation.» In *V European Conference on Computational Fluid Dynamics ECCOMAS CFD*. 2010.

[bon02] Bondalapati, K., and V.K. Prasanna. «Reconfigurable Computing Systems.» *Proceedings of the IEEE Comput. Digit. Tech.*, 90(7), (2002), 1201–1217.

[bra08] Brandvik, T., and G. Pullan. «Acceleration of a 3D Euler Solver Using Commodity Graphics Hardware.» In *46th AIAA Aerospace Sciences Meeting and Exhibit*. Reno, Nevada, 2008.

[bra05] Branke, J., and C. Schmidt. «Faster convergence by means of fitness estimation.» *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(1), (2005), 13–20.

[bre09] Brezillon, J., R. Dwight, and M. Widhalm. «Aerodynamic Optimization for Cruise and High-Lift Configurations.» *Numerical Fluid Mechanics and Multidisciplinary Design*, 107, (2009), 249–262.

[bro99] Brodersen, O., E. Monsen, A. Ronzheimer, R. Rudnik, and C. Rossow. «Computation of aerodynamic coefficients for the DLR-F6 configuration using

MEGAFLOW.» *New Results in Numerical and Experimental Fluid Mechanics II*, 72, (1999), 85 – 92. URL `http://elib.dlr.de/21090`.

[bro01] Brodersen, O., and A. Sturmer. «Drag Prediction of Engine - Airframe Interference Effects using Unstructured Navier-Stokes Calculations.» *AIAA Paper 2001-2414*.

[buc05] Buche, D., N. Schraudolph, and P. Koumoutsakos. «Accelerating evolutionary algorithms with Gaussian process fitness function models.» *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 35(2), (2005), 183–194.

[bul99] Bull, L. «On model-based evolutionary computation.» *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, 3(2), (1999), 76–82.

[car06] Carrier, G. «Single and multipoint aerodynamic optimisations of a supersonic transport aircraft wing using optimisation strategies involving adjoint method and genetic algorithm.» In *ERCOFTAC Conference on Optimisation*. 2006.

[cas07] Castro, C., C. Lozano, F. Palacios, and E. Zuazua. «Systematic Continuous Adjoint Approach to Viscous Aerodynamic Design on Unstructured Grids.» *AIAA Journal*, 45(9), (2007), 2125–2139.

[cat07] Catonguay, P., and S. Nadarajah. «Effect of Shape Parametrisation on Aerodynamic Shape Optimisation.» In *45th AIAA Aerospace Sciences Meeting an Exhibit. AIAA 2007-59*. 2007.

[che95] Chen, W. «A Robust Concept Exploration Method for Configuring Complex System.» In *Ph.D. Dissertation Thesis, Mechanical Engineering, Georgia Institute of Technology*. 1995.

[che04] Chen, W., P. Kosmas, M. Leeser, and C. Rappaport. «An FPGA implementation of the two-dimensional finite-difference time-domain (FDTD) algorithm.» *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, ACM Press*, (2004), 213–222.

[che11] Cheng, C. S., P. W. Chen, and K. K. Huang. «Estimating the shift size in the process mean with support vector regression and neural networks.» *Expert Systems with Applications*, 38, (2011), 10624–10630.

[cla05] Clarke, S. M., J. H. Griebsch, and T. W. Simpson. «Analysis of Support Vector Regression for Approximation of Complex Engineering Analyses.» *Transactions of ASME, Journal of Mechanical Design*, 127(6), (2005), 1077–1087.

[coh12] Cohen, K., and S. Siegel. «Nonlinear estimation of transient flow field low dimensional states using artificial neural nets.» *Expert Systems with Applications*, 39, (2012), 1264–1272.

[com11] Comission, European. «Work Programme for Transport including Aeronautics.» In *VII European Framework Programme*. 2011.

[com02] Compton, K., and S. Hauck. «Reconfigurable Computing: a Survey of Systems and Software.» *ACM Computing Surveys*, 34(2), (2002), 171–210.

[coo79] Cook, P., M. McDonald, and M. Firmin. «Aerofoil RAE 2822 - Pressure Distributions, and Boundary Layer and Wake Measurements. Experimental Data Base for Computer Program Assessment.» *Tech. rep.*, AGARD Report, 1979. URL `http://www.grc.nasa.gov/WWW/wind/valid/raetaf/raetaf.html`.

[dlr94] DLR. «Technical Documentation of the DLR TAU-Code.» In *TAU Technical Guide*. 1994.

[dom11] DOMINO. «DOMINO NURBS library.» https://sourceforge.net/projects/dominonurbs, 2011.

[dre89] Drela, M. «Xfoil: an analysis and design system for low Reynolds number airfoils.» *Tech. rep.*, University of Notre Dame, 1989.

[duv04] Duvigneau, R., and M. Visonneau. «Hybrid genetic algorithms and artificial neural networks for complex design optimisation in CFD.» *International Journal for Numerical Methods in Fluids*, 44(11), (2004), 1257–1278.

[dwi05] Dwight, Richard. «An implicit LU-SGS scheme for finite-volume discretizations of the Navier-Stokes equations on hybrid grids.» *DLR-FB-2005-05 ISSN 1434-8454*, DLR, 2005.

