Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018

ICCFD10-2018-322

Adjoint-based Pareto Front Tracing in Aerodynamic Shape Optimization

K. Gkaragkounis, E. Papoutsis-Kiachagias, V. Asouti and K. Giannakoglou
Corresponding author: kogkar@hotmail.com
National Technical University of Athens (NTUA), School of Mechanical Engineering
Parallel CFD & Optimization Unit
9, Iroon Polytechniou, NTUA Zografou Campus, 15780, Athens, Greece

Abstract: In multi-objective optimization problems, fronts of non-dominated solutions are traditionally computed either by multi-objective evolutionary algorithms or gradient-based methods. In the latter, the objective functions are usually concatenated into a single one using different value-sets of weights and the computation involves as many single-objective optimizations as the sought points on the front. Both have a relatively high computational cost, with that of evolutionary algorithms scaling with the number of design variables and the one of gradient-based methods scaling with the number of front points. This paper aims at developing and using gradient-based methods for populating the Pareto front at a low cost. To this end, after identifying one point on the front, a number of successive constrained single-objective optimizations are solved to compute the rest of the front points. The system of the KKT conditions for the above-mentioned optimization problem is solved in three different ways. Among other, the costly computation of the Hessian matrix is avoided a) through BFGS approximations which require only gradients computed with the continuous adjoint method and b) the computation of Hessian-vector products instead of the Hessian itself, by combining the adjoint method and the direct differentiation of the flow and adjoint equations. The cases used for demonstrating the developed algorithms pertain to the two-objective, constrained shape optimization of two isolated airfoils, targeting min. drag and max. lift, by retaining a constant airfoil area as a geometrical constraint.

Keywords: Pareto Front, Continuous Adjoint Method, Hessian matrix, Direct Differentiation.

1 Introduction

Optimization problems with more than one objective functions are becoming increasingly common, both in academic and industrial applications. These multi-objective optimization (MOO) problems are often tackled with Evolutionary Algorithms (EAs) which are able to compute the Pareto front of non-dominated solutions. The main advantages of using EAs are the use of "black-box" objective function evaluations and the ability of converging to globally optimal solutions, after a usually great number of generations. However, at least for standard EAs, the computational cost scales with the number N of design variables, [1].

On the other hand, gradient-based methods may tackle MOO problems by breaking them down into several single-objective optimization (SOO) ones. Most of the relevant publications refer to weighted-sum approaches in which a linear combination of the individual objectives is minimized. Different weight value-sets lead to different front members. However, such an algorithm can compute the Pareto front only if the weighted sum is convex w.r.t. the design variables [1–4]. Even in this case, the computational cost scales with the number of front points since a separate SOO problem should be solved for each of them. Another drawback of solving many SOO problems for different weights is the lack of an obvious mapping of the value-sets of weights and the front point positions, with the exception of the front edges [2–5].

In [3], instead of deciding the weight value-sets a-priori, a dynamic way to compute them is proposed. A small number of SOO problems are initially solved for a few weight value-sets and then, the euclidean

distance between the Pareto points is used to enrich less populated front parts. This method is also able to compute non-convex Pareto fronts, though it significantly depends on the choice of the starting point. Another method to compute non-convex and also discontinuous fronts is presented in [2], which relies on the steepest descent method and starts tracing the Pareto front from a point lying already on it.

In [5], a MOO approach in the form of a prediction-correction scheme using gradient methods is proposed. A constrained minimization problem is solved to move from one front point to another. In particular, equality constraints are imposed to all but one objective functions, by assigning user-defined target values while the remaining objective is minimized. This approach has the advantage of controlling the distance (in the objective space) between two Pareto points, at least for the constrained objective functions. The algorithm relies on the KKT conditions to formulate the equations to be solved for traversing from one point to the next. In specific, the KKT conditions yield a linear system, the solution of which gives the derivatives of the design variables and Lagrange multipliers w.r.t. the target objective values. The current front member is, then, updated based on these derivatives and a desired step in the constrained objectives. This so-called prediction step is used to move close to the next Pareto point. Due to the non-linearity of the underlying flow problem, an iterative correction step is additionally required to reach the front itself. The latter is based on a typical SQP approach for constrained optimization problems.

In this paper, a similar approach to the sequential constrained SOO problems discussed in [5] is adopted. Though the cost of the method still scales with the number of front points, the method is devised in the hope that each constrained SOO problem solved to move from one point to the next has a significantly lower cost than the alternative of starting each SOO problem from the initial geometry with a different weight value-set. The main interest of the paper is to utilize the KKT conditions in order to formulate systems of equations, the solution of which leads to the next Pareto point. In specific, two such systems are formulated. Both include, among other, the Hessian matrix of the Lagrangian. If this was to be computed, the total CPU cost would become proportional to N, [6]. Instead, a number of alternatives are proposed and evaluated. These include the Hessian approximation using BFGS, the computation of Hessian-vector products instead of the Hessian itself, [7], and the combined use of them.

Throughout this study, the continuous adjoint method [8, 9] is used to efficiently compute the gradients of the objective and constraint functions w.r.t. the design variables. To compute Hessian-vector products, a combination of Direct Differentiation (DD) and adjoint is utilized, which makes the cost independent of N. This method was inspired by the application of the truncated Newton method in shape optimization, [7, 10, 11], developed by the authors' group in the past.

The developed methods are evaluated in three aerodynamic shape optimization problems, with two objective functions each. All of the cases studied are for incompressible steady-state flows around isolated airfoils, targeting min. drag and max. lift, under an equality constraint on the airfoil area. The developed software was implemented within the OpenFOAM 2.3.1 open-source CFD toolbox framework.

2 The Pareto-tracing problem

Let us assume a MOO problem with $M=2$ objective functions f_1 and f_2, $\Lambda=1$ equality constraints ($c_g=0$) and N design variables ($b_n, n \in [1, N]$), for which $\mathbf{b} \in \mathcal{D}$, where $\mathcal{D}$ is a subdomain of $\mathbb{R}^N$; the presented method can readily be expanded to any other value of M and Λ. One way to determine the point on the Pareto front is by defining the targeted $M-1$ objective function values and using them as equality constraints while the algorithm minimizes the remaining one. Here, it is assumed that $\hat{f}_2$ is the target value for f_2, for which f_1 is to be minimized. To do so, a SOO problem must be solved by minimizing the following Lagrangian function

$$L\left(\mathbf{b}, \lambda_1, \lambda_2, \hat{f}_2\right) = f_1\left[\mathbf{U}(\mathbf{x}(\mathbf{b}), \mathbf{b}), \mathbf{x}(\mathbf{b}), \mathbf{b})\right] - \lambda_1 c_f\left[\mathbf{U}(\mathbf{x}(\mathbf{b}), \mathbf{b}), \mathbf{x}(\mathbf{b}), \mathbf{b}, \hat{f}_2\right] - \lambda_2 c_g[\mathbf{x}(\mathbf{b}), \mathbf{b}] \tag{1}$$

where λ_1, λ_2 are Lagrange multipliers,

$$c_f = f_2 - \hat{f}_2 \tag{2}$$

is the constraint associated with the desired f_2 value, $\mathbf{U}$ is the primal variables vector and $\mathbf{x}$ the nodal grid coordinates. The corresponding KKT first-order optimality conditions

$$\mathbf{K}(\mathbf{p}) = \begin{bmatrix} \frac{\partial L}{\partial \mathbf{b}}\big|_{\lambda_i, \hat{f}_2 = ct} \\ c_f \\ c_g \end{bmatrix} = \mathbf{0}. \tag{3}$$

should be met. By satisfying eqs. 3 for different $\hat{f}_2$ values, points on the Pareto front can be obtained in a sequential manner.

Let us define the variables $\mathbf{z} = [\mathbf{b}\ \lambda_1\ \lambda_2]^T \in I\!R^{N+2}$, $\mathbf{p} = \left[\mathbf{z}\ \hat{f}_2\right]^T \in I\!R^{N+3}$. Different approaches for satisfying the necessary KKT conditions are discussed in the next sections.

2.1 Linearizing the KKT conditions w.r.t. $\hat{f}_2$

According to [5], since eq. 3 is satisfied for any value of $\hat{f}_2$, its total derivative w.r.t. $\hat{f}_2$ is zero and hence,

$$\frac{\partial K_i}{\partial z_j}\bigg|_{\hat{f}_2 = ct} \frac{\partial z_j}{\partial \hat{f}_2} = -\frac{\partial K_i}{\partial \hat{f}_2}\bigg|_{\mathbf{z} = ct} \tag{4}$$

where repeated indices imply summation. Eq. 4 is nothing more than the application of the Implicit Function Theorem to eq. 3 and can be used to compute the derivatives of $\mathbf{z}$ w.r.t. $\hat{f}_2$, by solving the system

$$\underbrace{\begin{bmatrix} \mathcal{H} & -\mathbf{c'_f}^T & -\mathbf{c'_g}^T \\ \mathbf{c'_f} & 0 & 0 \\ \mathbf{c'_g} & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \mathbf{b''} \\ \lambda_1'' \\ \lambda_2'' \end{bmatrix}}_{\mathbf{X_1}} = \underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{\mathbf{Q_1}} \tag{5}$$

where $\mathcal{H}$ is the Hessian of L w.r.t. the design variables, $\mathcal{H} = \frac{\delta^2 L}{\delta \mathbf{b} \delta \mathbf{b}}$, $\mathbf{c'_\mu} = \frac{\partial c_\mu}{\partial \mathbf{b}}\big|_{\hat{f}_2 = ct}$, $\mu = f, g$, $\mathbf{b''} = \frac{\partial \mathbf{b}}{\partial \hat{f}_2}$ and $\boldsymbol{\lambda_i}'' = \frac{\partial \boldsymbol{\lambda_i}}{\partial \hat{f}_2}$, $i = 1, 2$. After solving eq. 5, b_n and λ_i are updated by using a first-order Taylor expansion, as follows

$$b_n^{new} = b_n^{old} + b_n'' \left(\hat{f}_2^{\ new} - \hat{f}_2^{\ old} \right) \tag{6a}$$

$$\lambda_i^{new} = \lambda_i^{old} + \lambda_i'' \left(\hat{f}_2^{\ new} - \hat{f}_2^{\ old} \right),\ i = 1,\ 2 \tag{6b}$$

approximating thus the next Pareto point. In eqs. 6, b_n^{old} and λ_i, $i = 1, 2$ correspond to the previously computed point on the front. Due to the non-linearity of the underlying fluid flow equations, the point computed through eqs. 6 has to be corrected in order for this to lay on the front (see section 2.3).

