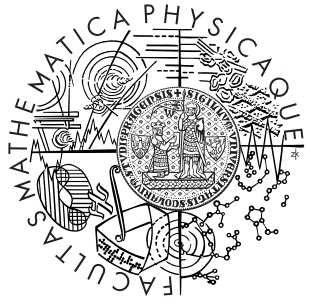
Charles University, Prague
Faculty of Mathematics and Physics
Department of Computational Mathematics

Ph.D. Thesis



Jaroslav Hájek

Aerodynamic optimization of airfoils and wings using fast solvers

Supervisor: Prof. Ing. Pavel Šafařík, CSc.
Study branch: Scientific and technical computing

Prague 2009

At this place I would like to give my sincere thanks to prof. Pavel Šafařík for his undying perseverance in guiding my work. My thanks are further directed towards Mgr. András Szöllös and Ing. Zdeněk Pátek, my colleagues from the Aeronautical Research and Test Institute, where most of this work was conducted, and to the Czech Ministry of Education. I also thank my wife for her patience while I was writing the thesis.

I hereby declare I have written my PhD thesis on my own using none but the referenced sources.

Contents

Title: Aerodynamic optimization of airfoils and wings using fast solvers
Author: Jaroslav Hájek
Department: Department of Numerical Mathematics
Supervisor: Prof. Ing. Pavel Šafařík, CSc.
Supervisor's e-mail address: pavel.safarik@fs.cvut.cz
Abstract: This thesis is concerned with aerodynamic optimization of airfoils and wings. It focuses at using very fast solvers based on hybrid potential methods to evaluate the aerodynamic performance of the airfoils and wings. Compared to more widely used CFD solvers which are more computationally demanding, use of fast solvers brings different possibilities and different problems to tackle, some of which are analyzed in this work. A state-of-the-art fast solver for airfoils is presented, and a fast solver for slender wings is developed, and some of its aspects are discussed. An innovative evolutionary optimization algorithm is presented and both solvers are then utilized to solve real-life airfoil and wing optimization problems.
Keywords: optimization, fast solver, airfoil, wing

Název: Aerodynamická optimalizace profilů a křídel s použitím rychlých řešičů
Autor: Jaroslav Hájek
Katedra: Katedra numerické matematiky
Vedoucí dizertační práce: Prof. Ing. Pavel Šafařík, CSc.
e-mail vedoucího: pavel.safarik@fs.cvut.cz
Abstrakt: Tato práce se zabývá aerodynamickou optimalizací křídel a jejich profilů. Zaměřuje se na použití velmi rychlých řešičů založených na hybridních potenciálních metodách pro vyhodnocování aerodynamické výkonnosti těchto profilů a křídel. Ve srovnání s šířeji používanými CFD řešiči, které jsou výpočetně náročnější, použití rychlých řešičů přináší nové možnosti a nové problémy k řešení, některé z nichž jsou analyzovány v této práci. Je prezentován zavedený rychlý řešič pro profily a vyvinut rychlý řešič pro štíhlá křídla, jehož některé aspekty jsou diskutovány. Je představen inovativní evoluční optimalizační algoritmus a oba řešiče jsou použity k řešení reálných optimalizací profilů a křídel.
Klíčová slova: optimalizace, rychlé řešiče, profil, křídlo

Selected notation

α angle of attack [1]

c_L lift coefficient [1]

c_D drag coefficient [1]

c_M momentum coefficient [1]

Γ, γ circulation $[m^2 s^{-1}]$

v velocity $[ms^{-1}]$

$\mathbf{v}$ velocity vector $[ms^{-1}]$

$v_x, v_y, v_z, v_r, v_\theta$ velocity components $[ms^{-1}]$

$\mathbf{f}$ objective vector

$\mathbf{x}$ design vector

f_i i-th objective

ψ streamline function (section 2) $[m^2 s^{-1}]$

ψ base function for parametrization (section 3)

$\mathbf{F}, \mathbf{J}, \mathbf{\Gamma}, \ldots$ vectors and matrices

$C_L(\alpha)$ lift coefficient as function of angle of attack

Dictionary of terms

Airfoil The shape of a vertical cut through a wing.

Chord The longest segment inside an airfoil.

Degree of freedom See Design variable.

Design variable A variable $(1 \dots N)$ of the parametrization.

Design vector A vector of design variables, element of $\mathbb{R}^N$.

Design space The set of all admissible design vectors.

Flowfield The vector field corresponding to the velocity of a flow.

Freestream The flow velocity at infinity; velocity of undisturbed flow.

GPARSEC Generalized PARSEC. A parametrization method for airfoils, see Sec. 3.

GNU Octave Software for numerical programming, data analysis and plotting. See `<http://www.octave.org>`

Lift Upwards force; acting in direction orthogonal to freestream velocity.

Drag Backwards force; acting in the direction of freestream velocity.

NLWing2 Free software for viscous analysis of slender wings. See Sec. 4.

Objective A decision-making function of the design.

Objective vector A vector of objectives.

Objective space The set of all possible objective vectors.

Pareto dominance A partial ordering between objective vectors, inequality in all components.

Population A working set of designs. Can denote either the set of design vectors, or their corresponding objective vectors.

Potential vortex Elemental inviscid irrotational incompressible flow. See Sec. 2.

Potential source Elemental inviscid irrotational incompressible flow. See Sec. 2.

Kutta-Joukowski law A fundamental relation between vorticity and induced lift.

Parametrization A mapping from $\mathbb{R}^N$ to geometric designs.

Population A finite set of design vectors.

Streamline function A scalar field determining an irrotational inviscid incompressible flow. See Sec. 2.

Superposition A flowfield obtained by summing several other flowfield.

Vortex thread, vortex line An element of 3D potential flow, generalization of 2D potential vortex. See Sec. 4.

Vortex segment A straight finite part of a vortex line or vortex thread.

XFOIL Free software for visual aerodynamic analysis of airfoils. See Sec. 2.

XFEVAL Software for automatic parallel evaluation of airfoils using XFOIL.

Wing section See Airfoil.

1 Introduction

In the area of 2-dimensional aerodynamic analysis, especially the analysis of airfoils (wing sections), it has long ago become possible to analyze hundreds or thousands of airfoils within relatively short time, thus opening the door to their optimization based on general nonlinear optimization algorithms.

Aerodynamic analysis of 3-dimensional flows is a different story. Most of the current attention is focused onto CFD methods. These methods are based on fundamental flow equations, like the Navier-Stokes equations, and attempt to numerically solve these equations by employing methods for solving partial differential equations. These methods usually employ meshes with many cells, leading to sometimes enormously large systems of equations, from thousands to many million unknowns. The performance of current computers and efficient methods for solving large sparse linear systems already allow these methods to be applied on real-world problems and evaluate aerodynamic properties of aerodynamic designs in acceptable time.

Looking at a typical CFD method, based on finite-volume (FV) or finite-element (FE) scheme from a physicist's perspective, we see that each cell is usually a simplistic physical model of the physical phenomenon in question, interacting in a simple manner with neighboring models. For instance, in the FV method, the physical quantities are assumed to be constant in the cell, reducing the governing equations to simple relations. It is the number of these elementary models that gives the finite volume and finite element methods their ability to model complex physical phenomena. The necessity of this approach stems from the fact that the governing partial differential equations are nonlinear.

It is well known that an irrotational, inviscid and incompressible flow admits a solution using velocity potential, satisfying the Laplace equation. We speak of a potential flow. The model of irrotational inviscid incompressible flow is very simple and almost unphysical; however, it is able to deliver surprisingly good approximate solutions for many real problems.

Moreover, there is a huge advantage in reducing the problem to a potential flow. A number of exact solutions for potential flow, corresponding to idealized models of physical phenomena, such as sources, sinks, vortexes and doublets, are known. And because the Laplace equation is linear, more complex potential flows can be obtained by superposition of the elementary flows. Usually, these flow elements can be only distributed along the *boundary* of the solution domain, i.e., when dealing with external aerodynamics, at object surfaces. Therefore, compared to the FV and FE approaches, the resulting discrete problem is much smaller. A necessary tradeoff is that the interaction of non-neighboring elements cannot be neglected, because they interact through the induced effect in the domain interior. As a result, instead of dealing with large sparse problems, these methods usually lead to much

smaller dense (non-sparse) problems. This approach has been extensively exploited before solving Navier-Stokes equations directly by FV and FE methods became computationally tractable.

Because the potential flow model is unable to model certain effects resulting from viscosity, in particular the boundary layer development, researchers were trying to combine the potential flow with supplementary models allowing to catch the viscous effects, yet retaining the computation advantages. As performance of computers increased, most of these research directions were abandoned in favor of focusing on CFD methods, that do not carry the theoretical limitations of potential flow.

However, the transition from mere performance evaluation to design optimization increases the computational complexity by several orders of magnitude, because instead of evaluating individual designs, hundreds or thousands are required. A CFD computation taking several hours is acceptable for fine tuning and assessing the performance of final designs. For preliminary stages of the optimization, where performance is preferred to accuracy, it is natural to ask whether we can utilize the fast potential flow methods to provide cheap yet sufficiently accurate predictions of aerodynamic performance, allowing us to locate the most promising regions of design space to proceed further with CFD methods.

Although analogical in theory, in practice aerodynamic optimization using a cheap solver is significantly different from that based on an expensive solver. An expensive solver may cause a single trial design evaluation to take hours or even days; usually, in that case the computational overhead of the algorithm itself becomes negligible, and the sole concern is to get an acceptable solution using as few trial designs as possible. A cheap solver, being orders of magnitude faster, allows us to operate with equivalently more trial designs. Usually we want to exploit this advantage to search our design space more thoroughly to escape possible local minima, and deliver more complete results (such as the Pareto front described in further sections). Obviously, this typically means a different sort of algorithms need to be employed for each case.

This thesis focuses on exploiting cheap aerodynamic computational methods based on potential flow in aerodynamic optimization. Such an optimization can serve as a preliminary stage for optimization using CFD methods; however, it is shown that for some real-world problems their results are directly applicable.

Four incremental steps are taken towards the final goal. The first section is concerned by the solution of flow around airfoils, explains basic concepts and theory and describes XFOIL, a state-of-the-art engineering software for airfoil design and performance evaluation. In the second section, an approach to airfoil optimization based on evolutionary algorithms is described in detail, including the discussion of design parametrization and practical aspects of parallel evaluation. The third section explains how to utilize results of two-dimensional computations in the nonlinear

lifting line model to allow a very efficient performance evaluation of slender wings. This method is then utilized in aerodynamic optimization of wings, described in the fourth section.

2 Airfoil flow solution

Airfoils are basic two-dimensional aerodynamic shapes forming cross sections of wing, turbine blades and other aerodynamic objects. Airfoils are studied in the frame of two-dimensional flow, and their aerodynamic properties are probably the most thoroughly explored and understood in the whole field of aerodynamics. This section describes the basic relations in two-dimensional potential flow and flow around airfoils, which will also become important in building the model for solving three-dimensional flows around wings.

The importance of studying two-dimensional flows around airfoils stems from the fact that such flows are equivalent to three-dimensional flows around wings of infinite span. Although in practice the span is always finite, the simplified two-dimensional models can still provide us with a lot of information about the true three-dimensional flows around a wing, especially for long-spanned wings.

There is a lot of existing software that can be used to evaluate aerodynamic performance of airfoils. The XFOIL code ([1]), which is a free software, ranks amongst the most highly regarded tools based on potential flow. In this section, capabilities of the XFOIL software for evaluating aerodynamic performance of airfoils, are described. These capabilities form the basis for airfoil optimizations described in the next section, and are also important for solving wing flows.

2.1 2D inviscid incompressible irrotational flow

The most simple model of flow is the inviscid incompressible irrotational flow. In two dimensions, denoting v_x and v_y the local velocity in the x, resp. y directions, the incompressibility condition is written as

$$\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} = 0 \tag{1}$$

and the irrotationality as

$$\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} = 0. \tag{2}$$

As a result, it is possible to introduce both a *velocity potential* and a *streamline function*. The velocity potential is a scalar field ϕ such that

$$\frac{\partial \phi}{\partial x} = v_x \quad \frac{\partial \phi}{\partial y} = v_y. \tag{3}$$

The streamline function ψ, on the other hand, satisfies

$$\frac{\partial \psi}{\partial x} = v_y \quad \frac{\partial \psi}{\partial y} = -v_x. \tag{4}$$

This equation implies that ψ has zero derivative in the flow direction in each point of the flow, i.e. that ψ is constant along streamlines.

From substituting Eqs. (3), (4) into Eqs. (1), (2) it follows that both the velocity potential and the streamline function are harmonic, i.e. they satisfy the Laplace equation:

$$\Delta \phi = \Delta \psi = 0 \tag{5}$$

Since we deal with inviscid flow, we impose the following condition on the airfoil boundary:

$$v_x n_x + v_y n_y = 0, \tag{6}$$

where $\mathbf{n} = (n_x, n_y)$ is the unit outer normal vector to the boundary. In terms of the potential and streamline function, this can be written as

$$\frac{\partial \phi}{\partial \mathbf{n}} = 0, \quad \frac{\partial \psi}{\partial \mathbf{t}} = 0 \tag{7}$$

i.e. the normal derivative of potential and the tangential derivative of the streamline function are zero.

For the following description of elementary potential flows, it is convenient to work with the polar coordinates, r and θ, defined by the standard relations $x = r\cos\theta$ and $y = r\sin\theta$. For a velocity field the transformation is $v_r = v_x \cos\theta + v_y \sin\theta$, $v_\theta = -v_x \sin\theta + v_y \cos\theta$. The elementary potential flow with greatest importance for the following sections is the *two-dimensional potential vortex*, given by the equations:

$$v_r = 0, \quad v_\theta = \frac{\gamma}{r}. \tag{8}$$

Its streamline function is given by

$$\psi = \gamma \log r. \tag{9}$$

The constant γ is sometimes called the *strength* of the vortex. The importance of a potential vortex is that it is fundamentally related to lift generated in two-dimensional flow. The relation is known as Kutta-Joukowski law and states that a potential flow obtained as a superposition of uniform flow and a potential vortex of strength γ generates a force at the vortex axis that is perpendicular to the freestream velocity (the velocity of the uniform flow component) with magnitude L (per unit span) given by

$$L = \rho v_\infty \gamma \tag{10}$$

ρ is the air density, v_∞ is the freestream speed. The lift effect of two-dimensional potential flow on any object is usually expressed by the dimensionless *lift coefficient* c_L:

$$L = \frac{1}{2}\rho v_\infty^2 c_L c, \tag{11}$$

where c is some characteristic length (for airfoils, it is the chord length). Similar definitions are used to measure *drag coefficient* c_D:

$$D = \frac{1}{2}\rho v_\infty^2 c_D c, \tag{12}$$

where D is the drag force per unit span, and *momentum coefficient* c_M:

$$M = \frac{1}{2}\rho v_\infty^2 c_M c^2, \tag{13}$$

where M is the rotational momentum per unit span (related to aerodynamic center described below).

Potential source is another important elementary flow. It is described by equations

$$v_r = \frac{\sigma}{r}, \quad v_\theta = 0. \tag{14}$$

Its streamline function is given by

$$\psi = \sigma\theta. \tag{15}$$

Yet another useful concept is the *aerodynamic center* of an airfoil, which can be defined as the unique point where the momentum is independent of lift, i.e. the point where the lift acts. Hence, for calculating lift, it makes sense to replace an airfoil by a potential vortex located at the airfoil's aerodynamic center. In thin airfoil theory, using a model of infinitely thin airfoil, it can be analytically derived that the aerodynamic center is located at one quarter of the chord (i.e. it divides the chord in 1:3 ratio, being closer to the leading edge). This approximation is widely used in practice for most airfoils, and forms the basis of the lifting line approximation of wing flow, as described in section 4.2. For more details, see [2].

