Aerodynamic Shape Optimization Using the Adjoint-based Truncated Newton Method

Evangelos M. Papoutsis-Kiachagias Mehdi Ghavami Nejad, and Kyriakos C. Giannakoglou

Abstract This paper presents the development and application of the truncated Newton (TN) method in aerodynamic shape optimization problems. The development is made for problems governed by the laminar flow equations of incompressible fluids. The method was developed in OpenFOAM© with the aim to stress its advantages over standard gradient-based optimization algorithms. The Newton equations are solved using the conjugate gradient (CG) method which requires the computation of the product of the Hessian of the objective function and a vector, escaping thus the need for computing the Hessian itself. The latter has a computational cost that scales with the number of design variables and becomes unaffordable in large-scale problems with many design variables. A combination of the continuous adjoint method and direct differentiation is used to compute all Hessian-vector products. A grid displacement PDE (Laplace equation) is also used to compute the necessary derivatives of grid displacements w.r.t. the design variables. The programmed method is used to optimize the sidewall shapes of $2D$ ducts for minimum total pressure losses.

1 Introduction to the Truncated Newton Method

An unconstrained optimization problem, in which the target is to minimize the objective function F by controlling the design variables b_i , $i = 1,...,N$ can be solved by means of the Newton method, according to which the design variables are updated as follows

Evangelos M. Papoutsis-Kiachagias e-mail: vaggelisp@gmail.com, Mehdi Ghavami Nejad e-mail: mehdi@mail.ntua.gr,
Kyriakos C. Giannakoglou e-mail: kgianna@central.ntua.gr
Parallel CFD & Optimization Unit, School of Mechanical Engineering, National Technical University of Athens, Greece

$$b_i^{n+1} = b_i^n + \delta b_i \tag{1a}$$

$$\frac{\delta^2 F}{\delta b_i \delta b_j}^n \delta b_j = -\frac{\delta F}{\delta b_i}^n \tag{1b}$$

where n is the Newton iteration counter, to be omitted hereafter. The direct solution of eq. 1b requires the computation of the Hessian of F, with computational cost that scales with N.

Considering eq. 1b as a linear system of equations of the form $Ax=q$, a possible way to solve it is through the Conjugate Gradient (CG) method, which is schematically given in Algorithm 1.

Algorithm 1 : The CG Method for the Solution of $Ax = q$

```
$m \leftarrow 0$
$x \leftarrow \text{init}()$
$r^m \leftarrow Ax - q$; $s \leftarrow -r^m$
while $r^m \neq 0$, (CG Iterations) do
    $\eta \leftarrow \frac{(r^m)^T r^m}{s^T As}$
    $x \leftarrow x + \eta s$
    $r^{m+1} \leftarrow r^m + \eta As$
    $\beta \leftarrow \frac{(r^{m+1})^T r^{m+1}}{(r^m)^T r^m}$
    $s \leftarrow -r^{m+1} + \beta s$
    $m \leftarrow m + 1$
end while
```

Based on Algorithm 1, the cost of each CG iteration is dominated by the cost of computing the matrix–vector product (As). In the so–called truncated variant (Truncated Newton, TN, [3]), the stopping criterion in Algorithm 1 becomes $m \leq M_{CG}$, where M_{CG} is a user–defined small integer (to be used instead of $r^m \neq 0$). Regarding eq. 1b, since the Hessian matrix stands for A, the use of the TN method in aerodynamic shape optimization problems means that the Hessian matrix itself is no more needed and only its product with a vector must be computed. On the other hand, the gradient of F must be available and the (continuous) adjoint method, [4], can be used for this. Its major advantage is its ability to compute the exact gradient of any function, at CPU cost which is independent of N,

2 The Continuous Adjoint Method for the Computation of $\frac{\delta F}{\delta b_i}$

The continuous adjoint method, [4], starts by differentiating the objective function F augmented by the field integral of the flow equations multiplied by the so–called adjoint fields, in order to derive the adjoint PDEs. The latter are, then, discretized and numerically solved to compute the adjoint fields. The gradient of F is expressed in the form of field or boundary integrals of quantities involving the previously computed flow and adjoint fields.

