ECCOMAS Congress 2016
VII European Congress on Computational Methods in Applied Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou, V. Plevris (eds.)
Crete Island, Greece, 5–10 June 2016

AERODYNAMIC SHAPE OPTIMIZATION USING THE TRUNCATED NEWTON METHOD AND CONTINUOUS ADJOINT

Mehdi Ghavami Nejad[1], **Evangelos M. Papoutsis-Kiachagias**[1], **and Kyriakos C. Giannakoglou**[1]

[1] National Technical University of Athens (NTUA), School of Mechanical Engineering, Parallel CFD & Optimization Unit, Greece
e-mail: mehdi@mail.ntua.gr, vaggelisp@gmail.com, kgianna@central.ntua.gr

Keywords: Truncated Newton, Continuous Adjoint, Aerodynamic Optimization, Losses Minimization

Abstract. *This paper presents the development and application of the Truncated Newton (TN) method for shape optimization problems based on continuous adjoint. The method is presented for laminar, incompressible flows. OpenFOAM® is chosen as the CFD toolbox in which the method is developed. The Newton equations are solved using the restarted linear GMRES algorithm which requires only the product of the Hessian matrix of the objective function (with respect to the design variables) with a vector. This overcomes the cost for computing the Hessian matrix itself, which unfortunately scales with the number of design variables. The computation of Hessian-vector products is conducted via the combination of continuous adjoint and direct differentiation that gives the minimum cost. The developed method is used for the shape optimization of two 3D ducts and the speed-up gained compared to rival methods is showcased.*

This research was funded from the People Programme (ITN Marie Curie Actions) of the European Union's 7th Framework Programme (FP7/2007-2013) under REA Grant Agreement 317006 *(AboutFLOW). The first author is an AboutFLOW Early Stage Researcher.*

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 ($b_i^{n+1} = b_i^n + \delta b_i$) after solving the Newton equations

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

where n is the Newton iteration counter, to be omitted hereafter. The direct solution of eq. 1 requires the computation of the Hessian of F, with a computational cost that scales with N [4].

Considering eq. 1 as a linear system of equations of the form $Ax = q$, a possible way to solve it is through an iterative solver which requires only the computation of matrix-vector products. Since the Hessian matrix is symmetric, a popular choice is the Conjugate Gradient (CG) method, [5, 1]. For reasons to be discussed in sections 8 and 9.1, the linear restarted GMRES method, [9], schematically given in Algorithm 1, is used herein instead.

Algorithm 1 : The Linear Restarted GMRES Method for the Solution of $Ax = q$

$r^0 = Ax^0 - q$, $s^1 = \frac{r^0}{\|r^0\|_2}$
for $j = 1,2,\ldots,M$ **do**
  $w^j = As^j$
  for $i = 1,2,\ldots,j$ **do**
    $h_{i,j} = (w^j, s^i)$
  end for
  $s^{j+1} = w^j - \sum_{i=1}^{j} h_{i,j} s^i$
  $h_{j+1,j} = \|s^{j+1}\|_2$
  $s^{j+1} = \frac{s^{j+1}}{h_{j+1,j}}$
end for
Compute $\beta_1,\ldots,\beta_M$ by solving the minimization problem $min\|Ax^M - q\|_2$
$x^M = x^0 + \sum_{i=1}^{M} \beta_i s^i$

Based on Algorithm 1, the cost of each GMRES iteration is dominated by the cost of computing the matrix–vector product (As), M times during the Arnoldi process, where M is the chosen number of basis vectors. Regarding eq. 1, since the Hessian matrix stands for A, the use of the Truncated Newton (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, for the r.h.s. of eq. 1, the gradient of F must be available and the (continuous) adjoint method, [6], is the less expensive way to compute it, at a CPU cost which is practically independent of N.

2 FLOW MODEL & OBJECTIVE FUNCTION

$3D$ laminar flows of incompressible fluids are governed by the continuity and momentum equations,

$$R^p = -\frac{\partial v_j}{\partial x_j} = 0 \tag{2}$$

$$R_i^v = v_j \frac{\partial v_i}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} + \frac{\partial p}{\partial x_i} = 0\,, \quad i = 1,2,3 \tag{3}$$

where 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.

In this paper, the development of the TN method will be demonstrated for the objective function

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

where $S = S_I \cup S_O \cup S_W$ is the domain boundary with S_I being the inlet, S_O the outlet, S_W the solid wall and $\mathbf{n}$ the outward unit normal vector to the surface. F stands for the volume–averaged total pressure losses of the flow inside a duct; the optimal duct shape is the one yielding the minimal value that F may take on, given the parameterization of S_W.

3 COMPUTATION OF $\frac{\delta F}{\delta b_i}$ VIA CONTINUOUS ADJOINT

It is beyond the scope of this paper to present the continuous adjoint method for the computation of $\delta F / \delta b_i$; the interested reader should refer to [6]. The adjoint continuity and adjoint momentum equations are

$$R^q = -\frac{\partial u_j}{\partial x_j} = 0 \tag{5}$$

$$R_i^u = u_j \frac{\partial v_j}{\partial x_i} - \frac{\partial \left(u_i v_j \right)}{\partial x_j} - \frac{\partial \tau_{ij}^a}{\partial x_j} + \frac{\partial q}{\partial x_i} = 0\,,\ i = 1,2,3 \tag{6}$$

where u_i are the adjoint velocity components, q the adjoint pressure and $\tau_{ij}^a = \nu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$ the adjoint stress tensor. By satisfying eqs. 5 and 6, $\delta F / \delta b_n$ becomes independent of $\delta v_i / \delta b_n$ and $\delta p / \delta b_n$ at the interior of the computational domain, [6].