[edw96] Edwards, J. R. «Comparison of eddy viscosity-transport turbulence models for three dimensional, shock-separated flowfields.» *AIAA Journal*, 34, nº 4, (1996), 756 – 763.

[ell95] Elliot, J., and J. Peraire. «Practical three-dimensional aerodynamic design by optimization.» *AIAA Journal*, 35(9), (1997), 1479–1485.

[ell97] Elliott, J., and J. Peraire. «Practical three-dimensional Aerodynamic Design and Optimisation using Unstructured Meshes.» *AIAA Journal*, 35(9), (1997), 1479–1485.

[emm06] Emmerich, M., K. Giannakoglou, and B. Naujoks. «Single- and multi-objective evolutionary optimisation assisted by Gaussian random field metamodels.» *IEEE Transactions on Evolutionary Computation*, 10(4), (2006), 421–439.

[eps09] Epstein, B., A. Jameson, S. Peigin, D. Roman, N. Harrison, and J. Vassberg. «Comparative study of three dimensional wing drag minimization by different optimization techniques.» In *46th AIAA Aerospace Science Meeting and Exhibit - AIAA paper 0326*. 2009.

[eps04] Epstein, B., and S. Peigin. «Robust Hybrid Approach to Multiobjective Constarined Optimization in Aerodynamics.» In *AIAA Journal*. 2004.

[eps05] —. «Constrained Aerodynamic Optimization of Three-Dimensional Wings Driven by Navier-Stokes Computations.» In *AIAA Journal*. 2005.

[fan00] Fang, K. T., D. K. Lin, P. Winker, and Y. Zhang. «Uniform Design: Theory and Application.» *Technometrics*, 39(3), (2000), 237–248.

[fog94] Fogel, D. B. «An introduction to simulated evolution.» *IEEE Transactions Neural Networks*, 5, (1994), 3–14.

[for09] Forrester, A., and A. Keane. «Recent Advances in Surrogate-Based Optimisation.» *Progress in Aerospace Sciences*, 45(1–3), (2009), 50–79.

[fus08] FUSIM, Project, C. Carreras, J. A. Lopez, R. Sierra, A. Rubio, G. Caffarena, V. Pejovic, O. Nieto, E. Andres, A. Baeza, and F. Palacios. «A Feasibility Study on the Application of FPGAs for the Hardware Acceleration of CFD acceleration.» *Tech. rep.*, Fusim-E, 2008.

[gad11] GADES, Consortium. «Global Aerodynamic Design capabilities for innovativE Shapes (GADES) proposal.», 2011.

[gar06] Gary, G., and S. Shan. «Review of Metamodeling Techniques in Support of Engineering Design Optimization.» In *ASME Transations, Journal of Mechanical Design*. 2006.

[ger97] Gerhold, T., M. Galle, O. Friedrichs, and J. Evans. «Calculation of Complex Three-Dimensional Configurations employing the DLR TAU-Code.» *AIAA-97-0167*.

[gia02] Giannakoglou, K. «Design of optimal aerodynamic shapes using stochastic optimisation methods and computational intelligence.» *Progress in Aerospace Sciences*, 38(1), (2002), 43–76.

[gia06] Giannakoglou, K. C., D.I. Papadimitriou, and I.C. Kampolis. «Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels.» *Computational Methods in Applied Mechanics Engineering*, 195, (2006), 6312–6329.

[gil03] Giles, M., and M. Duta. «Algorithm Developments for Discrete Adjoint Methods.» *AIAA Journal*, 41(2), (2003), 198–205.

[giu97] Giunta, A., V. Balabanov, D. Haim, B. Grossman, W. H. Mason, L. T. Watson, and R. Haftka. «Multidisciplinary Optimization of a Supersonic Transport Using Design of Experiments theory and Response Surface Modeling.» *Aeronautical Journal*, 101(1008), (1997), 347–356.

[gol89] Goldberg, D. E. «Genetic algorithms in search, optimization and machine learning.» *Addison-Wesley*.

[gou09] Gourdain, N., L. Gicquel, M. Montagnac, O. Vermorel, M. Gazaix, G. Staffelbach, M. Garcia, J. F. Boussuge, and T. Poinsot. «High performance parallel computing of flows in complex geometries: I. Methods.» In *Computational Science and Discovery*. 2009.

[gri00] Griewank, A. «Evaluating derivatives: Principles and Techniques of Algorithmic Differentiation.» In *SIAM Journal on Optimisation*. 2000.

[gro00] Gropp, W., K. Kaushik, Y. David, E. Keyes, and B. Smith. «Performance modeling and tuning of an unstructured mesh CFD application.» In *In Proceedings of SC2000. IEEE Computer Society*. 2000, 0 – 7803.

[guo04] Guo, Z., W. Najjar, F. Vahid, and K. Vissers. «A Quantitative Analysis of the Speedup Factors of FPGAs over Processors.» In *Proceedings of the International Symposium on FPGAs. ACM Press*. 2004.

[gup06] Gupta, S. «Performance evaluation and optimization of the unstructured CFD code UNCLE.» *Tech. rep.*, University of Kentucky, 2006.

