Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 204 (2005) 46–64
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Aerodynamic shape optimization using simultaneous pseudo-timestepping

S.B. Hazra [a,*], V. Schulz [a], J. Brezillon [b], N.R. Gauger [b]

[a] Department of Mathematics, University of Trier, D-54286 Trier, Germany
[b] Institute of Aerodynamics and Flow Technology, German Aerospace Center (DLR), D-38108 Braunschweig, Germany

Received 1 March 2004; received in revised form 26 August 2004; accepted 3 October 2004

Abstract

The paper deals with a numerical method for aerodynamic shape optimization. It is based on simultaneous pseudo-timestepping in which stationary states are obtained by solving the non-stationary system of equations representing the state, costate and design equations. The main advantages of this method are that it requires no additional globalization techniques and that a preconditioner can be used for convergence acceleration which stems from the reduced SQP method. A design example for drag reduction for an RAE2822 airfoil, keeping its thickness fixed, is included. The overall cost of computation is less than four times that of the forward simulation run.

Keywords: Shape optimization; Simultaneous pseudo-timestepping; Euler equations; Preconditioner; Reduced SQP methods; One-shot method; Airfoil

1. Introduction

Applications of numerical optimization techniques in the field of aerodynamics are an active area of research. With the advancement of computer technology and availability of fast solvers, the field of Computational Fluid Dynamics has made considerable progress. The FLOWer code [22,23] of the German Aerospace Center (DLR) presents one such example which we use for the solution of the Euler and the adjoint Euler equations. Despite many recent advances in the field of aerodynamic shape optimization, much important research remains to be done. Several works in this field have been reported in last three decades using different numerical techniques. Gradient methods are among the most commonly applied

* Corresponding author. Tel.: +49 651 201 3464; fax: +49 651 201 3966.
E-mail address: haza@uni.trier.de (S.B. Hazra).

0021-9991/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2004.10.007

Nomenclature

$(x,y) \in \mathbb{R}^2$	Cartesian coordinates	H	total enthalpy
$(\xi,\eta) \in [0,1]^2$	generalized coordinates	M	Mach number
Ω	flow field domain	$)_\infty$	values at free stream
$\partial\Omega = B \cup C$	flow field boundary	γ	ratio of specific heats
B	farfield boundary	C_{ref}	chord length
C	solid wall, airfoil	C_{p}	pressure coefficient
$\vec{n} := \begin{pmatrix} n_x \\ n_y \end{pmatrix}$	unit outward normal	C_{D}	drag coefficient
α	angle of attack	C_{L}	lift coefficient
ρ	density	C_{m}	pitching moment coefficient
$\vec{q} = un_x + vn_y$	velocity	I	cost unction
u	x-component of velocity	w	vector of state variables
v	y-component of velocity	q	vector of design variables
p	pressure	λ	vector of adjoint variables
E	total energy	J	Jacobian

methods in practical problems of this field. In this method, one of the main issues is the efficient computation of the sensitivity derivatives. Among various techniques reported for this purpose, the continuous adjoint method has gained considerable attention since its derivation by Jameson in [15].

The focus of the present work is on optimal control problems, and in particular the sub-class of shape design problems. Pioneering theoretical works on the methodology for solving such problems have been presented in [25,28–30]. These problems can be written in abstract form as

$$\begin{aligned} &\min I(w,q) \\ &\text{s.t.} \quad c(w,q) = 0, \end{aligned} \tag{1}$$

where $(w,q) \in X \times P$ (X,P are appropriate Hilbert spaces), $I{:}X \times P \to \mathbb{R}$ and $c{:}X \times P \to Y$ are twice Frechet-differentiable (with Y an appropriate Banach space). The Jacobian, $J = (\partial c)/(\partial w)$, is assumed to be invertible. Here, the equation $c(w,q) = 0$ represents the steady-state flow equations (in our case Euler equations) together with boundary conditions, w is the vector of dependent variables and q is the vector of design variables. The objective $I(w,q)$ is the drag of an airfoil for the purposes of this paper. Typically, there arise inequality constraints of the form

$$h(w,q) \geqslant 0,$$

which in practical applications, often pose severe restrictions on the validity region of the model or for the design construction. In the present work we are outlining a framework for unconstrained optimization, and the addition of constraints is addressed in the subsequent work [12].

The necessary optimality conditions can be formulated using the Lagrangian functional

$$L(w,q,\lambda) = I(w,q) - \lambda^* c(w,q), \tag{2}$$

where λ is the Lagrange multiplier or the adjoint variable from the dual Hilbert space. If $\hat{z} = (\hat{w},\hat{q})$ is a minimum, then there exists a $\hat{\lambda}$ such that

$$\nabla_z L(\hat{z},\hat{\lambda}) = \nabla_z I(\hat{z}) - \hat{\lambda}^* \nabla_z c(\hat{z}) = 0. \tag{3}$$

Hence, the necessary optimality conditions are

$$c(w,q) = 0 \quad \text{(State equation)}, \tag{4}$$

$$\nabla_w L(w,q,\lambda) = 0 \quad \text{(Costate equation)}, \tag{4a}$$

$$\nabla_q L(w,q,\lambda) = 0 \quad \text{(Design equation)}. \tag{4b}$$

Gradient methods, which are widely used in many practical applications, involve the solution of the state and the costate equations at each update of the design variables. These methods only act in the design space and assume that the state and costate or adjoint equations are solved exactly. Thus, they can be viewed as an explicit Euler approximation to the following evolution differential algebraic equations

$$\begin{aligned} &c(w,q) = 0, \\ &\nabla_w L(w,q,\lambda) = 0, \\ &\frac{\mathrm{d}q}{\mathrm{d}t} + \nabla_q L(w,q,\lambda) = 0. \end{aligned} \tag{5}$$

The disadvantage of these methods is their high computational cost due to the fact that state and costate equations have to be solved quite accurately in each iteration step. Computational results based on these methodologies have been presented in [7,9,16,17,32,33] on structured grids. An application of this method on unstructured grids has been presented in [1]. This approach with a less accurate state and costate solution has been performed in Iollo et al. [31].

In [38], Ta'asan proposed another approach in which pseudo-time embedding is suggested for the state and costate equations and the design equation is solved as an additional boundary condition, specially for boundary control problems. Using this method, one finds a steady state solution of the following system of equations

$$\begin{aligned} &\frac{\mathrm{d}w}{\mathrm{d}t} + c(w,q) = 0, \\ &\frac{\mathrm{d}\lambda}{\mathrm{d}t} + \nabla_w L(w,q,\lambda) = 0, \\ &\nabla_q L(w,q,\lambda) = 0. \end{aligned} \tag{6}$$

This is still a system of differential algebraic equations, where one has to provide some means to solve the design equation alone. In previous work of Hazra and Schulz [11], the above formulation was superseded by constructing a system consisting of only ODEs, which has been applied to an academic test problem (boundary control problem in elliptic equations).

The proposed new method for solving the above problem (4) is simultaneous pseudo-time stepping. It is well known that there is a strong correlation between iterative methods and pseudo-time stepping which has been exploited for the construction of a time-stepping method in the spirit of reduced SQP-methods. That is, to determine the solution of (4) we look for the steady state solutions of the following pseudo-time embedded evolution equations

$$\begin{aligned} &\frac{\mathrm{d}w}{\mathrm{d}t} + c(w,q) = 0, \\ &\frac{\mathrm{d}\lambda}{\mathrm{d}t} + \nabla_w L(w,q,\lambda) = 0, \\ &\frac{\mathrm{d}q}{\mathrm{d}t} + \nabla_q L(w,q,\lambda) = 0. \end{aligned} \tag{7}$$

This formulation is advantageous since the steady-state flow is obtained by integrating the pseudo-unsteady Euler (or Navier–Stokes) equations in this problem class. Therefore, one can use the same time-stepping scheme for the whole set of equations and preconditioners can be used to accelerate the convergence. Certain preconditioners are equivalent to SQP methods (see following section), whose mathematical background is well studied. In the present paper, we have implemented this method for the shape design example using Euler equations. The number of iterations required for the full optimization problem is less than four times that required for the forward simulation (only) problem. This means a drastic reduction of the computational cost compared to the usual gradient methods.