In principle, it is possible to solve the Laplace equation for the velocity potential or the streamline function using finite element or finite difference method. This approach leads to solving a single large sparse system of linear equations, for which efficient methods exist. On the contrary, solving the nonlinear Navier-Stokes equations usually requires many even larger sparse systems to be solved. Still, both approaches are based on direct local approximations to differential equations, and thus lead to systems with number of uknowns proportional to the *area* (or, in 3D, volume) of the problem domain.

Let us consider an arbitrary 2D function ξ defined *outside* a domain Ω, and let the gradient of the function be vanishing at infinity:

$$\nabla\xi(x,y) \to 0 \text{ as } x^2+y^2 \to \infty \tag{16}$$

Let also (x_0, y_0) be an arbitrary point outside Ω. Then, using the Green's third identity (see Appendix), we get:

$$\xi(x_0,y_0) = \frac{1}{2\pi}\oint_{\partial\Omega} -[\xi\,\nabla(\log r) - (\log r)\,\nabla\xi]\cdot\mathbf{n}\,dS \tag{17}$$

where

$$r = \sqrt{(x-x_0)^2+(y-y_0)^2}. \tag{18}$$

Let us see what happens if we substitute the streamline function ψ for ξ in this equation. Eq. (6) says that ψ is constant on $\partial\Omega$. Using again the Green's identity, the first integral then simplifies to a constant independent of x_0:

$$\oint_{\partial\Omega} \psi\,\nabla(\log r)\cdot\mathbf{n}\,dS = \psi\restriction_{\partial\Omega} \tag{19}$$

What remains is the integral

$$\oint_{\partial\Omega} \nabla\psi\cdot\mathbf{n}\log r\,dS. \tag{20}$$

Considering Eq. (9), it is apparent that the integrand of Eq. (20) is a streamline function of a potential vortex. Now, taking the derivatives of Eqs. (19), (20) with respect to x_0, y_0 will yield integral expressions for velocities v_x, v_y at (x_0, y_0).

Together, this shows that if a streamline function can be introduced for a flow in the whole domain with a constant limiting velocity at infinity, then the flow can be expressed as an integral superposition of potential vortices (of infinitesimal strength) along the domain boundary and a uniform flow field. Specifically, it can be shown that the streamline function exists for a simple inviscid incompressible irrotational flow around an airfoil. This result forms the basis for modeling flow by panel method in XFOIL, as described in the following section.

2.2 XFOIL: A panel method coupled with boundary layer model

XFOIL is free software aerodynamic code released under the General Public License. It is a visual tool for visual design and performance evaluation of airfoils. Its interactive nature, while useful for manual design, is actually an obstacle when use

as an automated optimization solver is intended. In the following section, means of overcoming this obstacle are described. The flow solution in XFOIL is based on vortex panel method, coupled with a boundary layer model.

The result in previous section suggests approximating the flow around an airfoil using a distribution of potential vortices along the airfoil surface (boundary), superimposed on an uniform flow given by the freestream velocity.

Figure 1 describes the model of flow used by XFOIL. The inviscid part is approximated by a piecewise linear distribution of potential vortices around the airfoil surface. This piecewise linear distribution is determined by the vorticity values in nodes distributed along the surface, $\gamma_1, \ldots, \gamma_N$. Segments connecting successive nodes are called panels; hence the name panel method. The boundary layer influence is modeled by the so-called "wall transpiration" model, using a piecewise constant distribution of potential sources. The zero normal flux boundary condition is imposed by requiring that the streamfunction be equal at all nodes to some constant ψ_0. Together, this gives a system of the form

$$\sum_{j=1}^{N} a_{ij}\gamma_j + \sum_{j=1}^{N+N_w-2} b_{ij}\sigma_j = \psi_0 + v_{x\infty}y_i + v_{y\infty}x_i, \tag{21}$$

where the influence coefficients a_{ij}, b_{ij} are determined from unit streamfunctions of a potential vortex and potential source, as given by Eqs. (9), (15).

The Kutta condition is imposed by requiring that $\gamma_1 + \gamma_N = 0$. A panel is also placed on the trailing edge if it has a nonzero thickness; the vorticity and source strength on this panel are determined by γ_1, γ_N. Another chain of panels, but with no vorticity, only sources, is used to model the wake.

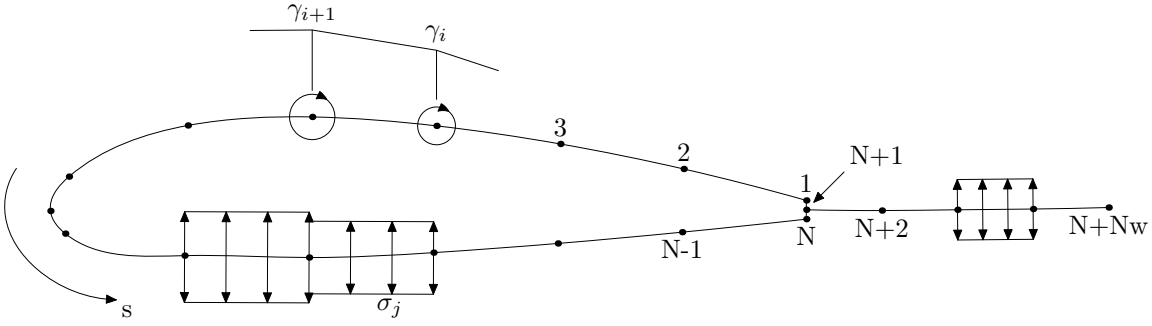


Figure 1: XFOIL flow model

The boundary layer model employed in XFOIL is quite complicated and its description is beyond the scope of this thesis. An interested reader may find more details in [3] and [4]. Let us just briefly summarize the nature of the method here:

The model is based on standard integral momentum and kinetic energy shape parameter equations for the boundary layer, forming a system of two ordinary differential equations. Empirical functional dependencies are assumed amongst the relevant boundary layer quantities to close the equations. Different dependencies are assumed for laminar and turbulent boundary layer region, respectively, giving rise to two different systems of equations. The transition points are determined using the e^9 method (see again the above mentioned papers [3] and [4] for further references).

The coupling between the inviscid potential model and the viscous boundary layer model is realized through the tangential velocity, u_e, which, taking into account Eq. (4) is assumed equal to $\nabla\psi \cdot \mathbf{n}$ on the suction side of the airfoil, and $-\nabla\psi \cdot \mathbf{n}$ on the pressure side of the airfoil. According to Eq. (20), in a control point on the airfoil surface we get the equations

$$\pm u_{ei} = \gamma_i + v_{x\infty} n_{xi} - v_{y\infty} n_{yi}, \tag{22}$$

where n_x, n_y are components of a unit normal vector of the airfoil. In the wake, there is no such simple relation, it is necessary to express the dependence of $\nabla\psi \cdot \mathbf{n}$ on the freestream and all vortex and source elements, to end up with a linear system

$$u_{ei} = v_{x\infty} n_{xi} - v_{y\infty} n_{yi} + \sum_{j=1}^{N} c_{ij}\gamma_j + \sum_{j=1}^{N+N_w-2} d_{ij}\sigma_j. \tag{23}$$

The influence coefficients c_{ij}, d_{ij} are again evaluated using the equations by Eqs. (8), (14). The normal direction to the wake is oriented so as to match the normal direction on the suction side of the airfoil.

3 Airfoil optimization

Airfoil optimization is usually used as a preliminary step to wing optimization. Given the relation to wings of infinite span (cf. Sec. 2, page 9), it is reasonable to expect that good aerodynamic performance of an airfoil will be reflected in good properties of a wing having this airfoil as a cross section.

In practice, usually when we speak of airfoil optimization, we can have two different flavors of the problem in mind: a refinement optimization or an explorative optimization. Refinement means the process starts from a baseline design, usually an existing industrial airfoil, and applies relatively small perturbations to its shape to optimize the desired objective. Since standard industrial airfoils are already well optimized with respect to typical flight conditions, only small deviations in both objectives and shape are expected. By "explorative" optimization, on the other hand, we mean a process where there is no baseline design, and the objectives are

such that standard airfoils are not close to satisfactory. In such case, we expect to search with less detail but broader scope, to cover a wide range of airfoil shapes. An explorative stage is often followed by a refinement.

While equivalent in pure theory, in practice explorative and refinement optimization differ fundamentally. The focus of this section will be on the explorative optimization which is less common and less understood.

3.1 Multi-objective optimization

We begin with the simplest mathematical formulation of a general optimization problem. That is, to find

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{24}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is a real function defined on n-dimensional vectors of real numbers, called *objective function*. Usually, we assume at least continuity of f, often first- or even second-order differentiability is required. This type of optimization problem is called *unconstrained optimization*. Equivalently, we may want to maximize f, but that is only a matter of convention, as maximizing f equals minimizing $-f$. The standard form of an optimization problem is minimization.

In real situations, the classical formulation of optimization problem, as introduced in the previous paragraph, is often too simplistic. For instance, when we optimize a wing section, we are not just interested in low drag; we also want high lift, low momentum, stability at high angles of attack, etc. Mathematically, we speak of *multi-objective* optimization. Instead of a single objective function F, we define a vector-valued function F, assigning to each design vector x an *objective vector* $f = F(x)$. The set of all possible objective vectors determines the *design space*. Unlike single-objective optimization, the solution of a multi-objective optimization problem is not a single design vector, but rather a set of so-called *non-dominated* or (*Pareto-non-dominated*) vectors, often called the *Pareto front*. The definition of this concept is as follows: Given two objective vectors, $\mathbf{f}^1$ and $\mathbf{f}^2$, we say that $\mathbf{f}^1$ dominates $\mathbf{f}^2$ if it holds

$$f_i^1 \leq f_i^2 \qquad \text{for all } i = 1 \dots D, \tag{25a}$$

$$f_i^1 < f_i^2 \qquad \text{for at least one } i = 1 \dots D. \tag{25b}$$

This is also called *strong dominance*. If we omit the condition (25b), we speak of *weak dominance*.

The Pareto front is simply the set of all objective vectors that are not dominated by any other vector. It constitutes a $(D-1)$-dimensional manifold in the objective space. Alternatively, we may speak of the set of the corresponding design vectors,

which is also a $(D-1)$-dimensional manifold, but in design space. The distinction is usually clear from context.

In theory, the solution of a multi-objective optimization problem is the Pareto front. In practice, however, individual solutions need to be picked from the Pareto front according to some *secondary objectives*. But unlike single-objective optimization based on some reduction of the multiple objectives (e.g., a weighted sum), the true multi-objective approach allows us to select *a posteriori*, i.e. we can use the knowledge of the shape of the true Pareto front in our decision. For more details, see [5].

3.2 Airfoil parametrization

When dealing with an airfoil optimization problem (or design optimization in general), we seek an optimum design from the set of all admissible designs, usually called the *design space*. Ideally, we would want to consider all airfoils. The set of all possible airfoil shapes forms an infinite-dimensional affine space; working with such a complex space is not practical. Since optimization algorithms are usually abstracted to the form given by Eq. (24), we need to constitute a mapping that maps vectors from $\mathbb{R}^D$ to airfoil shapes. This mapping is usually called *parametrization*. Parametrization inevitably constitutes a set of parametrized designs, which is a subset of all possible designs. Objective functions, however, are defined on airfoils directly - lift, drag, thickness and other aerodynamic or geometric properties. By choosing a parametrization for our optimization problem, we essentially form a new objective function (or functions) defined on $\mathbb{R}^D$.

It is, therefore, not surprising that the choice of parametrization has a fundamental impact on the optimization problem and its results.

In a refinement optimization, usually some flexible general parametrization of the shape difference is used. If, however, an explorative optimization is intended, with a novel combination objectives, often there is no baseline design to start from. Indeed, in such case, we want our parametrization to yield a wide range of airfoil shapes. Opposing to flexibility, however, there is the complexity of the parametrization, given by number of parameters. A more complex parametrization yields a more dimensional design space, that is more difficult to search for any general optimization algorithm. In the explorative optimization case, we want our parametrization to be a good compromise between simplicity and flexibility, to yield a wide range of varying airfoil shapes using a moderate number of variables.

Of course, best results can be achieved using a parametrization specially tailored to a problem, incorporating prior knowledge about the problem. The GPARSEC parametrization, described in the next subsection, is an example of such a specific-purpose parametrization designed for subsonic and transonic airfoils.

3.3 GPARSEC

Airfoil sections are traditionally and most easily described by coordinates of sample points on the airfoil. For the purposes of optimization, however, a simpler description of the airfoil shape is usually desirable. Various methods can be employed for this purpose, e.g. the extended Joukowski transformation [6], B-spline curves [7] or Hicks-Henne shape functions [8]. This work generalizes the PARSEC parametrization described in [9]. PARSEC differs from the above mentioned methods in the aspect that the design variables are real important geometric characteristics of the airfoil shape, which provides easier control over the generated airfoil shapes. The scheme is generalized to include arbitrary base functions and extended to provide more degrees of freedom. Further, several variants of the scheme are employed in a real airfoil optimization problem with four objectives to show that much better results than with the original PARSEC can be obtained.

3.3.1 Description of basic GPARSEC

The generalized PARSEC (GPARSEC) parametrization assumes the leading edge of the airfoil in the origin of two-dimensional plane and the trailing edge on the line $x = 1$. The upper (U) and lower (L) airfoil surfaces are given by the equations:

$$z_U = \sum_{i=1}^{6} a_{iU}\psi_{iU}(\sqrt{x}), \qquad z_L = \sum_{i=1}^{6} a_{iL}\psi_{iL}(\sqrt{x}). \tag{26}$$

Here ψ_{iU}, ψ_{iL} are suitable base functions with continuous second derivative on $(0, 1)$, satisfying the conditions:

$$\psi_{iU}(0) = \psi_{iL}(0) = 0. \tag{27a}$$

$$\psi_{iU}(1), \psi_{iL}(1), \psi'_{iU}(0), \psi'_{iL}(0), \psi'_{iU}(1), \psi'_{iL}(1) \text{ are finite.} \tag{27b}$$

The design variables for GPARSEC are given by 12 *geometric characteristics* of the airfoil shape, as described by Fig. 2. These geometric characteristics express the upper and lower radius of curvature of the leading edge r_{LELO}, r_{LEUP}, the positions of extrema of both surfaces X_{UP}, X_{LO}, Z_{UP}, Z_{LO}, the reciprocal curvatures (second derivatives) in these extrema Z_{XXLO}, Z_{XXUP}, the trailing edge angle β_{TE}, distortion α_{TE}, z-coordinate Z_{TE} and thickness ΔZ_{TE}.