Using the continuous adjoint method, the gradient of the F w.r.t. b_n becomes

$$\frac{\delta F}{\delta b_n} = \int_\Omega A_{jk} \frac{\partial}{\partial x_j} \left(\frac{\delta x_k}{\delta b_n} \right) d\Omega \tag{7}$$

where

$$A_{jk} = -u_i v_j \frac{\partial v_i}{\partial x_k} - u_j \frac{\partial p}{\partial x_k} - \tau_{ij}^a \frac{\partial v_i}{\partial x_k} + u_i \frac{\partial \tau_{ij}}{\partial x_k} + q \frac{\partial v_j}{\partial x_k} \tag{8}$$

A few comments on eq. 7 are due. According to [2], $\delta F / \delta b_i$ can either be expressed exclusively in terms of surface integrals or may also include field integrals. The two formulations are referred to as *SI* (*Surface Integral*) and *FI* (*Field Integral*), respectively. Eq. 7 is obviously based

on the FI formulation. A noticeable advantage of the latter is that it avoids the computation of second-order spatial derivatives along S which might become a source of error. The proposed TN method relies upon the FI formulation since, following this approach, the computation of even higher spatial gradients at the boundary is avoided during the evaluation of $\frac{\delta^2 F}{\delta b_n \delta b_m} s_m$.

4 BACKGROUND EXPRESSIONS

First of all, a clear distinction between total and partial derivatives should be made. 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

$$\frac{\delta\Phi}{\delta b_n} = \frac{\partial\Phi}{\partial b_n} + \frac{\partial\Phi}{\partial x_k}\frac{\delta x_k}{\delta b_n} \tag{9}$$

where the partial derivative $\partial\Phi/\partial b_n$ includes only the variation in Φ caused due to changes in the design variables, without considering space deformations.

The TN method makes extensive use of the products of total derivatives and any vector s_m. So, it is convenient to define

$$\overline{\Phi} = \frac{\delta\Phi}{\delta b_m} s_m \tag{10}$$

Eq. 10 is also valid for the grid coordinates, so $\overline{x_k} = \frac{\delta x_k}{\delta b_m} s_m$. Starting from

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

it can easily be proved that

$$\overline{\frac{\partial\Phi}{\partial x_j}} = \frac{\delta}{\delta b_m}\left(\frac{\partial\Phi}{\partial x_j}\right) s_m = \frac{\partial\overline{\Phi}}{\partial x_j} - \frac{\partial\Phi}{\partial x_k}\frac{\partial\overline{x_k}}{\partial x_j} \tag{11}$$

It can also be proved that, if Φ, Ψ is any pair of quantities, the following equation is also valid

$$\frac{\delta}{\delta b_m}\left(\Psi\frac{\partial\Phi}{\partial x_j}\right) s_m = \overline{\Psi}\frac{\partial\Phi}{\partial x_j} + \Psi\frac{\partial\overline{\Phi}}{\partial x_j} - \Psi\frac{\partial\Phi}{\partial x_k}\frac{\partial\overline{x_k}}{\partial x_j} \tag{12}$$

Also, as shown in [4], for either structured or unstructured grids,

$$\frac{\delta(d\Omega)}{\delta b_m} = \frac{\partial}{\partial x_\lambda}\left(\frac{\delta x_\lambda}{\delta b_m}\right) d\Omega \tag{13}$$

from which we get

$$\frac{\delta(d\Omega)}{\delta b_m} s_m = \frac{\partial}{\partial x_\lambda}\left(\frac{\delta x_\lambda}{\delta b_m} s_m\right) d\Omega = \frac{\partial\overline{x_\lambda}}{\partial x_\lambda} d\Omega \tag{14}$$

In what follows, the following abbreviation

$$\overline{\overline{x_{k,n}}} = \frac{\delta^2 x_k}{\delta b_n \delta b_m} s_m \tag{15}$$

will also be used.

5 COMPUTATION OF HESSIAN(F)–VECTOR PRODUCTS

The TN method requires the computation of $\frac{\delta^2 F}{\delta b_n \delta b_m} s_m$. Based on the background expressions presented in section 4, it is a matter of a rather lengthy development to show that

$$\begin{aligned}\frac{\delta^2 F}{\delta b_n \delta b_m} s_m &= \int_\Omega \overline{A_{jk}} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) d\Omega + \int_\Omega A_{jk} \frac{\delta}{\delta b_m}\left[\frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right)\right] s_m d\Omega \\ &+ \int_\Omega A_{jk} \frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right) s_m \frac{\delta(d\Omega)}{\delta b_m} \end{aligned} \tag{16}$$

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{17}$$

Based on eq. 15, the second integral on the r.h.s. of eq. 16 becomes

$$\int_\Omega A_{jk} \frac{\delta}{\delta b_m}\left[\frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right)\right] s_m d\Omega = \int_\Omega A_{jk} \frac{\partial \overline{x_{k,n}}}{\partial x_j} d\Omega - \int_\Omega A_{jk} \frac{\partial}{\partial x_\lambda}\left(\frac{\delta x_k}{\delta b_n}\right) \frac{\partial \overline{x_\lambda}}{\partial x_j} d\Omega \tag{18}$$

Computing $\overline{v_i}$ and $\overline{p}$ is straightforward since these are equal to the product of the directly differentiated flow variables and s_m. So, $\overline{v_i}$ and $\overline{p}$ result from

$$\overline{R^p} = \frac{\partial \overline{v_j}}{\partial x_j} - \frac{\partial v_j}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} = 0 \tag{19}$$

and

$$\begin{aligned}\overline{R_i^v} &= \frac{\partial(\overline{v_i} v_j)}{\partial x_j} + \frac{\partial(v_i \overline{v_j})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\nu \left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right)\right] + \frac{\partial \overline{p}}{\partial x_i} \\ &- \frac{\partial(v_i v_j)}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\nu \left(\frac{\partial v_i}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} + \frac{\partial v_j}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_i}\right)\right] \\ &+ \frac{\partial}{\partial x_k}\left[\nu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right] \frac{\partial \overline{x_k}}{\partial x_j} - \frac{\partial p}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_i} = 0 \end{aligned} \tag{20}$$

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

$$\overline{R^q} = \frac{\partial \overline{u_j}}{\partial x_j} - \frac{\partial u_j}{\partial x_k} \frac{\partial \overline{x_k}}{\partial x_j} = 0 \tag{21}$$

and

$$\begin{aligned}\overline{R_i^u} &= \overline{u_j}\frac{\partial v_j}{\partial x_i}+u_j\frac{\partial \overline{v_j}}{\partial x_i}-\frac{\partial(\overline{u_i}v_j)}{\partial x_j}-\frac{\partial(u_i\overline{v_j})}{\partial x_j}\\ &- \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \overline{u_i}}{\partial x_j}+\frac{\partial \overline{u_j}}{\partial x_i}\right)\right]+\frac{\partial \overline{q}}{\partial x_i}-u_j\frac{\partial v_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\\ &+ \frac{\partial(v_ju_i)}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j}+\frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_j}+\frac{\partial u_j}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}\right)\right]\\ &+ \frac{\partial}{\partial x_k}\left[\nu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)\right]\frac{\partial \overline{x_k}}{\partial x_j}-\frac{\partial q}{\partial x_k}\frac{\partial \overline{x_k}}{\partial x_i}=0\end{aligned} \tag{22}$$

from which $\overline{q}$ and $\overline{u}_i$ can be computed.