[har04] Harris, M. «Fast fluid dynamics simulation on the GPU.» *GPU Gems, Addison Wesley, 637-665.*

[hed99] Hedayat, A. S., N. J. Sloane, and J. Stufken. «Orthogonal Arrays: Theory and Applications.» In *Springer*. 1999.

[hen93] Hendrickson, B., and R. Leland. «A multilevel algorithm for partitioning graphs Technical.» In *Sandia National Laboratories Report*. 1993.

[hic78] Hicks, R. M., and P.A. Henne. «Wing design by numerical optimization,.» *Journal of Aircraft*, 15(7), (1978), 407–12.

[hos11] Hossain, A., and A. Rahmanb. «Prediction of aerodynamic characteristics of an aircraft model with and without winglet using fuzzy logic technique.» *Aerospace Science and Technology*, 15(8), (2011), 595–605.

[hu05] Hu, S.M., and J.A. Wallner. «A second order algorithm for orthogonal projection onto curves and surfaces.» *Computer Aided Geometric Design*, 22, (2005), 251–260.

[ige06] IGES, Organization. «The Initial Graphics Exchange Specification (IGES) Version 5.» *Tech. rep.*, IGES/PDES Organization, 2006.

[int12] Intel. «The parallel programming landscape.» In *State of Parallel Programming*. 2012.

[iul09] Iuliano, E., R. Donelli, D. Quagliarella, I. Salah, and D. Arnal. «Natural Laminar Flow Design of a Supersonic Transport Jet Wing-Body.» In *47th AIAA Aerospace Sciences Meeting. AIAA-2009-1279*. 2009.

[jah11] Jahangirian, A., and A. Shahrokhi. «Aerodynamic shape optimization using efficient evolutionary algorithms and unstructured CFD solver.» *Computers and Fluids*, 46, (2011), 270–276.

[jam88] Jameson, A. «Aerodynamic design via control theory.» *Journal of Scienctific Computing*, 3(3), (1988), 233–260.

[jam91] —. «Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings.» *AIAA Paper 91-1596.*

[jam03] —. «Aerodynamic Shape optimization using the Adjoint Method.» In *Lecture Series at the Von Karman Institute*. 2003.

[jam98] Jameson, A., J.J. Alonso, J.J Reuther, L. Martinelli, and J. C. Vassberg. «Aerodynamic shape optimization techniques based on control theory.» In *AIAA Paper 98-2538*. 1998.

[jam81] Jameson, A., W. Schmidt, and E. Turkel. «Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes.» *AIAA Paper 81-1259*.

[jam87] Jameson, A., and S. Yoon. «Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations.» *AIAA Journal*, 25, nº 7, (1987), 929–935.

[jep09] Jespersen, D. C. «Acceleration of a CFD Code with a GPU.» In *NASA technical report*. 2009.

[jia12] Jiang, H., and W. He. «Grey relational grade in local support vector regression for financial time series prediction.» *Expert Systems with Applications*, 39, (2012), 2256–2262.

[jin01] Jin, R., W. Chen, and T. W. Simpson. «Comparative Studies of Metamodeling Techniques Under Multiple Modeling Criteria.» *Structural and Multidisciplinary Optimization*, 23(1), (2001), 1–13.

[jin05] Jin, Y. «A Comprehensive Survey of Fitness Approximation in Evolutionary Computation.» *Soft Computing Journal*, 9(1), (2005), 3–12.

[jin02] Jin, Y., M Olhofer, and B. Sendhoff. «A framework for evolutionary optimisation with approximate fitness functions.» *IEEE Transactions on Evolutionary Computation*, 6(5), (2002), 481–494.

[jon98] Jones, D. R., M. Schonlau, and W.J. Welch. «Effcient Global Optimisation of Expensive Black-Box Functions.» *Journal of Global Optimisation*, 13(4), (1998), 455–492.

[kal97] Kalagnanam, J. R., and U. M. Diwekar. «An Efficient Sampling Technique for Off-Line Quality Control.» In *Technometrics Journal*. 1997.

[kam08] Kampolis, I. C., and K. C. Giannakoglou. «A multilevel approach to single and multiobjective aerodynamic optimization.» *Computational Methods in Applied Mechanics Engineering*, 197, (2008), 2963–2975.

[kam11] —. «Synergetic use of different evaluation, parametrisation and search tools within a hierarchical optimisation platform.» *Applied Soft Computing*, 11(1), (2011), 645–651.

[kam10] Kampolis, I.C., X.S. Trompoukis, V.G. Asouti, and K.C. Giannakoglou. «CFD-based Analysis and Two-level Aerodynamic Optimization on Graphics Processing Units.» *Computer Methods in Applied Mechanics and Engineering*, 199(9-12), (2010), 712–722.

[kar06] Karakasis, M., and K.C. Giannakoglou. «On the use of metamodel-assisted, multiobjective evolutionary algorithms.» *Journal of Engineering Optimisation*, 38(8), (2006), 941–957.

[kar98] Karypis, G., and V. Kumar. «Multilevel algorithms for multi-constraint graph partitioning.» In *University of Minnesota, Department of Computer Science/Army, HPC Research Center*. 1998.