2.2 Linearizing the KKT conditions w.r.t. z

Another way to locate the next Pareto point is to satisfy eqs. 3 by linearizing them w.r.t. $\mathbf{z}$, which is nothing else than the SQP method for constrained SOO [12]. In SQP, the update of $\mathbf{z}$ is computed by solving the system

$$\underbrace{\begin{bmatrix} \mathcal{H} & -\mathbf{c'_f}^T & -\mathbf{c'_g}^T \\ \mathbf{c'_f} & 0 & 0 \\ \mathbf{c'_g} & 0 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \mathbf{\Delta b} \\ \Delta\lambda_1 \\ \Delta\lambda_2 \end{bmatrix}}_{X_2} = \underbrace{\begin{bmatrix} -\frac{\delta L}{\delta \mathbf{b}} \\ -c_f \\ -c_g \end{bmatrix}}_{Q_2} \tag{7}$$

and updating according to

$$b_n^{new} = b_n^{old} + \Delta b_n \tag{8a}$$

$$\lambda_i^{new} = \lambda_i^{old} + \Delta\lambda_i,\ i=1,\ 2 \tag{8b}$$

Due to the non-linearity of the fluid-flow equations, a number of iterations is required to find the solution of each constrained SOO problem. A difference between eqs. 5 and 7 is that the first system is valid only if the KKT conditions are satisfied for the current $\mathbf{z}$ (i.e. if the current point belongs to the Pareto front) while the latter can be solved irrespective of the starting point.

2.3 Discussion

In both eqs. 5 and 7, the LHS matrices A are identical and include the Hessian of L. In this paper, Hessian computations are avoided, since the relevant cost is equal to at least $N+2$ EFS (Equivalent Flow Solutions, i.e. as if the flow PDEs were solved instead), [6]. Instead, two different methods are utilized to solve either eq. 5 or 7: (a) the damped version of BFGS (dBFGS) [12] is used to iteratively approximate the Hessian, by solely using the gradient of L w.r.t. $\mathbf{b}$, computed by the continuous adjoint method, section 3. Once the LHS of eq. 5 or 7 is known (or approximated), the latter can be solved using the Range-Space approach, [12], which involves the inversion of the Hessian matrix (b) Linear Restarted GMRES [13] (hereafter abbreviated to GMRES) is used to iteratively solve eq. 5 or 7. GMRES requires only matrix-vector products, without needing the LHS matrix itself and, hence, the computation of $\mathcal{H}$ can be avoided. Instead, its product with known vectors is computed by using a combination of DD and adjoint. Each inner iteration of the GMRES method (i.e. matrix-vector multiplication) during the computation of bases within the Arnoldi procedure comes at an additional cost of 2 EFS, see section 4.

Three different combinations of the two systems resulting from the KKT conditions (eqs. 5 and 7) and their above-mentioned solution algorithms are devised in this paper, in order to trace the Pareto front. These read:

1. ***Method 1:***
   Method 1 follows a two-step approach to move from point k to point $k+1$, both on the Pareto front. In the first (prediction) step, eq. 5 is solved once to update the initial $\mathbf{b}$ and λ values. We opt on using GMRES to (approximately) solve eq. 5 since, in this first step, no previous gradient exists in order to construct a reasonable approximation to the Hessian using dBFGS. Due to the non-linear nature of the problem, this update is unable to yield a point exactly on the Pareto front. Hence, some correction steps are required in order to bring this point back to the front. However, eq. 5 is valid only on the Pareto front and, therefore, the same equations cannot be used to compute corrections. Instead, eq. 7 is solved to bring the point on the front. During the correction step, eq. 7 utilizes the dBFGS method to approximate the Hessian matrix. The reason for not using GMRES to solve eq. 7 in the correction steps is that each inner GMRES iteration comes at a cost of 2 additional EFS, making it prohibitively expensive if used extensively.

2. ***Method 2:***
   The predictor step based on eq. 5 is skipped and eq. 7 is exclusively solved. Eq. 7 is solved using GMRES in the first step when moving from one Pareto point to another and the dBFGS-driven SQP method is used in the subsequent steps, following the rational presented in Method 1. Method 2 is devised in order to assess whether solving eq. 5 instead of eq. 7 in the first step has a significant merit.

3. ***Method 3:***
   In method 3, the movement from one Pareto front point to another is achieved by solving eq. 7 supported by dBFGS. The latter requires information derived from the previous optimization cycle ($\delta L/\delta b_n$, Δb_n) to update the Hessian matrix in the LHS of eq. 7. However, at the first optimization cycle, when moving between two Pareto points, this information is not available. Thus, the Hessian is initialized with the identity matrix which, practically, corresponds to a steepest descent step for minimizing f_1.

A point on the Pareto front should be available to use any of the aforementioned methods. The first Pareto point is computed by using SQP to solve a SOO problem in which the flow constraint is neglected. This is equivalent to solving a SOO problem with weights equal to 1 and 0 for f_1 and f_2, respectively.

In eqs. 5 and 7, all first-order derivatives w.r.t. b_n are computed using the continuous adjoint method, as presented in the next section. The treatment of the second-order derivative ($\frac{\delta^2 L}{\delta b_n \delta b_m}$) and the (iterative) solution scheme of eqs. 5 and 7 are presented in section 4.

3 The Continuous adjoint method for incompressible flows

The gradient of the objective functions w.r.t. the design variables is computed with the continuous adjoint method [8,9] at a computational cost independent of N. In what follows, the derivation of the adjoint PDEs is briefly presented, for steady-state laminar 2D incompressible flows. The primal system of equations is

$$R^p = -\frac{\partial v_j}{\partial x_j} = 0 \tag{9}$$

$$R^v_i = v_j \frac{\partial v_i}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} + \frac{\partial p}{\partial x_i} = 0 \ , \qquad i = 1,2 \tag{10}$$

where v_i and p are the velocity and pressure fields, $\tau_{ij} = \nu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right)$ are the stress tensor components and ν the constant viscosity. For the cases studied in this paper, f_1 and f_2 stand for the drag and lift exerted on an airfoil; a general expression for any component of the exerted forces is

$$F_{forces} = \int_{S_W} \left(p\delta^j_i - \tau_{ij} \right) r_i n_j dS \tag{11}$$

where $\mathbf{r}$ is the direction along which the force vector is projected, namely the direction of the freestream velocity for drag and its normal for lift. S_W stands for the airfoil wall and $\mathbf{n}$ is the outward (from the fluid domain) unit normal vector. The geometrical equality constraint, which in this paper stands for the preservation of the airfoil area, is given by

$$c_g = \int_\Omega d\Omega - \Omega_{initial} = 0 \tag{12}$$

with Ω being the volume of the fluid domain and $\Omega_{initial}$ its initial area. Maintaining the area of the fluid domain practically preserves the airfoil area, since the freestream boundaries are kept fixed. By imposing this constraint, the trivial solution of drag minimization which leads to an almost zero-thickness airfoil, is avoided.

Starting point of the adjoint method is the differentiation of the augmented objective function

$$L_{aug} = L + \int_\Omega q R^p d\Omega + \int_\Omega u_i R^v_i d\Omega \tag{13}$$

with L given by eq. 1, u_i being the adjoint velocity components and q the adjoint pressure. After a lengthy mathematical development, which is beyond the scope of this paper, see [8], field integrals involving $\delta v_i / \delta b_n$ and $\delta p / \delta b_n$ are eliminated by satisfying the adjoint PDEs, which read

$$R^q = -\frac{\partial u_j}{\partial x_j} = 0 \tag{14}$$

$$R^u_i = u_j \frac{\partial v_j}{\partial x_i} - \frac{\partial \left(u_i v_j \right)}{\partial x_j} - \frac{\partial \tau^a_{ij}}{\partial x_j} + \frac{\partial q}{\partial x_i} = 0 \ , \ i = 1,2 \tag{15}$$

where $\tau^a_{ij} = \nu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$ stand for the adjoint stress tensor components. The corresponding boundary conditions arise after proper mathematical treatment of the boundary integrals including the variations of p and v_i, which can also be found in [8]. Though $\delta L / \delta b_n$ could be computed by solving a single adjoint problem,

the necessity of acquiring $\delta c_f/\delta b_n$ separately (see eqs. 5 and 7), leads to the solution of two adjoint problems for computing $\delta f_1/\delta b_n$ and $\delta c_f/\delta b_n$. Note that c_g is a geometric constraint and, as such, its derivative is directly computed by an analytical expression. Hence, $\delta L/\delta b_n$ reads

$$\frac{\delta L}{\delta b_n} = \frac{\delta f_1}{\delta b_n} - \lambda_1 \frac{\delta c_f}{\delta b_n} - \lambda_2 \frac{\delta c_g}{\delta b_n} \tag{16}$$

Regarding f_1, f_2, the gradient of F_{forces} w.r.t. b_n is

$$\begin{aligned}\frac{\delta F_{forces}}{\delta b_n} = &\int_\Omega \underbrace{\left[-u_i v_j \frac{\partial v_i}{\partial x_k} - u_j \frac{\partial p}{\partial x_k} - \tau_{ij}^a \frac{\partial v_i}{\partial x_k} + u_i \frac{\partial \tau_{ij}}{\partial x_k} + q \frac{\partial v_j}{\partial x_k}\right]}_{A_{jk}} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega \\ &+ \int_{S_W} \left(p\delta_i^j - \tau_{ij}\right) r_i \frac{\delta\left(n_j dS\right)}{\delta b_n}\end{aligned} \tag{17}$$

Also, the flow constraint gradient is computed as $\frac{\delta c_f}{\delta b_n} = \frac{\delta f_2}{\delta b_n}$ and the gradient of c_g reads

$$\frac{\delta c_g}{\delta b_n} = \int_\Omega \frac{\partial}{\partial x_k}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega \tag{18}$$

where in this paper, $\delta x_k/\delta b_n$ can be derived by differentiating the volumetric B-Splines morpher w.r.t. b_n.

4 Computation of matrix-vector products

Let us write eqs. 5 and 7, in the general form of $A\mathbf{X_d} = \mathbf{Q_d}$, $d = 1, 2$. Such systems can be solved iteratively using a Krylov subspace method (in this case GMRES) which requires only matrix-vector products and avoids the computation of A which includes the Hessian of L. The product of A with a vector $\mathbf{s}$ reads

$$A\mathbf{s} = \begin{bmatrix} \mathbf{P} \\ \sum_{m=1}^{N} \frac{\delta c_f}{\delta b_m} s_m \\ \sum_{m=1}^{N} \frac{\delta c_g}{\delta b_m} s_m \end{bmatrix} \tag{19}$$

where the N components of $\mathbf{P}$ are computed as

$$P_n = \sum_{m=1}^{N} \frac{\delta^2 L}{\delta b_n \delta b_m} s_m - \frac{\delta c_f}{\delta b_n} s_{N+1} - \frac{\delta c_g}{\delta b_n} s_{N+2}, \ n = 1, ..., N \tag{20}$$

In eq. 20, the Hessian-vector product is computed as

$$\frac{\delta^2 L}{\delta b_n \delta b_m} s_m = \frac{\delta^2 f_1}{\delta b_n \delta b_m} s_m - \lambda_1 \frac{\delta^2 c_f}{\delta b_n \delta b_m} s_m - \lambda_2 \frac{\delta^2 c_g}{\delta b_n \delta b_m} s_m, \ n = 1, ..., N \tag{21}$$

and is the toughest and most expensive part of computing $A\mathbf{s}$.