Given the geometric characteristics, the GPARSEC coefficients a_{iU}, a_{iL} satisfy

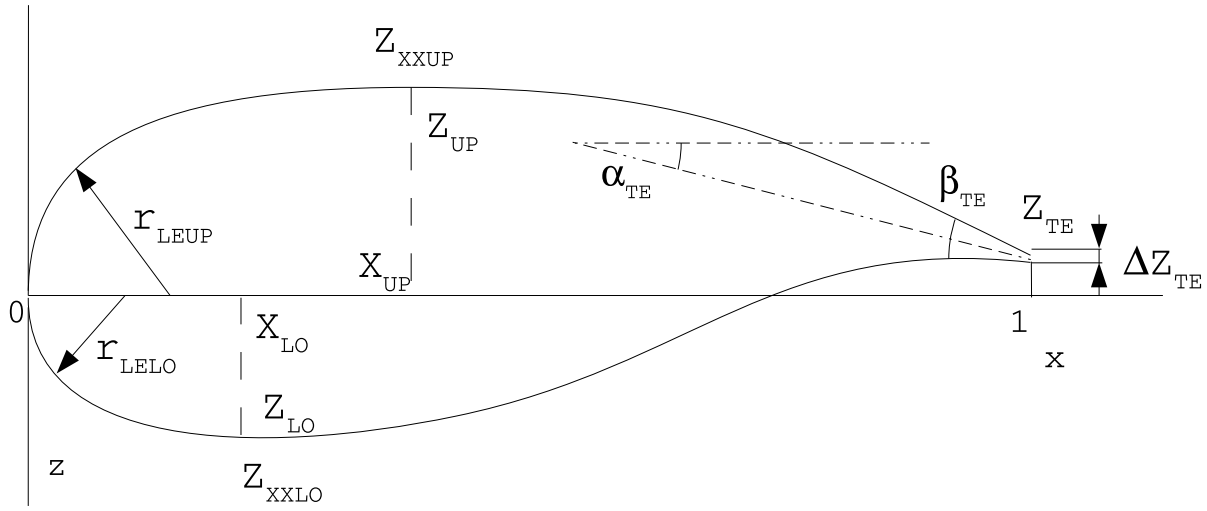


Figure 2: Design variables for GPARSEC

the following equations:

$$\sum_{i=1}^{6} \psi'_{iU}(0)\, a_{iU} = r_{\mathrm{LEUP}}, \tag{28a}$$

$$\sum_{i=1}^{6} \psi_{iU}(\sqrt{X_{\mathrm{UP}}})\, a_{iU} = Z_{\mathrm{UP}}, \tag{28b}$$

$$\sum_{i=1}^{6} \psi'_{iU}(\sqrt{X_{\mathrm{UP}}})\, a_{iU} = 0, \tag{28c}$$

$$\sum_{i=1}^{6} \psi''_{iU}(\sqrt{X_{\mathrm{UP}}})\, a_{iU} = 4 Z_{\mathrm{XXUP}} X_{\mathrm{UP}}, \tag{28d}$$

$$\sum_{i=1}^{6} \psi_{iU}(1)\, a_{iU} = Z_{\mathrm{TE}} + \Delta Z_{\mathrm{TE}}/2, \tag{28e}$$

$$\sum_{i=1}^{6} \psi'_{iU}(1)\, a_{iU} = -2 \tan(\alpha_{\mathrm{TE}} + \beta_{\mathrm{TE}}/2), \tag{28f}$$

$$\sum_{i=1}^{6} \psi'_{iL}(0)\, a_{iL} = -r_{\mathrm{LELO}}, \tag{29a}$$

$$\sum_{i=1}^{6} \psi_{iL}(\sqrt{X_{\mathrm{LO}}})\, a_{iL} = Z_{\mathrm{LO}}, \tag{29b}$$

$$\sum_{i=1}^{6} \psi'_{iL}(\sqrt{X_{\mathrm{LO}}})\, a_{iL} = 0, \tag{29c}$$

$$\sum_{i=1}^{6} \psi''_{iL}(\sqrt{X_{\mathrm{LO}}})\, a_{iL} = 4 Z_{\mathrm{XXLO}} X_{\mathrm{LO}}, \tag{29d}$$

$$\sum_{i=1}^{6} \psi_{iL}(1)\, a_{iL} = Z_{\mathrm{TE}} - \Delta Z_{\mathrm{TE}}/2, \tag{29e}$$

$$\sum_{i=1}^{6} \psi'_{iL}(1)\, a_{iL} = -2 \tan(\alpha_{\mathrm{TE}} - \beta_{\mathrm{TE}}/2). \tag{29f}$$

These two systems are square, and the a_{iU}, a_{iL} coefficients are fully determined. In this case, it is also possible to easily compute the derivatives of the coefficients with respect to the design variables, and, consequently, the sensitivity of the airfoil shape.

This scheme allows the construction of a wide family of parametrization by choosing various base functions satisfying the conditions (27a, 27b). Regardless of the choice of these functions, the shape of the airfoil is controlled by the twelve geometric characteristics described above. As has already been suggested, the principal advantage of this approach is that if convenient base functions are employed, design variables in reasonable boundaries always create meaningful airfoils, yet, at the same moment, very distinct shapes can be obtained. GPARSEC is thus especially suitable for global airfoil optimizations, when there is no initial or baseline shape to be perturbed. This is often true in multi-objective optimization. Another advantage is that necessary geometric constraints (such as minimum trailing edge thickness) can be directly expressed or approximated by simple constraints on design variables. This property can simplify the optimization process.

3.3.2 Choice of base functions

For the optimization problem presented in section 3.6, three sets of 12 base functions (6 for each surface) are considered:

1. The `sobieczky` set, according to the original PARSEC ([9]):

$$\psi_{iU}(t) = \psi_{iL}(t) = t^{2i-1}, \tag{30}$$

2. The `vzlu2` set, consisting of trigonometric functions,

3. The `vzlu3` set, consisting of polynomials.

The choice of base functions for GPARSEC can be a matter of experiment. The base functions should be sufficiently independent to produce reasonable airfoil shapes for most GPARSEC geometric characteristics. In particular, the linear systems (28a-28f), (29a-29f) should not become too ill-conditioned. The left-hand side matrices of these systems depend only on the variables X_{UP}, X_{LO}. It is, thus, a good idea to inspect the dependence of the condition number of these matrices on X_{UP}, X_{LO} in advance, in order to grant reasonable behavior in the area of interest (e.g. $0.1 \leq X_{\mathrm{UP}} \leq 0.5$).

An interesting option is employing other airfoil sections as base functions. In this way, we could obtain a parametrization that can exactly represent selected airfoil sections of particular interest; therefore, one might expect that the resulting parametrization could inherit the properties of these airfoils to some extent. This idea has not yet been experimentally tested, however.

3.3.3 Limitations of GPARSEC

The principal limitation of GPARSEC is that leading edge of the airfoil is assumed at the origin, while the chord should be situated approximately along the x-axis. For certain types of airfoils (e.g. strongly curved turbine blades), these assumptions are unsatisfiable and such airfoils are thus unsuitable for GPARSEC. An example of such an airfoil is given in Fig. 3. Another issue with GPARSEC might be its limited number of design variables. When GPARSEC is included in more complex parametrization schemes for whole wings or airplanes, which typically contain several airfoils, this is more likely an advantage. But for a pure airfoil optimization, especially when restricted to symmetric airfoils (in which case the effective number of GPARSEC design variables equals six), more degrees of freedom might be desired. This is addressed in the following section.

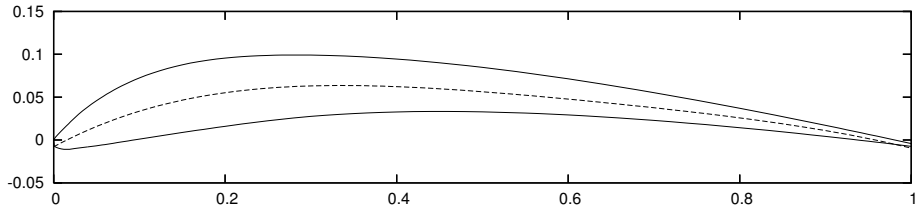


Figure 3: An airfoil unsuitable for GPARSEC

3.3.4 Extension to more DOFs

The basic 12 degrees of freedom may be insufficient for some applications. Therefore, it is desirable to have a way of extending the basic GPARSEC scheme with more design variables if this proves necessary, while not losing the ability to directly control the geometric characteristics. To achieve this goal, a modification of the Hicks-Henne shape functions [10] are employed and four types of additive bump functions are introduced:

$$H_{FU}(x,\beta) = H(\frac{\sqrt{x}}{\sqrt{X_{\mathrm{UP}}}},\beta) \qquad H_{BU}(x,\beta) = H(\frac{1-\sqrt{x}}{1-\sqrt{X_{\mathrm{UP}}}},\beta) \tag{31}$$

$$H_{FL}(x,\beta) = H(\frac{\sqrt{x}}{\sqrt{X_{\mathrm{LO}}}},\beta) \qquad H_{BL}(x,\beta) = H(\frac{1-\sqrt{x}}{1-\sqrt{X_{\mathrm{LO}}}},\beta) \tag{32}$$

where

$$H(t,\beta) = \begin{cases} \sin^3(\pi t^\beta) & \text{for } 0 \le t < 1 \\ 0 & \text{for } t \ge 1 \end{cases} \tag{33}$$

and $\beta > 1/3$ is a parameter that controls the position of the maximum of the bump. The key property of these functions is that adding small multiples of H_{FU}, H_{BU} to the upper airfoil surface or H_{FL}, H_{BL} to the lower airfoil surface does not interfere with any of the GPARSEC geometric characteristics of the airfoil shape (i.e., these are preserved). It is thus possible to easily augment the basic GPARSEC scheme which controls the main geometric characteristics of the airfoil with these additive bump functions that provide fine tuning of the airfoil. This can be referred to as the extended GPARSEC parametrization.

3.4 Genetic algorithms

Genetic optimization algorithms are a class of optimization algorithms based on random generation and recombination of design vectors, called individuals. They typically maintain a set of individuals called a population; using random generation and recombination "operators" (mutation, crossover), and comparing according to the objective (fitness), the population is let to evolve towards the most promising individuals, corresponding to optimal designs.

Compared to local search methods, genetic algorithms typically lack the strong local convergence properties; indeed, they require much more objective evaluations when applied to nice quadratic-like functions. Their strength, however lies in their robustness, i.e. their ability to overcome local minima, failed evaluations, and noisy objectives.

3.4.1 The μ-ARMOGA algorithm

The μ-ARMOGA algorithm is a micro-genetic algorithm targeted at multi-objective optimization using small populations and an elitist Pareto archive. Pareto archive is a set of non-dominated individuals.

A simplistic description of the algorithm follows: In each cycle, an initial population is randomly initialized using the population statistics (see below). Afterwards, selected members from the Pareto archive are injected into the population. Then, crossover is applied to the population, combining the existing vectors in pairs and replacing them. The individuals of the updated population are then evaluated for objectives. The evaluated individuals are used for possibly updating the Pareto archive, and (once per several generations), they are also used to update the population statistics. Population statistics is a nonlinear transformation used to focus the evolution to a certain area of the design space. For each design variable x_i, $i = 1, \ldots, D$, we introduce the mean μ_i and deviation σ_i, and define the trans-

formed variable $\hat{x}_i$ by

$$\hat{x}_i = \frac{2}{\sqrt{\pi}} \int_0^{x_i} e^{-(u-\mu_i)^2/(2\sigma_i^2)} du \tag{34}$$

The transformed variables satisfy $-1 < \hat{x}_i < 1$. Note that some authors use an alternative transformation

$$\hat{x}_i = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{x_i} e^{-(u-\mu_i)^2/(2\sigma_i^2)} du \tag{35}$$

so that $0 < \hat{x}_i < 1$. The former approach has the advantage that the center is around zero, where machine numbers have finer resolution. The transformed variables are used instead of the true design variables in evolution operations such as crossover and reinitialization. Inverse transformation is applied to the newly created design vectors to get the true (untransformed) design variables suitable for evaluation. The mechanism for updating the population statistics is called *adaptation*.

The above basic descriptions allows a number of further choices and modifications. The precise way how to do crossovers, adaptation and how to inject archive members into the resulting population is something that has no fixed optimum and can vary from problem to problem.

The μ-ARMOGA algorithm has been implemented in VZLU in the ARMOGA software. A new, improved implementation is being worked on. In the next section, one of the novel features of the μ-ARMOGA algorithm is described in detail.

3.4.2 Updating the Pareto archive

As has been stated above, a key component of our evolutionary algorithm is the *Pareto archive*. It acts as a collector of promising individuals during the evolution, and is also used to actually yield the resulting Pareto front approximation when the algorithm terminates. The archive is informed each time new population are evaluated, optionally allowed to store its individuals if they are non-dominated. Information is retrieved back from the archive by injecting archive members into the reinitialized population, prior to crossovers. This simple interaction makes it possible to try various archiving strategies and to study them in their own right. In this section, a novel strategy for updating the Pareto archive in a (micro-)evolutionary algorithm is briefly described. The details are described in [12].

In a real updating strategy, putting a limit on the number of archive members is needed, not only to keep the complexity under control, but also to achieve a good diversity of the final approximation of the Pareto front. In the presented strategy, a fixed upper limit on the number of Pareto archive members is used. The aim is to terminate the algorithm with an archive of Pareto-optimal solutions that is "well spread" over the true Pareto front of the problem. This approach is dealing with

a single new individual at a time. This makes it suitable for micro-evolutionary approaches, where only several new individuals appear in each generation.

When a new individual arrives, it is first checked for Pareto dominance with all existing members of the archive. Then, the following three cases are distinguished:

- The new individual is dominated by one or more individuals from the archive. In this case, the new individual is discarded.
- The new individual dominates one or more individuals in the archive. The dominated individuals are removed, and the new individual is added to the archive and the internal information of the archive is updated (see below).
- The new individual is non-dominated and non-dominating. If the number of individuals in the archive has not yet reached the upper limit, the new individual is added as in the previous case. In the opposite case, at least one individual needs to be discarded (either the newcomer or one from archive), but this can not be decided by Pareto dominance. That is when the algorithm proceeds to the secondary decision procedure, described below.

When the case that can not be resolved by Pareto dominance occurs, the secondary goal is to maximize the distance between neighboring individuals, based on some distance measure in the objective space. In this thesis, the standard Euclidean distance is used.

First, consider the minimum pairwise distance, i.e.

$$\min_{\substack{i,j\in P\\ i\neq j}} \|\mathbf{f}_i - \mathbf{f}_j\|, \tag{36}$$

where P denotes the set of archived individuals and $\mathbf{f}_i$ stands for the vector of objective values of individual i. Take the pair of individuals that achieves the minimum in the above expression. If there are multiple such pairs, take any of them. Without loss of generality, assume that the minimum pair is $\mathbf{f}_1, \mathbf{f}_2$. Further, denote $\mathbf{g}$ the vector of objective values of the new individual. If

$$\min_{\substack{k\in P\\ k\neq 1}} \|\mathbf{f}_k - \mathbf{g}\| \geq \|\mathbf{f}_1 - \mathbf{f}_2\|, \tag{37}$$

one can replace $\mathbf{f}_1$ by $\mathbf{g}$. Alternatively, if

$$\min_{\substack{k\in P\\ k\neq 2}} \|\mathbf{f}_k - \mathbf{g}\| \geq \|\mathbf{f}_1 - \mathbf{f}_2\|, \tag{38}$$

one can replace $\mathbf{f}_2$ by $\mathbf{g}$. If either of the above conditions is satisfied, the overall minimum pairwise distance will be improved by the substitution or, if there were

multiple minimal pairs, it will stay the same but the number of minimal pairs will reduce. Let us call this the *global improvement* check.

If neither of these conditions is satisfied, consider the closest archived individual to $\mathbf{g}$, say, $\mathbf{f}_c$ instead. If

$$\min_{\substack{k\in P\\ k\neq c}}\|\mathbf{f}_k-\mathbf{g}\| > \min_{\substack{k\in P\\ k\neq c}}\|\mathbf{f}_k-\mathbf{f}_c\|, \tag{39}$$

replace $\mathbf{f}_c$ by $\mathbf{g}$. If this condition holds, there is a certain subset of the archived individuals whose pairwise minimum will improve. This is the *local improvement* check.

If neither check is successful, the new individual is discarded.

Searching for the minimum-distance pair of the archive afresh each time an individual is considered would be too costly. To make the procedure efficient, it is necessary to maintain for each archived individual a pointer to its closest neighbor (or any of them). Thus, searching for the pairwise minimum in Eq. (36) requires only one pass through the archive. Similarly, the right-hand side of Eq. (39) is simply the distance of $\mathbf{f}_c$ to its closest neighbour. Thus, these two checks only require computing the distances of the new individual to all archived individuals, and computing the minima on left-hand sides of Eqs. (37),(38),(39). Thus, *deciding* whether to add a new individual has linear complexity in terms of number of archived individuals (evaluating mutual pairwise dominance is also linear).