Let us assume a $2D$ laminar flow of an incompressible fluid governed by the continuity ($R^p=0$) and the momentum ($R_i^v=0$) equations, where

$$R^p=-\frac{\partial v_j}{\partial x_j} \tag{2}$$

$$R_i^v=v_j\frac{\partial v_i}{\partial x_j}-\frac{\partial \tau_{ij}}{\partial x_j}+\frac{\partial p}{\partial x_i}\ , \quad i=1,2 \tag{3}$$

Here, v_i are the velocity components, p the static pressure divided by the constant density, $\tau_{ij}=\nu\left(\frac{\partial v_i}{\partial x_j}+\frac{\partial v_j}{\partial x_i}\right)$ the stress tensor and ν the constant viscosity.

Without loss in generality, let us assume that the objective function F to be minimized is the volume–averaged total pressure losses (for the flow inside a duct; internal aerodynamics), namely

$$F=\int_{S_{I,O}}F_{S,i}n_i dS \quad , \qquad F_{S,i}=-\left(p+\frac{1}{2}v_k^2\right)v_i \tag{4}$$

where $S=S_I\cup S_O\cup S_W$ is the domain boundary with S_I being the inlet, S_O the outlet and S_W the wall boundary and $\mathbf{n}$ the outward unit normal vector on the surface. Recall that, for any flow quantity Φ, the total derivative $\delta\Phi/\delta b_n$, which represents the total change in Φ caused by variations in b_n, is

$$\frac{\delta \Phi}{\delta b_n}=\frac{\partial \Phi}{\partial b_n}+\frac{\partial \Phi}{\partial x_k}\frac{\delta x_k}{\delta b_n} \tag{5}$$

In eq. 5, the partial derivative $\partial\Phi/\partial b_n$ represents only the variation in Φ caused due to changes in the design and flow variables, without considering space deformations.

Then, the differentiation of F w.r.t. b_n gives

$$\frac{\delta F}{\delta b_n}=\int_S\frac{\partial F_{S,i}}{\partial b_n}n_i dS+\int_S\frac{\partial F_{S,i}}{\partial x_k}\frac{\delta x_k}{\delta b_n}n_i dS+\int_S F_{S,i}\frac{\delta\left(n_i dS\right)}{\delta b_n} \tag{6}$$

The development of the augmented objective function

$$F_{aug}=F+\int_\Omega u_i R_i^v d\Omega+\int_\Omega qR^p d\Omega \tag{7}$$

leads to the adjoint continuity ($R^q=0$) and adjoint momentum ($R_i^u=0$) equations,

$$R^q=-\frac{\partial u_j}{\partial x_j} \tag{8}$$

$$R_i^u=u_j\frac{\partial v_j}{\partial x_i}-\frac{\partial\left(u_i v_j\right)}{\partial x_j}-\frac{\partial \tau_{ij}^a}{\partial x_j}+\frac{\partial q}{\partial x_i}\ , \ i=1,2 \tag{9}$$

where $\tau^a_{ij} = \nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ is the adjoint stress tensor. By satisfying eqs. 8 and 9, all field integrals in $\delta F_{aug}/\delta b_n$ which depend on $\delta v_i/\delta b_n$ and $\delta p/\delta b_n$ are eliminated. The adjoint boundary conditions are derived by eliminating the total derivatives of the flow variables along the boundaries, while also considering the flow boundary conditions. In this paper, we will refrain from further developing the adjoint boundary conditions, see [4].

After satisfying the adjoint PDEs, eqs. 8 and 9, the expression for the gradient of F is

$$\frac{\delta F}{\delta b_n} = \int_\Omega A_{jk} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega \tag{10}$$

where

$$A_{jk} = -u_i v_j \frac{\partial v_i}{\partial x_k} - u_j \frac{\partial p}{\partial x_k} - \tau^a_{ij} \frac{\partial v_i}{\partial x_k} + u_i \frac{\partial \tau_{ij}}{\partial x_k} + q \frac{\partial v_j}{\partial x_k} \tag{11}$$

3 Computation of Hessian(F)–Vector Products

As explained in Section 1, the TN method requires the computation of $\frac{\delta^2 F}{\delta b_n \delta b_m} s_m$, where s_m might be the components of any vector.