[ken95] Kennedy, J., and R.C. Eberhart. «Particle swarm optimisation.» In *IEEE International Conference on Neural Networks*. 1995.

[khu09] Khurana, M., H. Winarto, and A. Sinha. «Airfoil optimization by swarm algorithms with mutation and artificial neural networks.» *American Institute of Aeronautics and Astronautics Conference*, (2009), 1278–1289.

[kim09] Kim, J., and V. N. Rao. «Development of an efficient aerodynamic shape optimization framework.» *Mathematics and Computers in Simulation*, 79, (2009), 2373–2384.

[kio96] Kiock, R. «The ALVAST Model of DLR.» *DLR Institusbericht 129-96/22*, DLR, 1996.

[kon11] Kontoleontos, E. A., V.G. Asouti, and K.C Giannakoglou. «An Asynchronous Metamodel–Assisted Memetic Algorithm for CFD–based shape optimisation.» In *Engineering Optimisation*. 2011.

[kru03] Krüger, J., and R. Westermann. «Linear algebra operatoes for GPU implementation of numerical algorithms.» In *ACM Transactions on Graphics*. 2003.

[kro01] Kroll, N., T. Gerhold, S. Melber, R. Heinrich, T. Schwarz, and B. Schoning. «Parallel Large Scale Computations for Aerodynamic Aircraft Design with the German CFD System MEGAFLOW.» *Parallel Computational Fluid Dynamics - Practice and Theory*. URL `http://elib.dlr.de/12496/`.

[kru08] Krumbein, A. «personal communication.»

[kul07] Kulfan, B. M. «A Universal Parametric Geometry Representation Method "CST".» In *AIAA-2007-0062*. 2007.

[kul06] Kulfan, B. M., and J. Bussoletti et al. «Fundamental Parametric Geometry Representations for Aircraft Component Shapes.» In *AIAA-2006-6948*. 2006.

[kwo03] Kwok, J.T., and I.W. Tsang. «Linear dependency between ϵ; and the input noise in ϵ-support vector regression.» *IEEE Transactions on Neural Networks*, 14, (2003), 544–553.

[laf05] Laflin, K., S. Klausmeyer, T. Zickuhr, J. Vassberg, R. Wahls, J. Morrison, O. Brodersen, M .Rakowitz, E. Tinoco, and J. Godard. «Data Summary from Second AIAA Computational Fluid Dynamics Drag Prediction Workshop.» *Journal of Aircraft*, 42, n$^{\underline{o}}$ 5, (2005), 1165–1178.

[lan11] Langer, S. «Application of a line implicit method to fully coupled system of equations for turbulent flow problems.» *Journal of Computer and Fluids*, (2011), DOI: 10.1002/fld2561.

[lep01] Lepine, J., F. Guibault, J. Trepanier, and F. Pepin. «ptimized nonuniform rational b-spline geometrical representation for aerodynamic design of wings.» In *AIAA Journal, 39(11)*. 2001.

[lep00] Lepine, J., and J. Trepanier. «Wing Aerodynamic Design Using an Optimized NURBS Geometrical Representation.» In *38th Aerospace Science Meeting and Exibit. AIAA paper 2000-669*. 2000.

[li98] Li, P., A. R. Seebass, and Sobieczky. «Manual aerodynamic optimization of an oblique flying wing.» *American Institute of Aeronautics and Astronautics Conference*, (1998), 98–108.

[lia08] Liakopoulos, P., I.C. Kampolis, and K.C. Giannakoglou. «Grid enabled, hierarchical distributed metamodel-assisted evolutionary algorithms for aerodynamic shape optimization.» *Future Computer Systems*, 24, (2008), 701–708.

[lia10] Lian, Y., A. Oyama, and M. S. Liou. «Progress in design optimization using evolutionary algorithms for aerodynamic problems.» *Progress in Aerospace Sciences*, 46, (2010), 199–223.

[lia03] Liang, Y.M, and W.T. Hewitt. «Point inversion and projection for NURBS curve and surface: Control polygon approach.» *Computer Aided Geometric Design*, 20, (2003), 79–99.

[lim10] Lim, D., Y. Jin, Y.S. Ong, and B. Sendho. «Generalizing Surrogate-Assisted Evolutionary Computation.» *IEEE Transactions of Evolutionary Computation*, 14(3), (2010), 329–355.

[lin04] Lin, Y. «An Efficient Robust Concept Exploration Method and Sequential Exploratory Experimental Design.» In *Ph.D. Dissertation Thesis, Mechanical Engineering, Georgia Institute of Technology, Atlanta*. 2004.

[loz09] Lozano, C. «DOVRES Task 2.1/INTA3 Final Report: Assessment of Mesh Sensitivities in Continuous Adjoint Gradients for Aerodynamic Shape Optimization in the TAU Code.» INTA technical report, 2009.

[loz12] —. «Discrete Surprises in the Computation of Sensitivities from Boundary Integrals in the Continuous Adjoint Approach to Inviscid Aerodynamic Shape Optimization.» *Computers and Fluids*, 56, (2012), 118–127.