If the new individual is to be added, the existing closest-neighbor links need to be updated. Each resulting archive member is considered in turn. If the link is valid (i.e. the closest neighbor in the archive was not discarded), we simply check if the newcomer is closer, and possibly update the link. This takes only constant time. However, if the link became invalid (the former closest neighbor was discarded), the closest neighbor needs to be computed afresh by computing objective distances of the updated individual to all others.

This case is experimentally analyzed in [11], where it is asserted that in real runs, only a few invalid links occur on average per update. Hence, the algorithm runs in linear time in practice.

3.5 Practical aspects

Let us now consider some practical aspects of an optimization process by a micro-genetic algorithm, i.e. a genetic algorithm exploiting small populations. In particular, we will be concerned with a key parameter of any micro-genetic algorithm, the micro-population size.

First, we will note that there are at least three ways to measure an algorithm's performance, i.e. speed of convergence. First, we can count the number of objective

evaluations (i.e. candidate vectors) needed to achieve a certain progress; second, we can count the populations needed; and third, measure the actual time consumed by the computation. In general, the convergence speed measured by number of populations will increase with the micro-population size, because more information is gained per population. Measured by number of evaluations, however, the performance will tend to degrade with increasing population size, because less information is shared, on average, between successively evaluated individuals. To describe the second phenomenon more closely: if a population of size N is split into two subpopulations of size $N/2$, then the information from the first subpopulation can be used for better selection of the second subpopulation, thus leading to a faster convergence in terms of number of evaluated individuals.

From practical point of view, of course, the actual time taken by the optimization process is the most important quantity. Let us for a moment assume that evaluating any individual takes up a constant time on a single processor; and that we have M processors available for parallel evaluation. Using a similar reasoning as in the previous paragraph, we can conclude that the optimal population size is M (or the minimal population size required by the optimization algorithm if that is greater than M). Indeed, if N is the population size, then if $N < M$ we can supply the remaining $M - N$ individuals at random, providing additional information for the algorithm. Based on our assumption that the algorithm can utilize information effectively, it is apparent that population size M is at least as good as that of size M. Conversely, if $N > M$, let us consider evaluating MN successive individuals

1. using M populations of size N,
2. using N populations of size M.

In a genetic algorithm, the information available for selecting an individual is that of all previously evaluated populations. Considering the above two schemes, denoting k_M, k_N, the number of individuals providing information for selecting the k-th individual from the first and second scheme, respectively, we see that k_M is the greatest multiple of M less than k, and k_N is the greatest multiple of N less than k; thus, $k_M \geq k_N$ for all k, and $k_M > k_N$ for some of them. Again, based on our assumption of efficient utilization of information by the algorithm, we conclude that a population of size M provides at least as good performance as size N. The final conclusion is that in the case of constant-time evaluation of each individual, the optimal micro-population size is equal to the number of available processors.

In practice, the situation is more difficult, because time consumed by evaluating an individual may vary. Generally, when a population of size M is evaluated using N parallel processors, we can define the *efficiency index* as

$$\eta = \frac{\sum_m t_m}{N \max_n T_n}. \tag{40}$$

Here t_m denotes the time consumption by the m-th job, and T_n is the time to complete the n-th process, computed as

$$T_n = \sum_{i \in I_n} t_i + \text{ scheduling overhead}, \tag{41}$$

where I_n denotes the set of job indices assigned to the n-th processor.

Using a similar reasoning as above, we can show that if $\eta = 100\%$, for a certain $M = M_0$, the optimum population is at most M_0, i.e. using population size greater than M brings no additional benefit. On the other hand, it is apparent that if the time taken to evaluate an individual is variable, $M = N$ will generally give $\eta < 100\%$. The situation is easily to visualize: individuals are distributed evenly amongst processors, and one individual that is being evaluated slowly blocks the whole process, while the rest of the population is already evaluated. When $M > N$, one possibility how to distribute the M jobs amongst the processors is to split them into N equal (or approximately equal) groups and assign each group to a single processor. In parallel computing, this approach is usually referred to as *static scheduling*. On the contrary, *dynamic scheduling* loops sequentially through the jobs, waiting until a processor becomes idle; the job is then assigned to the processor and the cycle proceed to the next job. It is apparent that dynamic scheduling will tend to yield better efficiency indices, due to the broader ability to balance processor loads. However, dynamic scheduling is also harder to implement, because it requires interprocess communication.

Parallel automatic evaluation of airfoils using dynamic scheduling has been implemented in VZLU in the XFEVAL software, for the purpose of solving multi-objective aerodynamic optimization problems using the μ-ARMOGA algorithm. XFEVAL provides an encapsulation for XFOIL, which does not itself support batch or parallel processing. The requested aerodynamic characteristics (computable by XFOIL) are specified using a simple "evaluation scripts". Examples include lift at certain angle of attack, drag at certain lift (or angle), momentum, or maximum lift in a certain range of angles of attack. XFEVAL works by forking a requested number of threads, feeding new designs to idle threads and collecting results. Each thread repeatedly requests designs from the main thread and evaluates them. During the process, XFEVAL calculates the evaluation efficiency as described in the previous paragraphs. Experiments with utilizing the computed efficiency for adapting the population size are currently in progress.

3.6 Example: Hybrid regime airfoil optimization

Using the techniques described in this section, we solved a multi-objective optimization problem, optimizing an airfoil shape for a rear wing section of a small business

aircraft. Eventually, two utility models were produced from solving this problem: [13] and [14].

The objectives were given as follows:

- Flight regime (`obj1`): minimize C_D at $\alpha = 0°$, $Re = 6 \cdot 10^6$, $M = 0.3$.
- Maneuvering regime (`obj2`): minimize C_D at $\alpha = 5°$, $Re = 6 \cdot 10^6$, $M = 0.3$.
- Takeoff and landing with a side wind (`obj3`): maximize $C_L^{\max}$ at $Re = 2e6$, $M = 0.12$.

Here, Re denotes the Reynolds number, and M denotes the Mach number of the flow. All of these objectives can be considered either with a free transition of laminar boundary layer (denoted `obj1x`, `obj2x`, `obj3x`), or with forced transition at 7% of the chord (denoted `obj1y`, `obj2y`, `obj3y`).

We solved three optimization problems: one with the above objectives with free transition, one with forced transition, and one with both free and forced transition (giving a total of six objectives). These are called the laminar, turbulent, and hybrid case, respectively.

The airfoils were parametrized using the GPARSEC parametrization. The optimization has been carried by the μ-ARMOGA algorithm. The populations of candidate designs were transformed to coordinate representation by the GPARSEC utilities, and then efficiently evaluated by means of the XFEVAL evaluator. We have experimented with various settings of the μ-ARMOGA algorithm, including the population size, which was adjusted according to the evaluation efficiency index calculated by XFEVAL, to ensure full loading of 8 processors used in parallel.

Figure 4 shows the resulting Pareto front of the free transition problem optimization problem, Figure 5 shows the Pareto front of the forced transition problem. For the hybrid problem, two projections of the (six-dimensional) Pareto front are shown: projection on the free trans. objectives in Fig. 6 and projection on the forced trans. objectives in Fig. 7.

4 Wing flow solution

4.1 3D potential flow

The potential (or locally potential) flow model can also be applied in 3 dimensions. To facilitate studying complex vortex structures that can arise in 3-dimensional flows, it is useful to introduce the notion of vorticity (rotation)

$$\omega = \operatorname{rot} \mathbf{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right). \tag{42}$$

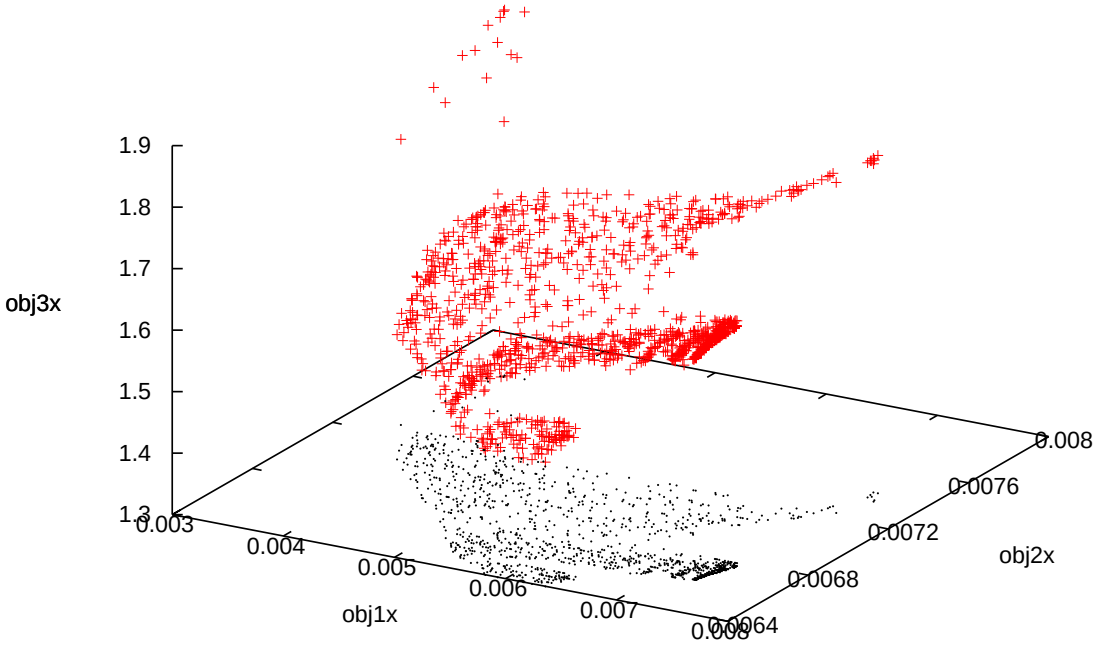


Figure 4: Pareto front of the free transition problem

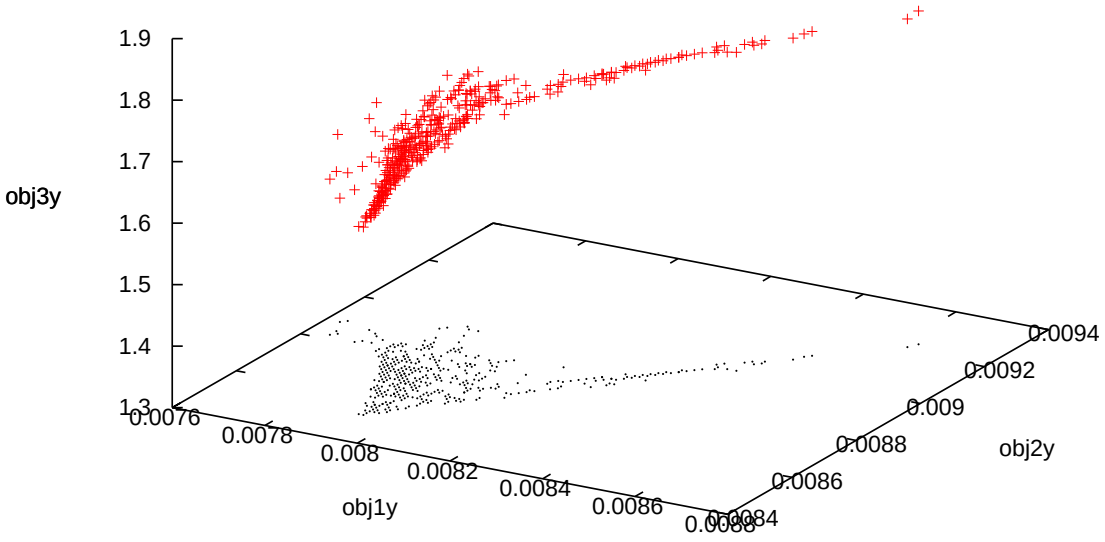


Figure 5: Pareto front of the forced transition problem

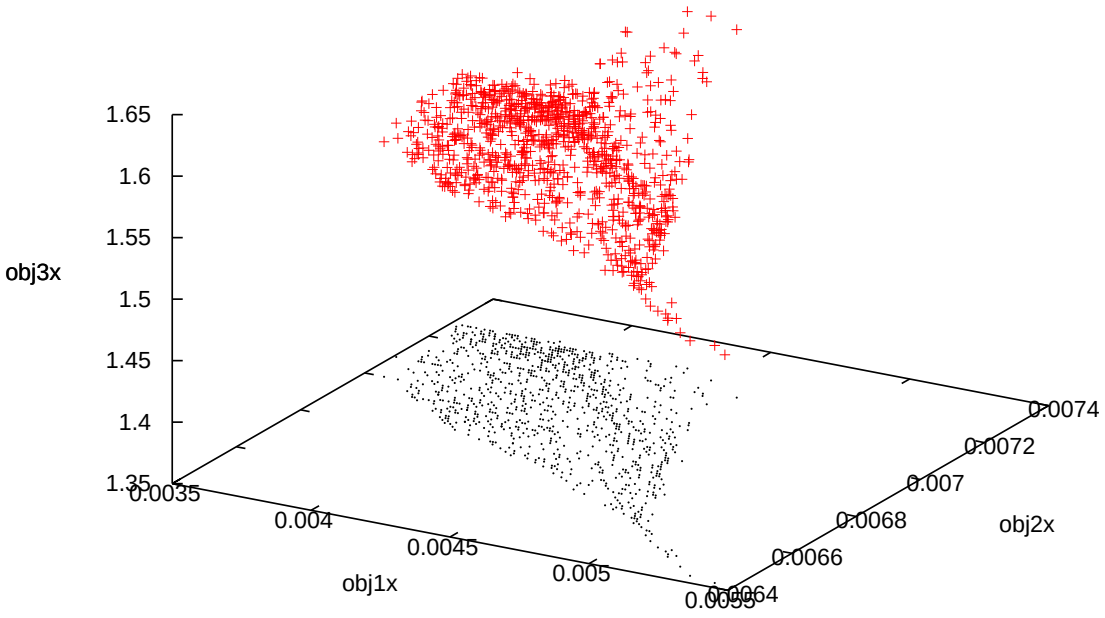


Figure 6: Pareto front of the hybrid problem - free trans. projection

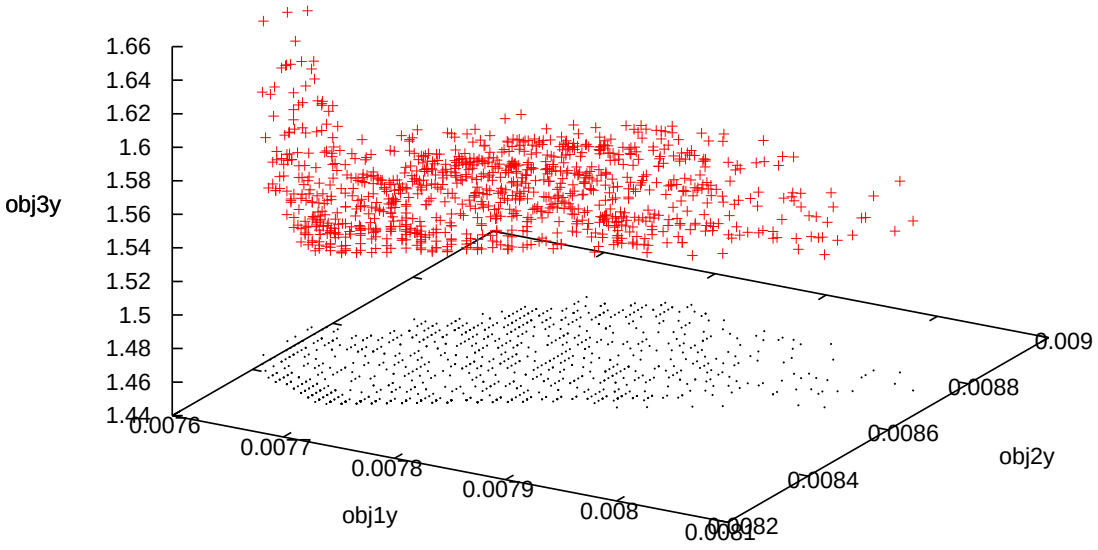


Figure 7: Pareto front of the hybrid problem - forced trans. projection

Zero vorticity in a certain domain means a locally potential flow in that domain. The vorticity is fundamentally related to circulation (around a closed curve) by the Kelvin-Stokes theorem:

$$\oint_{\partial A} \mathbf{v} \cdot d\mathbf{s} = \int_A \operatorname{rot} \mathbf{v} \cdot d\mathbf{A}. \tag{43}$$

In other words, normal flux of vorticity through a surface A is equal to the circulation along its boundary ∂A. This also implies that the vorticity is a conservative quantity, i.e. that $\operatorname{div} \operatorname{rot} \mathbf{v} = 0$. By *vortex tube* we usually denote the flow that is irrotational everywhere outside a thin tube. Using the Kelvin-Stokes theorem and the conservativity of vorticity, we can easily conclude that the normal flux of vorticity through any cross-section of the tube is constant, and around any simple closed curve encircling the vortex tube there is a constant circulation. A limiting case, vortex tube of infinitesimal thickness, is called a *vortex thread* or *vortex filament*. Again, there must be a constant (nonzero) circulation around the filament, no matter what curve is used. This also means that a vortex thread cannot have free endpoints inside the fluid. More complex systems of vortex filaments with branches and crossings are possible, however; such systems must satisfy the Kirchhoff's law: the amount of circulation incoming to a node must equal to the total circulation coming out.