Let us use overbar to denote the product of the total gradient $\frac{\delta \Phi}{\delta b_m}$ of any quantity Φ and s_m, namely

$$\overline{\Phi} = \frac{\delta \Phi}{\delta b_m} s_m \tag{12}$$

It can be proved that

$$\overline{\frac{\partial \Phi}{\partial x_j}} = \frac{\delta}{\delta b_m}\left(\frac{\partial \Phi}{\partial x_j}\right) s_m = \frac{\partial \overline{\Phi}}{\partial x_j} - \frac{\partial \Phi}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} \tag{13}$$

Also, for any pair of Φ and Ψ,

$$\frac{\delta}{\delta b_m}\left(\Psi \frac{\partial \Phi}{\partial x_j}\right) s_m = \overline{\Psi} \frac{\partial \Phi}{\partial x_j} + \Psi \frac{\partial \overline{\Phi}}{\partial x_j} - \Psi \frac{\partial \Phi}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} \tag{14}$$

Based on the above, it is a matter of mathematical development to show that

$$\begin{aligned} \frac{\delta^2 F}{\delta b_n \delta b_m} s_m = & \int_\Omega \overline{A_{jk}} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega + \int_\Omega A_{jk} \frac{\delta}{\delta b_m}\left[\frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right)\right] s_m d\Omega \\ & + \int_\Omega A_{jk} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) s_m \frac{\delta(d\Omega)}{\delta b_m} \end{aligned} \tag{15}$$

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{16}
$$

and, [2],

$$\frac{\delta(d\Omega)}{\delta b_m} s_m = \frac{\partial}{\partial x_\lambda} \left(\frac{\delta x_\lambda}{\delta b_m} s_m \right) d\Omega = \frac{\partial \overline{x_\lambda}}{\partial x_\lambda} d\Omega \tag{17}$$

since $\overline{x_\lambda} = \frac{\delta x_\lambda}{\delta b_m}$. By denoting

$$\overline{\overline{x_{k,n}}} = \frac{\delta^2 x_k}{\delta b_n \delta b_m} s_n \tag{18}$$

it can be proved that

$$\int_\Omega A_{jk} \frac{\delta}{\delta b_m} \left[\frac{\partial}{\partial x_j} \left(\frac{\delta x_k}{\delta b_n} \right) \right] s_m d\Omega = \int_\Omega A_{jk} \frac{\partial \overline{\overline{x_{k,n}}}}{\partial x_j} d\Omega - \int_\Omega A_{jk} \frac{\partial}{\partial x_\lambda} \left(\frac{\delta x_k}{\delta b_n} \right) \frac{\partial \overline{x_\lambda}}{\partial x_j} d\Omega \tag{19}$$

4 Computation of $\overline{v_i}$ and $\overline{p}$

Computing $\overline{v_i}$ and $\overline{p}$ is straightforward and can be done by formulating the product of the direct differentiation (DD, i.e. derivation w.r.t. b_n) of the flow equations and s_m. It is

$$\overline{R^p} = \frac{\delta R^p}{\delta b_m} s_m = 0 \quad , \qquad \overline{R_i^v} = \frac{\delta R_i^v}{\delta b_m} s_m = 0 \tag{20}$$

where

$$\overline{R^p} = \frac{\partial \overline{v_j}}{\partial x_j} - \frac{\partial v_j}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} \tag{21}$$

and

$$\begin{aligned}\overline{R_i^v} = & \frac{\partial(\overline{v_i} v_j)}{\partial x_j} + \frac{\partial(v_i \overline{v_j})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right)\right] + \frac{\partial \overline{p}}{\partial x_i} \\ & - \frac{\partial(v_i v_j)}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial v_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right] \\ & + \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_k}}{\partial x_j} - \frac{\partial p}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\end{aligned} \tag{22}$$