6 COMPUTATION OF $\overline{x_k}$ AND $\overline{\overline{x_{k,n}}}$

In order to compute the Hessian-vector product of eq. 16, the grid sensitivities $\delta x_k/\delta b_n$ as well as their first-($\overline{x_k}$) and second-order ($\overline{\overline{x_{k,n}}}$) projections to $\mathbf{s}$ must be computed. These computations depend upon the method used to deform the computational grid after the update of the design variables. In [1], the Laplace equation was used as the grid displacement model and the corresponding PDEs for computing the aforementioned terms were presented. Here, a different grid displacement model is employed, based on volumetric B–Splines, details for which can be found in [7, 3]. In brief, the grid points coordinates x_l are given by

$$x_l(u,v,w) = U_{i,pu}(u)V_{j,pv}(v)W_{k,pw}(w)B_l^{ijk} \tag{23}$$

Here, $B_l^{ijk}, l\in[1,3], i\in[0,I], j\in[0,J], k\in[0,K]$ are the Cartesian coordinates of the ijk-th control point of a $3D$ structured control grid (acting also as the design variables of the optimization problem), I,J and K stand the number of control points per control grid direction, $\mathbf{u}=[u_1,u_2,u_3]^T=[u,v,w]^T$ are the CFD grid point parametric coordinates, U,V,W are the B–Splines basis functions and pu,pv,pw their respective degrees, which may be different per control grid direction. Details about B–Splines basis definitions and properties can be found in [8].

Obtaining grid sensitivities and their projections to $\mathbf{s}$ is just a matter of analytically differentiating eq. 23 w.r.t. the coordinates of the control grid points. Let $b_m = B_t^{\lambda\mu\xi}$. Then, the grid sensitivities are given by

$$\frac{\delta x_l(u,v,w)}{\delta b_m} = U_{\lambda,pu}(u)V_{\mu,pv}(v)W_{\xi,pw}(w)\delta_l^t \tag{24}$$

where δ_l^t is the Kronecker symbol. Eq. 24 states that grid sensitivities for each CFD grid point with parametric coordinates $\mathbf{u}$ are given by the product of the basis functions, evaluated at $\mathbf{u}$, corresponding to the $\lambda\mu\xi$ control point. After computing the N components of $\delta x_l/\delta b_m$, computing $\overline{x_l}$ is a matter of a simple summation. It should be noted that $\overline{\overline{x_{l,n}}}=0$, since the grid displacement model depends linearly on the design variables. This further simplifies eq. 18, by eliminating the first term on its r.h.s.

7 THE TN ALGORITHM – COMMENTS ON THE CPU COST

Using eqs. 14 to 18, eq. 16 can be written as

$$\frac{\delta^2 F}{\delta b_n \delta b_m}s_m = \int_\Omega\left[\overline{A_{jk}}+A_{jk}\frac{\partial \overline{x_\lambda}}{\partial x_\lambda}-A_{\lambda k}\frac{\partial \overline{x_j}}{\partial x_\lambda}\right]\frac{\partial}{\partial x_j}\left(\frac{\delta x_k}{\delta b_n}\right)d\Omega \tag{25}$$

where $\overline{A_{jk}}$ is given by eq. 17. 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 $R^v_i = 0$ (eqs. 2 and 3) yields the flow fields (p, v_i). The solution of $R^q = 0$ and $R^u_i = 0$ (eqs. 5 and 6) 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 an Equivalent Flow Solution, i.e. the cost for solving the flow equations).

Before solving for $\overline{p}$ and $\overline{v_i}$, $\overline{x_k}$ must be computed by evaluating eq. 24 for $m \in [1, N]$ and contracting with the components of the projection vector $\mathbf{s}$. The latter has a cost of N GDE (GDE stands for Grid Displacement Evaluations, i.e. the cost of evaluating $\delta x_k / \delta b_m$ for a single m), since $\delta x_k / \delta b_m$ has to be evaluated separately for each design variable. It should be mentioned that 1 GDE is significantly cheaper than 1 EFS since $\delta x_k / \delta b_m$ is computed analytically through eq. 24. $\overline{x_k}$ has to be evaluated once per GMRES iteration, contributing a total cost of MN GDE per optimization cycle.

Computing $\overline{p}$ and $\overline{v_i}$ requires the numerical solution of eqs. 19 and 20. Similarly, to compute $\overline{q}$ and $\overline{u_i}$ requires the numerical solution of eqs. 21 and 22. Both systems of equations should be solved within the GMRES loop (i.e. M times) and contribute $2M$ EFS to the overall cost of a Newton iteration or cycle.

Within each GMRES iteration, the computation of $\overline{A_{jk}}$ also requires the availability of the $\delta x_k / \delta b_n$. These fields, however, have already been computed for the evaluation of $\overline{x_k}$ and contribute no extra cost.

Based on the above, the overall CPU cost per Newton iteration is equal to $2+2M$ EFS plus NM GDE. However, since the cost of a GDE is significantly lower than that of an EFS, the GDE part can be considered negligible for a moderate number of design variables. This leads to a cost per Newton cycle that is, practically, independent of the number of design variables N.