The following notation is introduced for any arbitrary quantity Φ

$$\overline{\Phi} = \frac{\delta \Phi}{\delta b_m} s_m \tag{22}$$

with its gradient being equal to, [10],

$$\overline{\frac{\partial \Phi}{\partial x_k}} = \frac{\partial \overline{\Phi}}{\partial x_k} - \frac{\partial \Phi}{\partial x_\lambda} \overline{\frac{\partial x_\lambda}{\partial x_k}} \tag{23}$$

Since

$$\frac{\delta^2 c_f}{\delta b_n \delta b_m} s_m = \frac{\delta^2 f_2}{\delta b_n \delta b_m} s_m \tag{24}$$

the first two terms on the RHS of eq. 21 require the products of the second derivative of eq. 11 and $\mathbf{s}$. After differentiating the latter equation w.r.t. $\mathbf{b}$ and taking the inner product with $\mathbf{s}$, we obtain

$$\begin{aligned}\frac{\delta^2 F_{forces}}{\delta b_n \delta b_m} s_m = &\int_\Omega \overline{A_{jk}} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega + \int_\Omega A_{jk} \frac{\partial}{\partial x_j}\left(\frac{\delta^2 x_k}{\delta b_n \delta b_m}\right) s_m d\Omega - \int_\Omega A_{jk} \frac{\partial}{\partial x_\lambda}\left(\frac{\delta x_k}{\delta b_n}\right) \frac{\partial \overline{x_\lambda}}{\partial x_j} d\Omega \\ &+ \int_\Omega A_{jk} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) \frac{\partial \overline{x_\lambda}}{\partial x_\lambda} d\Omega + \int_{S_W} \left[\overline{p}\delta_i^j - \nu\left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right) + \nu\left(\frac{\partial v_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right] r_i \frac{\delta(n_j dS)}{\delta b_n} \\ &+ \int_{S_W} \left(p\delta_i^j - \tau_{ij}\right) r_i \overline{\frac{\delta(n_j dS)}{\delta b_n}}\end{aligned} \tag{25}$$

where

$$\begin{aligned}\overline{A_{jk}} = &-\overline{u_i} v_j \frac{\partial v_i}{\partial x_k} - u_i \overline{v_j} \frac{\partial v_i}{\partial x_k} - u_i v_j \overline{\frac{\partial v_i}{\partial x_k}} + u_i v_j \frac{\partial v_i}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_k} - \overline{u_j}\frac{\partial p}{\partial x_k} - u_j \frac{\partial \overline{p}}{\partial x_k} + u_j \frac{\partial p}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_k} - \nu\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\frac{\partial v_i}{\partial x_k} \\ &+ \nu\left(\frac{\partial u_i}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_j} + \frac{\partial u_j}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_i}\right)\frac{\partial v_i}{\partial x_k} - \nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial \overline{v_i}}{\partial x_k} + \nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\frac{\partial v_i}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_k} \\ &+ \overline{u_i}\frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] + u_i \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right)\right] - u_i \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial v_i}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_j} + \frac{\partial v_j}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_i}\right)\right] \\ &- u_i \frac{\partial}{\partial x_\lambda}\left[\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_\lambda}}{\partial x_k} + \overline{q}\frac{\partial v_j}{\partial x_k} + q\frac{\partial \overline{v_j}}{\partial x_k} - q\frac{\partial v_j}{\partial x_\lambda}\frac{\partial \overline{x_\lambda}}{\partial x_k}\end{aligned} \tag{26}$$

The governing equations for $\overline{v_i}$ and $\overline{p}$ are formulated by directly differentiating the flow PDEs and projecting them to $\mathbf{s}$, i.e.

$$\overline{R^p} = \frac{\partial \overline{v_j}}{\partial x_j} - \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} = 0 \tag{27}$$

$$\begin{aligned}\overline{R_i^v} = &\frac{\partial(\overline{v_i} v_j)}{\partial x_j} + \frac{\partial(v_i \overline{v_j})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right)\right] + \frac{\partial \overline{p}}{\partial x_i} - \frac{\partial(v_i v_j)}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} \\ &+ \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial v_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_k}}{\partial x_j} - \frac{\partial p}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i} = 0\end{aligned} \tag{28}$$

Following the same process for the adjoint equations yields

$$\overline{R^q} = \frac{\partial \overline{u_j}}{\partial x_j} - \frac{\partial u_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} = 0 \tag{29}$$

$$\begin{aligned}\overline{R_i^u} = &\overline{u_j}\frac{\partial v_j}{\partial x_i} + u_j \frac{\partial \overline{v_j}}{\partial x_i} - \frac{\partial(\overline{u_i} v_j)}{\partial x_j} - \frac{\partial(u_i \overline{v_j})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\right] + \frac{\partial \overline{q}}{\partial x_i} - u_j \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i} \\ &+ \frac{\partial(v_j u_i)}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right] + \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_k}}{\partial x_j} - \frac{\partial q}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i} = 0\end{aligned} \tag{30}$$

which can be solved for $\overline{q}$ and $\overline{u}_i$. The boundary conditions for $\overline{v_i}, \overline{p}, \overline{u_i}, \overline{q}$ result from the inner product of the DD of the primal and adjoint boundary conditions with $\mathbf{s}$ in a straightforward manner. As already discussed, two different systems of adjoint PDEs are solved in each optimization cycle (separately for f_1 and c_f). Nevertheless, the directly differentiated adjoint equations (eqs. 29 and 30) are solved only once per matrix-vector computation (i.e. GMRES inner iteration). This is achieved by taking advantage of the linearity of the adjoint equations (eqs. 14, 15) which allows the combination of the two different vectors of

adjoint variables corresponding to f_1 and c_f in a single adjoint variable vector, as

$$\boldsymbol{\Psi}\big|_{un} = \boldsymbol{\Psi}\big|_{f_1} - \lambda_1 \boldsymbol{\Psi}\big|_{c_f} \tag{31}$$

where $\boldsymbol{\Psi} = [u_i \; q]^T$ and the subscripts f_1, c_f and un (standing for unified) denote the adjoint variables computed by solving the adjoint equations for f_1, c_f and their linear combination respectively. Once the adjoint variables Ψ_{un} are assembled, they are used to solve eqs. 29, 30. Then, the projected second derivatives of F_{forces} (eq. 25) can be computed. Finally, the product of the Hessian of c_g and $\mathbf{s}$ is computed as

$$\frac{\delta^2 c_g}{\delta b_n \delta b_m} s_m = \int_\Omega \frac{\partial}{\partial x_k}\left(\frac{\delta x_k}{\delta b_n}\right) \frac{\partial \overline{x_\lambda}}{\partial x_\lambda} d\Omega + \int_\Omega \frac{\partial}{\partial x_k}\left(\frac{\delta^2 x_k}{\delta b_n \delta b_m}\right) s_m d\Omega - \int_\Omega \frac{\partial}{\partial x_\lambda}\left(\frac{\delta x_k}{\delta b_n}\right) \frac{\partial \overline{x_\lambda}}{\partial x_k} d\Omega \tag{32}$$

5 Overview of the three methods

In this section, an overview of Methods 1 to 3, as described in section 2.3 is presented, in order to compute N_P points on the Pareto front. The number of GMRES inner iterations is denoted by N_{bases}.

5.1 Method 1

- **Step 1:**
  Perform a SOO to locate the first point on the Pareto front, which requires N_{SOO} optimization cycles. If we neglect c_f, i.e. perform a drag minimization under the constraint of constant airfoil area, the cost of this step is equal to $(2N_{SOO} - 1)$ EFS.
- **Step 2:**
  A new value is given to $\hat{f}_2$.
- **Step 3:**
  Solve the primal, 9, 10, and adjoint PDEs, 14, 15, (once for f_1 and once for c_f) at the cost of 3 EFS.
- **Step 4–Prediction Step:**
  The GMRES method is utilized to solve eq. 5 using N_{bases} bases. For each basis:
  1. Solve the DD of the primal (eqs. 27, 28) and adjoint PDEs (eqs. 29, 30) at the cost of 2 EFS.
  2. Compute the matrix-vector product, see eqs. 19.
  3. Perform one iteration of the GMRES method.
- **Step 5:**
  Update b_n and λ_i, $i = 1, 2$ using eqs. 6.
- **Step 6:**
  Solve eqs. 9, 10 and evaluate the objective function eq. 11 and the constraints 2 and 12. If constraints are satisfied and the objective function does not change, then this is a Pareto point and the algorithm moves to **Step 2**; else, continues.
- **Step 7 – Correction Steps:**
  Perform N_C correction steps, by solving the eq. 7 with the following substeps:
  1. Solve adjoint eqs. 14, 15 PDEs at the cost of 2 EFS.
  2. Compute $\delta L / \delta b_n$, using eqs. 16–18.
  3. Update b_n and λ_i using eqs. 8.
- **Step 8:**
  Return to **Step 6**.

The overall cost $N_{Method1}$ is equal to

$$N_{Method1} = \left[2N_{SOO} - 1 + (N_P - 1)\left(3 + 2N_{bases} + 3\overline{N_C}\right)\right] EFS \tag{33}$$

with $\overline{N_C}$ being the average number of correction steps needed to reach each Pareto point.

5.2 Method 2

In Method 2, the prediction step used in **Step 4** of section 5.1 is substituted by solving eq. 7 once, using the GMRES method. The cost is quantified in the same way, i.e.

$$N_{Method2} = \left[2N_{SOO} - 1 + (N_P - 1)\left(3 + 2N_{bases} + 3\overline{N_C}\right)\right] EFS \tag{34}$$

The computational cost for Methods 1 and 2 is the same. However, due to solving different equations in **Step 4**, the two methods can follow different convergence paths and $N_{Method1}$ and $N_{Method2}$ may differ in general.

5.3 Method 3

- **Step 1:**
  Perform a SOO to find the first point on the Pareto front in N_{SOO} cycles, at a cost of $2N_{SOO}-1$ EFS.
- **Step 2:**
  Redefine $\hat{f}_2$.
- **Step 3:**
  Solve the primal (eqs. 9, 10) and adjoint (eqs. 14, 15) PDEs (for f_1) at the cost of 2 EFS.
- **Step 4:**
  Update b_n using steepest descent, i.e. $b_n^{new} = b_n^{old} - \eta \frac{\delta f_1}{\delta b_n}$, where η is the step length.
- **Step 6:**
  Solve eqs. 9, 10 and evaluate the objective function eq. 11 and the constraints, eqs. 2 and 12. If the constraints are satisfied and the objective function does not change, then the Pareto point has been identified and the algorithm moves to **Step 2**; else, continues.
- **Step 7 – Correction Steps:**
  Perform N_C steps, by solving eq. 7
  1. Solve adjoint eqs. 14, 15 PDEs (for f_1 and c_f) at the cost of 2 EFS.
  2. Compute $\delta L/\delta b_n$, using eqs. 16–18.
  3. Update b_n and λ_i, using eqs. 8.
- **Step 8:**
  Return to **Step 6**.