The paper is organized as follows. In the following Section we discuss the preconditioners which stem from RSQP methods. Section 3 presents the preconditioned pseudo-unsteady optimization problem. In Section 4, we present the state, costate and design equations whose discretization is discussed in Section 5. Numerical results are presented in Section 6. We draw our conclusions in Section 7.

2. Preconditioners

For the solution of problem (1), we recall a straight forward reduced SQP (RSQP) method. A detailed discussion of this approach can be found in [34,35]. Industrial applications of the RSQP concepts are discussed in [2,26,36,37]. Here we present the outline of the method.

Reduced SQP methods are related to projected Lagrangian methods (cf. [10]) and are most advantageous in cases, where the number of degrees of freedom (here the design parameters) is small compared to the number of state variables. The variable steps in each iteration can be considered linear combinations of steps towards optimality and steps towards feasibility of the constraints. The constraints are linearized by a Taylor expansion up to first order terms, so that all steps towards optimality lie in the tangent space of c of the current approximation (w,q):

$$c(w,q) + J(w,q)\,\Delta w + \frac{\partial c}{\partial q}(w,q)\,\Delta q = 0.$$

The optimization problem is projected to this tangent space and approximated by a quadratic problem with the projected Hessian of the Lagrangian given by (2).

In this method, the computationally expensive operation of computing the exact projected Hessian is typically avoided by using appropriate update formulas. It can be proven that under mild conditions the reduced SQP method with the update formulas for the reduced Hessian shows super-linear local convergence properties (s. [34]).

In the above method, it is necessary to invert the Jacobian J of the constraints. In many cases, that is not a viable approach. The following considerations allow for replacing J with an approximate operator A which is invertible with comparatively low cost.

To this end, one employs an inexact reduced SQP method as introduced in [35]. Although it is not inverted, the Jacobian is still used for the computation of the correct adjoint variables and the correct state increments in the sense of defect correcting iterations. The algorithm of this method reads as follows:

Algorithm 1. The RSQP method with an approximate Jacobian.

(0) Set $k:=0$, $\lambda_0 = 0$; start at some initial guess w_0, q_0.
(1) Compute the increment of the adjoint variables from the linear system with the adjoint operator A^*
$A^*(w_k,q_k)\Delta\lambda_k := \nabla_w I(w_k,q_k) - J^*(w_k,q_k)\lambda_k$;
compute the reduced gradient
$\gamma_k := \nabla_q I(w_k,q_k) - \left(\frac{\partial c}{\partial q}(w_k,q_k)\right)^*(\lambda_k + \Delta\lambda_k)$;
determine some approximation B_k of the projected Hessian of the Lagrangian.

(2) Solve $B_k \Delta q_k = -\gamma_k$.
(3) Compute step on w from the linear system
$A(w_k, qk)\Delta wk := -\frac{\partial c}{\partial q}(w_k, q_k)\Delta q_k - c(w_k, q_k)$.
(4) Set $w_{k+1} := w_k + \Delta w_k$, $q_{k+1} := q_k + \Delta q_k$ and $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$.
(5) $k := k + 1$; go to (1) until convergence.

A step of this method can also be interpreted as an approximate Newton step for the necessary conditions of finding the extremum of problem (1), since the updates of the variables are computed according to the linear system3

$$\begin{pmatrix} 0 & 0 & A^* \\ 0 & B & \left(\frac{\partial c}{\partial q}\right)^* \\ A & \frac{\partial c}{\partial q} & 0 \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta q \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -c \end{pmatrix}. \tag{8}$$

This is the basic formulation of the inexact reduced SQP methods that we are using in the subsequent sections.

3. Pseudo-timestepping for optimization problems

Corresponding to the presentation in Section 1, the necessary conditions (4) for optimization can be considered as an overall nonlinear equation

$$F(w, \lambda, q) = \begin{pmatrix} \nabla_w L(w, \lambda, q) \\ \nabla_q L(w, \lambda, q) \\ c(w, \lambda, q) \end{pmatrix} = 0,$$

to be solved. In this way, it would be possible to perform an optimization strategy in a consistent way with the time stepping method that is often used for the solution of the design equation alone.

The pseudo-time embedded system (7) is usually a stiff system of ODEs. Therefore, explicit time-stepping schemes may converge very slowly or might even diverge. In order to accelerate convergence, this system needs some preconditioning. In this paper, we use the inverse of the matrix in Eq. (8) as a preconditioner for the time-stepping process. The pseudo-time embedded system of ODEs that we consider is

$$\begin{pmatrix} \dot{w} \\ \dot{q} \\ \dot{\lambda} \end{pmatrix} = \begin{bmatrix} 0 & 0 & A^* \\ 0 & B & \left(\frac{\partial c}{\partial q}\right)^* \\ A & \frac{\partial c}{\partial q} & 0 \end{bmatrix}^{-1} \begin{pmatrix} -\nabla_w L \\ -\nabla_q L \\ -c \end{pmatrix}. \tag{9}$$

This seems natural since Eq. (8) can be considered as an explicit Euler discretization for the corresponding time-stepping that we envision. Also, due to its block structure, it is computationally inexpensive. The preconditioner employed is similar to the preconditioners for KKT-systems discussed in [4,3] in the context of Krylov subspace methods and in [5] in the context of Lagrange–Newton–Krylov–Schur methods.

Within the inexact reduced SQP-preconditioner, one has to look for an appropriate approximation of the reduced Hessian, B. In particular, when dealing with partial differential equations constituting the state equations, the reduced Hessian can often be expressed as a pseudo-differential operator. Pseudo-differential

operators [14,20] are characterized by their, so-called, symbol in terms of Fourier analysis. This can be exploited for preconditioning purposes as in [11].

4. Detailed equations of the aerodynamic shape optimization problem

In this section, we explain briefly the state, costate and design equations represented in Eqs. (4) for the shape optimization problem.

4.1. State equations

Since we are interested in the steady flow, a proper approach for numerical modeling is to integrate the unsteady Euler equations in time until a steady state is reached. These equations in Cartesian coordinates (x,y) for two-dimensional flow can be written in integral form for the region Ω with boundaries $\partial\Omega$ $(=B \cup C)$ (see Fig. (1)) as

$$\frac{\partial}{\partial t}\int_{\Omega} w \, d\Omega + \int_{\partial\Omega} F \cdot \vec{n} \, ds = 0, \tag{10}$$

where $\vec{n}$ denotes the unit outward normal to $\partial\Omega$ and

$$w := \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad F := [f, g], \quad f := \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix} \quad \text{and} \quad g := \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{bmatrix}.$$

For a perfect gas the pressure and total enthalpy is given by

$$p = (\gamma - 1)\rho\left\{E - \frac{1}{2}(u^2 + v^2)\right\}, \quad H = E + \frac{p}{\rho},$$

respectively. The boundary conditions used to solve these equations are the zero normal velocity on the solid wall C, and the farfield boundary B is treated by considering the incoming and outgoing characteristics based on the one dimensional Riemann invariants.

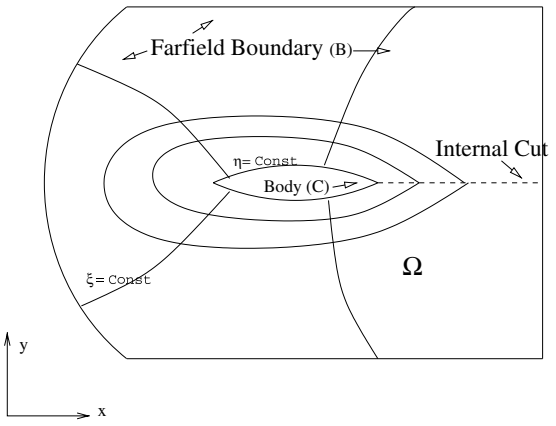

Fig. 1. Physical domain of the problem.