8 CHOICE OF THE LINEAR SOLVER

The TN method can be coupled with any iterative linear solver that relies on the computation of matrix-vector products, without requiring the knowledge of the Hessian matrix itself. In previous publications, [5, 1], the CG method was used as the linear solver, since the Hessian matrix is symmetric in theory. However, the Hessian expression obtained through the use of the AV-DD approach (i.e., use the adjoint variable (AV) method for the computation of $\delta F / \delta b_n$ and DD for the computation of the variations of the primal and adjoint fields; the equivalent of tangent-on-reverse in the Automatic Differentiation terminology) is not symmetric (eq. 16, neglecting the multiplication with s_m) and produces a symmetric matrix only upon the convergence of all equations to machine accuracy (as discussed in Appendix A). This non-symmetry of the Hessian expression is essential for the application of TN methods, since it allows the computation of Hessian-vector products at a cost which is independent of N. In CFD-based optimization, it is quite common not to converge the primal and adjoint equations to machine accuracy in each optimization cycle in order to reduce the total CPU cost. This can deteriorate, to an extent, the symmetry of the Hessian matrix, rendering CG inappropriate for the solution of the Newton equations. To avoid this inconsistency, the linear GMRES solver can be used in the context of TN methods. The impact of the linear solver is investigated in section 9.1.

9 APPLICATIONS

The applications section consists of two parts. In the first, the impact of the linear solver used to iteratively solve the Newton equations is investigated. In the second, optimization problems concerning the total pressure losses minimization in two 3D duct geometries are tackled; the results obtained by using the TN approach are compared to those computed by using Steepest Descent (SD), the Fletcher-Rives Conjugate Gradient (CG) and BFGS methods for updating the design variables.

9.1 Impact of the linear solver

To investigate the impact of the linear solver, an optimization problem with only 5 design variables was devised, making the computation of the Hessian matrix feasible. The shape optimization of a 2*D* U-bend duct is considered, targeting minimum total pressure losses. The upper part of the U-bend is parameterized using Bézier–Bernstein polynomials and the y coordinates of 5 control points are used as the design variables, fig. 1. The flow is laminar with $Re = 667$ based on the inlet length. The update of the design variables is driven by a number of different methods, among which the TN method coupled with the CG and GMRES solvers. Their convergence histories are presented in fig. 2. It can be seen that as the number of linear solver iterations M increases, the GMRES-based TN method greatly outperforms the CG-based one. If fact, when M is chosen to be equal to the Hessian matrix dimension, the GMRES-based TN method has exactly the same convergence with the pure Hessian method, as expected. On the contrary, the CG-based TN method requires approximately 4 times more EFS in order to reach the optimal solution. This can be attributed to the fact that CG is used to iteratively solve a slightly non-symmetric system. In detail, the symmetric, in theory, elements of the Hessian matrix computed using the AV-DD approach have a maximum difference of 0.8%, a mean difference of 0.1% and a standard deviation of 0.2%. Based on the above, for the remainder of this article, the GMRES solver is used in conjunction with the TN method.

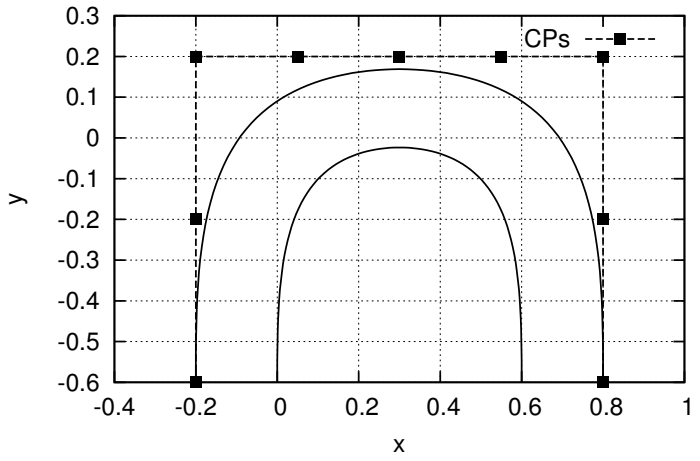
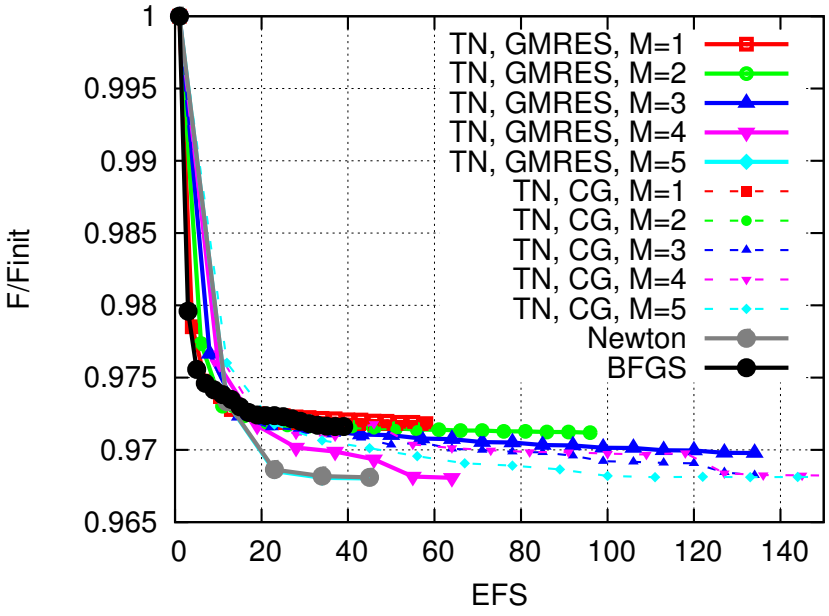


Figure 1: 2*D* U-bend duct optimization: dust shape and the Bézier–Bernstein control points parameterizing it. Only the y coordinates of the top 5 control points (CP) are allowed to vary during the optimization.



