Computers & Fluids 74 (2013) 102–113

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

On the handling of turbulence equations in RANS adjoint solvers

Andre C. Marta [a,*], Sriram Shankaran [b]

[a] Center for Aerospace Science and Technology, Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal
[b] Global Research Center, General Electric, 1 Research Circle, Niskayuna, NY 12309, USA

ARTICLE INFO

Article history:
Received 10 May 2011
Received in revised form 7 January 2013
Accepted 10 January 2013
Available online 30 January 2013

Keywords:
Adjoint method
Discrete approach
Turbulence models
Approximation models
Constant eddy viscosity
Shape optimization

ABSTRACT

Recent developments in numerical design tools have made practical the use of gradient-based optimization using high-fidelity computational fluid dynamic simulations. Such has been made possible with the use of adjoint solvers, that can efficiently provide gradients of functions of interest with respect to design variables. However, in the presence of flows modeled by the Reynolds-Averaged Navier–Stokes (RANS) equations, the corresponding adjoint might become too complex to be fully derived or run. This has led to the use of many simplifications in the implementation of such adjoint solvers. In this paper, the constant eddy viscosity (CEV) approximation is explained and its validity tested. Two cases are used, a two-dimensional turbine vane blade and a three-dimensional transonic compressor rotor blade. The gradients computed using both the full RANS and the CEV approximation adjoints are verified against finite-differences. It is shown that the gradients differ slightly but when used in an optimization problem, the optimal solution found is nearly identical. Therefore, the CEV approximation in RANS adjoint solvers proved to be valid for engineering design problems, bringing significant advantages, such as faster implementation and less computational resources needed in terms of CPU and memory size, when compared to the full RANS adjoint solver.

1. Introduction

The continuous growth of computational power has made external and internal flow simulations to be routinely performed using high-fidelity computational fluid dynamic (CFD) models. The emerging trend is to use optimization techniques as part of the design tools, with numerical design optimization becoming common practice not only in academia but also in industry.

Among the several optimization methods developed by the operations research field [1], and considering that CFD flow simulations can take hours, if not days, to perform, the most efficient methods are gradient-based, which require a minimal number of cost function evaluations. However, these methods require an estimate of the cost function derivatives. To address this, the designer faces the problem of evaluating the derivatives [2]. Finite-difference (FD) approximations have always been popular due to their simplicity but they rapidly become computationally prohibitive when the number of variables greatly exceeds the number of functions. In this case, an adjoint method is the best-suited approach to efficiently estimate function gradients since the cost involved in calculating sensitivities using the adjoint method is practically independent of the number of design variables.

The application of adjoint methods to CFD was pioneered by Pironneau [3] and it was later revisited and extended by Jameson to perform airfoil [4] and wing [5] design. More recent successful applications include multipoint aerodynamic shape optimization problems [6], aero-structural design optimization [7], and even magnetohydrodynamics flow control [8].

The major drawback of using adjoint-based gradients has always been the necessity of implementing an additional solver – the adjoint system of equations solver, that is generally of the same complexity as the flow solver. Thus, in the presence of flows modeled by the Reynolds-Averaged Navier–Stokes (RANS) equations, the corresponding adjoint system might become far too complex to be fully derived. This has led to the use of many approximations and simplifications in the implementation of such adjoint solvers. The different approaches found in the literature are:

- Euler equations
  Both the flow and adjoint solvers only account for the inviscid flow effects. The argument being that, in some external flows, such as in clean aircraft configurations, and in some internal flows, such as in some turbine blades, the viscous effects can be neglected since there are no regions of flow separation [9].
- RANS with algebraic turbulence models
  The adjoint solver is consistent with the flow solver, but a simplistic turbulence model is used to expedite the development of

[*] Corresponding author. Tel.: +351 218419466.
E-mail address: andre.marta@ist.utl.pt (A.C. Marta).

0045-7930/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.compfluid.2013.01.012

the former solver. Often used when the viscous and turbulent effects needs to be accounted for, but the development effort is kept to a minimum [10].

- RANS with constant eddy viscosity (CEV) approximation
  The flow solver uses proper two-equation turbulence models, such as κ–ϵ or κ–ω, but the adjoint solver assumes frozen eddy viscosity. In this case, the flow is properly solved but it is assumed that the variation of eddy viscosity can be neglected in the adjoint [11,12].
- RANS flow and adjoint solver
  This corresponds to the exact derivation of the adjoint solver, regardless of the complexity of the turbulence model used. The dual (adjoint) solver is perfectly consistent with the primal (flow) solver. This approach is made feasible if one uses the hybrid *ADjoint* methodology to develop the adjoint solver [13].