The cost function that we choose in the present optimization problem is drag reduction (with the geometric constraint of constant thickness of the airfoil). Hence, the cost function reads as

$$I(w,q) := C_{\mathrm{D}} = \frac{1}{C_{\mathrm{ref}}} \int_C C_{\mathrm{p}} \left(n_x \cos\alpha + n_y \sin\alpha\right) \mathrm{d}s, \tag{11}$$

where the surface pressure coefficient is defined by

$$C_{\mathrm{p}} := \frac{2(p - p_\infty)}{\gamma M_\infty^2 p_\infty}. \tag{12}$$

The other constraint on constant thickness is maintained as we replace the airfoil by its camberline representation. The geometry is modeled by Hicks–Henne functions. The y-coordinates of the surface are written in parametric form. These parameters are the design variables of the optimization problem.

4.2. Costate equations

The costate or adjoint Euler equations are given by (see, for example, [8])

$$\frac{\partial}{\partial t} \int_\Omega \lambda \,\mathrm{d}\Omega + \int_{\partial\Omega} \bar{F} \cdot \vec{n} \,\mathrm{d}s = 0, \tag{13}$$

where the vector λ contains the components of the adjoint variable and $\bar{F}$ is the matrix of adjoint flux density, defined as

$$\lambda := \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix}, \quad \bar{F} := \left[\left(\frac{\partial f}{\partial w}\right)^{\mathrm{T}} \lambda, \left(\frac{\partial g}{\partial w}\right)^{\mathrm{T}} \lambda \right].$$

The boundary conditions for the adjoint Euler equations on the solid body are of Neumann-type and for the above mentioned cost function they are given by

$$n_x \lambda_2 + n_y \lambda_3 = -\frac{2}{\gamma M_\infty^2 p_\infty C_{\mathrm{ref}}} \left(n_x \cos\alpha + n_y \sin\alpha\right), \quad \text{on } C. \tag{14}$$

The farfield boundary conditions are based upon incoming and outgoing characteristics and free-stream conditions apply there as well. It is important to note that the adjoint Euler equations are linear in λ and the wall boundary conditions depend on the cost function.

4.3. Design equation

For the design Eq. (4b), we need an expression for the derivative of the Lagrangian with respect to the geometry of the airfoil. All the computations are carried out in a generalized coordinate system. Therefore, a transformation is used to transform the physical (x, y)-domain to the computational (ξ, η)-domain. In the computational domain, the components of the gradient $(\partial L)/(\partial q)$ can be determined by integrating the adjoint solutions multiplied by the metric sensitivities as follows

$$\begin{aligned} \left(\frac{\partial L(q+\epsilon\tilde{q})}{\partial q}\right)\bigg|_{\epsilon=0} = &- \int_C p(-\lambda_2 \tilde{q}_\xi^y + \lambda_3 \tilde{q}_\xi^x) \,\mathrm{d}s - \int_\Omega \left(\lambda_\xi^{\mathrm{T}} \left(\tilde{q}_\eta^y f - \tilde{q}_\eta^x g\right) + \lambda_\eta^{\mathrm{T}} \left(-\tilde{q}_\xi^y f + \tilde{q}_\xi^x g\right)\right) \mathrm{d}\Omega \\ &+ \frac{1}{C_{\mathrm{ref}}} \int_C C_{\mathrm{p}} \left((\tilde{q}^\perp)^x \cos\alpha + (\tilde{q}^\perp)^y \sin\alpha\right) \mathrm{d}s, \end{aligned} \tag{15}$$

where $\tilde{q}$ is the variation in the geometry of the airfoil and $\tilde{q}^x, \tilde{q}^y$ are its x- and y-components, $((\tilde{q}^{\perp})^x, (\tilde{q}^{\perp})^y)$ are the components of the unit normal to $\tilde{q}$.

5. Discretization

5.1. State equations

The governing equations are discretized following the method of lines. Space discretization of the compressible Euler equations is carried out using a cell centered finite volume scheme. The physical domain is subdivided into a large number of quadrilateral cells ($\Omega_{i,j}$) as shown in Fig. 2. Since the conservation laws, Eq. (10), are valid for any arbitrary control volume, they also hold locally for each cell ($\Omega_{i,j}$). Hence,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega i,j} w\,\mathrm{d}\Omega + \int_{\partial\Omega_{i,j}} F\cdot\vec{n}\,\mathrm{d}s = 0, \tag{16}$$

where the boundary $\partial\Omega_{i,j}$ consists of the four sides of the quadrilateral, and $\vec{n}$ is the unit outward normal to the surface. The flow quantities w are taken to be volume averaged at the center (i, j) of the cell $\Omega_{i,j}$ (see Fig. 2), that is,

$$w_{i,j} := \frac{1}{V_{i,j}}\int_{\Omega_{i,j}} w\,\mathrm{d}\Omega, \tag{17}$$

where $V_{i,j}$ is the volume of the cell $\Omega_{i,j}$. If the mesh is time independent and the second integral of (16) is approximated using the mid-point rule, the discrete analog of Eq. (16) is written as

$$V_{i,j}\left(\frac{\mathrm{d}}{\mathrm{d}t}w_{i,j}\right) + Q_{i,j} = 0, \tag{18}$$

where $Q_{i,j}$ represents the net flux out of a cell (i, j) and is given by

$$Q_{i,j} := F_{i,j+\frac{1}{2}}\vec{S}_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}}\vec{S}_{i,j-\frac{1}{2}} + F_{i+\frac{1}{2},j}\vec{S}_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}\vec{S}_{i-\frac{1}{2},j}, \tag{19}$$

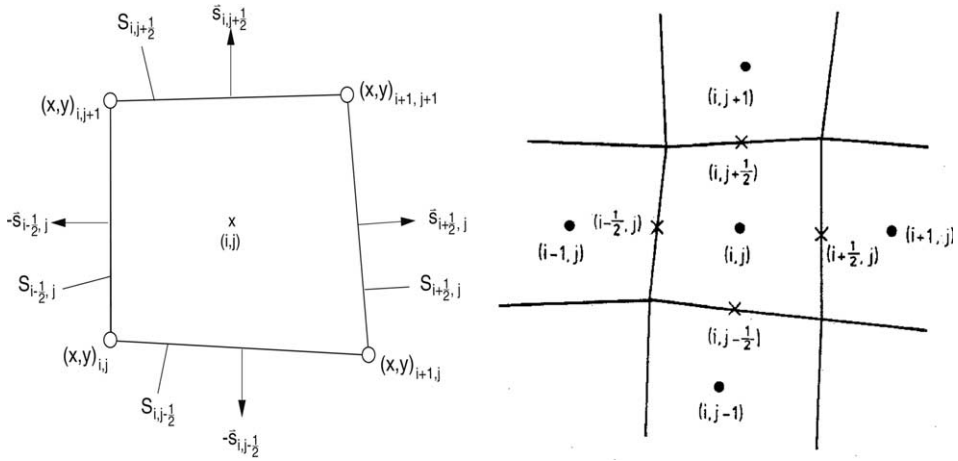

Fig. 2. Quadrilateral cell (i, j) (left) and location of dependent variables (●) and flux values (×) (right).

with $\vec{S}_{i,j-\frac{1}{2}}$ being the normal to the side $S_{i,j-\frac{1}{2}}$ and $F_{i,j-\frac{1}{2}}$ is calculated using the average of w at the cell centers (i,j) and $(i,j-1)$. The details of the flux computation can be found in [21]. The semi-discrete equations are augmented with 1st and 3rd order artificial dissipations $D_{i,j}$, as described in [19],

$$V_{i,j}\left(\frac{\mathrm{d}}{\mathrm{d}t}w_{i,j}\right)+Q_{i,j}-D_{i,j}=0. \tag{20}$$

These equations are then integrated in time using a 5-stage Runge–Kutta type scheme. This scheme takes the following form for Eq. (20) at time level n:

$$\begin{aligned} w_{i,j}^{(0)} &= w_{i,j}^{n} \\ w_{i,j}^{(1)} &= w_{i,j}^{(0)}-\alpha_1\Delta t P_{i,j}^{(0)} \\ &\vdots \\ w_{i,j}^{(5)} &= w_{i,j}^{(0)}-\alpha_5\Delta t P_{i,j}^{(4)} \\ w_{i,j}^{(n+1)} &= w_{i,j}^{(5)}, \end{aligned} \tag{21}$$

where the residuals

$$P_{i,j}^{(k)} := \frac{1}{V_{i,j}}\left(Q_{i,j}^{(k)}-D_{i,j}^{(k)}\right), \quad k=0,1,2,3,4$$

and the values of the constant coefficients are $\alpha_1 = 1/4$, $\alpha_2 = 1/6$, $\alpha_3 = 3/8$, $\alpha_4 = 1/2$, $\alpha_5 = 1$.