[loz11] Lozano, C., E. Andrés, M. Martín, and P. Bitrián. «Domain versus boundary computation of flow sensitivities with the continuous adjoint method for aerodynamic shape optimization problems.» *International Journal for Numerical Methods in Fluids*, DOI 10.1002/fld.2743.

[luk04] Luk, W., P.Y.K. Cheung, and N. Shirazi. «Configurable Computing.» In *Electrical Engineer's Handbook*. 2004.

[mah04] Mahmoud, B., M. H. Bedoui, R. Essababah, and H. Essabbah.

[mar10] Marinus, B. G., M. Rogery, and R. Braembusschez. «Aeroacoustic and Aerodynamic Optimisation of Aircraft Propeller Blades.» In *AIAA/CEAS Aeroacoustics Conference. AIAA 2010-3850*. 2010.

[mar08] Martin, M., E. Andres, and C. Carreras. «Solving the Euler Equations for Unstructured Grids on Graphical Processor Units.» In *9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*. Trondheim, Norway, 2008.

[mar11] Martin, M., E. Andrés, M. Widhalm, P. Bitrián, and C. Lozano. «NURBS-based aerodynamic shape design optimisation with the DLR TAU code.» In *Aerospace Engineering, Proceedings of the Institution of Mechanical Engineers Part G*. 2011.

[mag11] Martin, M., G. Guirao, and E. Andrés. «CORESIMULAERO Task 5.2/INTA1 Mid Term report: Optimal NURBS for design.» INTA technical report, 2011.

[mar09] Marzocca, P. «The NACA airfoil series.» *Tech. rep.*, Clarkson University, 2009.

[mav98] Mavriplis, D. «Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes.» *Journal of Computational Physics*, 145, n$^{\underline{o}}$ 1, (1998), 141–165.

[mav02] —. «Parallel Performance Investigation of an Unstructured Mesh Navier-Stokes Solver.» *International Journal of High Performance Computing*, 2(16), (2002), 395–406.

[mav06] —. «Multigrid Solution of the Discrete Adjoint for Optimization Problems on Unstructured Meshes.» *AIAA Journal*, 44, (2006), 42–50.

[mav07] —. «Unstructured Mesh Discretizations and Solvers for Computational Aerodynamics.» *AIAA paper 2007-3955*.

[mea04] Meaux, M., M. Cormery, and G. Voizard. «Viscous aerodynamic shape optimization based on the discrete adjoint state for 3d industrial configurations.» In *Technical report, European Congress on Compurational Methods in Applied Sciences and Engineering*. 2004.

[mec01] Meckesheimer, M. «A Framework For Metamodel-Based Design: Subsystem Metamodel Assessment and Implementation Issues.» In *Ph. D. Dissertation Thesis, Industrial Engineering, The Pennsylvania State University*. 2001.

[mec02] Meckesheimer, M., A. J. Booker, R. R. Barton, and T. W. Simpson. «Computationally Inexpensive Metamodel Assessment Strategies.» *AIAA Journal*, 40(10), (2002), 2053–2060.

[men94] Menter, F. «Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications.» *AIAA Journal*, 32, n$^{\underline{o}}$ 8, (1994), 1598–1605.

[met97] METIS. «METIS: unstructured graph partitioning and sparse matrix ordering system.» *Tech. rep.*, http://glaros.dtc.umn.edu/gkhome/views/metis, 1997.

[mit74] MITCHELL, T. J. «An Algorithm for the Construction of "D-Optimal" Experimental Designs.» *Technometrics*, 16(2), (1974), 203–210.

[mit92] MITCHELL, T. J., and M. D. MORRIS. «Bayesian Design and Analysis of Computer Experiments: Two Examples.» *Statistica Sinica*, 2, (1992), 359–379.

[moh01] MOHAMMADI, B., and O. PIRONNEAU. «Comparative study of three dimensional wing drag minimization by different optimization techniques.» In *Applied Shape Optimization for Fluids, Oxford: Oxford University Press*. 2001.

[mom02] MOMMA, M., and K. P. BENNETT. «A pattern search method for model selection of support vector regression.» In *SIAM International Conference on Data Mining*. 2002.

[mor08] MORRIS, A., C. ALLEN, and T. RENDALL. «CFD-based optimisation of airfoils using radial basis functions for domain element parametrisation and mesh deformation.» *International Journal for Numerical Methods in Fluids*, 58(8), (2008), 827–860.

[mou07] MOUSAVI, A., P. CASTONGUAY, and S. NADARAJAH. «Survey of shape parameterization techniques and its effect on three-dimensional aerodynamic shape optimization.» In *AIAA 37th Fluid Dynamics Conference and Exhibit. AIAA paper 2007-3837*. 2007.

[mul02] MULLER, S.D., J. MARCHETTO, S. AIRAGHI, and P. KOUMOUTSAKOS. «Optimisation Based on Bacterial Chemotaxis.» *IEEE Transactions on Evolutionary Computation*, 6(1), (2002), 16–29.

[myr95] MYERS, R. H., and D. MONTGOMERY. «Response Surface Methodology: Process and Product Optimization Using Designed Experiments.» In *John Wiley and Sons, Inc.* 1995.