The computational advantage of using vortex filament systems lies in the fact that the local velocity of the locally potential flow determined by the vortex system S, usually called the *induced velocity*, can be computed using the *Biot-Savart law*:

$$\mathbf{v}(\mathbf{x}_0) = \frac{\Gamma}{4\pi} \int_S \frac{\mathbf{x} - \mathbf{x}_0}{\|\mathbf{x} - \mathbf{x}_0\|^3} \times d\mathbf{x}. \tag{44}$$

In the case when S is composed of segments, lines and half-lines, the integrals corresponding to the components can even be evaluated analytically (see Appendix).

4.2 Prandtl's lifting line model

Armed by the theory of three-dimensional potential vortices presented in the previous section, we consider Prandtl's lifting line model of the wing. The fact that in two-dimensional flow, airfoil can be approximated by a potential vortex leads naturally to an attempt to approximate a wing by a vortex segment stretched along the span. The Helmholtz's law, however, prohibits the existence of an isolated vortex segment. Following real-life observations, Prandtl modeled the wing by a horseshoe vortex, consisting of one segment along the wing span and two half-lines carrying the vorticity to infinity in the freestream direction. Due to Helmholtz's law, such a model yields constant circulation along the span. Prandtl thus later refined his model to a distribution of half-line vortices along the span and variable distribution

of vorticity on the bound segment. The free vortex half-lines actually model the *wake* behind the wing, which is observed in real flows.

Prandtl considered continuous distribution, yielding an integral equation (Prandtl's integral equation) that can be solved numerically:

$$\frac{\Gamma(y)}{c(y)} = \frac{1}{4}\int_{-b/2}^{b/2} \frac{1}{y-\bar{y}}\,\mathrm{d}\Gamma(\bar{y}) \tag{45}$$

Similarly as before, we will consider instead an early discretization scheme, using discrete distribution of vortices in the wake. As we already know, every valid vortex system in three dimensions must obey the Helmholtz's law and, consequently, the Kirchhoff's law at vortex crossings. One elegant way to ensure this is to build our system from small horseshoe vortices, consisting of a small bound segment (position on the wing). The resulting model is shown in figure 8. Like in the two-dimensional case, the wing moving through the air generates circulation; and, therefore, lift. This is approximated by the bound vortex segments positioned along the wing's span. The broken line consisting of these segments is usually called the *lifting line*, hence the model's name. The free vortices, shedding from the bound vortex segments toward infinity in the freestream direction, ensure the Helmholtz law is satisfied by detracting circulation from the lifting line. They model the *wake* formed by the wing. As mentioned above, an elegant way to automatically satisfy Helmholtz law is to view the model as a superposition of N horseshoe vortices. The strengths (circulations) of these vortices form unknown variables for our problem. If these variables are known, the velocity of the flow can be determined in arbitrary point using the Biot-Savart law.

4.3 The nonlinear lifting line method

The method starts from the assumption that the flow around the wing can be modeled by a system of horseshoe vortices, as shown on Fig. 8. The bound segments of these vortices model the circulation around the wing, while the free segments form the wake. Given a single horseshoe vortex with known strength (circulation), its induced velocity in any point in the 3D space can be calculated using the Biot-Savart law, (44).

Given the vortex strengths, we can, using the above method, calculate induced velocities in the midpoints of the bound vortex segments (the influence of a bound segment on itself is set to zero). The induced velocity is a vector in three-dimensional space; however, we will only be interested in its projection into a section plane (perpendicular to the wing):

$$v_{xi} = \sum g_{xik}\Gamma_k \tag{46}$$

$$v_{yi} = \sum g_{yik}\Gamma_k \tag{47}$$

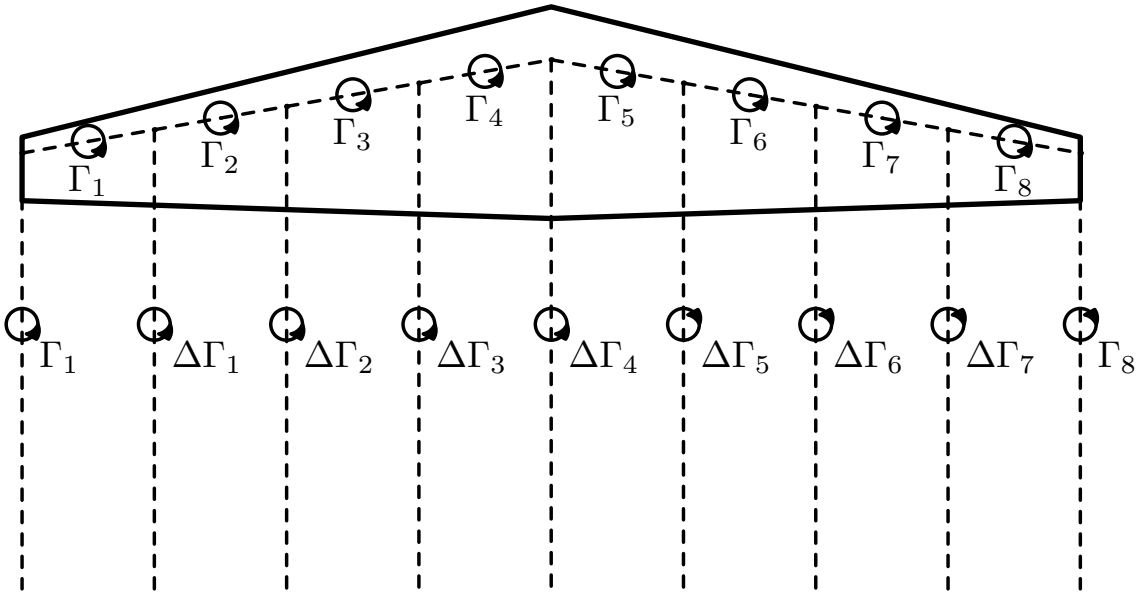


Figure 8: A horseshoe vortex model of a wing

Here, v_{xi} is the chordwise velocity component and v_{yi} is the component perpendicular to the chord in i-th section (panel). g_{xik} and g_{yik} define the *induced velocity tensor*. These induced velocities are superimposed onto the freestream velocity to get local stream velocity and local angle of attack (in the following equations, we omit the subscript i for brevity):

$$v = \sqrt{(v_x + v_{x\infty})^2 + (v_y + v_{y\infty})^2}, \tag{48}$$

$$\alpha = \operatorname{arctg}\left((v_y + v_{y\infty})/(v_x + v_{x\infty})\right) - \alpha_{\text{twist}}. \tag{49}$$

The local lift coefficient is interpolated from the local lift curves:

$$c_L = C_L(\alpha) \tag{50}$$

and the equations are closed using the following equation:

$$c_L v c = 2\Gamma, \tag{51}$$

which results from combining Eqs. (10) and (11). Choosing the local circulations as unknown variables, one ends with a system of nonlinear equations of the form

$$\Gamma_i = F_i\left(\sum g_{xik}\Gamma_k, \sum g_{yik}\Gamma_k\right) \tag{52}$$

where F_i are scalar nonlinear functions of two variables. These functions depend not only on the wing geometry and the section polars, but also on the global angle of attack.

4.4 Implementation: NLWing2

NLWing2 ([15]) is an implementation of the nonlinear lifting line method developed at VZLÚ. In vector notation (bold symbols denote vector or matrix variables and functions), we can write this nonlinear system in the form

$$\mathbf{F}(\mathbf{\Gamma}(\alpha), \alpha) = 0. \tag{53}$$

The computation proceeds by continuously tracking $\Gamma(\alpha)$ using a *predictor-corrector* approach. At any angle of attack α, we choose a step $\Delta\alpha$ and use approximate differentiation of Eq. (53) with respect to α to obtain the *predictor* in the form:

$$\nabla\mathbf{F}(\mathbf{\Gamma}, \alpha) \cdot \Delta\mathbf{\Gamma} = -\Delta\alpha\, \mathbf{F}(\mathbf{\Gamma}, \alpha + \Delta\alpha) \tag{54}$$

with $\Delta\mathbf{\Gamma}$ unknown, obtained as a solution of the linear system above. Afterwards, we replace $\mathbf{\Gamma} + \Delta\mathbf{\Gamma} \to \mathbf{\Gamma}$, $\alpha + \Delta\alpha \to \alpha$ and use the new value of $\mathbf{\Gamma}$ as a starting point for the corrector applied to the system of nonlinear equations (53). The corrector consists in iterative solution of a system of nonlinear equations, discussed below. In this manner, the parametrized solution, constituting a multi-dimensional curve, is tracked with increasing angle of attack. NLWing2 implements an adaptive stepping mechanism that can choose a smaller step when local conditions approach the maximum local lift or if the corrector fails. This will be subject of further research.

NLWing2 is able to calculate integral as well as span-wise local quantities, including lift, viscous drag, induced drag and momentum. Of particular interest in our optimizations was the point of the initial stall of local lift, which is also output.

Figures 9 and 11 show a comparison between NLWing2 and `Glauert`, another software used at VZLÚ. `Glauert` solves the linear Prandtl's integral equation using Fourier series approximations, called Glauert's method. Figure 9 shows the integral lift curve; the differences between the linear and nonlinear approach are clearly visible. Figure 10 shows the polar curves (lift/drag dependences). For the nonlinear method, both inviscid (i.e. induced drag only) and viscous polars are shown. In Figure 11 the span-wise distributions of local lift are compared; again, it is clearly visible that at lower angles of attack the differences are small but at higher angles the linear method significantly over-estimates the lift.

In the two sections below, we shall touch two particular areas regarding the details of the method implemented in NLWing2.

4.4.1 Solving nonlinear equations

Let us consider a general nonlinear system

$$\mathbf{F}(\mathbf{x}) = 0. \tag{55}$$

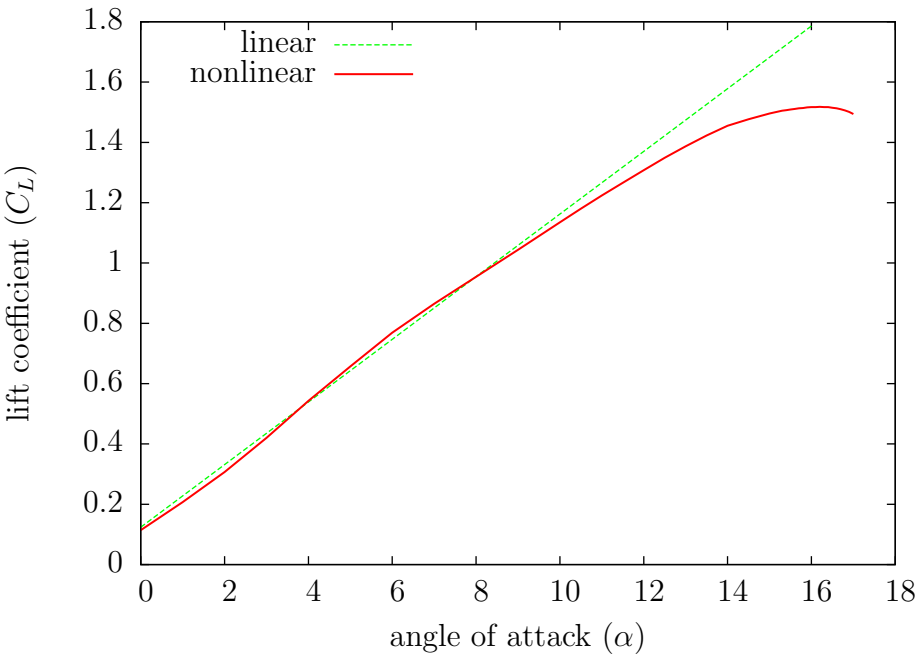


Figure 9: Comparison of lift curves of linear vs. nonlinear method

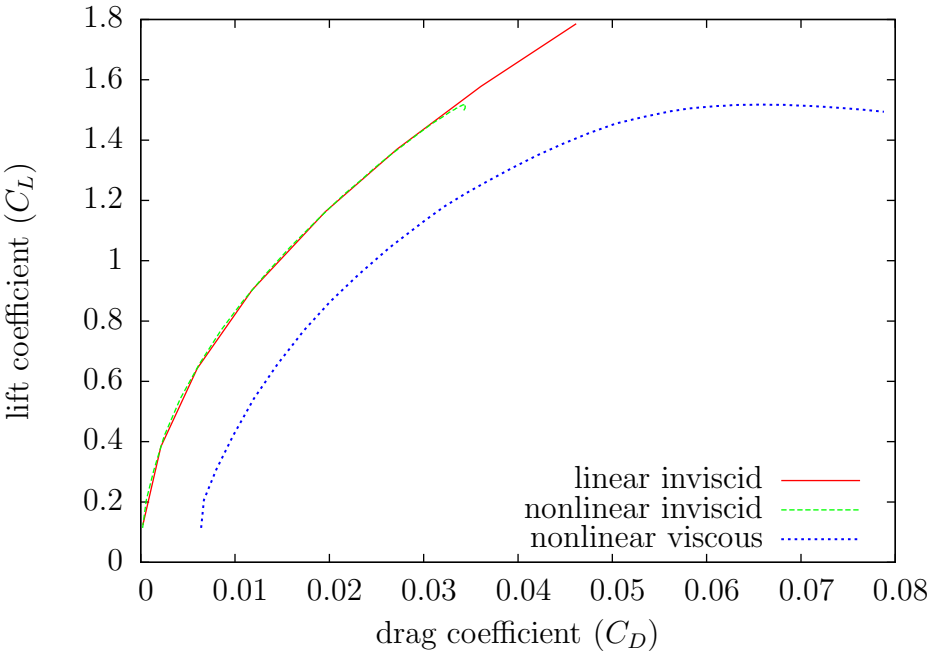


Figure 10: Comparison of polar curves of linear vs. nonlinear method

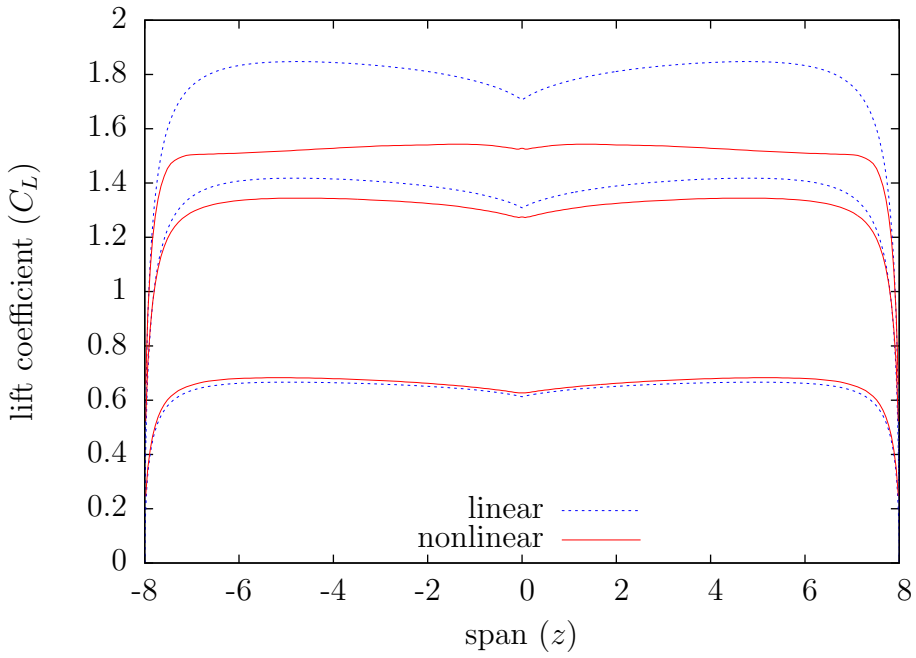


Figure 11: Comparison of span-wise lift distribution of linear vs. nonlinear method