(a)

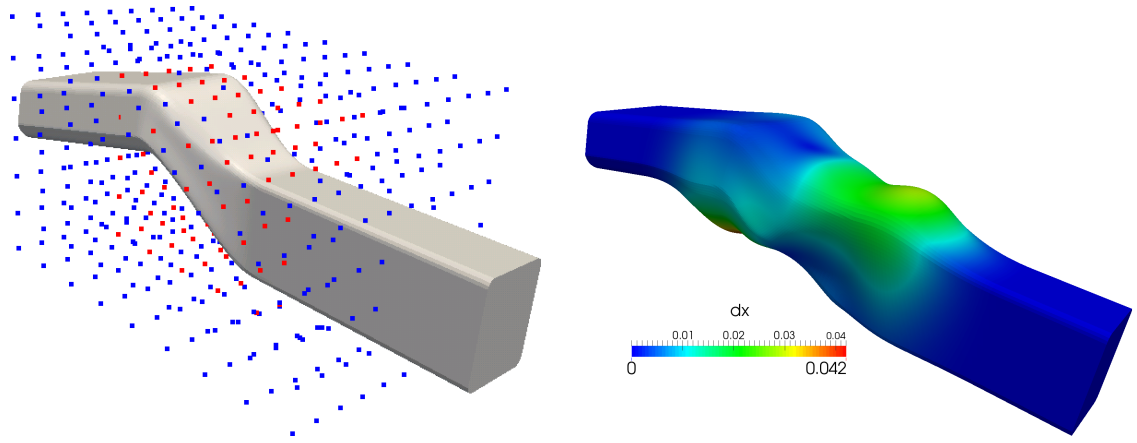
Figure 2: $2D$ U-bend duct optimization: Convergence of the BFGS, Newton and TN optimization algorithms. Convergence of the TN method is included with both CG and GMRES, with linear solver iterations in the range of $M \in [1,5]$. Results of GMRES-based TN are plotted with a continuous line while CG-based results with a dashed one. As expected, the convergence of the GMRES-based TN with $M=5$ coincides with the convergence of the Newton method (the two curves are hardly distinguishable) since the dimension of the problem is $N=5$.

9.2 3D shape optimization

In this section, two applications of the developed TN optimization algorithm are presented. The first one deals with the shape optimization of a $3D$ S-bend duct. The geometry and flow conditions are provided as one of the cases of the AboutFLOW ITN programme. The flow is laminar with a Reynolds number of $Re=400$ based on the inlet hydraulic diameter and the mesh is comprised of 474000 hexahedrals. A $9\times7\times9$ control grid is used to parameterize part of the duct which, after disregarding fixed control points, results to 375 design variables, fig. 3. In fig. 4, the convergence history of the developed TN algorithm is compared to those of the SD, CG and BFGS methods. Comparisons are presented twice, in terms of the cycles required to reach the minimum and the corresponding EFS. It can be observed that TN outperforms the other methods, since it computes the optimized duct shape using less optimization cycles and, especially, by requiring slightly less EFS. In fig. 5, the flow streamlines on the reference and optimized geometries are compared, indicating the significant reduction of the flow recirculation that leads to a total pressure losses reduction of $\sim 60\%$.

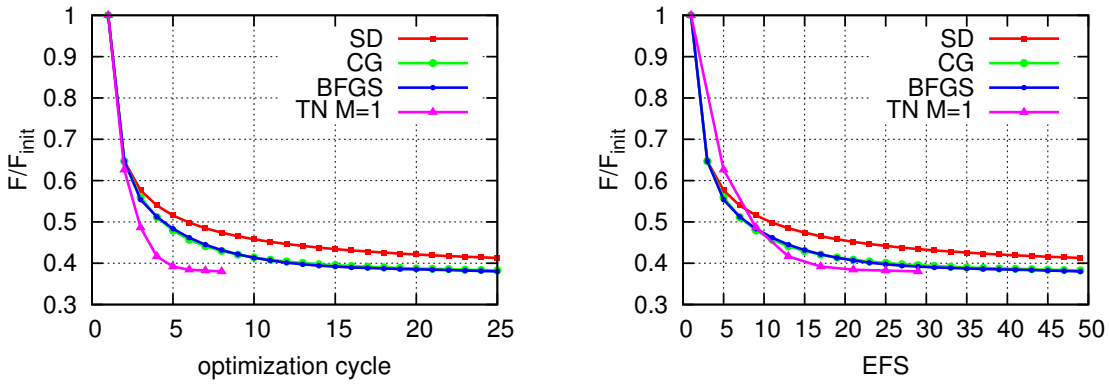
The second case is concerned with the optimization of a 3D U-bend duct. The flow Reynolds number is $Re=400$ and a mesh consisting of 7×10^5 hexahedrals is used. The reference geometry and the $4\times5\times2$ control grid parameterizing it are depicted in fig. 6. Since only two rows of control points are used in the z direction and the reference geometry was generated by stacking a $2D$ profile, the shape parameterization is practically $2D$. In fig. 7, the total pressure field along with velocity vectors are plotted for two slices, located at 10% and 50% of the duct height. It can be seen that the flow recirculation downstream of the U-shaped formation has been suppressed, leading to a reduction of about 18% in the objective function value. In fig. 8, the convergence histories of the TN, SD and CG methods are illustrated. For this case, TN and CG reach the

optimal solution at approximately the same CPU cost. The TN method outperforms CG during the initial phase of the optimization run, providing a better solution if the entire CPU cost of the optimization can not be afforded.



(a) (b)

Figure 3: S-bend duct optimization: (a) duct shape and the control grid parameterizing it. Control points in red are allowed to vary during the optimization while blue ones are kept fixed, (b) optimal shape coloured based on the cumulative displacement. Flow from right to left.



(a) (b)

Figure 4: S-bend duct optimization: comparison of the convergence of SD, CG, BFGS and TN w.r.t. (a) optimization cycles and (b) EFS. The TN method outperforms all other methods in both comparisons.