These approaches are all used today by the adjoint-based design community, but there is no clear evidence of what are the penalties associated with approximation models compared to the exact adjoint solver, when using the adjoint solution to drive a realistic gradient-based optimization problem. This paper intends to shed some insight about the trade-off's between the last two approaches listed, that is to say, to evaluate the constant eddy viscosity approximation model compared to the full RANS adjoint solver when performing design optimization problems.

2. Background

The underlying theory of adjoint-based high-fidelity CFD design optimization is presented next.

2.1. Generic design problem

A generic CFD design problem can be formally described as

$$\begin{aligned} \text{Minimize } & Y(\boldsymbol{\alpha}, \mathbf{q}(\boldsymbol{\alpha})), \\ \text{w.r.t. } & \boldsymbol{\alpha}, \\ \text{subject to } & \mathbf{R}(\boldsymbol{\alpha}, \mathbf{q}(\boldsymbol{\alpha})) = 0, \\ & \mathbf{C}(\boldsymbol{\alpha}, \mathbf{q}(\boldsymbol{\alpha})) = 0, \end{aligned} \tag{1}$$

where Y is the cost function, $\boldsymbol{\alpha}$ is the vector of design variables, $\mathbf{q}$ is the flow solution, which is typically of function of the design variables, and $\mathbf{C} = 0$ represents additional constraints that may or may not involve the flow solution. The flow governing equations expressed in the form $\mathbf{R} = 0$ also appear as a constraint, since the solution $\mathbf{q}$ must always obey the flow physics.

When using a gradient-based optimizer to solve the design problem (1), the evaluation of the cost and constraint functions, and their gradients with respect to the design variables are also required, that is, $\mathrm{d}Y/\mathrm{d}\boldsymbol{\alpha}$ and $\mathrm{d}C_i/\mathrm{d}\boldsymbol{\alpha}$ have to be estimated.

2.2. Flow governing equations

The governing equations used in the present work are the RANS equations. In conservation form, the Navier–Stokes system of equations may be written in index notation as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0, \tag{2a}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j + p\delta_{ij} - \tau_{ji}) = 0, \quad i = 1,2,3, \tag{2b}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho E u_j + p u_j - u_i \tau_{ij} + q_j) = 0, \tag{2c}$$

where ρ, u_i and E are respectively the density, mean velocity and total energy, τ_{ij} is the viscous stress and q_j is the heat flux.

A turbulence model needs to be used to model the Reynolds stresses. In this paper, a two-equation turbulence model was used, in particular the k–ω model of Wilcox [14],

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho k u_j) = \tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta_k \rho k \omega + \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_k \frac{\rho k}{\omega}\right)\frac{\partial k}{\partial x_j}\right], \tag{3a}$$

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x_j}(\rho \omega u_j) = \frac{\gamma \omega}{k}\tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta_\omega \rho \omega^2 + \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_\omega \frac{\rho k}{\omega}\right)\frac{\partial \omega}{\partial x_j}\right], \tag{3b}$$

where k is the turbulence kinetic energy and ω is the specific dissipation rate. The turbulent eddy viscosity is computed from $\mu_T = \rho k/\omega$ and the constants are $\gamma = 5/9$, $\beta_k = 9/100$, $\beta_\omega = 3/40$, $\sigma_k = 1/2$ and $\sigma_\omega = 1/2$. The effective viscosity used in the Navier–Stokes equations (2) is then computed as $\mu = \mu_m + \mu_T$, where μ_m is the molecular (laminar) viscosity.

In semi-discrete form, the RANS governing equations (2, 3) can be expressed as

$$\frac{\mathrm{d}q_{ijk}}{\mathrm{d}t} + \mathrm{R}_{ijk}(\mathbf{q}) = 0, \tag{4}$$

where $\mathbf{q} = (\rho, \rho\mathbf{u}, \rho E, \rho k, \rho\omega)^T$ is the vector of conservative variables, $\mathbf{R}$ is the residual with all of its components (inviscid, viscous and turbulent fluxes, boundary conditions and artificial dissipation), and the triad ijk represents the three computational directions. The unsteady term of (4) is dropped out since only the steady solution of the equation is of interest in this work.