5 Computation of $\overline{u_i}$ and $\overline{q}$

Similarly, the product of the DD of the adjoint equations and s_m yields

$$\overline{R^q} = \frac{\delta R^q}{\delta b_m} s_m = 0 \quad , \qquad \overline{R_i^u} = \frac{\delta R_i^u}{\delta b_m} s_m = 0 \tag{23}$$

where

$$\overline{R^q} = \frac{\partial \overline{u_j}}{\partial x_j} - \frac{\partial u_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} \tag{24}$$

and

$$\begin{aligned}\overline{R_i^u} = & \overline{u_j}\frac{\partial v_j}{\partial x_i} + u_j \frac{\partial \overline{v_j}}{\partial x_i} - \frac{\partial(\overline{u_i} v_j)}{\partial x_j} - \frac{\partial(u_i \overline{v_j})}{\partial x_j} \\ & - \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}\right)\right] + \frac{\partial \overline{q}}{\partial x_i} - u_j \frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i} \\ & + \frac{\partial(v_j u_i)}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial u_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right] \\ & + \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_k}}{\partial x_j} - \frac{\partial q}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\end{aligned} \tag{25}$$

6 Computation of $\overline{x_k}$ and $\overline{\overline{x_{k,n}}}$

In aerodynamic shape optimization problems, a widely used grid displacement model, i.e. a mathematical model that propagates variations in the boundary shape to the internal computational nodes, is based on the Laplace equation with Dirichlet boundary conditions. Written for the derivatives of the grid coordinates X_K w.r.t. the design variables, it takes the form

$$R_i^x = \frac{\partial^2}{\partial x_j^2}\left(\frac{\delta x_k}{\delta b_n}\right) = 0 \tag{26}$$

from which it can readily be deduced that

$$\frac{\partial^2 \overline{x_k}}{\partial x_j^2} = 0 \tag{27}$$

It can also be proved that

$$\frac{\partial^2 \overline{\overline{x_{k,n}}}}{\partial x_j^2} = 2\frac{\partial^2}{\partial x_j \partial x_\lambda}\left(\frac{\delta x_k}{\delta b_m}\right)\frac{\partial \overline{x_\lambda}}{\partial x_j} \tag{28}$$

which can numerically be solved to compute $\overline{\overline{x_{k,n}}}$ with appropriate boundary conditions depending also on the adopted parametrization model.

7 The TN Algorithm – Comments on the CPU Cost

Using eqs. 17 to 19, eq. 15 can be written as

$$\begin{aligned}\frac{\delta^2 F}{\delta b_n \delta b_m} s_m = & \int_\Omega \left[\overline{A_{jk}} + A_{jk}\frac{\partial \overline{x_\lambda}}{\partial x_\lambda} - A_{\lambda k}\frac{\partial \overline{x_j}}{\partial x_\lambda}\right]\frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega \\ & + \int_\Omega A_{jk}\frac{\partial \overline{\overline{x_{k,n}}}}{\partial x_j} d\Omega\end{aligned} \tag{29}$$

where $\overline{A_{jk}}$ is given by eq. 16. To compute $\overline{A_{jk}}$, apart from the flow and adjoint fields, the "overbar" fields ($\overline{v_i}, \overline{u_i}, \overline{p}, \overline{q}$, as well as $\overline{x_i}$ and their spatial derivatives must be available.

So, in each Newton cycle, the numerical solution of $R^p = 0$ and $R_i^v = 0$ (where R^p and R_i^v are given by eqs. 2 and 3) yields the flow fields (p, v_i). The solution of $R^q = 0$ and $R_i^u = 0$ (where R^q and R_i^u are given by eqs. 8 and 9) yields the adjoint fields (q, u_i). So, far, the computational cost is approximately equal to that of twice solving the flow equations or 2 EFS (EFS stands for Equivalent Flow Solutions, i.e. a way to measure computational cost).

Before solving for $\overline{p}$ and $\overline{v_i}$, $\overline{x_k}$ must be computed by solving eq. 27 at the cost of 1 GDS (GDS stands for Grid Displacement Solutions, i.e. the cost of solving the grid displacement PDE or any of the PDEs that result from its differentiation). It should be mentioned that the cost of 1 GDS is significantly lower than the cost of 1 EFS. Eq. 27 has to be solved once per CG iteration, contributing a total cost of M_{CG} GDS per optimization cycle.