The cost of the Method 3 is

$$N_{Method3} = \left[2N_{SOO} - 1 + (N_P - 1)\left(2 + 3\overline{N_C}\right)\right] EFS \tag{35}$$

In **Step 4** of Method 3, steepest descent is used, neglecting the optimization constraints. Even though **Step 4** of Methods 1 and 2 requires the solution of additional PDEs, these methods can take constraints into consideration and are used to hopefully provide a good approximation of the next front point. The validity of this hypothesis is examined in section 6.

6 Applications

Methods 1 to 3, as presented in section 2.3 are used to compute the fronts of non-dominated solutions in two-objective optimization problems that use the NACA0012 and NACA4412 as the starting geometries. Three cases are studied, in which the minimization of drag and maximization of lift are targeted. The already described geometrical constraint of constant airfoil area is used in order to avoid non-realistic solutions. The flow is laminar and the parameterization is based on volumetric B-Splines control boxes [14], each consisting of 5×4 control points, with a basis-function degree of 3 for both directions.

Before examining the cases, a note on the computational cost of Method 1 should be made. For a zero initialization, due to the form of the RHS of eq. 5, the first GMRES iteration corresponds to a multiplication of the Hessian with the zero vector, which is free of cost. Thus, if N_{bases} are selected, $2\,(N_{bases}-1)$ PDEs

have to be solved. In each case, the same convergence criteria are used for all Methods, to have a fair comparison between them.

6.1 Pareto front computation starting from the NACA4412 airfoil

6.1.1 Case 1: Performance of Methods 1 to 3 for a small design space

In case 1, Methods 1 to 3 are used to compute the Pareto front, starting from the NACA4412 airfoil. The mesh consists of $\sim 60k$ cells, the Reynolds number is $Re = 1000$, the freestream flow angle is 1^o and the target lift change between two Pareto points is $\Delta\hat{f}_2 = 10\%$ of the current $\hat{f}_2$ value.

The purpose of this study is to examine the capability of Step 4 of Methods 1 and 2 to compute a good approximation of the next Pareto point when eqs. 5 and 7 are solved exactly using GMRES. To do so, $N_{bases} = N+2$. To have an affordable CPU cost, the design space is chosen to consist of 2 design variables ($N = 2$, $N_{bases} = 4$), which are the displacements of the 2 control points depicted in fig. 1, along the y axis. The Pareto front, along with the dominated point pertaining to the baseline geometry are presented in fig. 2. All methods produce practically the same points on the front. Also, in figs. 3 to 5, all intermediate

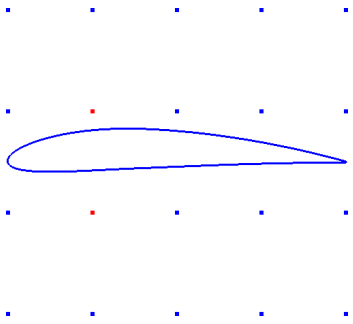


Figure 1: Case 1: Baseline geometry along with the parameterization. Control points in blue are kept fixed during the optimization while the red ones are allowed to move along the y axis.

steps when moving from one Pareto point to another, computed by Methods 1 to 3 are presented. It can be observed that, in Methods 1 and 2, Step 4 points to the direction of the next point, while this does not happen for Method 3.

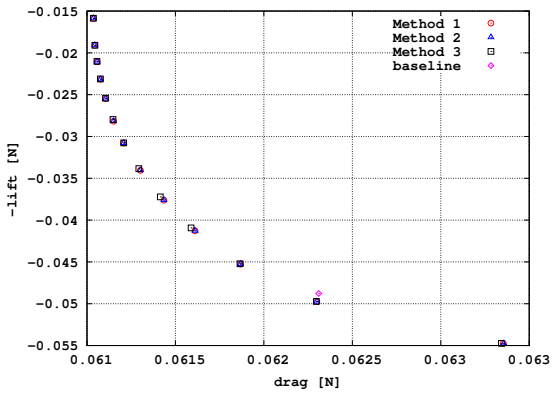


Figure 2: Case 1: Baseline geometry ("baseline") and Pareto fronts computed by Methods 1 to 3.

However, in terms of computational cost, Methods 1 and 2 are more expensive than Method 3, see table 1. Also, for the reason discussed in the introduction of this section, the cost of Method 2 is higher than that of Method 1, since two more PDEs are solved at Step 4. The shapes corresponding to the 1st (min. drag), 7th and 13rd (max. lift) Pareto points are presented in fig. 6. To minimize drag, the part of the airfoil close to the leading edge moves downwards whereas, to maximize lift, the same part moves upwards.

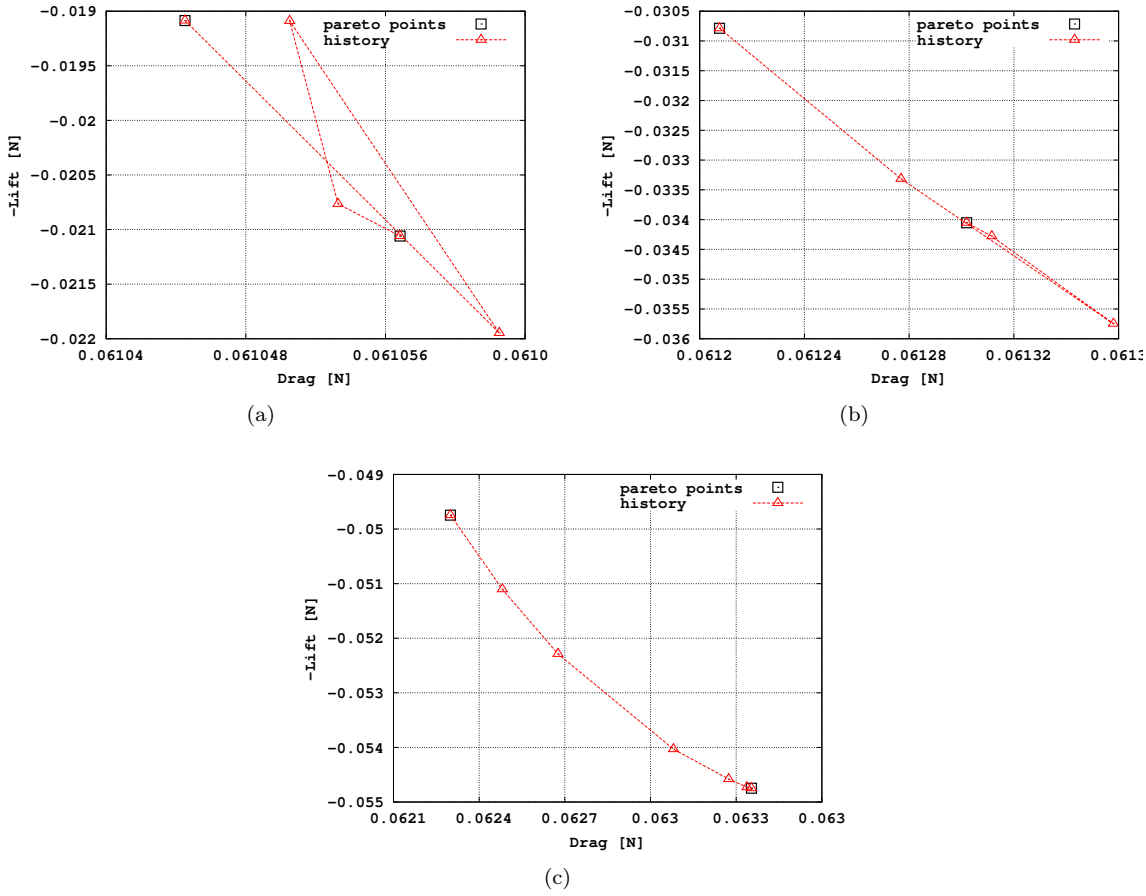


Figure 3: Case 1: Transitions from (a) 2nd to 3rd, (b) 7th to 8th, (c) 12th to 13th Pareto points, using Method 1. Black squares represent the Pareto points and red triangles the intermediate steps to move from one Pareto point to the next. Transition from the top Pareto point to the one at the bottom.

6.1.2 Case 2: Performance of Methods 1 to 3 for a larger design space

In Case 1, eqs. 5 and 7 were exactly solved using GMRES at the first step when moving to the next Pareto point, since the small number of design variables ($N = 2$) made it possible to afford the max. number of N_{bases}. In Case 2, the optimization problem of Case 1 is revisited to investigate the performance of Methods 1 and 2, when fewer than the max. number of bases are used in GMRES. The first and last control points along the x and y directions of the control box are kept fixed and the remaining 6 control points are allowed to move along the x and y axis. Thus, $N = 12$ and 3 bases are used in the GMRES solver to perform Step 4, in both Methods 1 and 2. The points on the Pareto front, as computed with Methods 1 to 3, are given in fig. 7. In figs. 8 to 10, the transitions from one Pareto point to another, as computed by Methods 1 to 3 is presented for a sample of the front points. It can be observed that the solution provided at Step 4 of Methods 1 to 2 manages to lead close to the next Pareto point, in contrast to Method 3 in which Step 4 fails to do so, since the constraints are not taken into consideration. However, Method 3 still outperforms the others in terms of CPU cost as seen in table 2. In figs. 11, the baseline geometry is compared with three points of the Pareto front; the two extreme points and one between them. In figs. 12, the velocity magnitude fields are depicted for the geometries of fig. 11 and in figs. 13, the corresponding C_p and C_f distributions are presented. Regarding lift, the point where the local load (pressure difference between the pressure and suctions sides) becomes negative moves from 1/10 of the chord length to 8/10; a similar trend is observed for minimizing drag. Also, the C_f of the suction side close to the trailing edge becomes higher than zero (as opposed to the baseline geometry), increasing local lift production. In order to minimize drag,