[nad07] NADARAJAH, S. K., and A. JAMESON. «Optimun Shape Design for Unsteady Flows with Time-Accurate Continuous and Discrete Adjont Methods.» *AIAA Journal*, 45(7), (2007), 1478–1491.

[nak11] NAKASHIMA, T., and J. GIMÉNEZ. «Performance Analysis and Optimization of A CFD Code on A Large-Scale Parallel Computer.» In *ParCFD2011*. 2011.

[nem11] NEMILI, A., E. OEZKAYA, N. R. GAUGER, A. CARNARIUS, and F. THIELE. «Optimal Control of Unsteady flows using Discrete Adjoints.» In *41st AIAA Fluid Dynamics Conference and Exhibit. AIAA 2011-3720*. 2011.

[nor05] NORIEGA, L. «Multilayer Perceptron Tutorial.» *Tech. rep.*, School of Computing. Staffordshire University, 2005. URL `http://www.acare4europe.org/docs/Towards2050.pdf`.

[nvi07] NVIDIA, Corporation. «Compute Unified Device Architecture Programming Guide (CUDA).» In *http://www.nvidia.com*. 2007.

[obe00] Oberkampf, W. L., and T. G. Trucano. «Validation Methodology in Computational Fluid Dynamics.» In *AIAA 2000-2549*. 2000.

[ong06] Ong, Y. S., M.H. Lim, N. Zhu, and K.W. Wong. «Classification of adaptive memetic algorithms: a comparative study.» *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 36(1), (2006), 141–152.

[ong03] Ong, Y.S., P.B. Nair, and A. J. Keane. «Evolutionary optimisation of computationally expensive problems via surrogate modelling.» *AIAA Journal*, 41(4), (2003), 687–696.

[orr96] Orr, M. «Introduction to Radial Basis Functions (RBF) Network.» *Tech. rep.*, Center for Cognitive Science. University of Edinburg, 1996. URL `http://www.acare4europe.org/docs/Towards2050.pdf`.

[ort09] Ortiz-Garcia, E., S. Salcedo-Sanz, A. Perez-Bellido, and J. A. Portilla-Figueras. «Improving the training time of support vector regression algorithms through novel hyper-parameters search space reductions.» *Neurocomputing*, 72, (2009), 3683–3691.

[pai04] Painchaud-Ouellet, S., C. Tribes, J. Y. Trepanier, and D. Pelletier. «Airfoild Shape Optimization Using NURBS Representation Under Thickness Constraint.» In *42nd AIAA Aeroespace Sciences Meeting and Ehibit, AIAA 2004-1095*. 2004.

[pal07] Palki, A. B. «Cache optimization and performance evaluation of a structured CFD code.» *Tech. rep.*, University of Kentucky, 2007.

[par02] PARMETIS. «PARMETIS: parallel graph partitioning and fill-reducing matrix ordering.» *Tech. rep.*, http://glaros.dtc.umn.edu/, 2004.

[pei04] Peigin, S., and B. Epstein. «Robust Handling of Non-Linear Constraints for GA Optimization of Aerodynamic Shapes.» In *International Journal for Numerical Methods in Fluids*. 2004.

[pet10] Peter., J., and R. Dwight. «Numerical sensitivity analysis for aerodynamic optimisation: A survey of approaches.» *Computers and Fluids*, 39(3), (2010), 373–391.

[pet07] Peter, J., and M. Marcelet. «Comparison of surrogate models for turbomachinery design.» In *7th WSEAS International Conference on Simulation, Modelling and Optimisation*. 2007.

[pie97] Piegl, L., and W. Tiller. *The NURBS book*. Springer-Verlag, 1997.

[pra09] Praveen, C., and R. Duvigneau. «Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design.» *Computational Methods in Applied Mechanics Engineering*, 198, (2009), 1087–1096.

[pts06] PTSCOTCH. «Software package and libraries for sequential and parallel graph partitioning.» *Tech. rep.*, http://www.labri.fr/perso/pelegrin/scotch, 2004.

[put03] Puttegowda, K., W. Worek, N. Pappas, A. Dandapani, P. Athanas, and A. Dickerman. «A Run-Time Reconfigurable System for Gene-Sequence Searching.» *International Conference on VLSI Design*, 0, (2003), 561.

[pyt08] Python, Community. «Python Open Source.» http://www.python.org.

[rak05] Rakowitz, M., S. Heinrich, A. Krumbein, B. Eisfeld, and M. Sutcliffe. «Computation of Aerodynamic Coefficients for Transport Aircraft with MEGAFLOW.» In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design* (Springer, ed.). Springer Verlag, 2005, volume 89, 135–150.

[red09] Redondo, D. «Inversión de punto sobre un panel NURBS.» *Universidad Carlos III de Madrid.*

[roa98] Roache, P. J. «Verification and Validation in Computational Science and Engineering.» In *Hermosa Publishers.* 1998.

[ron05] Ronzheimer, A. «Shape Parameterization in Multidisciplinary Design Optimization Based on Freeform Deformation.» In *Evolutionary and Deterministic Methods for Design (EUROGEN).* 2005.