A general solution of this system by an iterative method with a starting point produces a sequence of approximate solutions $\mathbf{x}_k, k = 1, 2, \ldots$. We will also denote $\mathbf{F}_k = \mathbf{F}(\mathbf{x}_k)$, $\mathbf{J}_k = \nabla\mathbf{F}(\mathbf{x}_k)$ and $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$. Various methods differ in the way the sequence x_k is constructed. The most basic method for solving systems of nonlinear equations is the Newton method:

$$\mathbf{s}_k = -\mathbf{J}_k^{-1}\mathbf{F}_k. \tag{56}$$

The Newton method exhibits quadratic convergence in a sufficiently close neighborhood of the true solution with an invertible jacobian. It's global convergence properties, however, are poor if started too far off the true solution. In NLWing2 the corrector method can be chosen amongst a tuned Levenberg-Marquardt method and Powell's hybrid method.

The Levenberg-Marquardt (see [16]) method uses the steps

$$\mathbf{s}_k = -(\mathbf{J}_k^T\mathbf{J}_k + \lambda\mathbf{I})^{-1}\mathbf{J}_K^T\mathbf{F}_k, \tag{57}$$

where λ is a positive parameter adapted in each step, representing a smooth transition between a Newton step ($\lambda = 0$) and a steepest descent step (λ large). Instead of (57), the L-M corrector in NLWing2 solves the following equivalent problem:

$$\begin{pmatrix} \mathbf{J}_k \\ \sqrt{\lambda}\mathbf{I} \end{pmatrix} \mathbf{s}_k = \begin{pmatrix} -\mathbf{F}_k \\ \mathbf{0} \end{pmatrix} \tag{58}$$

in the least squares sense. This is somewhat slower but avoids dealing with the squared matrix $\mathbf{J}_k^T\mathbf{J}_k$ which is twice as ill-conditioned as $\mathbf{J}_k$. A step is only taken if $\|\mathbf{F}_{k+1}\| < \|\mathbf{F}_k\|$. The λ parameter is increased after a successful step and decreased after an unsuccessful step. For the precise strategy that has been somewhat tuned to the real problems being solved by NLWing2, we defer the reader to the source code.

We have implemented the Powell's hybrid method in Octave's `fsolve` function. In each step, this method approximately solves the following so-called trust-region subproblem to determine the step length:

$$\min_{\mathbf{D}_k\mathbf{s}_k \le \Delta} \|\mathbf{J}_k\mathbf{s}_k + \mathbf{F}_k\|, \tag{59}$$

where $\mathbf{D}_k$ is a diagonal scaling matrix that is automatically adjusted according to column norms of $\mathbf{J}_k$. This is a difficult problem in general, requiring either an iterative method or a singular value decomposition (see [17]). To avoid this, $\mathbf{s}_k$ is only searched along a special path in the 2-dimensional subspace

$$\operatorname{span}\left\{\mathbf{J}_k^{-1}\mathbf{F}_k, \mathbf{D}_k^{-2}\mathbf{J}_K^T\mathbf{F}_k\right\}, \tag{60}$$

called the double dogleg path. The details can be found in [18]. The first of the above basis vectors represents the Newton direction, while the second is the so-called Cauchy point. In cases when $\mathbf{J}$ is computed by finite differences or is significantly more expensive to compute than $\mathbf{F}$ (this is usually true in real-life problems, and is also true in NLWing2), the Powell's hybrid method can avoid recalculating the Jacobian in certain steps by performing a Broyden (secant) update of the Jacobian, i.e. adding a matrix of rank 1 to $\mathbf{J}_k$ so that $\mathbf{J}_{k+1}$ satisfies the secant equation:

$$\mathbf{J}_{k+1}\mathbf{s}_k = \mathbf{F}_{k+1} - \mathbf{F}_k. \tag{61}$$

Amongst all such possible matrices, the one with the smallest scaled Frobenius norm is selected, namely

$$\mathbf{J}_{k+1} = \mathbf{J}_k + \frac{(\mathbf{F}_{k+1} - \mathbf{F}_k)\,\mathbf{D}_k^2\,\mathbf{s}_k^T}{\|\mathbf{D}_k\mathbf{s}_k\|^2}. \tag{62}$$

Even with Broyden's update, the Jacobian must be factorized in each step to get the Newton direction for the dogleg trust-region problem. To reduce the computational cost even further, the method maintains and updates a QR factorization of the Jacobian $\mathbf{Q}_k\mathbf{R}_k$ rather than $\mathbf{J}_k$ directly. The Newton direction is then obtained as $\mathbf{J}_k^{-1}\mathbf{F}_k = \mathbf{R}_k^{-1}\mathbf{Q}_k^T\mathbf{F}_k$ which is much more efficient because $\mathbf{R}_k$ is triangular. Updating the QR decomposition after an additive rank-1 update can be done efficiently by using Givens rotations. For details, see [17]. Since version 1.1.0, NLWing2 is able to optionally employ the more efficient `fsolve` algorithm as a corrector. This will probably become the default once the algorithm becomes well-tested (its implementation in Octave is still fresh).

4.4.2 Interpolating polars

In Eq. (50), we used the notion of "local lift curve". That means, at the collocation points along the span, we need a function transforming local angle of attack (given by Eq. (49)) into local lift coefficient. These lift curves are supposed to be supplied by external means; e.g., by a prior viscous two-dimensional computation or an experiment. In real life, however, we won't have the lift curve supplied for each collocation point; because we do not know them beforehand and their number may be relatively high. It is, therefore, necessary to have a means to combine two lift curves C_L^A, C_L^A, given at span-wise coordinates z_A, z_B to a single lift curve C_L at z. We shall refer to this procedure as *polar interpolation*, because it can be actually used to interpolate also the drag and moment curve. In NLWing2, two methods of polar interpolation were tested: the *trivial interpolation* and *feature interpolation*.

Trivial interpolation calculates a linear interpolation coefficient

$$\eta = \frac{z - z^A}{z^B - z^A} \tag{63}$$

and then defines the combined lift curve using a simple linear interpolation:

$$C_L(\alpha) = \eta\, C_L^A(\alpha) + (1 - \eta) C_L^B(\alpha) \tag{64}$$

Feature interpolation is a more elaborate approach: For each lift curve, we determine the angles of zero lift and maximum lift:

$$C_L^i(\alpha_0^i) = 0, \quad i = A, B, \tag{65a}$$

$$C_L^i(\alpha_{\max}^i) = \max_\alpha C_L^i(\alpha), \quad i = A, B. \tag{65b}$$

Now, we define

$$\alpha_0 = \eta\alpha_0^A + (1 - \eta)\alpha_0^B, \tag{66}$$

$$\alpha_{\max} = \eta\alpha_{\max}^A + (1 - \eta)\alpha_{\max}^B, \tag{67}$$

and

$$C_L(\alpha) = \eta\, C_L^A \left[\tilde{\alpha}(\alpha_{\max}^A - \alpha_0^A) + \alpha_0^A\right] + (1 - \eta) C_L^B \left[\tilde{\alpha}(\alpha_{\max}^B - \alpha_0^B) + \alpha_0^B\right], \tag{68}$$

where

$$\tilde{\alpha} = \frac{\alpha - \alpha_0}{\alpha_{\max} - \alpha_0}. \tag{69}$$

Now, substituting Eqs. (66), (69) into Eq. (68), we get

$$C_L(\alpha_0) = \eta\, C_L^A(\alpha_0^A) + (1 - \eta) C_L^B(\alpha_0^B) = 0, \tag{70}$$

which means that α_0 is again a zero lift angle for the interpolated lift curve.

Also, plugging the relations

$$C_L^A(\bar{\alpha}) \leq C_L^A(\alpha_{\max}^A) \text{ for any } \bar{\alpha}, \tag{71a}$$

$$C_L^B(\bar{\alpha}) \leq C_L^B(\alpha_{\max}^B) \text{ for any } \bar{\alpha}, \tag{71b}$$

into Eq. (68) we get

$$C_L(\bar{\alpha}) \leq \eta\, C_L^A(\alpha_{\max}^A) + (1-\eta) C_L^B(\alpha_{\max}^B) \text{ for any } \bar{\alpha}. \tag{72}$$

On the other hand, substituting Eq. (67) into Eq. (68) we obtain

$$C_L(\alpha_{\max}) = \eta\, C_L^A(\alpha_{\max}^A) + (1-\eta) C_L^B(\alpha_{\max}^B). \tag{73}$$

Together with Eq. (72), this yields

$$C_L(\bar{\alpha}) \leq C_L(\alpha_{\max}) \text{ for any } \bar{\alpha}, \tag{74}$$

which shows us that $\alpha_{\max}$ is the maximum lift angle for the interpolated lift curve, and the maximum lift itself is given by the simple interpolation in Eq. (73).

In other words, each lift curve is split into three components: the zero lift angle, the maximum lift angle, and the normalized lift curve. Interpolation is performed on these components separately and the results are then combined back into a single lift curve. In this manner, three key "features" of the interpolated lift curves, i.e. the zero lift angle, the maximum lift angle, and the maximum lift itself, can be obtained by interpolation separately and are thus kept under strong control. Also, the interpolated lift curves are guaranteed to have a single maximum. Trivial interpolation provides no such guarantee; the interpolated polar may end up having have multiple maxima.

Computational experiments also suggest that feature interpolation yields results closer to true lift curves that would be obtained by re-evaluating each inner section.

Figure 12 compares interpolated lift curves created by trivial vs. feature interpolation, using lift curves of two industrial airfoils as a basis. It is clearly visible that the maximum lift position varies significantly more smoothly with feature interpolation.

5 Wing optimization - a case study

Armed by the tools to evaluate aerodynamic performance of wings, we can proceed towards wing optimization. We do so in the form of a case study. The particular optimization problem analyzed in this section is concerned with optimizing the

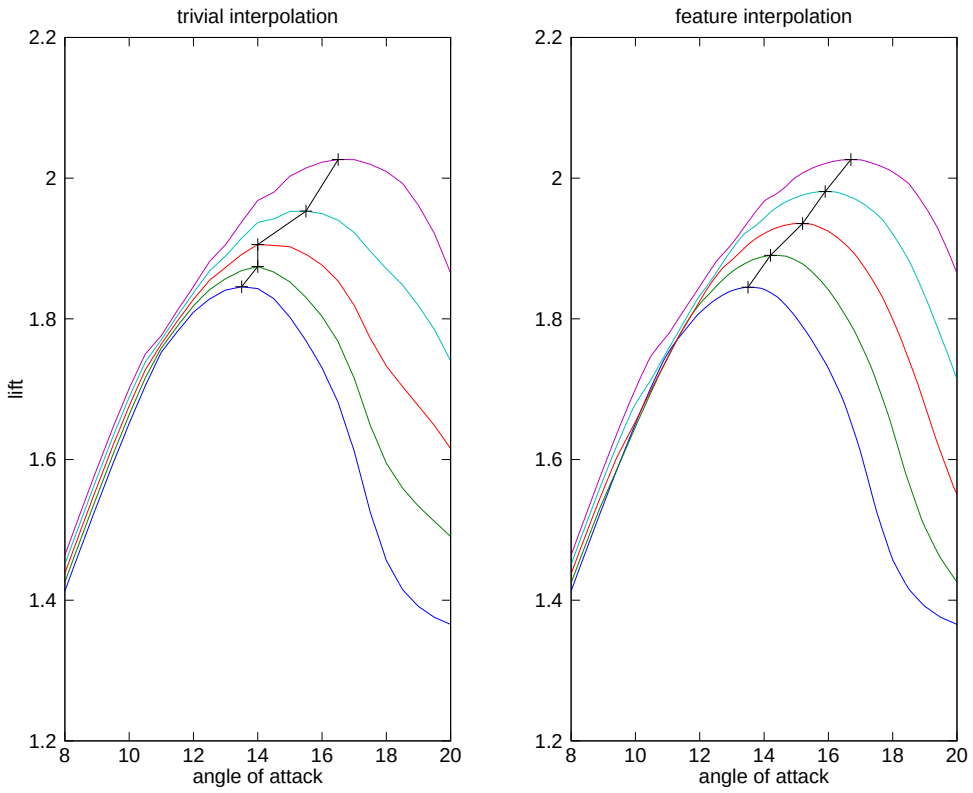


Figure 12: Comparison of polar interpolation methods

	32 designs		1024 designs	
# Processors	time	speed-up	time	speed-up
1	100.7s	1.00	55m 14s	1.00
2	50.0s	2.01	27m 28s	2.01
4	25.5s	3.95	13m 44s	4.02
8	13.4s	7.51	6m 52s	8.05

Table 1: Time consumption by NLWing2 evaluating wing designs

geometry of a symmetric wing for a standard-class glider, with respect to two and later three objectives and one primary constraint. This constraint is what uniquifies this study to a strong extent, because it is quite difficult to employ in alternative optimization approaches.

5.1 Parallel evaluation

With an optimized build of Octave, using `fsolve` as a corrector, NLWing2 works very fast. Still it is possible to speed-up evaluations by parallelizing them. Because NLWing2 is an Octave package, we aimed to create an efficient general parallel evaluation tool.

To this end, we contributed the `parcellfun` function into the `general` package of OctaveForge. This function work analogously to Octave's built-in `cellfun` function, which allows to conveniently evaluate an arbitrary function multiple times with different arguments. The difference is that `parcellfun` can perform the jobs in parallel using a specified number of processors. Parcellfun works using the POSIX `fork` call, which can clone running processes. At parcellfun startup, the Octave interpreter process is cloned multiple times, and individual jobs are distributed to the forked processes using POSIX pipes. Results are collected back again via pipes. This particular implementation overcomes the difficulty that the Octave interpreter is, by design, not thread-safe.

Table 1 shows timings of parallel evaluation of wings from the design space of the wing optimization problem (described in next section). It is apparent that parcellfun scales extremely well; for larger design sets, essentially the maximum theoretical scalability is achieved. The tiny excess over the theoretical maxima is caused by extra initialization work done in the first evaluation.

5.2 Design variables

In this section we describe a real wing optimization problem carried out as part of the CESAR project at VZLU.