In this case the steady state is independent of the time step Δt and is amenable to a variety of techniques for rapid convergence. For stability, a modified condition, as in [21],

$$\Delta t_{i,j} \leqslant \kappa V_{i,j}\left[\mid \bar{q}_{i,j}\cdot\vec{S}_{i+\frac{1}{2},j}\mid+\mid \bar{q}_{i,j}\cdot\vec{S}_{i,j+\frac{1}{2}}\mid+a_{i,j}\left\{\mid \vec{S}_{i+\frac{1}{2},j}\mid+\mid \vec{S}_{i,j+\frac{1}{2}}\mid\right\}\right]^{-1}, \tag{22}$$

has been used to determine the time step for each cell. Here, κ is the Courant number and $\bar{q}_{i,j}$ and $a_{i,j}$ represent the velocity vector and the velocity of sound, respectively, at (i,j). Thus, the stability limit on Δt for a time accurate calculation is

$$\Delta t = \min_{i,j}\Delta t_{i,j}. \tag{23}$$

The solution is advanced in time using the local time step $\Delta t_{i,j}$ as in Eq. (22), based on κ, instead of Eq. (23). This allows for faster signal propagation and thus faster convergence. The details of the grid generation and solution methodology can be found in [22,23].

5.2. Costate equations

Due to structural similarity of the state and costate equations, it is obvious that one can use the same solver for both sets of equations. These equations are also discretized in space using a cell centered finite volume scheme on the same computational grid as described for the Euler equations. The adjoint equations in each cell (i,j) are written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{i,j}}\lambda\,\mathrm{d}\Omega+\int_{\partial\Omega_{i,j}}\bar{F}\cdot\vec{n}\,\mathrm{d}s. \tag{24}$$

Analogous to Eq. (19), the adjoint flux $\tilde{Q}_{i,j}$ is computed as

$$\tilde{Q}_{i,j} := \bar{F}_{i,j+\frac{1}{2}}\vec{S}_{i,j+\frac{1}{2}}-\bar{F}_{i,j-\frac{1}{2}}\vec{S}_{i,j-\frac{1}{2}}+\bar{F}_{i+\frac{1}{2},j}\vec{S}_{i+\frac{1}{2},j}-\bar{F}_{i-\frac{1}{2},j}\vec{S}_{i-\frac{1}{2},j},$$

where the averaged tensors of flux density is computed as

$$\bar{F}_{i,j+\frac{1}{2}} := \left[\left(\frac{\partial f}{\partial w}\right)^{\mathrm{T}}_{i,j} \frac{\lambda_{i,j+1}+\lambda_{i,j}}{2}, \left(\frac{\partial g}{\partial w}\right)^{\mathrm{T}}_{i,j} \frac{\lambda_{i,j+1}+\lambda_{i,j}}{2} \right],$$

with $\lambda_{i,j}$ being the volume averaged value of λ as defined in Eq. (17). Averaging the cell face normals $\vec{S}$ leads to

$$\tilde{Q}_{i,j} = \sum_{\substack{l\in\{i,j\} \\ m\in\{i,j\}\setminus\{l\}}} \left(\bar{s}^{(l)}_x \left(\frac{\partial f}{\partial w}\right)^{\mathrm{T}}_{i,j} + \bar{s}^{(l)}_y \left(\frac{\partial g}{\partial w}\right)^{\mathrm{T}}_{i,j} a \right) \frac{\lambda_{m,l+1}-\lambda_{m,l-1}}{2},$$

where

$$\begin{pmatrix} \bar{s}^{(l)}_x \\ \bar{s}^{(l)}_y \end{pmatrix} := \frac{\vec{S}_{m,l+\frac{1}{2}} + \vec{S}_{m,l-\frac{1}{2}}}{2},$$

and enables to evaluate the adjoint flux efficiently by introducing the transformation

$$T^{-1} = \begin{pmatrix} 1 & -u & -v & \frac{1}{2}(u^2+v^2) \\ 0 & 1 & 0 & -u \\ 0 & 0 & 1 & -v \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

so that

$$\tilde{Q}^{(l)}_{i,j} = T \left(\bar{s}^{(l)}_x \left(\frac{\partial f}{\partial w}\right)^{\mathrm{T}}_{i,j} + \bar{s}^{(l)}_y \left(\frac{\partial g}{\partial w}\right)^{\mathrm{T}}_{i,j} \right) T^{-1}.$$

Introducing $\hat{\lambda}^{(l)}_{i,j} := T \frac{\lambda_{m,l+1}-\lambda_{m,l-1}}{2}$, one gets

$$\tilde{Q}_{i,j} = \sum_{l\in\{i,j\}} T^{-1} \tilde{Q}^{(l)}_{i,j} \hat{\lambda}^{(l)}_{i,j}, \tilde{Q}^{(l)}_{i,j} = \begin{pmatrix} u\bar{s}^{(l)}_x + v\bar{s}^{(l)}_y & 0 & 0 & 0 \\ \bar{s}^{(l)}_x & u\bar{s}^{(l)}_x + v\bar{s}^{(l)}_y & 0 & \frac{\gamma}{\gamma-1}\frac{p}{\rho}\bar{s}^{(l)}_x \\ \bar{s}^{(l)}_y & 0 & u\bar{s}^{(l)}_x + v\bar{s}^{(l)}_y & \frac{\gamma}{\gamma-1}\frac{p}{\rho}\bar{s}^{(l)}_y \\ 0 & (\gamma-1)\bar{s}^{(l)}_x & (\gamma-1)\bar{s}^{(l)}_y & u\bar{s}^{(l)}_x + v\bar{s}^{(l)}_y \end{pmatrix}.$$

These matrices $\tilde{Q}^{(l)}_{i,j}$ are easy to evaluate and one finally obtains the finite volume scheme

$$V_{i,j}\left(\frac{\mathrm{d}\lambda_{i,j}}{\mathrm{d}t}\right) + \tilde{Q}_{i,j} - D_{i,j} = 0,$$

where $D_{i,j}$ is again the artificial dissipation as described in [19] for the adjoint field vector λ. These equations are integrated in time using the same Runge–Kutta scheme as described above. The details of the spatial discretization and the adjoint flux computations are described in [7].

5.3. Surface parameterization

The airfoil is represented by its camberline so that the constant thickness is maintained during the optimization (otherwise the drag reduction problem will result in a flat geometry). In the current study, the geometry is modeled by Hicks–Henne functions [13]. In this representation the y-coordinates of the surface are written in parametric form. These parameters are the design variables of the optimization problem.