10 CONCLUSIONS

A Truncated Newton method for computing an approximation to the second-order correction of the design variables by iteratively solving the Newton equations using GMRES was presented. The method builds on previous work of the authors, by extending the mathematical formulation for a different grid displacement model and investigating the impact of the linear solver used to iteratively solve the Newton equations. It was observed that due to a slight non-symmetry of the Hessian matrix, caused by the lack of convergence of the adjoint equations to machine accuracy, CG may become inefficient when computing the solution of the Newton equations. Using GMRES as the linear solver within the TN loop significantly accelerated

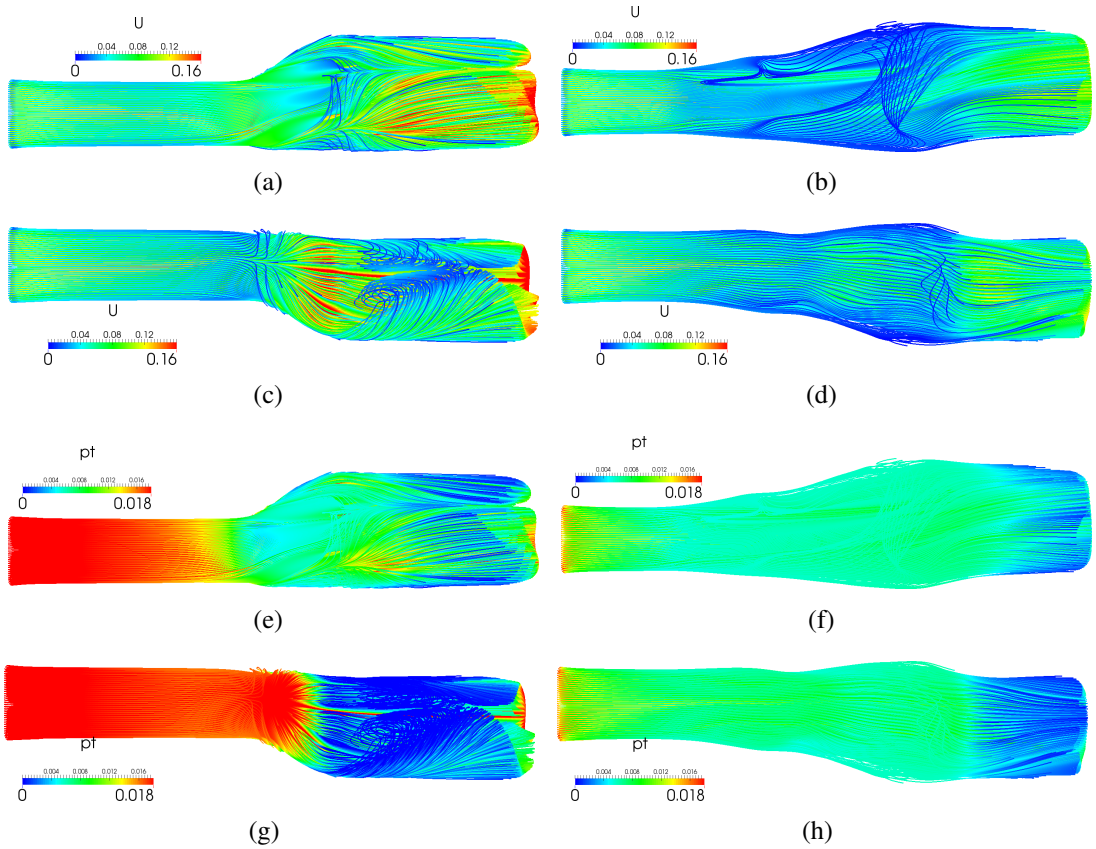


Figure 5: S-bend duct optimization: Velocity streamlines plotted for the reference (left column) and optimized (right column) geometries. In the top four figures, streamlines are coloured based on the flow velocity while, in the bottom four, on the total pressure values. The intense flow recirculation close to the bottom side of the wall (figs. c and g) has drastically been reduced (figs. d and h), leading to a reduction of about 60% in the objective function.

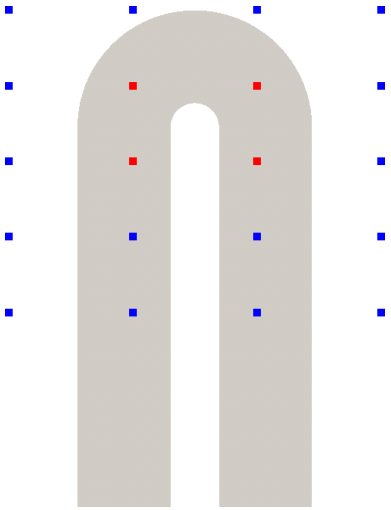
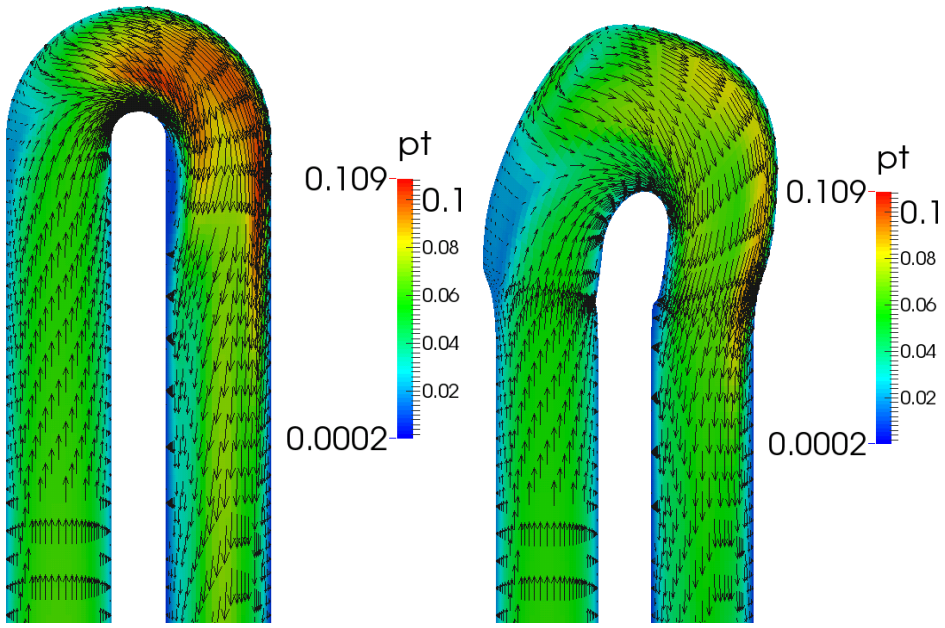
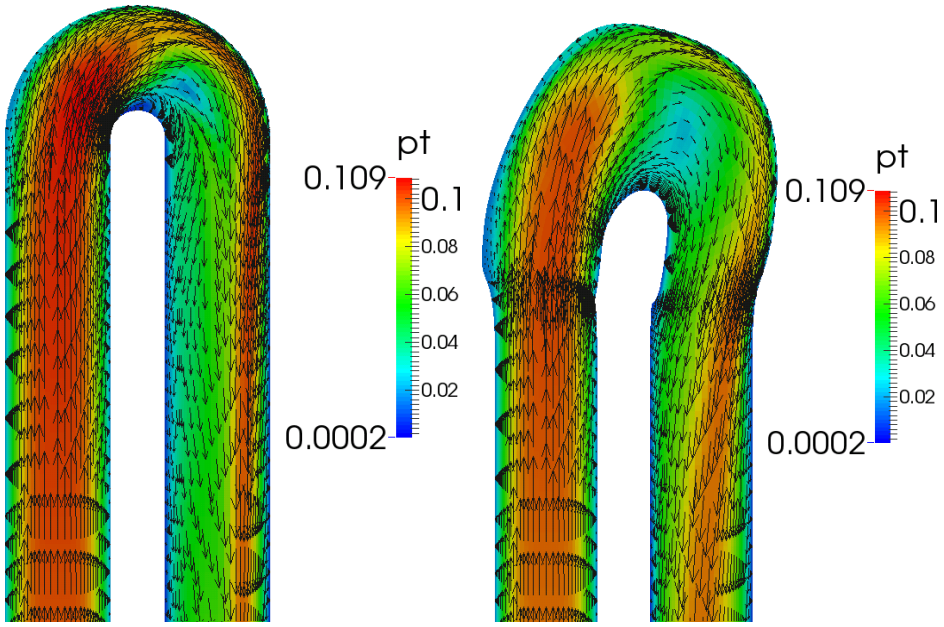