Computing $\overline{p}$ and $\overline{v_i}$ requires the numerical solution of equations 20 (considering also eqs. 21 and 22). Similarly, to compute $\overline{q}$ and $\overline{u_i}$ requires the numerical solution of equations 23 (considering also eqs. 24 and 25). Both systems of equations should be performed within the CG loop (i.e. M_{CG} times) and contribute $2M_{CG}$ EFS to the overall cost of a Newton iteration or cycle.

Within each CG iteration, the computation of $\overline{A_{jk}}$ also requires the availability of the $\delta x_k / \delta b_n$ and $\overline{\overline{x_{k,n}}}$ fields. To this end, eqs. 26 and 28 must be solved for $n \in [1, N]$. This results to $2M_{CG}N$ GDS per optimization cycle.

Based on the above, the overall CPU cost per Newton iteration is equal to $2+2M_{CG}$ EFS and $1+2M_{CG}N$ GDS. However, since the cost of a GDS is significantly lower than that of an EFS, the GDS part can be considered negligible for a moderate number of design variables. This leads to a cost per Newton cycle that is independent of the number of design variables N.

8 Applications

In this section, two applications of the developed TN optimization algorithm are presented.

The first one deals with the shape optimization of an S-bend duct. The flow is laminar with a Reynolds number of $Re=785$ based on the inlet height and a mesh consisting of 27500 quadrilaterals is used. Each of the upper and lower sides are parameterized using 9 Bézier–Bernstein control points, fig. 1. The first and last two control points per side are kept fixed while the x and y coordinates of the rest are allowed to vary, giving rise to a total of 20 design variables. In fig. 2, the convergence history of the developed TN algorithm is compared to those of steepest descent (SD) and the Fletcher-Rives Conjugate Gradient (CG), [1], method. Both the iterations required to reach a minimum and the corresponding EFS are compared. In addition, an investigation of the effect of the M_{CG} number can be seen in the same figures. It can be observed that TN outperforms SD and CG, since it computes the optimized duct shape using less optimization cycles and, especially, by requiring less EFS. In addition, it can be seen that even though increasing the M_{CG} number reduces the number of optimization cycles required to reach the minimum, there is no obvious gain from the EFS point of view. In fig. 3, the flow velocity magnitude in the initial and optimized ducts is presented.

The second case is concerned with the optimization of a divergent duct. The flow Reynolds number is $Re=475$ and a mesh consisting of 20000 quadrilaterals is used. The initial and optimized geometries along with the Bézier–Bernstein control points used to parameterize the duct shape are depicted in fig. 4. In fig. 5, the flow velocity magnitude in the initial and optimized ducts is presented. In fig. 6, the convergence history of the TN, SD and CG are illustrated. In this case as well, TN outperforms SD and CG from the optimization cycles point of view; regarding EFS, TN and CG compute the optimal solution almost at the same cost. Increasing M_{CG} has the same effect as in the first case, i.e. the optimized geometry is computed in less optimization cycles but without a significant advantage in CPU cost. This seems to indicate that, at least for the cases studied, a low M_{CG} number should be chosen.