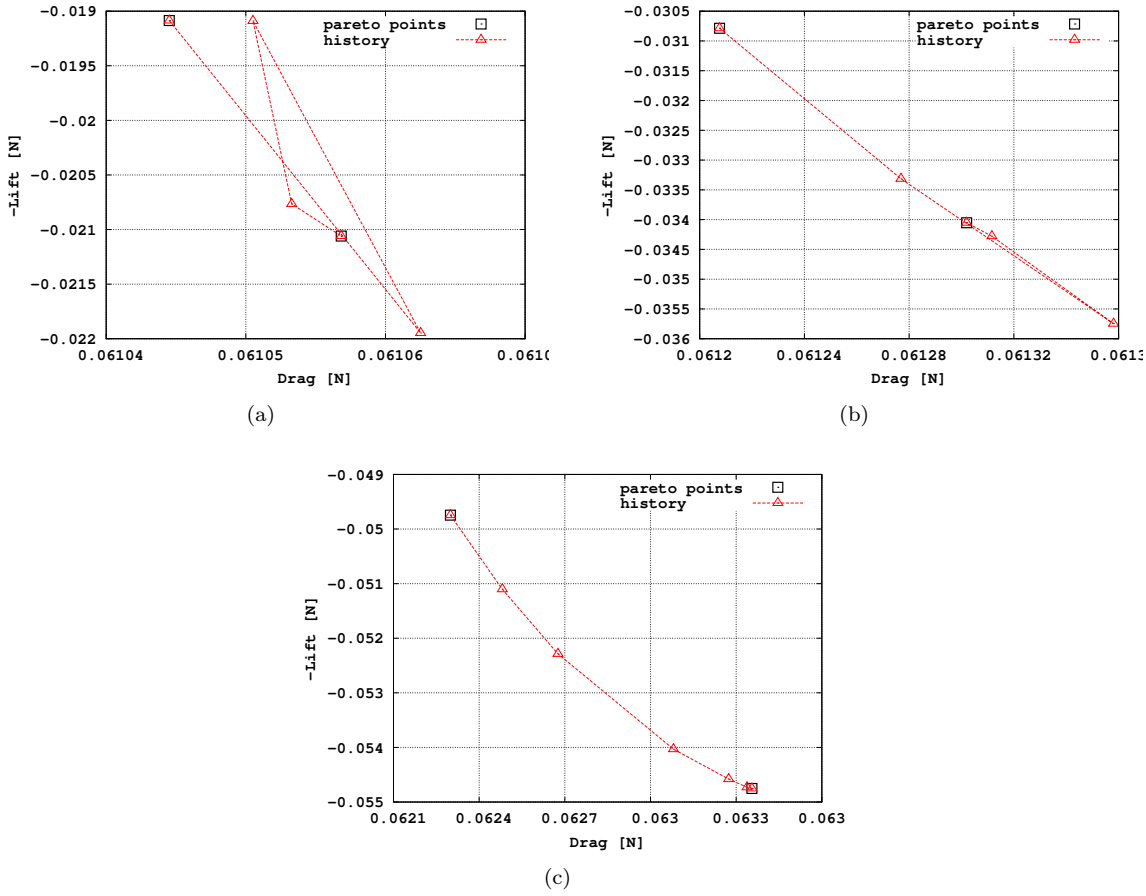


Figure 4: Case 1: Transitions from (a) 2nd to 3rd, (b) 7th to 8th, (c) 12th to 13th Pareto points, using Method 2. Notation as in fig. 3.

the C_p difference becomes smaller than in the baseline geometry and from the 4/10 of the chord length on, it becomes almost equal to zero.

6.2 Case 3: Pareto front computation starting from the NACA0012 airfoil

In Case 3, the study conducted in Case 2 is repeated for a second airfoil, the NACA0012 one. In this case, the mesh consists of $\sim 38k$ cells, the Reynolds number is $Re = 1000$, the farfield velocity angle is 2 degrees and, after computing the first front point, a constant change of $\Delta\hat{f}_2 = 5\%$ imposed at the current $\hat{f}_2$ value is applied to move from one Pareto point to another. In order to solve eqs. 5 and 7 with GMRES, 2 and 1 bases are used respectively (see discussion in the introduction of section 6).

The fronts computed by Methods 1 to 3 are presented in fig. 14, along with the point corresponding to the baseline geometry. The computational cost is summarized in table 3 and the intermediate steps when moving between Pareto points is presented for a sample of points along the front in figs. 15 to 17.

Even though a small number of bases are used, the solution provided by GMRES at Step 4, in both Methods 1 and 2, manages to approach the next Pareto point. However, even in this case, Method 3 has a smaller computational cost even though the first step in each transition to the next point moves to the wrong direction. In fig. 18, the geometries for three Pareto points compared with the baseline geometry are depicted and in 20, the corresponding C_p and C_f distributions are illustrated. For the same geometries, the velocity magnitude fields are presented in figs. 19. In order to reduce drag, the suction side moves downwards at the first half of the chord length and upwards at the second half. Smaller changes occur at the pressure side. In this way, the difference in the static pressure between the pressure and suction side decreases from

Method	Computational cost [EFS]		Average Computational cost [EFS]	
	1st SOO included	1st SOO excluded	1st SOO included	1st SOO excluded
Method 1	211	193	21.1	16.08
Method 2	235	217	23.5	18.08
Method 3	124	106	12.4	8.83

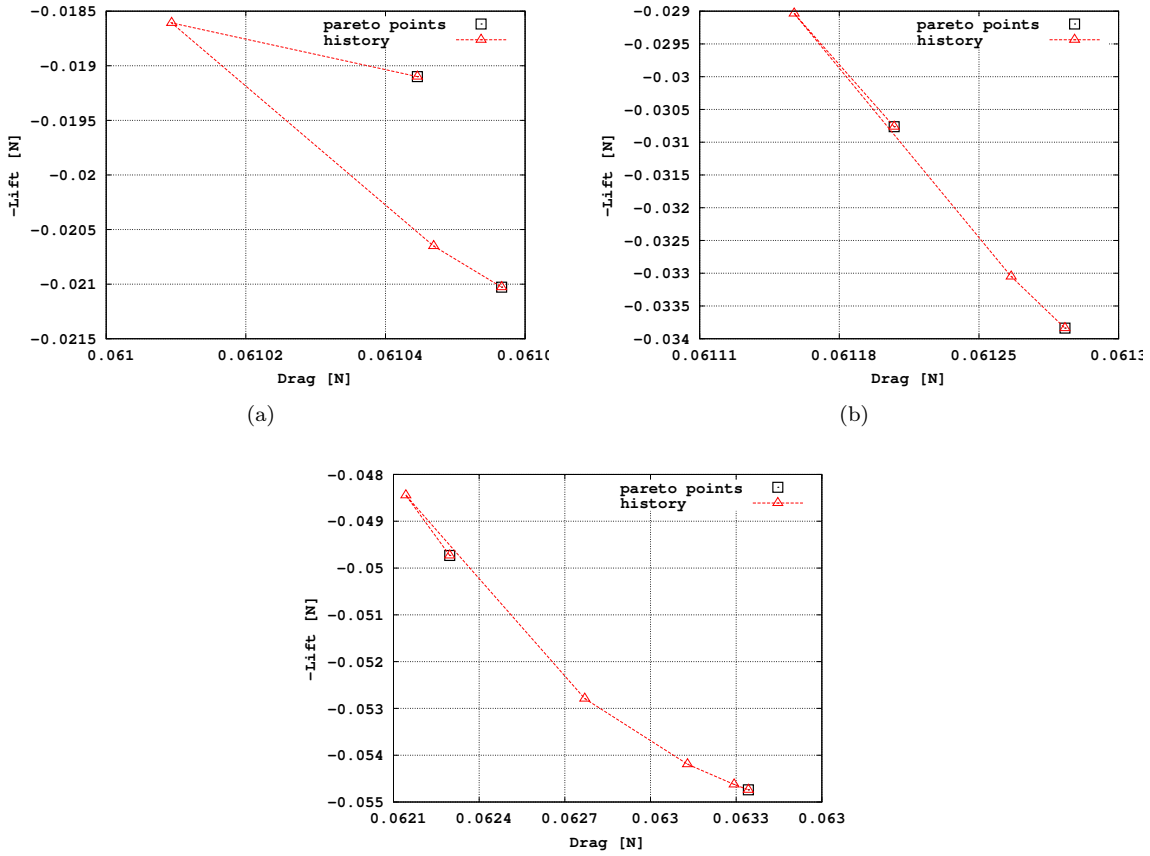
Table 1: Case 1: Computational cost (in EFS) of Methods 1 to 3, to compute 13 points on the Pareto front, with (2nd column) and without (3rd) counting the EFS to reach the first point of the Pareto front. The average cost concerns the EFS needed to reach each point on the Pareto front (4th and 5th columns). The SOO to compute the first front point costed 19 EFS.

Method	Computational cost [EFS]		Average Computational cost [EFS]	
	1st SOO included	1st SOO excluded	1st SOO included	1st SOO excluded
Method 1	156	124	15.6	13.78
Method 2	174	142	17.4	15.78
Method 3	102	70	10.2	7.78

Table 2: Case 2: Computational cost (in EFS) of Methods 1 to 3, to compute 10 points on the Pareto front, with (2nd column) and without (3rd) counting the EFS to reach the first point of the Pareto front. Notation as in table 1. The SOO to compute the first point of the front costed 33 EFS.

Method	Computational cost [EFS]		Average Computational cost [EFS]	
	1st SOO included	1st SOO excluded	1st SOO included	1st SOO excluded
Method 1	141	123	12.81	12.3
Method 2	153	135	13.91	13.5
Method 3	97	79	8.81	7.9

Table 3: Case 3: Computational cost in EFS of Methods 1 to 3, to compute 11 points on the Pareto front, with (2nd column) and without (3rd) counting the EFS to reach the first point of the Pareto front. Notation as in table 1. The SOO to compute the first point of the front costed 19 EFS.



(a) (b)

(c)

Figure 5: Case 1: Transitions from (a) 2nd to 3rd, (b) 7th to 8th, (c) 12th to 13th Pareto points, using Method 3. Notation as in fig. 3.

(a) (b) (c)

Figure 6: Case 1: Comparison of the baseline geometry (blue) with the optimized ones (red) referring to (a) the 1st (min. drag), (b) the 7th and (c) the 13th (max. lift) Pareto points.

the 1/5 of the chord length on. To increase lift, the airfoil becomes thinner close to the leading edge and the part of the pressure side closer to the trailing edge moves downwards. That change in the geometry resulted in increasing the difference in the static pressure between the pressure and suction sides along the whole chord length.

7 Conclusion

In this paper, three gradient-based algorithms, namely Methods 1 to 3, were compared for computing the front of non-dominated solutions in multi-objective aerodynamic optimization problems. In the cases studied, it was observed that the first step of an SQP-based method initialized with the identity matrix (Method 3) moved in the opposite direction of the next Pareto point, due to the inability of taking into consideration constraints. In an attempt to mitigate this effect, Methods 1 and 2 were implemented. The latter use a combination of Direct Differentiation and adjoint to compute the Hessian-vector products existing on the

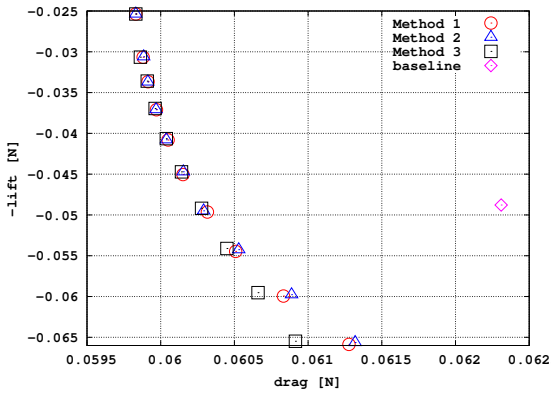


Figure 7: Case 2: Baseline geometry point ("baseline") and Pareto fronts computed by Methods 1 to 3.