[sae12] Saeed, A. K. «Acceleration of CFD and Data Analysis Using Graphics Processors.» *Tech. rep.*, University of Massachusetts, 2012.

[sal11] Salcedo-Sanz, S., E. Ortiz-Garcia, A. Perez-Bellido, and A. Portilla. «Short term wind speed prediction based on evolutionary support vector regression algorithms.» *Expert Systems with Applications*, 38, (2011), 4052–4057.

[sam04] Samareh, J. A. «Aerodynamic Shape Optimization Based on Free-Form Deformation.» In *AIAA 2004-4630.* 2004.

[sam08] —. «A survey of Shape Parametrisation Techniques.» In *CAES/AIAA 2008 ATIO Conference. AIAA 2008-8879.* 2008.

[san11] Sanchez, D., G. Sutter, S. Lopez, I. Gonzalez, F. J. Gomez, J. Aracil, and F. Palacios. «Using high level languages and floating-point arithmetic in FPGA-based acceleration of aeronautical CFD simulations.» In *IEEE Design and Test of Computers.* 2011.

[san08] Santos, M., B. De Mattos, and R. Girardi. «Aerodynamic Coefficient Prediction of Airfoils using Neural Networks.» In *AIAA 2008-887.* 2008.

[sch95] Schenk, M. «Modification of the ALVAST geometry for CFD calculations.» DLR IB 129-95/25, 1995. URL `http://elib.dlr.de/36394`.

[sch08] Schmidt, S., C. Ilic, V. Schulz, and N. Gauger. «Shape Gradients and their Smoothness for Practical Aerodynamic Design Optimization.» *Tech. rep.*, DFG Forschungsberich SPP1253-10-03, 2008.

[sch79] Schmitt, V., and F. Charpin. «Pressure Distributions on the ONERA M6 Wing at Transonic Mach Numbers.» *Experimental Database for Computer Program Assessment. AGARD-AR-138, pp. B1-1-B1-44*, May 1979.

[sch06] Schwamborn, D., T. Gerhold, and R. Heinrich. «The DLR Tau-code: recent applications in research and industry.» In *Proceeding of ECCOMAS CFD 2006*. Netherlands, 2006.

[sel06] Selimovic, I. «Improved algorithms for the projection of points on NURBS curves and surfaces.» *Computer Aided Geometric Design*, 23(5), (2006), 439–445.

[she94] Shephard, M., J. Flahertya, H. de Cougny, C. Ozturan, C. Bottasso, and M. Beall. «Parallel automated adaptive procedures for unstructured meshes.» *AGARD R-807, pages 6.1-6.49*, In Parallel Comput. in CFD, Neuilly-Sur-Seine, 1994.

[sil05] Sillen, M. «Evaluation of Parallel Performance of an Unstructured CFD-Code on PC-Clusters.» In *17th AIAA Computational Fluid Dynamics Conference*, AIAA, Toronto, Ontario, Canada, 2005-4629.

[sim01] Simpson, T. W., J. Peplinski, P. N. Koch, and J. K. Allen. «Metamodels for Computer-Based Engineering Design: Survey and Recommendations.» *Engineering with Computers*, 17(2), (2001), 129–150.

[smo98] Smola, A. J., N. Murata, B. Scholkopf, and K. Muller. «Asymptotically optimal choice of ϵ-loss for support vector machines.» *Proc. of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing.*

[smo99] Smola, A. J., and B. Scholkopf. «A tutorial on support vector regression.» In *Statistics and Computing*. 2004, volume 14(3).

[sob98] Sobieczky, H. «Parametric Airfoils and Wings.» *Numerical Fluid Mechanics*, 16, (1998), 71–88.

[son04] Song, W., and A.J. Keane. «A study of shape parameterization methods for airfoil optimisation.» In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimisation Conference, AIAA paper 2004-4482*. 2004.

[spa92] Spalart, P. R., and S. R. Allmaras. «A one-equation turbulence model for aerodynamic flows.» *AIAA Paper 92-0439.*

[sri10] Sripawadkul, V., M. Padulo, and M. Guenov. «A Comparison of Airfoil Shape Parametrisation Techniques for Early Design Optimisation.» In *13th AIAA/ISSMO Multidisciplinary Analysis Optimisation Conference. AIAA 2010-9050*. 2010.

[stu11] Stück, A., and T. Rung. «Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation.» *Computers and Fluids*, 47, (2011), 22–32.

[ste03] Stefano, P. Di, and L. Di Angelo. «Neural Network Based Geometric Primitive for Airfoil Design.» In *Shape Modeling International*. 2003.

[sto97] Storn, R., and K. Price. «Differential evolution: a simple and efficient heuristic for global optimisation over continuous spaces.» *Journal of Global Optimisation*, 11, (1997), 341–359.

[str06] Strenski, D. «Computational Bottlenecks and Hardware Decisions for FPGAs.» *FPGA and Programmable Logic Journal*.

[sut11] Sutter, G., S. Lopez-Buedo, I. Gonzalez, F. J. Gomez-Arribas, and J. Aracil. «An Euler solver accelerator in FPGA for computational fluid dynamics applications.» In *2011 VII Southern Conference on Programmable Logic (SPL)*. 2011.