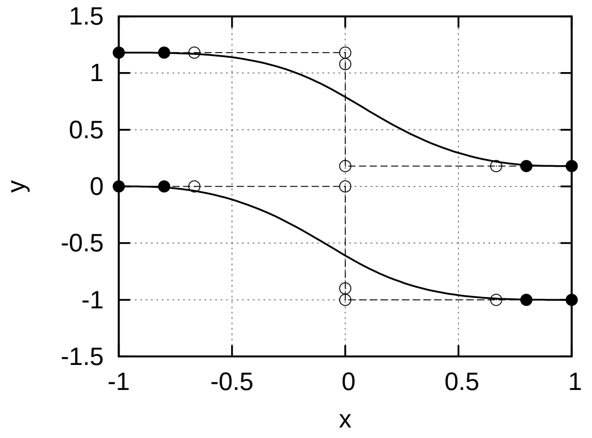


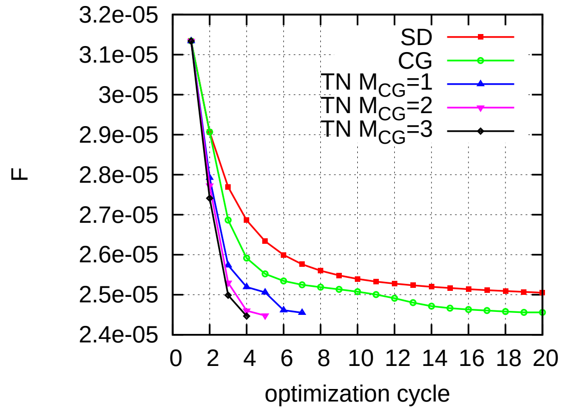
Fig. 1 S-bend duct optimization: dust shape and the Bézier–Bernstein control points parameterizing it. Axes not in scale. Control points depicted with a dark cycle remain fixed during the optimization.

9 Conclusions

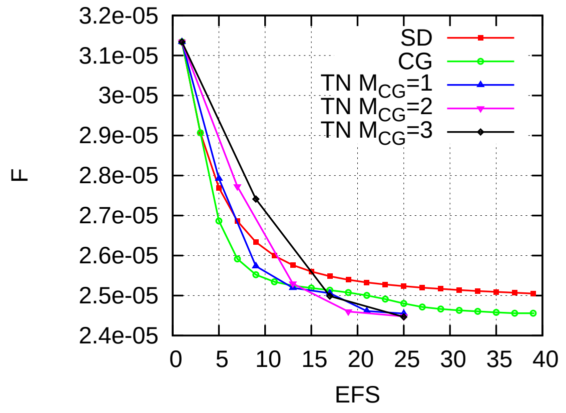
A Truncated Newton method for computing an approximation to the second-order correction of the design variables by iteratively solving Newton's equation using Conjugate Gradient was presented. The method built on previous work of the authors for Euler flows and extended the mathematical background for incompressible, laminar flows. The proposed Truncated Newton method computes the required Hessian-vector products by utilizing a combination of (continuous) adjoint and direct differentiation. The cost per optimization cycle is approximately equal to $2+2M_{CG}$ equivalent flow solutions, where M_{CG} is the number of CG iterations used to approximate the solution of Newton's equation; this cost is practically independent of the design variables number. In the two applications presented, each with a moderate number of design variables, it was shown that Truncated Newton outperforms other optimization methods in terms of optimization cycles and is, at least, as fast as Conjugate Gradient in terms of CPU cost. A parametric study for M_{CG} has also shown that its value should remain as low as possible.

Acknowledgment

The second author is an Early Stage Researcher and his research is funded from the People Programme (ITN Marie Curie Actions) of the European Union's 7^{th} Framework Programme (FP7/2007-2013) under REA grant agreement n^{o} 317006 (AboutFLOW project).



(a)



(b)

Fig. 2 S-bend duct optimization: Convergence of the steepest descent (SD), Conjugate Gradient (CG) and Truncated Newton (TN) optimization algorithms, w.r.t. optimization cycles (a) and EFS (b).

References

1. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Computer Journal **7**, 149–154 (1964)
2. Papadimitriou, D., Giannakoglou, K.: A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. Computers & Fluids **36**, 325–341 (2007)
3. Papadimitriou, D., Giannakoglou, K.: Aerodynamic design using the truncated Newton algorithm and the continuous adjoint approach. International Journal for Numerical Methods in Fluids **68**, 724–739 (2012)
4. Papoutsis-Kiachagias, E., Giannakoglou, K.: Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. Archives of Computational

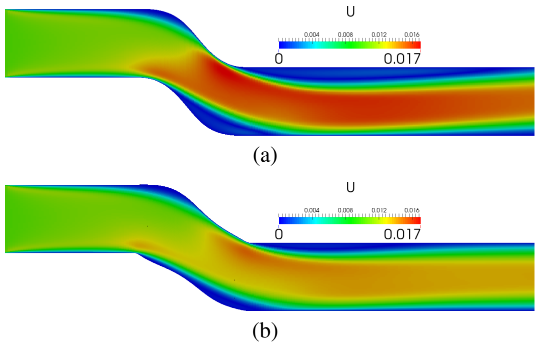


Fig. 3 S-bend duct optimization: Velocity magnitude for the initial (a) and optimized (b) geometries.