5.4. Gradient computation

As an efficient method of calculating the gradient $(\delta I)_{m=1,\ldots,n} := \nabla_{qI}$, by evaluating the integrals (15), we use, as in [8], the so called 'grid moving technique' based on Reuther's approach (s.[18]). These integrals (15) are dependent on the adjoint field vector λ and the metric sensitivities generated by the perturbation of the geometry (by the n design variables). The idea is to allow the geometry perturbation in the whole flow field (the whole grid) while keeping the far field boundary fixed. For this, one introduces a distance function $R(\xi,\eta) = 1 - \frac{\eta}{\eta_B(\xi)}$, where $\eta_B(\xi)$ denotes the values of η at the far field corresponding to the points $(\xi, 0)$ at the wall (see Fig. 1). Then it holds that

$$R(\xi,0) = 1 \quad \text{at the wall and} \quad R(\xi,\eta_B(\xi)) = 0, \quad \text{at the far field.}$$

The grid for the perturbed (new) geometry is defined by

$$x_{\text{new}} - x_{\text{old}} := R \cdot (x_{\text{new}_s} - x_{\text{old}_s}),$$

$$y_{\text{new}} - y_{\text{old}} := R \cdot (y_{\text{new}_s} - y_{\text{old}_s}),$$

where the index s refers to the values at the surface and $x_{\text{new}}, y_{\text{new}}$ correspond to $x_{\text{new}_s}, y_{\text{new}_s}$, respectively. Thus, the perturbations are

$$\delta x = R\delta x_s, \quad \delta y = R\delta y_s,$$

where $\tilde{q}^x = \delta x_s, \tilde{q}^y = \delta y_s$ in Eq. (15), and the metric sensitivities are

$$\delta x_\xi = R \cdot (\delta x_s)_\xi + R_\xi \delta x_s, \quad \delta y_\xi = R \cdot (\delta y_s)_\xi + R_\xi \delta y_s,$$

$$\delta x_\eta = R \cdot (\delta x_s)_\eta + R_\eta \delta x_s, \quad \delta y_\eta = R \cdot (\delta y_s)_\eta + R_\eta \delta y_s.$$

The second term of each of these equalities is very small (s.[8]) and, therefore, could be neglected from the above expressions. The effect of this simplification has been studied and justified in [8]. In that case the metric sensitivities can be expressed as

$$\delta x_\xi \approx R \cdot (\delta x_s)_\xi, \quad \delta y_\xi \approx R \cdot (\delta y_s)_\xi,$$

$$\delta x_\eta \approx R \cdot (\delta x_s)_\eta, \quad \delta y_\eta \approx R \cdot (\delta y_s)_\eta,$$

and one obtains the components of the gradient (for the cost function (11)) as integrals along the geometry C

$$\begin{aligned}\delta I = &- \int_C \left((\delta y_s)_\eta \int_\eta \frac{\partial \lambda^{\mathrm{T}}}{\partial \xi} (R(\eta) \cdot f)\, \mathrm{d}\eta \right) \mathrm{d}\xi + \int_C \left((\delta x_s)_\eta \int_\eta \frac{\partial \lambda^{\mathrm{T}}}{\partial \xi} (R(\eta) \cdot g)\, \mathrm{d}\eta \right) \mathrm{d}\xi \\ &+ \int_C \left((\delta y_s)_\xi \int_\eta \frac{\partial \lambda^{\mathrm{T}}}{\partial \eta} (R(\eta) \cdot f)\, \mathrm{d}\eta \right) \mathrm{d}\xi - \int_C \left((\delta x_s)_\xi \int_\eta \frac{\partial \lambda^{\mathrm{T}}}{\partial \eta} (R(\eta) \cdot g) \mathrm{d}\eta \right) \mathrm{d}\xi \\ &- \int_C p\Big(-\lambda_2 (\delta y_s)_\xi + \lambda_3 (\delta x_s)_\xi \Big)\, \mathrm{d}\xi + \frac{1}{S_{\text{ref}}} \int_C C_{\mathrm{p}} \Big((\delta y_s)_\xi \cos\alpha - (\delta x_s)_\xi \sin\alpha \Big)\, \mathrm{d}\xi.\end{aligned}$$

The integrals in η have to be evaluated once. Then, for each design variable only the integration in ξ along C remains to be evaluated.

5.5. Grid-perturbation strategy

As the shape of the airfoil changes during the optimization process, the location of the grid nodes has to be adjusted. This can be done by generating a new grid after each design iteration or by using a grid-perturbation strategy after each design iteration.

The strategy follows the idea in [27], but in the present study, a specific property of structured mesh is used. A finite number of cells, namely j_0, surrounding the airfoil are defined, and all nodes belonging to this area are moved exactly as the nodes at the boundary. The remaining unchanged cells are smoothly moved until the farfield. The modified grid is then given by, for each cell i,

$$y(i,j)_{(\text{new})} = y(i,j)_{(\text{old})} + Dy(i) \quad \text{if} \quad j \leqslant j_0$$
$$y(i,j)_{(\text{new})} = y(i,j)_{(\text{old})} + 0.5 * Dy(i)(1 + \cos(pi * Sj)) \quad \text{if} \quad j > j_0$$

where $Dy(i)$ represents the deformation of the surface of airfoil at the cell i, i.e., $Dy(i) = y(i,1)_{(\text{new})} - y(i,1)_{(\text{old})}$ and Sj is given by $Sj = (j - j_0)/(j_{\max} - j_0)$, which represents the distance in index notation between the deformed cell (i,j) and the last non-deformed cell belonging to the same i indices, i.e., (i,j_0) cell. Here, $j_{\max}$ represents the total number of cells in j-direction.

The overall algorithm reads as follows:

Algorithm 2. The simultaneous pseudo-timestepping for the preconditioned system

(0) Set $k := 0$; start at some initial guess $\mathbf{x}_0$, λ_0, q_0.
(1) Compute λ^{k+1} using (21) with Δt from Eq. (22).
(2) Determine some approximation B_k of the projected Hessian of the Lagrangian.
(3) March in time one step, using Δt from Eq. (23), the design equation.
$q^{k+1} = q^k - \Delta t\{B_k^{-1}\nabla_q L - B_k^{-1}(\frac{\partial c}{\partial q})^*(A^*)^{-1}\nabla_w L\}$.
(4) Compute $\mathbf{w}^{k+1}$ using (21) with Δt from Eq. (22).
(5) k:=k + 1; go to (1) until convergence.

Step (1) represents a Runge–Kutta-version of the first step $(-(A^*)^{-1}\nabla_w L)$ of the reduced SQP-method (8). The block matrices A and A^* corresponding to the state and costate equations in the preconditioner are just identity matrices in the current implementation.

The algorithm above is a 'one-shot' method since we perform one time-step for each design update. However, it is different than the 'one-shot' methods used in [24,39] in which the design variables are updated in a hierarchical manner.

6. Numerical results and discussion

The optimization method is applied to a test case of the RAE 2822 airfoil. The physical domain is discretized using an algebraically generated (193 × 33) C-grid. On this grid the preconditioned pseudo-stationary equations are solved. The camberline representation of the airfoil is parameterized by 21 Hicks–Henne parameters. The complete optimization cycle is performed under the optimization platform SynapsPointerPro [6]. We start the optimization iteration (i.e., w_0 and λ_0) with the solution obtained after 500 time steps of the state and costate equations. We use the FLOWer code of the German Aerospace Center (DLR) for solving the forward and adjoint equations. These codes are very robust and tested in many applications including flow computations for the company AIRBUS-Germany. Therefore, no modifications were made to this code (which is a typical requirement with most of the industries).

The design equation is integrated in time using an explicit Euler scheme. Therefore the time step used for the three sets of equations are not the same. In the current implementation of FLOWer the time steps are determined independently for each discretization cell according to the local stability.

One of the main issues of using this kind of preconditioned pseudo-timestepping is the approximation of the reduced Hessian. A better approximation will lead to faster convergence of the optimization algorithm. In the current study, we compare the use of two different approximations.

6.1. Case 1

The Hessian is approximated by

$$B_k = \beta\delta_{ij},$$

where β ($\gg 1$) is a constant and δ_{ij} is the Kronecker symbol representing the identity matrix. This kind of approximation is used as initial value for (iterative) BFGS updates of the Hessian. As convergence criterion of the optimization iteration we use the discrete 2-norm of the increments of the profile parameters ($\|q_k - q_{k-1}\|_2$) less than 0.0017.

In this case, convergence of the optimization is achieved after 3700 time-steps. The convergence history is presented in Fig. 3. After convergence is achieved for optimization, we perform another 600 time iterations for state and costate solvers to reduce the residual of these two variables further to get more accurate values of the surface pressure, force coefficients and gradients (which are comparable to the values obtained by other methods).