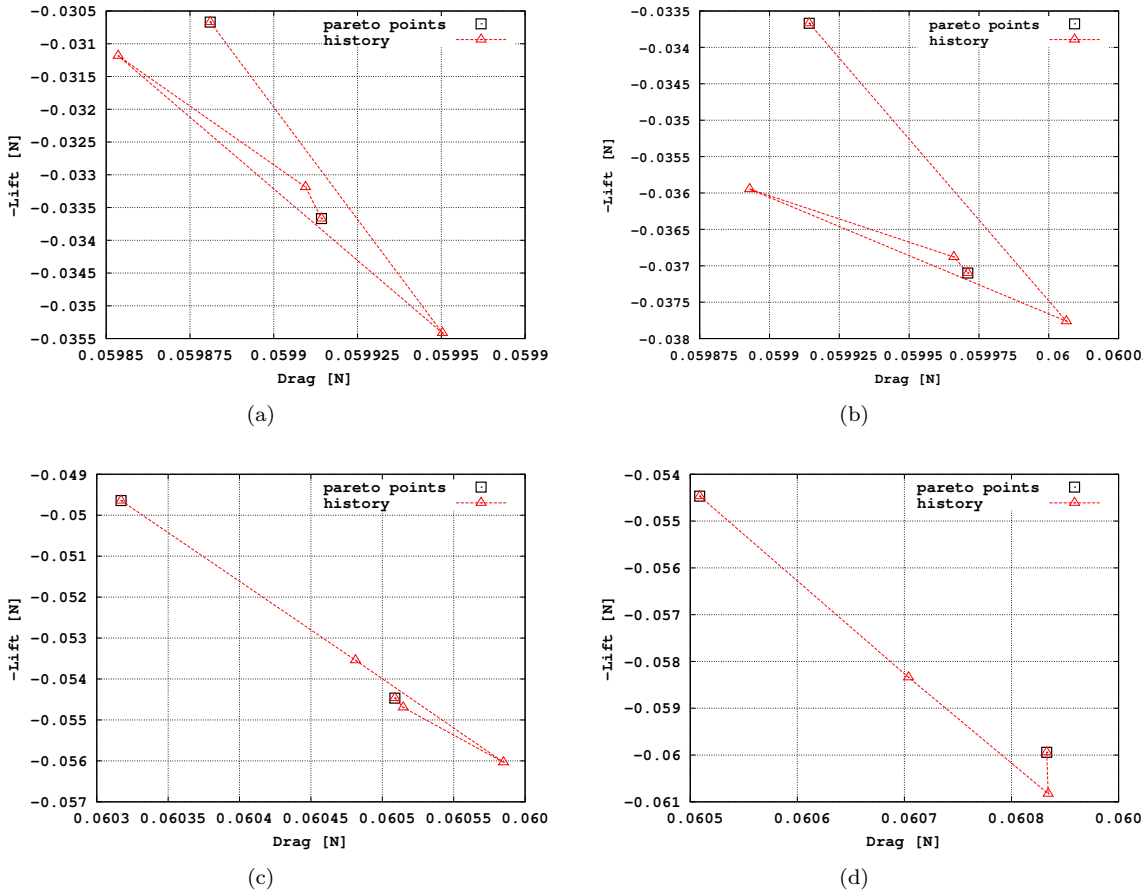


Figure 8: Case 2: Transitions from (a) the 2nd to the 3rd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 8th to the 9th Pareto points, using Method 1. Notation as in fig. 3.

LHS of two linear systems emerging from the KKT conditions and (approximately) solve the aforementioned systems using GMRES. Methods 1 and 2 are devised in an attempt to take constraints into consideration even in the first step of moving from a Pareto point to the next and, hence, speed up the whole process. For all cases examined, Method 1 indeed manages to move in the vicinity of the next Pareto point, even if the corresponding linear system is not solved exactly (i.e. even if the number of GMRES inner iterations

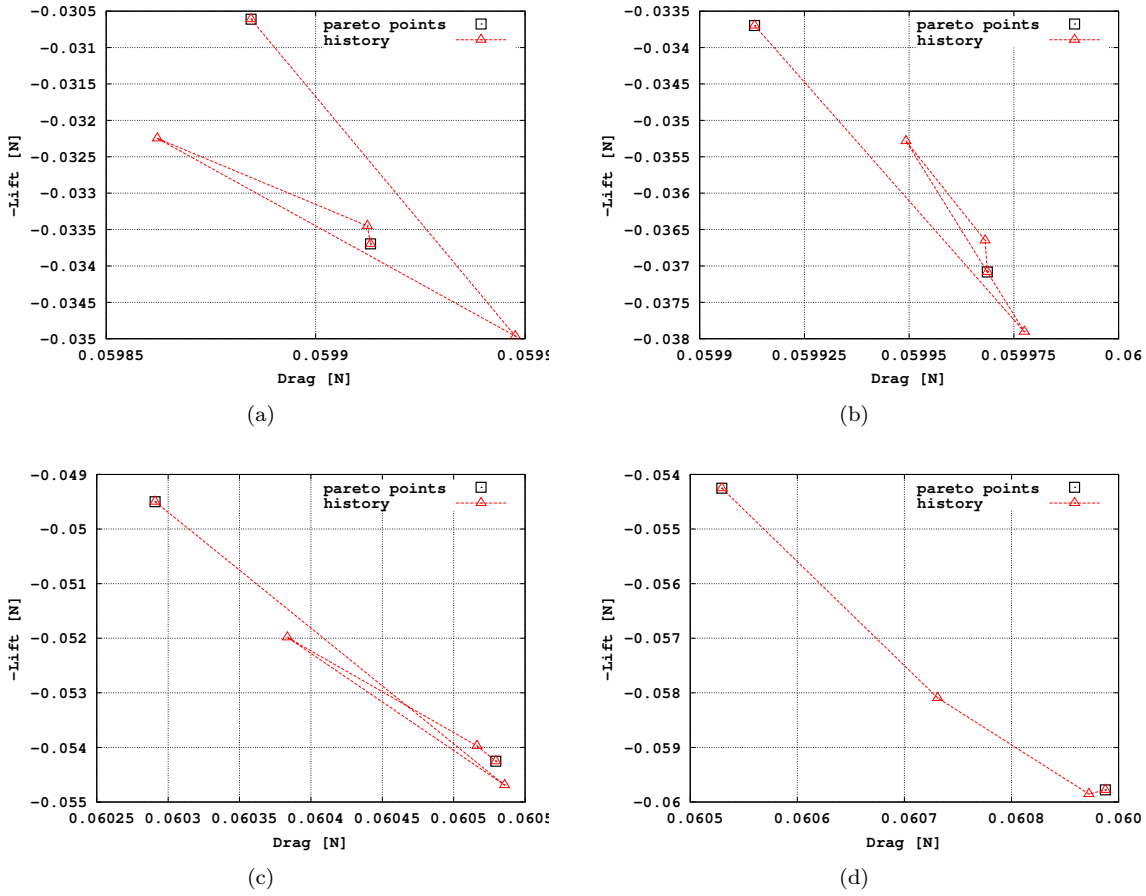


Figure 9: Case 2: Transitions from (a) the 2nd to the 3rd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 8th to the 9th Pareto points, using Method 2. Notation as in fig. 3.

used is less than the size of the linear system). Method 2 does so for two out of the three examined cases. However, each inner iteration of GMRES requires the computation of an Hessian-vector product that costs 2 Equivalent Flow Solutions. In the end, this additional cost makes Methods 1 and 2 more expensive than Method 3, even if the latter is based in a much worse initialization each time a new Pareto point is computed.

Acknowledgments

The PhD thesis of the first author is funded by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI).

References

[1] D.W. Zingg, M. Nemec, and T.H. Pulliam. A comparative evaluation of genetic and gradient- based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics*, 17:103–126, 2012.

[2] S. Shankaran and B. Barr. Efficient gradient-based algorithms for the construction of pareto fronts. In *ASME Paper GT2011-45069*, Vancouver, Canada, 6-10 June 2011.

[3] J. Fliege and B.F. Svaiter. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29:149–158, 2005.

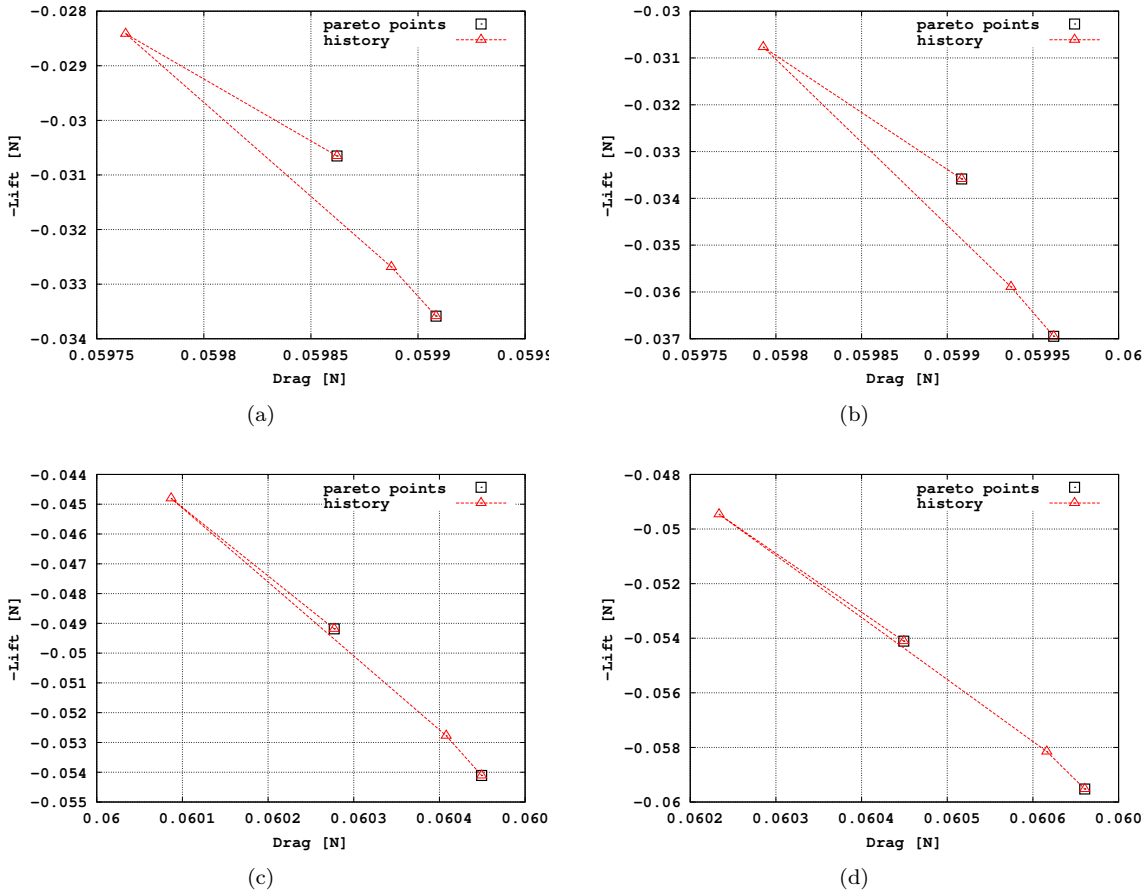


Figure 10: Case 2: Transitions from (a) the 2nd to the 3rd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 8th to the 9th Pareto points, using Method 3. Notation as in fig. 3.

[4] I. Das and J.E. Dennis. A closer look at the drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14:63–69, 1997.