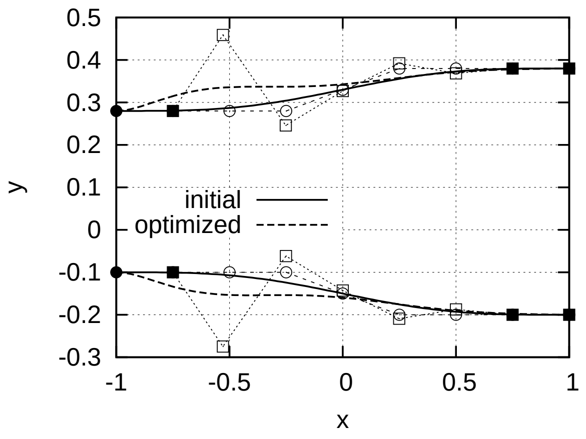


Fig. 4 Divergent duct optimization: dust shape and the Bézier–Bernstein control points parameterizing it. Axes not in scale. Control points depicted with a dark cycle remain fixed during the optimization.

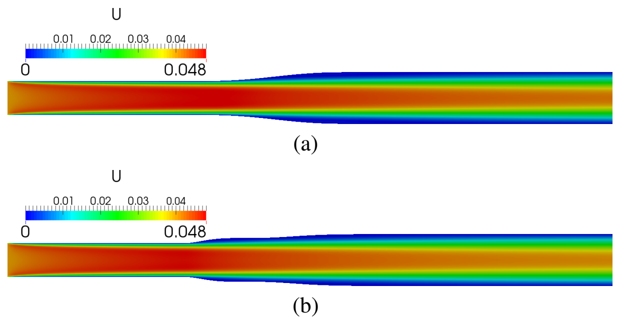
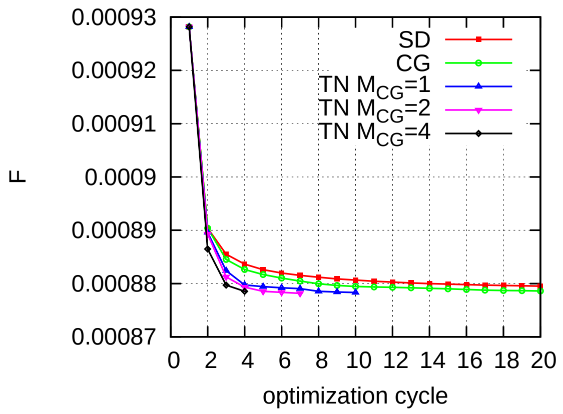
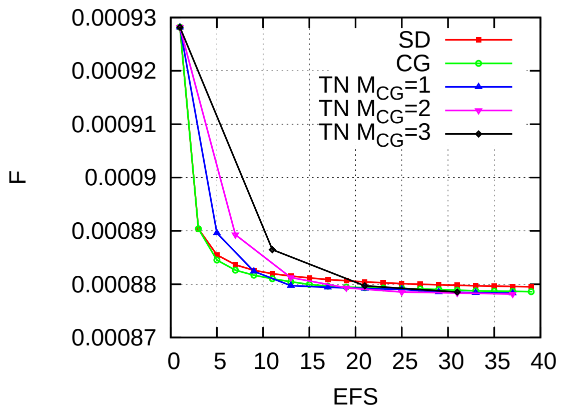


Fig. 5 Divergent duct optimization: Velocity magnitude for the initial (a) and optimized (b) geometries.



(a)



(b)

Fig. 6 Divergent duct optimization: Convergence of the steepest descent (SD), Conjugate Gradient (CG) and Truncated Newton (TN) optimization algorithms, w.r.t. optimization cycles (a) and EFS (b).