For the sake of comparison, we computed more exact drag values on the optimized geometry obtained in every 100 iterations (initially in every 10 iterations). This drag is compared with the inexact drag from the optimization cycle in Fig. 4. We observe that the final objectives coincide.

Fig. 5 presents the initial and final sensitivities of the parameters (upper left), airfoils (upper middle), and camber lines (upper right). The surface pressure distributions of the initial and optimized airfoils (obtained by the current method and that obtained by steepest descent methods) are compared in the same figure (lower). Both optimized pressure distributions almost coincide.

6.2. Case 2

In this case, the reduced Hessian approximation is based on second order information, as in the case of BFGS optimization methods. We define $s_k := (q_{k+1} - q_k)$ and $z_k := (\nabla I_{k+1} - \nabla I_k)$, where k represents the

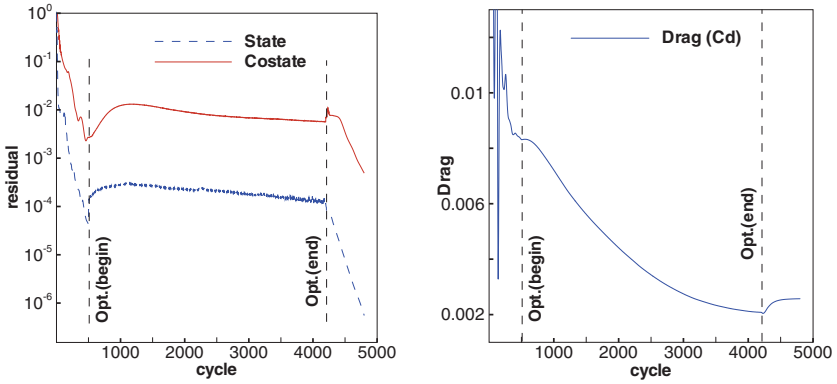

Fig. 3. Convergence history of the optimization iterations.

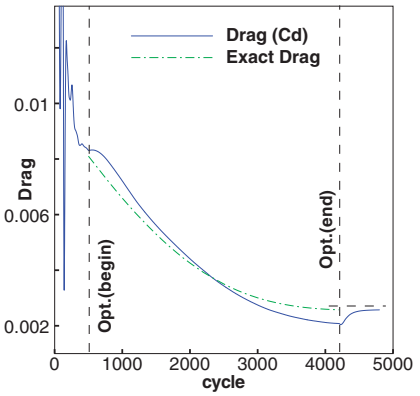

Fig. 4. Comparison of inexact and exact drag reduction during optimization iterations.

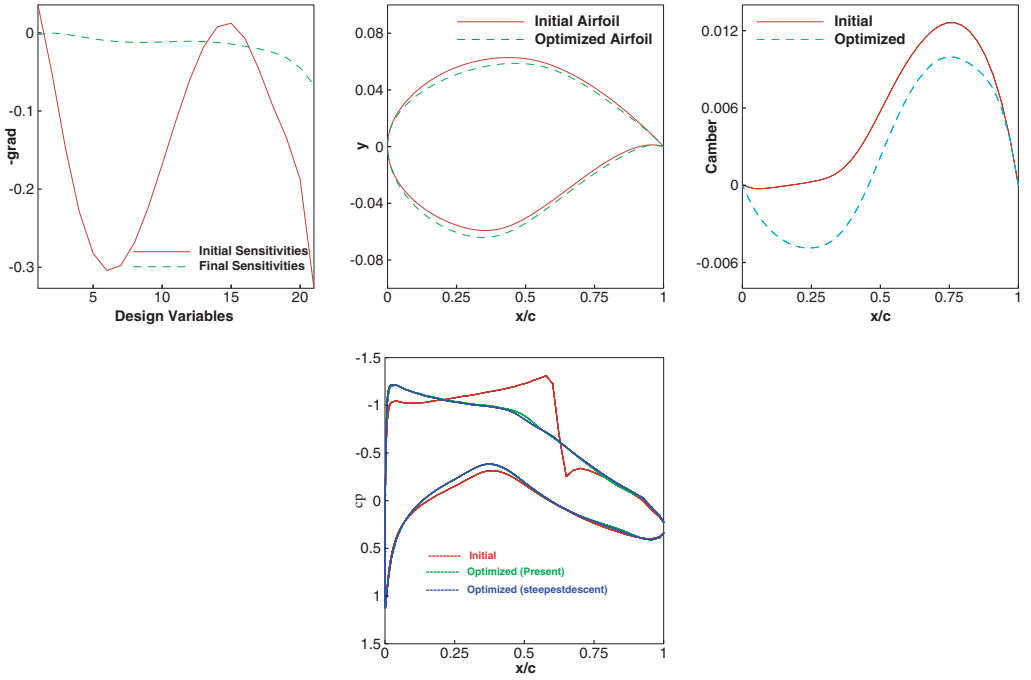

Fig. 5. Comparison of initial and final quantities.

iteration number. Then the curvature in the direction s_k is obtained from the product $(z_k^{\mathrm{T}} s_k)$. If the curvature is positive, the reduced Hessian is approximated by

$$B_k = \bar{\beta} \frac{z_k^{\mathrm{T}} s_k}{z_k^{\mathrm{T}} z_k} \delta_{ij},$$

where $\bar{\beta}$ is a constant. Otherwise, it is approximated by $\beta \delta_{ij}$, where β is a constant as before. Additionally, we impose upper and lower limits on the factor so that

$$\beta_{\min} < \bar{\beta}\frac{z_k^{\mathrm{T}} s_k}{z_k^{\mathrm{T}} z_k} < \beta_{\max}.$$

This prevents the optimizer from taking steps that are too small or too large. The constants $\beta_{\min}$ and $\beta_{\max}$ can be chosen, e.g., depending on the accuracy achieved in one time step by the forward and adjoint solver. This gives the flexibility of using different codes (e.g., a multigrid forward and adjoint solver).

In this case, we started the optimization with the same initial conditions as Case 1. The iteration is stopped when the optimized drag is less than 0.0025 (which is the optimized drag obtained in Case 1). The optimization requires 1225 iterations to reach this criterion, which is less than one-third of that required in Case I. This reflects the fact that a better approximation of the reduced Hessian in the preconditioner leads to faster convergence of the optimization problem. Fig. 6 presents the optimization convergence history of this case. Also presented in Fig. 7 is the comparison of exact drag and the inexact drag during the optimization. Here, also we see that the final values are same in both cases.

Fig. 8 presents the comparisons of initial and final gradients (upper left), airfoils (upper middle), and camber lines (upper right). The optimized surface pressure distribution obtained using the current method and that obtained with steepest descent method are also compared in the same Figure (lower).

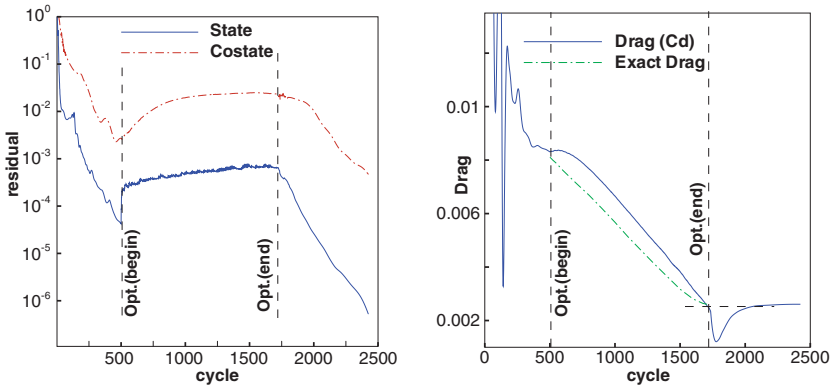

Fig. 6. Convergence history of the optimization iterations.