Optimize the planform of a symmetric wing, consisting of four (two for each half-wing) trapezoids. The geometry is to be determined by the following five design variables:

- `Sw`: The wing planform area $23 - 27m^2$
- `TR`: Taper ratio (tip depth / root depth) $0.4 - 0.6$
- `ttw`: Tip twist $0 - 4°$
- `xbr`: Relative break position $0 - 0.5$
- `btr`: Deviation from straight wing $-1 - 1$

The last two variables deserve a more precise description. Say that the root chord length is c_{rw}, c_{br} is the chord length separating the root (inner) and tip (outer) trapezoid, and c_{tw} is the tip chord length. Further, let b_r and b_t be the root and tip trapezoid span, respectively.

$$\texttt{xbr} = \frac{b_r}{b_r + b_t} \tag{75}$$

and

$$\texttt{btr} = \frac{c_{br} - c_{b0}}{c_{rw} - c_{b0}}, \quad c_{b0} = \texttt{xbr}\, c_{tw} + (1 - \texttt{xbr}) c_{rw}. \tag{76}$$

Further, $c_{rw} = 1.955m$, dihedral angle is $1.5°$, and the quarter-chord line is assumed straight. These constraints determine the geometry of the wing. The somewhat strange definition of `btr` is chosen to avoid creating geometries with too sharp angles, which are never feasible and for which NLWing2 may even fail to produce a meaningful result. Also, the fact that if $\texttt{btr} = 0$, the wing simplifies to a single trapezoid, is useful in the following investigations.

5.3 Objectives

The aerodynamic performance of the wing is measured by the following two quantities:

- `cdsw`: $C_D \cdot \texttt{Sw}$ at $C_L \cdot \texttt{Sw} = 7.558$,
- `cmcsw`: $C_M \cdot c_{\mathrm{MAC}} \cdot \texttt{Sw}$ at $C_L \cdot \texttt{Sw} = 7.558$.

where C_L, C_D, C_M are the lift, drag and momentum coefficients of the wing, respectively. c_{MAC} is the mean aerodynamic camber of the wing. Multiplying C_L and C_D by `Sw` is used to achieve independence of the varying geometry; the result is

essentially a normalized force. The role of $c_{\mathrm{MAC}} \cdot \mathtt{Sw}$ for C_M is analogous. The optimization is not, however, concerned only with performance. An important factor for glider wings design is flight safety, especially in the take-off and landing regimes. In these regimes, the wing typically operates close to its maximum lift and its critical angle of attack, where separation of the boundary layer starts to develop on the wing. During the flight, exceeding the critical angle of attack leads to a sudden drop of lift. To facilitate flight safety, it is important to keep the separated region away from flight control devices positioned at the wing, such as ailerons and flaps, typically located near the tip of the wing. Therefore, we introduced the primary constraint requiring that

- $\mathtt{sep} \leq 65\%$: where $\mathtt{sep} = \frac{Z_{\mathrm{SEP}}}{b/2}$

where b is the wing span and Z_{SEP} is the smallest (in absolute value) span-wise coordinate where, as the global angle of attack increases, the local lift coefficient first reaches its maximum (i.e. critical angle of attack of the local wing section) and, presumably, the separated region is formed and starts to spread. The global angle of attack where this occurs is referred to as α_{crit}. For reasons described later, the following additional objective needs to be introduced:

- `clmaxsw`: $C_L\mathtt{Sw}$ at α_{crit}.

5.4 Results

Given the outstanding performance of our evaluation mechanism (NLWing2 combined with `parcellfun`), we could even afford using a brute force approach. That means, sample the whole design space using a cartesian grid, evaluate all designs, and then simply pick the non-dominated designs satisfying our constraints, according to the definitions given by Eqs. (25a), (25b). In this simple manner, we can obtain a crude approximation of the Pareto front. The result of brute force grid sampling is shown in Fig. 13. Red markers indicate all sample designs, blue markers denote the Pareto front approximation. The 5-dimensional design space was sampled using an $8 \times 8 \times 8 \times 8 \times 8$ grid, giving 32768 total designs, whose evaluation lasted approximately 3.5 hours.

The grid sampling approach can be viewed as a very simplistic optimization algorithm. Despite its simplicity, it possesses certain attractive properties not provided by more elaborate algorithms: First, it is very robust in the sense that a sufficiently broad global optimum is guaranteed to be found; and second, the objectives and

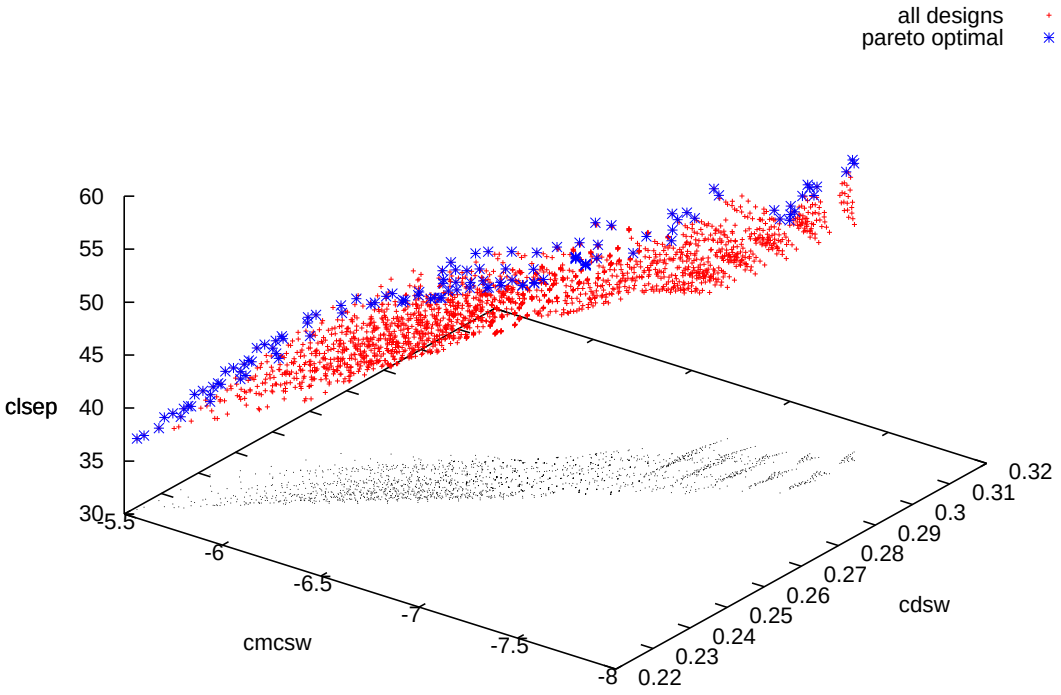


Figure 13: Wing optimization results - grid sampling

constraint can be tweaked to some extent without actually doing any new computations. For instance, in the grid sampling we evaluate the objectives and `sep` for all designs, but we only proceed with those designs satisfying $\texttt{sep} \leq 65\%$. If, later, we want to tighten this constraint to 60% instead of 65%, no new computations are needed; we simply start over with the complete grid data.

Figure 14 compares the approximate Pareto front obtained from grid sampling with one delivered by μ-ARMOGA(as described in section 3).

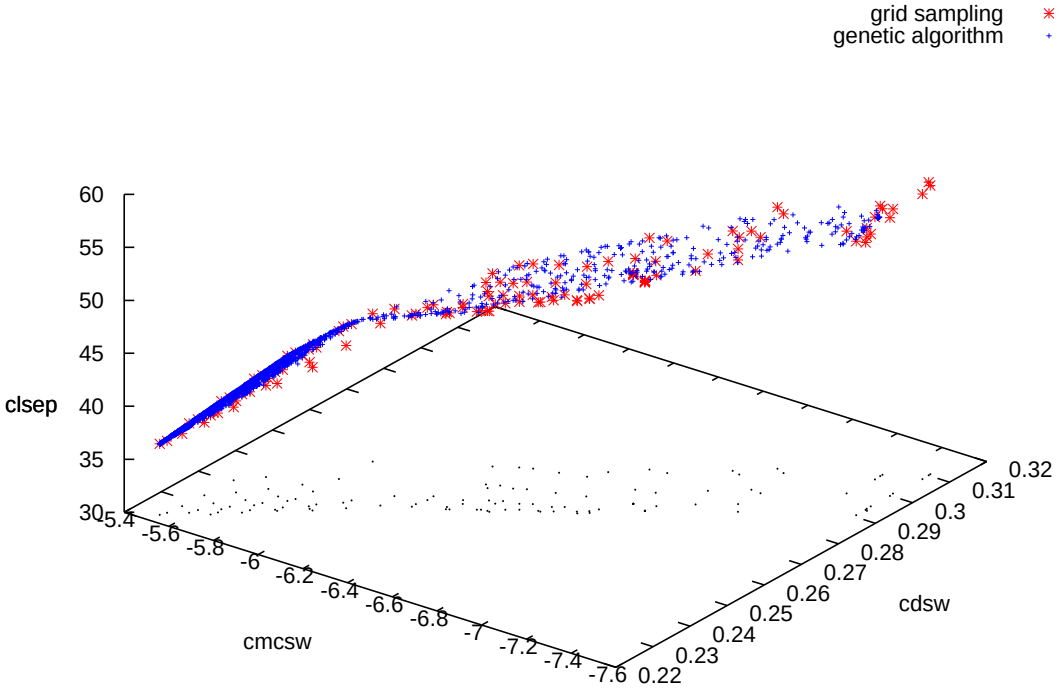


Figure 14: Wing optimization results - grid sampling vs. μ-ARMOGA

5.5 Problem degeneration

We now reveal the motivation for adding the third objective, `clsep`. Let us consider our optimization problem with `clsep` dropped. Further, let us fix `btr` to zero, restricting the design space to wings consisting of a single trapezoid. The resulting Pareto front for this simplified optimization problem is shown as red in Fig. 15.

Now, let us put `btr` back to the game, allowing it to span the range from -1 to 1. Amazingly enough, the Pareto front degenerates to a single optimum point, shown as green in Fig. 15.

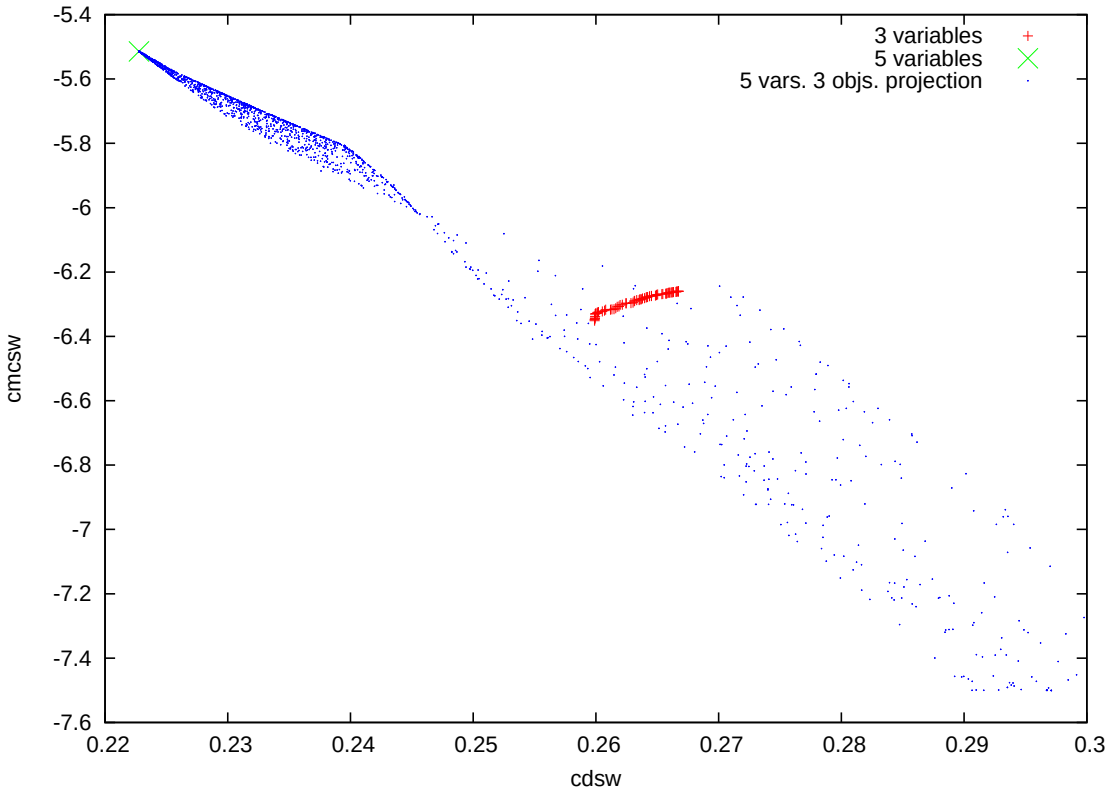


Figure 15: Wing optimization results - simplified problem

What happened? In the one-trapezoid case, the primary mechanism that can push the point of separation `sep` towards the root (and thus towards satisfying the primary constraint) is increasing the tip twist. By allowing a "broken" wing, we introduced another, completely different mechanism that allows to accomplish the same - creating a spike on the local critical lift distribution. This is pictured by Fig. 16. The left part displays the span-wise lift distribution of a design with nonzero `btr`. On the right side `btr` is set to zero, while other variables remain the same. Now it is clearly visible how the mechanism works: the spike in local chord length distribution induces a spike in local maximum lift distribution. The span-wise position of the spike determines the separation point, which can then be directly controlled. Apparently, an adverse effect is that the separation will occur earlier, i.e. at a lower

total lift and angle of attack. This is a hidden tradeoff that does not occur when only the single-trapezoid designs (3 variables) are allowed. Hence the motivation to add the third objective. This objective explicitly expresses this tradeoff, turning a degenerate 2-dimensional Pareto front into a regular 3-dimensional one.

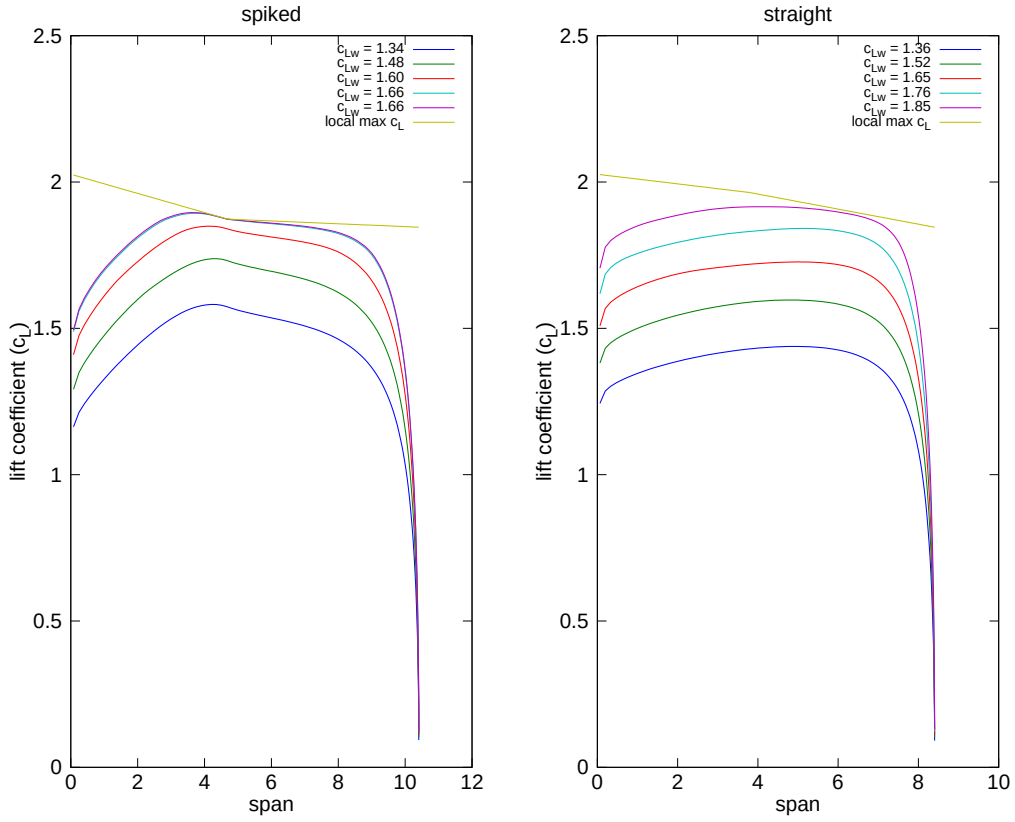


Figure 16: The effect of chord length spike on the point of separation

6 Conclusions and future work

This work develops a methodology for optimizing airfoils and wings for subsonic flow conditions using fast aerodynamic performance evaluation methods (solvers). The basis of aerodynamic performance evaluation of airfoils is represented by the state-of-the-art free software XFOIL. The underlying computational model used by XFOIL is briefly described in this thesis, especially the key concepts of inviscid incompressible

irrotational flow. Since XFOIL is designed for human-driven visual design, the Python module XFEVAL was developed, encapsulating XFOIL, to allow automatic parallel evaluation of multiple airfoils in symmetric multiprocessor machines. The aspects of this parallel evaluation are discussed in this work.