[5] S. Schmidt and V. Schulz. Pareto-curve continuation in multi-objective optimization. *Pacific Journal of Optimization*, 4(2):243–257, 2008.

[6] D.I. Papadimitriou and K.C. Giannakoglou. Direct, adjoint and mixed approaches for the computation of Hessian in airfoil design problems. *International Journal for Numerical Methods in Fluids*, 56(10):1929–1943, 2008.

[7] D.I. Papadimitriou and K.C. Giannakoglou. Aerodynamic design using the truncated newton algorithm and the continuous adjoint approach. *International Journal for Numerical Methods in Fluids*, 68(6):724–739, 2012.

[8] E.M. Papoutsis-Kiachagias and K.C. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: industrial applications. *Archives of Computational Methods*

Figure 11: Case 2: Comparison of the baseline geometry (blue) with the optimized ones (red) referring to (a) the 1st (i.e. min. drag), (b) the 6th and (c) the 10th Pareto points.

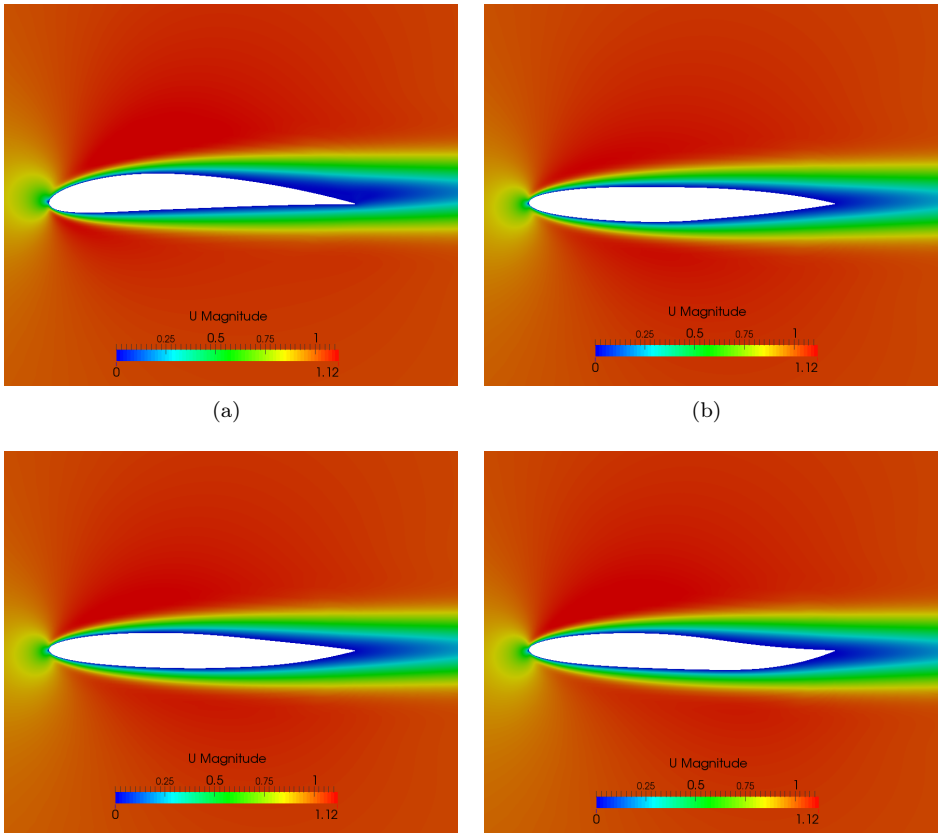


Figure 12: Case 2: Velocity fields for (a) the baseline geometry and those referring to (b) the 1st (c) 6th and (f) 10th Pareto points.

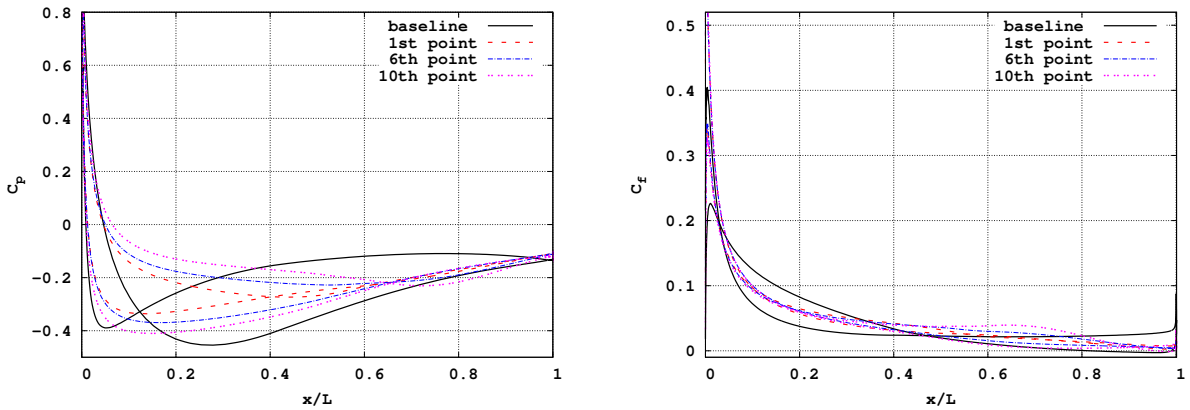


Figure 13: Case 2: (a) C_p and (b) C_f distributions along the pressure and suction sides of the baseline (black), the min. drag (red) and max. lift (blue) airfoils.

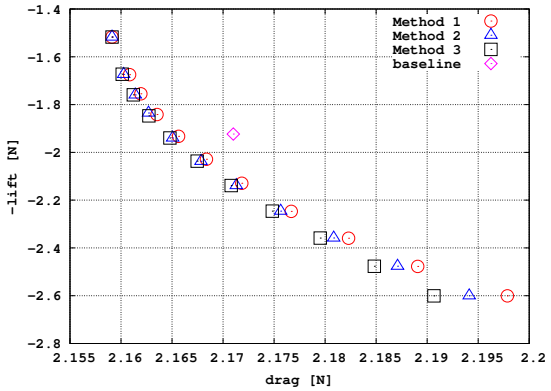


Figure 14: Case 3: Baseline geometry point ("baseline") and Pareto fronts computed by Methods 1 to 3.

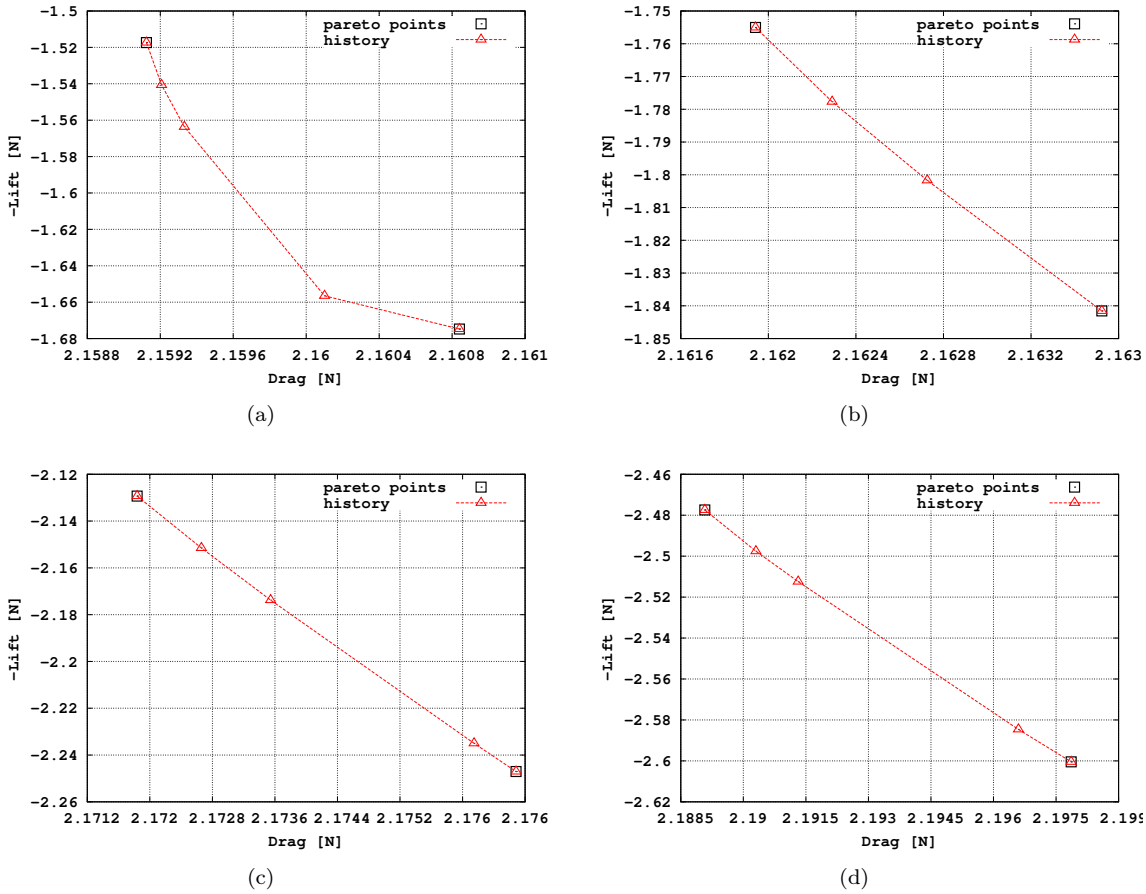


Figure 15: Case 3: Transitions from (a) the 1st to the 2nd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 10th to the 11th Pareto points, using Method 1. Notation as in fig. 3.

in Engineering, 23(2):255–299, 2016.

[9] I.S. Kavvadias, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. On the proper treatment of grid sensitivities in continuous adjoint methods for shape optimization. *Journal of Computational Physics*, 301:1–18, 2015.