Figure 6: 3*D* U-bend optimization: Part of the duct shape along with one of two iso-z control point planes. Red control points are allowed to vary while blue ones are kept fixed during the optimization.

the convergence. The proposed TN 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 $2+2M$ equivalent flow solutions, where M is the number of



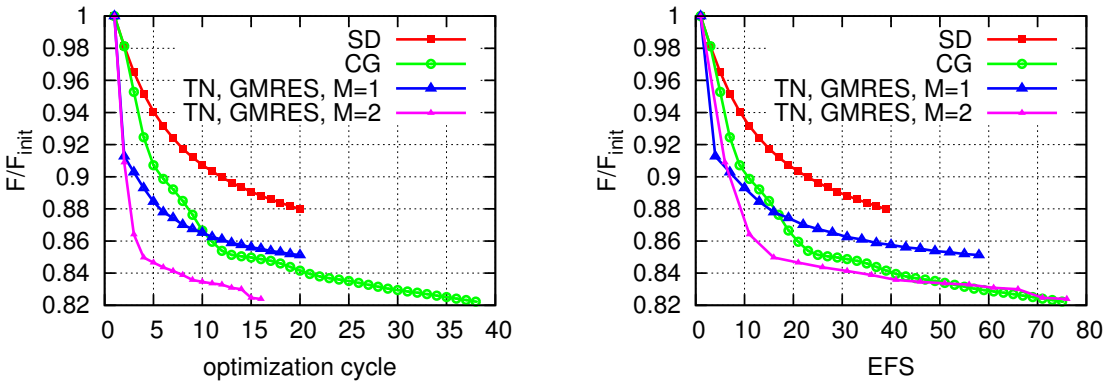
(a) reference, 10% thickness (b) optimized, 10% thickness



(c) reference, 50% thickness (d) optimized, 50% thickness

Figure 7: 3*D* U-bend optimization: Total pressure field plotted for the reference (left column) and optimized (right column) geometries, for a slice residing at 10% (top) and 50% of the duct height.

GMRES iterations used to approximate the solution of the Newton equations; this cost is practically independent of the design variables number. In the two applications presented, it was shown that TN outperforms other optimization methods in terms of optimization cycles and is, at least, as fast in terms of CPU cost. On going research, including the preconditioning of the Newton system and appropriate initialization, for further improving the TN method speed-up is performed.



(a) (b)

Figure 8: 3*D* U-bend optimization: Convergence of the SD, CG and TN methods wrt (a) optimization cycles and (b) EFS. For this case, $M=2$ has to be used for TN to outperform the CG method.

A ON THE SYMMETRY OF THE HESSIAN MATRIX

In this appendix, the symmetry of the Hessian matrix computed using the AV-DD approach (i.e. the approach used to also compute the Hessian-vector product in the TN approach) is examined. Since the continuous gradient and Hessian expressions are quite lengthy, the discrete approach is going to be used in this appendix. The conclusions, however, can be extended to the continuous formulation as well.

Let the objective function F and discretized residuals $\mathbf{R}$ be functions of the design, $\mathbf{b}$, and flow variables, $\mathbf{U}(\mathbf{b})$, i.e. $F=F(\mathbf{b},\mathbf{U}(\mathbf{b}))$ and $\mathbf{R}=\mathbf{R}(\mathbf{b},\mathbf{U}(\mathbf{b}))$. After introducing the augmented objective function as $F_{aug}=F+\Psi_k R_k$ and differentiating it w.r.t. to $\mathbf{b}$, we get

$$\frac{dF_{aug}}{db_i}=\frac{\partial F}{\partial b_i}+\Psi_k\frac{\partial R_k}{\partial b_i}+\left(\frac{\partial F}{\partial U_m}+\Psi_k\frac{\partial R_k}{\partial U_m}\right)\frac{dU_m}{db_i} \tag{26}$$

from which the adjoint equations and sensitivity derivatives are derived as

$$R_m^{\Psi}=\frac{\partial F}{\partial U_m}+\Psi_k\frac{\partial R_k}{\partial U_m}=0 \tag{27}$$

$$\frac{dF}{db_i}=\frac{\partial F}{\partial b_i}+\Psi_k\frac{\partial R_k}{\partial b_i} \tag{28}$$