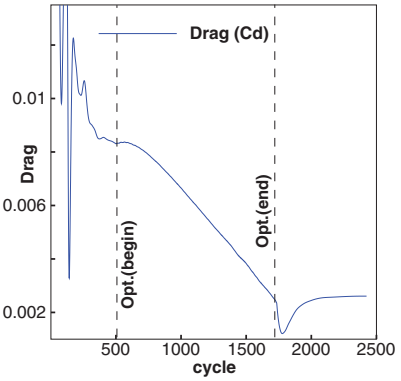

Fig. 7. Comparison of inexact and exact drag reduction during optimization iterations.

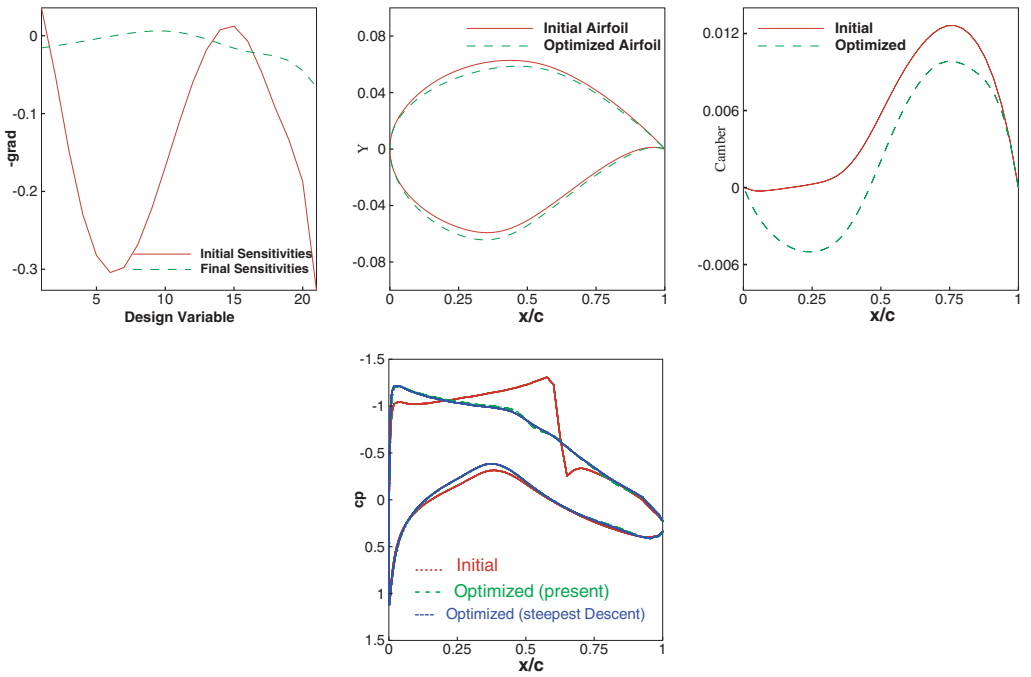

Fig. 8. Comparison of initial and final quantities.

In the pseudo-time optimization iteration, the initial drag of 0.0081012 is reduced to 0.0025996 in the optimization process which is a reduction of about 68%. The lift and pitching moment coefficient history is presented in Fig. 10. Since there is no constraint on these two quantities, they are also reduced by about 10% and 20%, respectively.

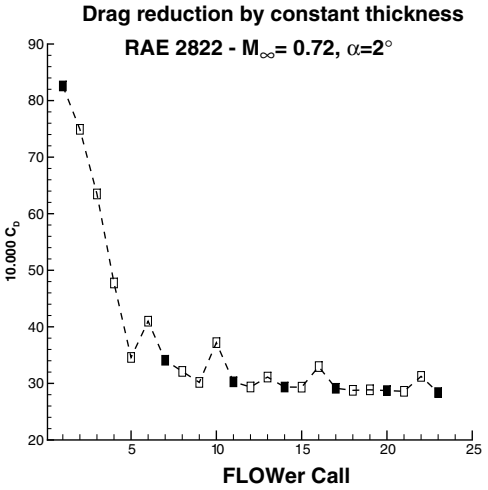

Fig. 9. Reduction of force coefficients during steepest descent optimization iteration.

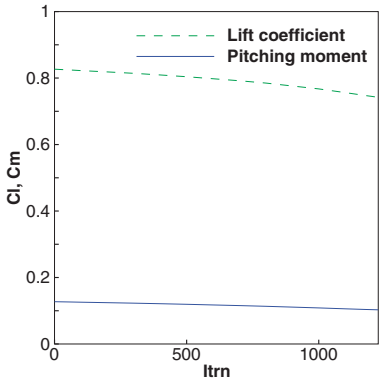

Fig. 10. Lift and pitching moment coefficients during optimization iteration.

Fig. 11 represents the contour plots of the Mach and surface pressure at the initial condition and after optimization. As we see, the initial shock, which causes the major drag in the transonic range, has disappeared completely after the optimization.

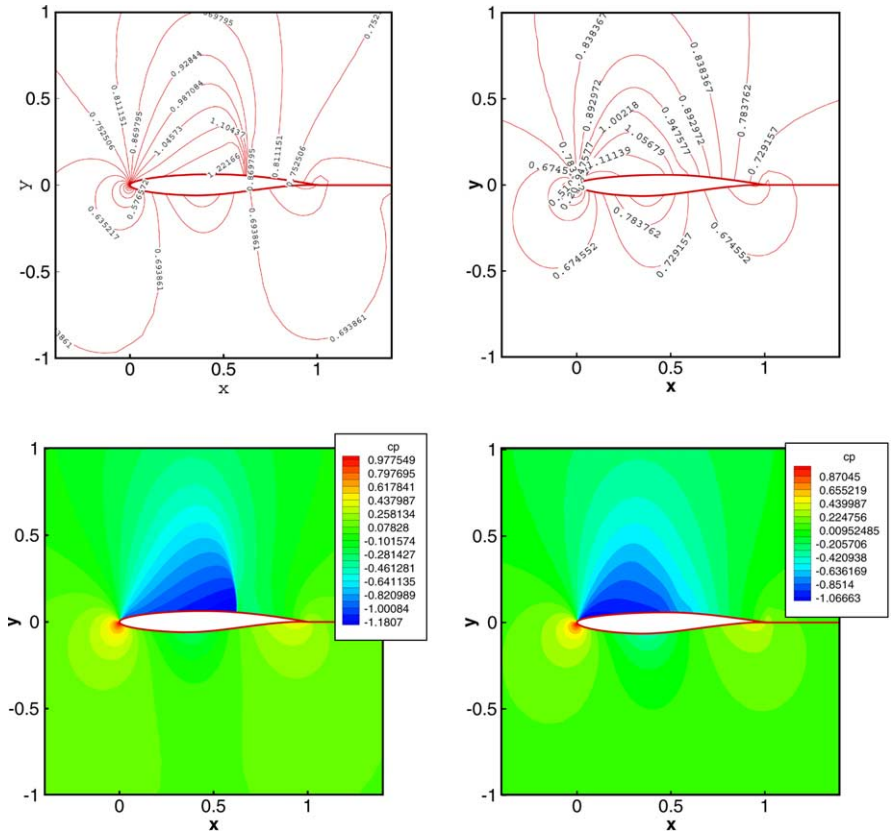

Fig. 11. Comparison of initial (left column) and final (right column) Mach (top) and pressure (bottom) contours.

Both the optimized surface pressures almost coincide with that obtained by steepest descent method. However, the steepest descent method required 23 forward runs (white squares in Fig. 9) and 6 adjoint runs (black squares) together with a line search. Each forward and adjoint run requires approximately 1500 iterations in time. That means that the steepest descent method needs an effort of about 29 forward runs whereas the present method needs an effort of little more than 3 forward runs. Additionally, the simultaneous pseudo-time method needs a new grid, obtained by grid perturbation as discussed earlier, after each optimization iteration. Additional time is required to write the output after every iteration and read the same before each iteration, as the iterations start with solution values from the previous iteration. However, the total time required for this overhead is negligible compared to one complete forward run. If we add all these efforts together, the time taken is still less than 4 forward simulation runs. In terms of CPU time, the complete optimization cycle needs about 40 minutes on an Intel(R) Xeon(TM) CPU 1700 MHz machine. Steepest descent method with restart or a higher order method (e.g., nonlinear CG or SQP) could also be applied which would result in an improvement over the steepest descent results as well. Comparison with these strategies is beyond the scope of this paper. However, it is possible to obtain an impression of this comparison if one imagines what optimization progress could be achieved in 2 optimization iterations (2 forward + 2 adjoint simulation runs) of any gradient based methods, since this takes the equivalent computational effort as a whole optimization run of our one-shot approach.