[10] M. Ghavami Nejad, E.M. Papoutsis-Kiachagias, and K.C Giannakoglou. Aerodynamic shape opti-

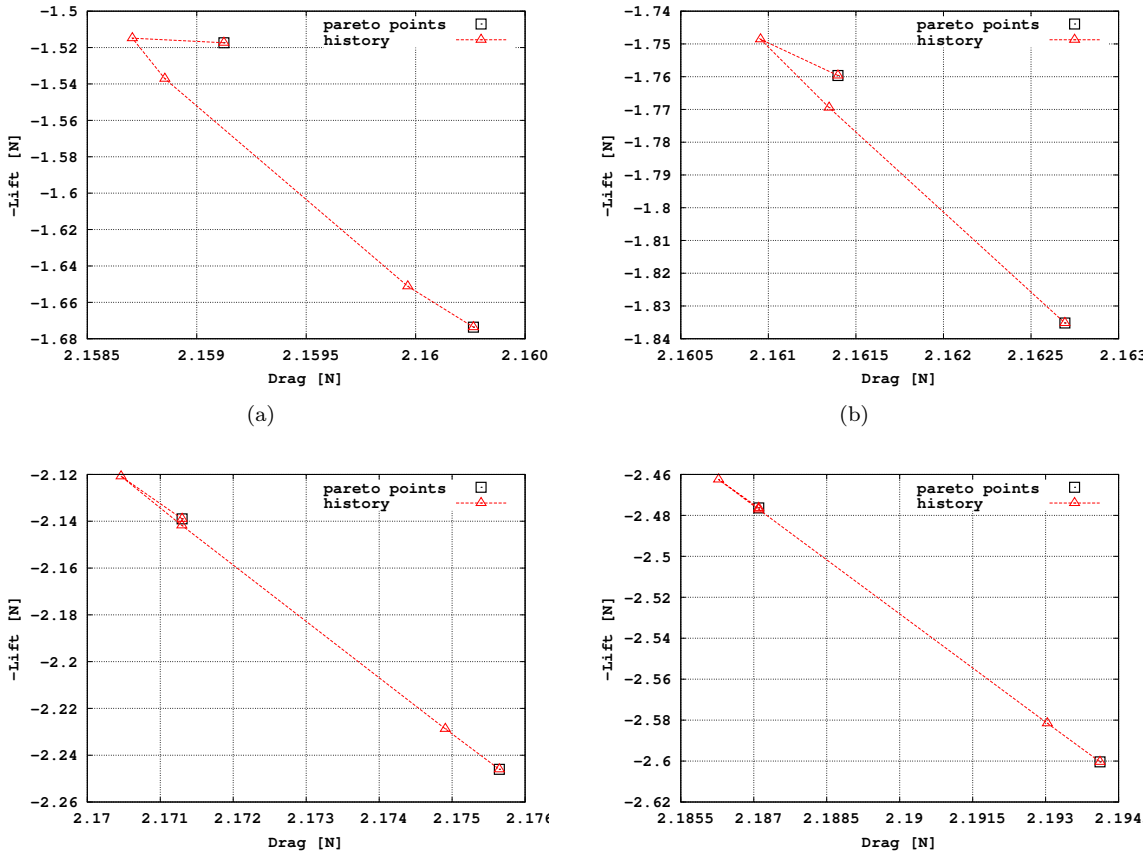


Figure 16: Case 3: Transitions from (a) the 1st to the 2nd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 10th to the 11th Pareto points, using Method 2. Notation as in fig. 3.

mization using the adjoint-based truncated newton method. In *EUROGEN 2015, 11th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, Glasgow, UK, September 14-16 2015.

[11] M. Ghavami Nejad, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. Aerodynamic shape optimization using the truncated Newton method and continuous adjoint. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, Crete island, Greece, June 5-10 2016.

[12] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer, New York, 1999.

[13] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[14] E.M. Papoutsis-Kiachagias, N. Magoulas, J. Mueller, C. Othmer, and K.C. Giannakoglou. Noise reduction in car aerodynamics using a surrogate objective function and the continuous adjoint method with wall functions. *Computers & Fluids*, 122:223–232, 2015.

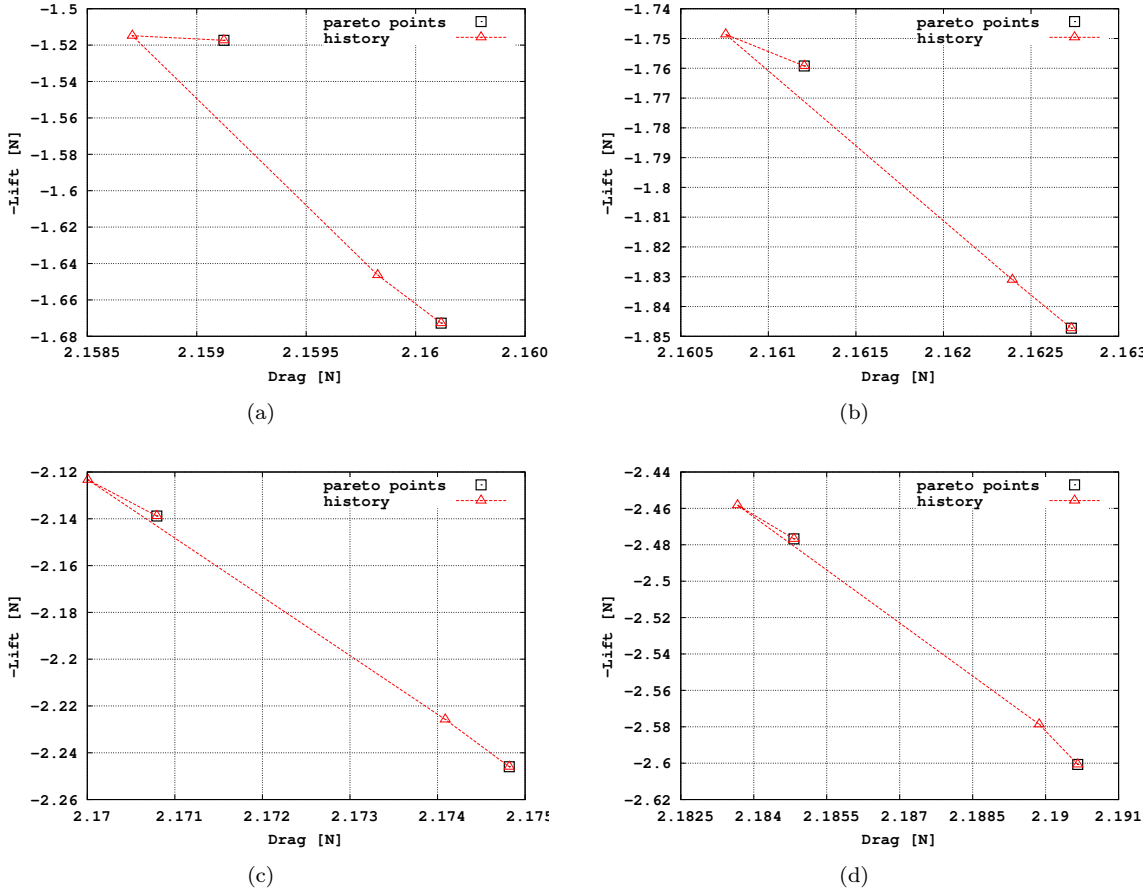


Figure 17: Case 3: Transitions from (a) the 1st to the 2nd, (b) the 3rd to 4th, (c) the 7th to 8th and (d) the 10th to the 11th Pareto points, using Method 3. Notation as in fig. 3.

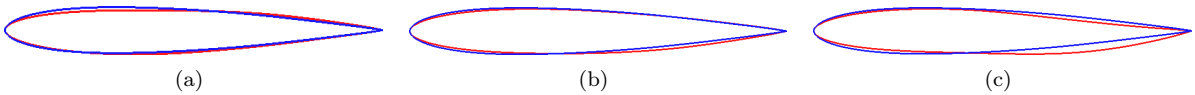
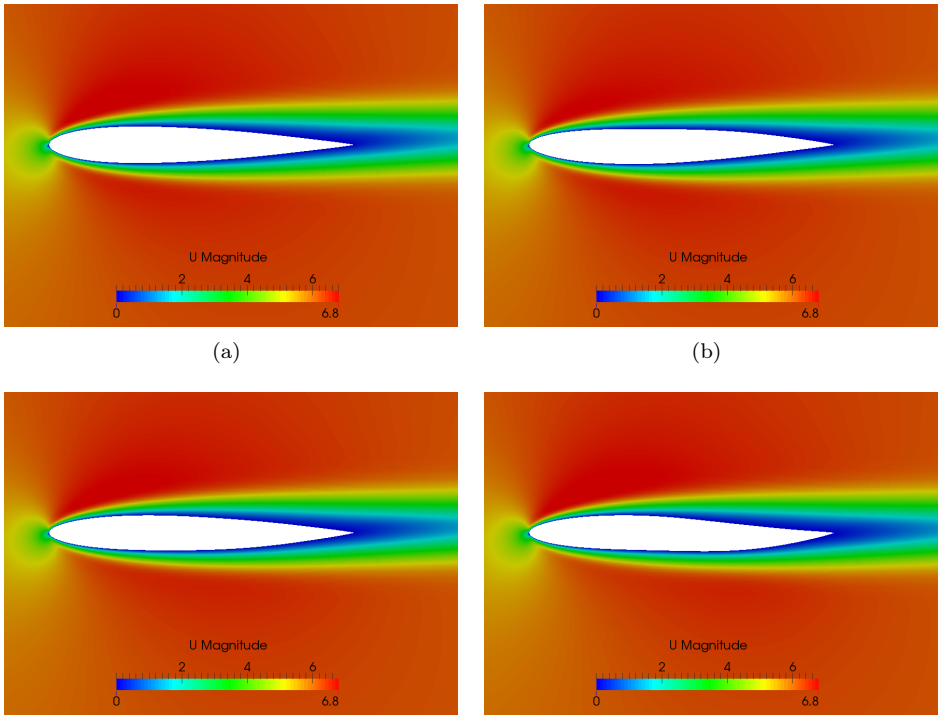


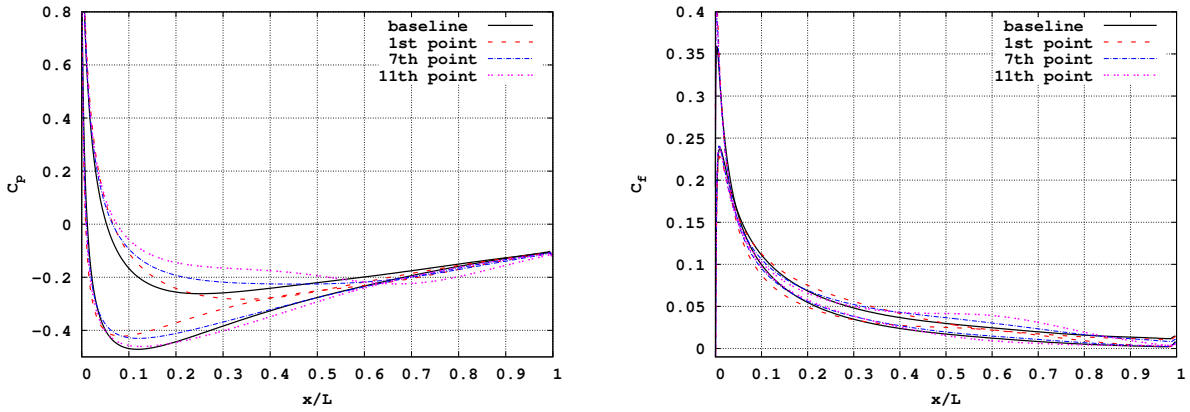
Figure 18: Case 3: Comparison of the baseline geometry (blue) with the optimized ones (red) referring to (a) the 1st Pareto point, i.e. min drag (left), (b) the 6th (centre) and (c) the 11th (right).



(a) (b)

(c) (d)

Figure 19: Case 3: From left to right: velocity fields for the flow around (a) the baseline geometry and those referring to (b) the 1st (i.e. for min. drag), (c) 6th and (d) 11th Pareto points.



(a) (b)

Figure 20: Case 3: C_p (left) and C_f (right) distributions along the pressure and suction sides of the baseline (black) geometry and of those referring to the 1st (red), 6th (blue) and 11th (magenta) Pareto points.