Differentiating eq. 28 once more w.r.t. the components of $\mathbf{b}$ gives

$$\frac{d^2F}{db_i db_j}=\frac{\partial^2 F}{\partial b_i\partial b_j}+\Psi_k\frac{\partial^2 R_k}{\partial b_i\partial b_j}+\frac{\partial^2 F}{\partial b_i\partial U_k}\frac{dU_k}{db_j}+\Psi_k\frac{\partial^2 R_k}{\partial b_i\partial U_m}\frac{dU_m}{db_j}+\frac{\partial R_k}{\partial b_i}\frac{d\Psi_k}{db_j} \tag{29}$$

The Hessian expression given by eq. 29 is not symmetric, since permuting i and j does not yield $\frac{d^2F}{db_i db_j}=\frac{d^2F}{db_j db_i}$. This non-symmetric expression actually allows the computation of Hessian-vector products with a cost that does not depend on the design variables number in TN methods. However, since eq. 26 and eq. 28 are equivalent (upon the convergence of the residuals of the

adjoint equations to machine accuracy) for computing dF/db_i, differentiating eq. 26 yields

$$\begin{aligned}\frac{d^2F}{db_i db_j} &= \frac{\partial^2 F}{\partial b_i \partial b_j} + \Psi_k \frac{\partial^2 R_k}{\partial b_i \partial b_j} + \frac{\partial^2 F}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \frac{\partial^2 F}{\partial b_j \partial U_k}\frac{dU_k}{db_i} + \Psi_k \frac{\partial^2 R_k}{\partial b_i \partial U_m}\frac{dU_m}{db_j} \\ &+ \Psi_k \frac{\partial^2 R_k}{\partial b_j \partial U_m}\frac{dU_m}{db_i} + \frac{\partial^2 F}{\partial U_m \partial U_k}\frac{dU_m}{db_i}\frac{dU_k}{db_j} + \Psi_k \frac{\partial^2 R_k}{\partial U_m \partial U_l}\frac{dU_m}{db_i}\frac{dU_l}{db_j} \\ &+ \left(\frac{\partial R_k}{\partial b_i} + \frac{\partial R_k}{\partial U_m}\frac{\partial U_m}{\partial b_i}\right)\frac{d\Psi_k}{db_j} + \left(\frac{\partial F}{\partial U_m} + \Psi_k \frac{\partial R_k}{\partial U_m}\right)\frac{d^2 U_m}{db_i db_j}\end{aligned} \tag{30}$$

The sum of the first eight terms on the r.h.s. of eq. 30 is symmetric while the last two terms are zero since they include dR_k/db_i and R_k^Ψ, respectively. Hence, since the expression in eq. 30 is symmetric and eqs. 29 and 30 are equivalent, upon the convergence of the adjoint equations to machine accuracy, the Hessian matrix obtained through eq. 29 is symmetric as well.

However, if eq. 27 is not converged to machine accuracy, i.e. if

$$\tilde{R}_m^{\tilde{\Psi}} = \frac{\partial F}{\partial U_m} + \tilde{\Psi}_k \frac{\partial R_k}{\partial U_m} + r_m^a = 0, \ , \ r_m^a \neq 0 \tag{31}$$

where $\tilde{\Psi}_k$ is the slightly non-converged adjoint solution and r_k^a the adjoint residual, differentiation of the equivalent of eq. 26 would yield

$$\begin{aligned}\frac{d^2F}{db_i db_j} &= \frac{\partial^2 F}{\partial b_i \partial b_j} + \tilde{\Psi}_k \frac{\partial^2 R_k}{\partial b_i \partial b_j} + \frac{\partial^2 F}{\partial b_i \partial U_k}\frac{dU_k}{db_j} + \frac{\partial^2 F}{\partial b_j \partial U_k}\frac{dU_k}{db_i} + \tilde{\Psi}_k \frac{\partial^2 R_k}{\partial b_i \partial U_m}\frac{dU_m}{db_j} \\ &+ \tilde{\Psi}_k \frac{\partial^2 R_k}{\partial b_j \partial U_m}\frac{dU_m}{db_i} + \frac{\partial^2 F}{\partial U_m \partial U_k}\frac{dU_m}{db_i}\frac{dU_k}{db_j} + \Psi_k \frac{\partial^2 R_k}{\partial U_m \partial U_l}\frac{dU_m}{db_i}\frac{dU_l}{db_j} \\ &+ \frac{dR_k}{db_i}\frac{d\tilde{\Psi}_k}{db_j} + \tilde{R}_m^{\tilde{\Psi}}\frac{d^2 U_m}{db_i db_j} + \frac{dr_k^a}{db_i}\frac{d\tilde{\Psi}_k}{db_j}\end{aligned} \tag{32}$$

Eq. 32 states that if the adjoint equations are not converged to machine accuracy, eqs. 29 and 30 are no longer equivalent due to the last, non-symmetric term in eq. 32.

REFERENCES

[1] M. Ghavami Nejad, EM. Papoutsis-Kiachagias, and KC. Giannakoglou. Aerodynamic shape optimization 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.

[2] IS. Kavvadias, EM. Papoutsis-Kiachagias, and KC. Giannakoglou. On the proper treatment of grid sensitivities in continuous adjoint methods for shape optimization. *Journal of Computational Physics*, 301:1–18, 2015.

[3] MJ. Martin, E. Andres, C. Lozano, and E. Valero. Volumetric B-splines shape parametrization for aerodynamic shape design. *Aerospace Science and Technology*, 37:26–36, 2014.

[4] DI. Papadimitriou and KC. Giannakoglou. Aerodynamic shape optimization using first and second order adjoint and direct approaches. *Archives of Computational Methods in Engineering*, 15(4):447–488, 2008.

[5] DI. Papadimitriou and KC. Giannakoglou. Aerodynamic design using the truncated Newton algorithm and the continuous adjoint approach. *International Journal for Numerical Methods in Fluids*, 68:724–739, 2012.

[6] EM. Papoutsis-Kiachagias and KC. Giannakoglou. Continuous adjoint methods for turbulent flows, applied to shape and topology optimization: Industrial applications. *Archives of Computational Methods in Engineering, DOI 10.1007/s11831-014-9141-9*, 2014.

[7] EM. Papoutsis-Kiachagias, N. Magoulas, J. Mueller, C. Othmer, and KC. 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.

[8] L. Piegl and W. Tiller. *The NURBS book*. Springer, 1997.

[9] 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.