[szo09] Szollos, A., M. Smid, and J. Hájek. «Aerodynamic Optimisation via Multi-objective Micro-genetic Algorithm with Range Adaptation: Knowledge-Based Reinitialization, Crowding and Dominance.» *Advances in Engineering Software*, 40(6), (2009), 419–430.

[tay95] Taylor, V. E., and B. Nour-Omid. «A Study of the Factorization Fill-in for a Parallel Implementation of the Finite Element Method.» *International Journal for Numerical Methods in Engineering*, 37, (1995), 3809–3823.

[ter04] Teresco, J. D., J. Faik, and J. E. Flaherty. «Hierarchical Partitioning and Dynamic Load Balancing for Scientific Computation.» In *Workshop on State-Of-The-Art in Scientific Computing: 7th International Conference*. 2004.

[tod05] Todman, T.J., G.A. Constantinides, S.J.E. Wilson, O. Mencer, W. Luk, and P.Y.K. Cheung. «Regonfigurable Computing: Architectures and Design Methods.» *IEEE Proc. Comput. Digit. Tech.*, 152(2).

[van10] Vankan, J., and R. Maas. «Surrogate Modelling For Efficient Design Optimisation of Composite Aircraft Fuselage Panels.» In *27th International Congress Of The Aeronautical Sciences, ICAS*. 2010.

[vap98] Vapnik, V. «Statistical Learning Theory.» In *Wiley*. 1998.

[vas02] Vassberg, J., and A. Jameson. «Aerodynamic Shape Optimization of a Reno Race Plane.» In *International Journal of Vehicle Design*. 2002.

[vas06] —. «Aerodynamic Shape Optimization.» In *Von Karman Institute for Fluid Dynamics, Lecture Series - Introduction to Optimization and Multidisciplinary Design*. 2006.

[vas07] Vassberg, J., E. Tinoco, M. Mani, O. Brodersen, B. Eisfeld, R. Wahls, J. Morrison, T. Zickuhr, K. Laflin, and D. Mavriplis. «Summary of the Third AIAA CFD Drag Prediction Workshop.» *Paper 2007-0260*, AIAA, Reno, Nevada, January 2007.

[wan05] Wang, X., C. Yang, B. Qin, and W. Gui. «Parameter selection of support vector regression based on hybrid optimization algorithm and its application.» *Journal of Control Theory and Applications*, 3, (2005), 371–376.

[wee05] de Weerdt, E., Q.P. Chu, and J. A. Mulder. «Neural Network Aerodynamic Model Identification for Aerospace Reconfiguration.» In *AIAA*. 2005.

[won05] Won, K. S., and T. Ray. «A Framework for Design Optimisation using Surrogates.» *Engineering Optimisation*, 37(7), (2005), 685–703.

[wyl07] Wylie, N., J. Brian, M. Geimer, M. Nicolai, and M. Probst. «Performance analysis and tuning of the XNS CFD solver on BlueGene/L.» In *Proceedings of the 14th European PVM/MPI User's Group Meeting*. Springer, 2007, volume LNCS 4757, 107–116.

[xin99] Xin, Y., Y. Liu, and G. Lin. «Evolutionary programming made faster.» *IEEE Transactions on Evolutionary Computation*, 3, (1999), 82–102.

[ye00] Ye, K. Q., W. Li, and A. Sudjianto. «Algorithmic Construction of Optimal Symmetric Latin Hypercube Designs.» *Journal of Statistical Planning and Inferences*, 90, (2000), 145–159.

[yim08] Yim, J., B.J Lee, and C. Kim. «Multi-Stage Aerodynamic Design of Multi-Body Geometries via Global and Local Optimisation Methods.» In *46th AIAA Aerospace Sciences Meeting and Exhibit. AIAA 2008-134.* 2008.

[you08] Younis, A., G. Jichao, D. Zuomin, and L. Guangyao. «Trends, Features, and Test of Common and Recently Introduced Global Optimisation Methods.» In *2th AIAA/ISSMO Multidisciplinary and Optimisation Conference. AIAA 2008-5853.* 2008.

[zho10] Zhong-Hua, H., R. Zimmermann, and S. Görtz. «A New Cokriging Method for Variable-Fidelity Surrogate Modeling of Aerodynamic Data.» In *AIAA*. 2010.

[zho07] Zhou, Z., Y.S. Ong, M. Lim, and B. Lee. «Memetic algorithms using multi-surrogates for computationally expensive optimisation problems.» *Journal of Soft Computing*, 11(10), (2007), 957–971.

[zol08] Zoltan. «ZOLTAN Parallel Partitioning, Load Balancing and Data-Management Services.» http://www.cs.sandia.gov/Zoltan.

[zym09] Zymaris, A., D. Papadimitriou, K. Giannakoglou, and C. Othmer. «Continuous Adjoint Approach to the Spalart-Alllmaras Turbulence Model for Incompressibe Flows.» *Computers and Fluids*, 38(8), (2009), 1528–1538.

[zym10] —. «Continuous Adjoint wall functions: A new concept for use in aerodynamic shape optimisation.» *Journal of Computational Physics*, 229(13), (2010), 5228–5245.