To express the shape of an airfoil using a set of numbers, which is necessary for optimization, the novel GPARSEC parametrization method is developed. This method uses a set of geometric characteristics of an airfoil as the design variables, allows employing arbitrary basis functions and extension to more DOFs in a manner not interfering with the geometric meaning of the basic set of design variables. To carry out the actual optimization, an evolutionary algorithm, μ-ARMOGA, was developed. A detailed description of μ-ARMOGAis given in a cited paper ([11]); in this thesis, only a brief summary is presented, pointing out the algorithm's key features. However, one of its most innovative parts, a novel strategy for updating the Pareto archive, is described and discussed in detail. This approach relies on maintaining and updating nearest-neighbor information in the Pareto archive, and using it as a secondary (to standard Pareto dominance) criterion for accepting or discarding a new individual. The approach is particularly suitable for micro-evolutionary approaches (such as μ-ARMOGA). Further details as well as numerical experiments are given in a cited paper ([11]).

These methods are then applied to a real-life problem - optimizing a subsonic airfoil subject to standard flight conditions in both laminar and turbulent boundary layer regime. The objectives are discussed, and numerical results of the optimization are presented.

Concerning performance evaluation of wings, a special-purpose solver, NLWing2, was developed in the form of an extension package to the GNU Octave environment ([20]). It is based on an extension of the Prandtl's lifting line method, allowing curved and swept wings with dihedral angle, and incorporating nonlinear local dependencies of lift on angle of attack. For the arising system of nonlinear parametric equation, a solution method based on the predictor-corrector paradigm was developed. Originally, the corrector was a simple Levenberg-Marquardt method; later, a general nonlinear equation solver was developed for Octave, based on a trust-region technique with factorized Broyden updating. This solver (`fsolve`) can be employed as a more efficient and robust corrector component for the NLWing2 algorithm. A problem basically unique to the nonlinear lifting-line method is interpolation of airfoil polars. Two possible approaches, trivial and feature interpolation, are described in detail and advantages of the latter are highlighted.

Finally, optimization of wings is considered, in the form of a case study for a certain multi-objective optimization problem. Parallel evaluation with NLWing2 using the capabilities of GNU Octave is also addressed. Design variables and objectives for the optimization problem are discussed in detail, and it is shown how the choice

of objectives influences the nature of the optimization problem. Numerical results of optimization are also presented.

The tools and methodology developed and described in this thesis forms a useful framework to approach complex design optimization problems of simple subsonic airfoils and wings, in particular problems with relatively simple geometry yet difficult objectives. The diagram presented in Fig. 17 shows how all the tools can work together and the flow of information between them. The component showing a question mark is of particular concern for future work: the transition from suitable design variables to a wing geometry (i.e., a wing parametrization) is not yet addressed by any general software tool; rather, custom program (GNU Octave scripts) is created for each particular problem. This will be a subject of future research. Other areas providing possible future research directions include the polar interpolation problem, better automatic continuation method for NLWing2, extending the nonlinear lifting-line method to work with less slender wings, and improving the μ-ARMOGA algorithm. So far, the methodology was successfully used for a couple of commercial and research jobs at VZLÚ. Two utility models were registered, with more planned.

7 Appendix

7.1 Green's identities

The Green's second identity states that for ξ, χ defined inside a domain Ω, it holds:

$$\int_\Omega (\xi\,\Delta\chi - \chi\,\Delta\xi)dV = \oint_{\partial\Omega} (\xi\nabla\chi - \chi\nabla\xi)\cdot\mathbf{n}\,dS \tag{77}$$

This identity is also valid if ξ is smooth but $\Delta\chi$ only exists in the sense of distributions. This allows us to make the following choice of χ, also known as the Green's function for the Laplace equation:

$$\chi(x) = \begin{cases} \frac{1}{2\pi} - \log\|x - x_0\| & \text{if } n = 2, \\ \frac{1}{4\pi}\frac{1}{\|x-x_0\|} & \text{if } n = 3. \end{cases} \tag{78}$$

where n is the spatial dimension. Formula for general n also exists. With this definition, it can be shown that

$$\Delta\chi = \delta(x - x_0) \tag{79}$$

in the sense of distributions, where δ is the Dirac delta distribution. Substituting this into Green's second identity, we obtain the Green's third identity:

$$\xi(x_0) = \int_\Omega (\chi\,\Delta\xi)dV + \oint_{\partial\Omega} (\xi\nabla\chi - \chi\nabla\xi)\cdot\mathbf{n}\,dS \tag{80}$$

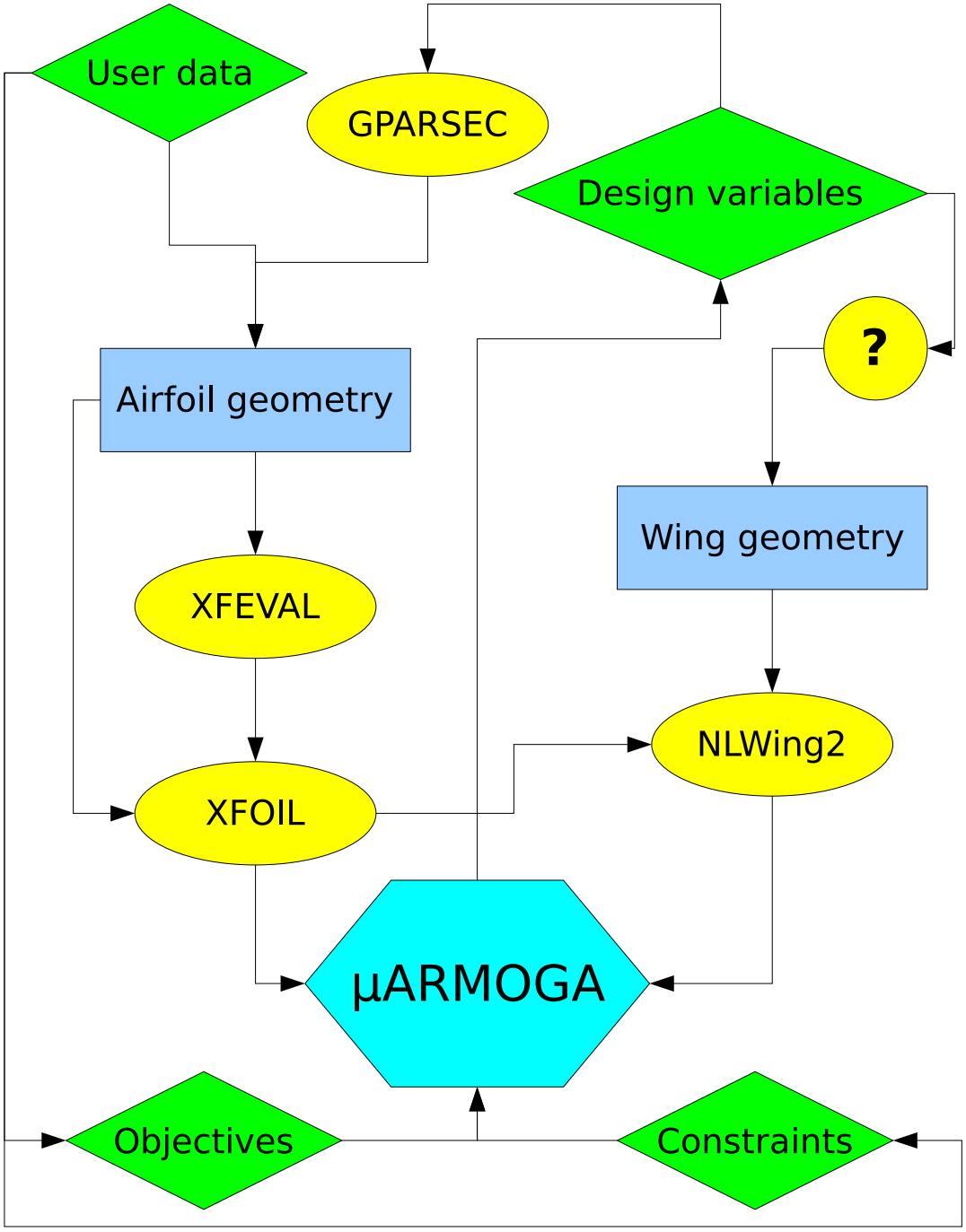


Figure 17: Workflow diagram of complex optimization framework

allowing us to express the value of ξ in any point as an integral involving its Laplacian inside Ω plus certain integrals over the boundary. If $\Delta\xi$ is known or is zero in Ω (solution of Poisson's or Laplace's equation), then the values ξ inside the domain are uniquely determined by its values on the boundary.

7.2 Induced velocities calculation

Let us consider a vortex segment with endpoints A, B and of strength Γ. Let R_A, R_B be the radiusvectors from a collocation point C to A, B, respectively. Then the induced velocity in C can be expressed analytically by integrating the infinitesimal Biot-Savart law (Eq. (44)). We get

$$\mathbf{v}_I = \frac{\Gamma}{4\pi} \int_0^1 \frac{(t\mathbf{R}_B + (1-t)\mathbf{R}_A) \times (\mathbf{R}_B - \mathbf{R}_A)}{\|(t\mathbf{R}_B + (1-t)\mathbf{R}_A)\|^3}\, dt \tag{81}$$

which simplifies to (the numerator is constant)

$$\mathbf{v}_I = \frac{\Gamma}{4\pi}(\mathbf{R}_A \times \mathbf{R}_B) \int_0^1 \frac{dt}{\|(t\mathbf{R}_B + (1-t)\mathbf{R}_A)\|^3}. \tag{82}$$

Using Lagrange's identity, one can find the antiderivative of the integrand to be

$$\frac{(t\mathbf{R}_B + (1-t)\mathbf{R}_A) \cdot (\mathbf{R}_B - \mathbf{R}_A)}{\|(t\mathbf{R}_B + (1-t)\mathbf{R}_A)\| \left(\|\mathbf{R}_B\|^2 \|\mathbf{R}_A\|^2 - \mathbf{R}_B \cdot \mathbf{R}_A\right)} \tag{83}$$

which after some manipulations leads to the final expression

$$\mathbf{v}_I = \frac{\Gamma}{4\pi} \frac{(\|\mathbf{R}_A\| + \|\mathbf{R}_B\|)(\mathbf{R}_A \times \mathbf{R}_B)}{\|\mathbf{R}_A\| \|\mathbf{R}_B\| \left(\|\mathbf{R}_A\| \|\mathbf{R}_B\| + \mathbf{R}_A \cdot \mathbf{R}_B\right)}. \tag{84}$$

This result can be written in multiple forms, but this form has the advantage of not being singular if C lies on the line AB but outside the segment AB. This property is important in the nonlinear lifting line method.

The velocity induced by a half-infinite ray starting at A going in the direction of $\mathbf{v}$ can be found by substituting $\mathbf{R}_B = \mathbf{R}_A + t\,\mathbf{v}$ into Eq. (84) and taking the limit $t \to +\infty$, yielding

$$\mathbf{v}_I = \frac{\Gamma}{4\pi} \frac{\mathbf{R}_A \times \mathbf{v}}{\|\mathbf{R}_A\| \left(\|\mathbf{R}_A\| \|\mathbf{v}\| + \mathbf{R}_A \cdot \mathbf{v}\right)}. \tag{85}$$

References

[1] http://web.mit.edu/drela/Public/web/xfoil/

[2] Katz, J., Plotkin, A., Low-speed Aerodynamics, Cambridge University Press, 2001, ISBN 0521665523, 9780521665520

[3] Drela, M., Giles, M.B., Viscous-Inviscid Analysis of Transonic an Low Reynolds Numer Airfoils, AIAA Journal, 1986, Vol. 25, No. 10

[4] Drela, M., Low-Reynolds-Number Airfoil Design for the M.I.T. Daedalus Prototype: A Case Study, Journal of Aircraft, 1988, Vol. 25, No. 8

[5] Stewart, T.J.: A Critical Survey on the Status of Multiple Criteria Decision Making and Practice, International Journal of Management Science, Vol. 20 No. 5/6, 1992

[6] Douglas Aircraft Company: Study of High-Speed Civil Transports, NASA-CR4236, 1990.

[7] Jones, R.T.: Wing Theory, Princeton University Press, Princeton, NJ, 1990 (Chap. 7)

[8] Wu, H.Y., Yang, S.C., Liu, F. and Tsai, H.M., Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization, AIAA 2003-4095, 16th AIAA CFD Conference, June 23-26, Orlando, FL, 2003

[9] Sobieczky, H.: Parametric Airfoils and Wings, Notes on Numerical Fluid Mechanics, edited by K. Fuji and G.S. Dulikravich, Vol. 68, Vieweg Verlag, 1998, pp. 71-88

[10] Samareh, J.A., Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization, AIAA Journal, Vol. 39, No. 5, 2001, pp. 877-884

[11] Hájek, J., Šístek, J., Szöllös, A.: A New Mechanism for Maintaining Diversity of Pareto Archive in Multiobjective Optimization, submitted to Journal of Evolutionary Computation.

[12] Hájek,J., Szöllös,A., Šmíd,M.: Aerodynamic Optimization via Multi-Objective Micro-genetic Algorithm with Range-Adaptation, Knowledge-Based Reinitialization, eps-Dominance and Crowding, Advances in Engineering Software, Vol. 40, 2009, pp. 419-430

[13] Pátek, Z., Szöllös, A., Hájek, J.: Aerodynamic airfoil of glider wing without flaps (in Czech), Utility model CZ 17179 U1, Czech Republic, Appl. no. 2006-17864, Int. Cl. B64C 3/14. Owner Výzkumný a Zkušební Letecký Ústav, a.s.

[14] Hájek, J., Pátek, Z., Szöllös, A.: Aerodynamic airfoil (in Czech), Utility model CZ 18600 U1, Czech Republic, Appl. no. 2007-19280, Int. Cl. B64C 3/14, F15D 1/10. Owner Výzkumný a Zkušební Letecký Ústav, a.s.

[15] http://octave.sourceforge.net/packages/nlwing2.html

[16] Nocedal, J., Wright, S. J., Numerical Optimization, Springer Verlag, 2000, ISBN 0387987932, 9780387987934

[17] Golub, G. H., Van Loan, C. F., Matrix Computations (3rd edition), Johns Hopkins University Press, 1996, ISBN 0801854148

[18] Dennis, J. E., Schnabel, R. B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, 1996, ISBN 0898713641, 9780898713640

[19] Saffman, P. G. (1992), Vortex Dynamics, Cambridge University Press, ISBN 9780521477390

[20] http://www.octave.org