7. Conclusions

A new method has been proposed for aerodynamic shape optimization which is based on simultaneous pseudo-timestepping. The preconditioned pseudo-stationary state, costate and design equations are integrated simultaneously in time until a steady state is reached. The preconditioner used in this study is motivated by a continuous re-interpretation of reduced SQP methods. A better approximation of the reduced Hessian in the preconditioner leads to faster convergence of the optimization problem. The overall cost of computation is approximately 15% of that of a straight forward application of the steepest descent method. The generalization of the proposed strategy to problems with state constraints (e.g., drag reduction with constant lift) is done in the subsequent work. Applications in 3D is our future goal.

Acknowledgement

We thank the anonymous referees for their invaluable comments and suggestions on this work.

References

[1] W.K. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, AIAA 97-0643, 1997.
[2] H.G. Bock, W. Egartner, W. Kappis, V. Schulz, Practical shape optimization for turbine and compressor blades, Optimiz. Eng. 3 (2002) 395–414.
[3] A. Battermann, M. Heinkenschloss, Preconditioners for Karush–Kuhn–Tucker systems arising in the optimal control of distributed systems, in: W. Desch, F. Kappel, K. Kunisch (Eds.), Optimal Control of PDE, Vorau 1996, Birkhäuser Verlag, 1996, pp. 15–32.
[4] A. Battermann, E.W. Sachs, Block preconditioners for KKT systems in PDE-governed optimal control problems, in: K.H. Hoffmann, R.H.W. Hoppe, V. Schulz (Eds.), Fast Solution of Discretized Optimization Problems, Birkhäuser Verlag, 2001, pp. 1–18.
[5] G. Biros, O. Ghattas, Parallel Lagrange-Newton-krylov-schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver, Tech. Rep., Laboratory for Mechanics, Algorithms and Computing, Carnegie Mellon University, 2000.

[6] O. Fromman, SynapsPointerPro v2.50, Synaps Ingenieure Gesellschaft mbH, Bremen, Germany, 2002.
[7] N.R. Gauger, Aerodynamic shape optimization using the adjoint Euler equations. in: Proceedings of the GAMM Workshop on Discrete Modelling and Discrete Algorithms in Continuoum Mechanics (ISBN 3-89722-683-9) Berlin: Logos Verlag, 2001, pp. 87–96.
[8] N.R. Gauger, Das Adjungiertenverfahren in der aerodynamischen Formoptimierung, DLR-Report No. DLR-FB-2003-05 (ISSN 1434-8454), 2003.
[9] N.R. Gauger, J. Brezillon, Aerodynamic shape optimization using adjoint method, J. Aero. Soc. India 54 (3) (2002).
[10] P.E. Gill, W. Murry, M.H. Wright, Practical Optimization, Academic Press, London, 1981.
[11] S.B. Hazra, V. Schulz, Simultaneous pseudo-timestepping for PDE-model based optimization problems. To appear in BIT 2004.
[12] S.B. Hazra, V. Schulz, Simultaneous pseudo-timestepping for Aerodynamic shape optimization problems with state constraints, Forschugsbericht Nr.04-6, Department of Mathematics/Computer Science, University of Trier, Germany, ISSN 0944-0488, SIAM J. Opt., 2004, submitted.
[13] R.M. Hicks, P.A. Henne, Wing design by numerical optimization, J. Aircraft 15 (1978) 407–412.
[14] L. Hörmander, Pseudo-differential operators, Comm. Pure Appl. Math. 18 (1965) 501–517.
[15] A. Jameson, Aerodynamic design via control theory, J. Scientific Comput. 3 (1988) 23–260.
[16] A. Jameson, Automatic design of transonic airfoils to reduce shock induced pressure drag, MAE Report 1881, presented in at the 31st Israel Annual Conference on Aviation and Aeronautics, February, 1990.
[17] A. Jameson, Optimum aerodynamic design using CFD and control theory, in: AIAA 12th Computational Fluid Dynamics Conference, AIAA 95-1729-CP, June, 1995.
[18] A. Jameson, J. Reuther, Control theory based airfoil design using the Euler equations, AIAA in: Proceedings of 94-4272-CP, 1994.
[19] A. Jameson, W. Schmidt, E. Turkel, Numerical solution of the Euler equation by finite volume methods using Runge–Kutta time-stepping schemes, AIAA 81-1259, 1981.
[20] J.J. Kohn, L. Nirenberg, On the algebra of pseudo-differential operators, Comm. Pure Appl. Math. 18 (1965) 269–305.
[21] N. Kroll, R.K. Jain, Solution of two-dimensional Euler equations – Experience with a finite volume code, DFVLR-IB-129-84/19, 1984.
[22] N. Kroll, C.C. Rossow, K. Becker, F. Thiele, The MEGAFLOW – a numerical flow simulation system, in: 21st ICAS Symposium, Paper 98-2.7.4, Melbourne, Australia, 1998.
[23] N. Kroll, C.C. Rossow, K. Becker, F. Thiele, The MEGAFLOW project, Aerosp., Sci. Technol. 4 (2000) 223–237.
[24] G. Kuruvila, S. Ta'asan, M. Salas, Airfoil optimization by the one-shot method, AGARD-FDP-VKI, Special Course on Optimum Design Methods in Aerodynamics, April 1994.
[25] J.L. Lions, Optimal Control of Systems Governed by Partial Differential Equations, Springer-Verlag, New York, 1971.
[26] D. Logashenko, B. Maar, V. Schulz, G. Wittum, Optimal geometrical design of Bingham parameter measurement devices, International Series of Numerical Mathematics (ISNM) 138 (2001) 167–183.
[27] M. Nemec, D.W. Zingg, From analysis to design of high-lift configurations using a Newton-Krylov algorithm, ICAS Congress 2002.
[28] O. Pironneau, On optimum design in fluid mechanics, J. Fluid Mech. 64 (1974) 97–110.
[29] O. Pironneau, On optimum profiles in Stokes flow, J. Fluid Mech. 59 (1973) 117–128.
[30] O. Pironneau, Optimal shape design for elliptic systems, Springer-Verlag, New York, 1982.
[31] A. Iollo, G. Kuruvila, S. Ta'asan, pseudo-time method for optimal shape design using Euler equations, ICASE Report No. 95-59, 1995.
[32] J. Reuther, A. Jameson, Aerodynamic shape optimization of wing and wing-body configurations using control theory, AIAA 95-0123, January, 1995.
[33] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders, Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation, AIAA 96-0094, January, 1996..
[34] V.H. Schulz, Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots, Ph.D. Thesis, Universität Heidelberg, 1996.
[35] V.H. Schulz, Solving discretized optimization problem by partially reduced SQP methods, Comput. Vis. Sci. 1 (1998) 83–96.
[36] V. Schulz, SQP-based direct discretization methods for practical optimal control problems, (guest editor) Special issue of the Journal of Computational and Applied Mathematics 2000, 120, 1–2.
[37] M. von Schwerin, O. Deutschmann, V. Schulz, Process Optimization of Reactive Systems by Partially Reduced SQP Methods, Computers Chem. Eng. 24 (2000) 89–97.
[38] S. Ta'asan, Pseudo-time methods for constrained optimization problems governed by PDE, ICASE Report No. 95-32, 1995.
[39] S. Ta'asan, G. Kuruvila, Aerodynamic design and optimization in one shot. AIAA 92-0025, January 1992.