2.3. Adjoint equations

The derivation of the adjoint equations for systems of PDEs follows the work by Giles and Pierce [15]. The adjoint equations can be expressed as

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^T \boldsymbol{\psi} = \left[\frac{\partial Y}{\partial \mathbf{q}}\right]^T, \tag{5}$$

where $\boldsymbol{\psi}$ is the adjoint vector.

If geometric parameters not handled directly by the CFD solver are chosen as design variables $\boldsymbol{\alpha}$, it is necessary to use the chain rule of differentiation to express the gradient of the cost function with respect to the design variables as

$$\frac{\mathrm{d}Y}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\mathrm{d}Y}{\mathrm{d}\mathbf{x}}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\alpha}}, \tag{6}$$

being the total gradient of the cost function with respect to the computational grid coordinates of each node $\mathbf{x}$, based on the adjoint solution $\boldsymbol{\psi}$, given by

$$\frac{\mathrm{d}Y}{\mathrm{d}\mathbf{x}} = \frac{\partial Y}{\partial \mathbf{x}} - \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}. \tag{7}$$

The evaluation of the gradient of each cost or constraint function in the optimization problem (1) requires solving (5) with a new right-hand side vector. On the other hand, the computational cost of the total sensitivity (7) is almost independent of the number of grid coordinates $\mathbf{x}$, which is the feature that makes the adjoint method so attractive for gradient-based optimization involving a large number of variables and a few functions.

3. Implementation

The development of the flow and adjoint solvers and their integration into a design system are described next.

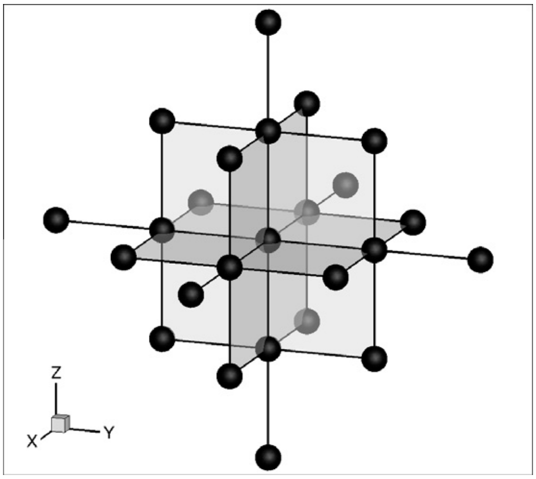


Fig. 1. Computational flow stencil: 25 nodes.

3.1. Flow solver

The proprietary flow solver used in this work supports three-dimensional, multi-block, structured grids, and it uses a finite-volume formulation of the non-linear and linear RANS equations. Several turbulence models are available, such as k–ω, k–ϵ and SST, having the option to use wall functions or wall integration for boundary layer resolution. This solver is typically employed in the solution of turbomachinery blade rows and it is capable of efficiently performing three-dimensional analysis for aeromechanics, aerodynamic design, parametric studies, and robust design applications.

As typical for most iterative CFD flow solvers, the residual calculation is done in a subroutine that loops through the three-dimensional domain and accumulates the several fluxes and boundary conditions contributions in the residual **R**. However, the residual at each computational cell only depends on the flow variables at that cell and at the cells adjacent to it, which define the stencil of dependence, as shown in Fig. 1.

3.2. Adjoint solver

The simple mathematical form of (5) can be very misleading since, depending on the approach, their numerical implementation can be quite complex, if derived by manual differentiation, or quite costly, if derived using finite-differences.

A discrete adjoint approach formulation is chosen because it can be applied to any set of governing equations and it can treat arbitrary cost functions. As such, and in contrast to the continuous approach, no simplifications have to be made during the derivation: the effects of viscosity and heat transfer and the turbulence equations can be easily handled when deriving the discrete adjoint.

But the most interesting feature of the discrete approach is that it allows the use of automatic differentiation (AD) tools [16] in its derivation, expediting considerably the process of obtaining the differentiated form of the discretized governing equations necessary to assemble the adjoint system of equations.

As such, the approach used in this work is hybrid and it follows the work of Marta et al. [13]. The discrete adjoint solver is derived with the aid of an automatic differentiation tool that is selectively applied to the CFD source code that handles the residual and function evaluations. This hybrid approach retains the accuracy of the adjoint methods, while it adds the ease of implementation of the automatic differentiation methods. If the AD tool was directly applied to the original CFD solver, it would produce an inefficient adjoint code that would require unaffordable memory resources and computational time. Therefore, it is necessary to rewrite the residual and function evaluation code, transforming the original flux calculation loop over the computational control volume faces to a complete single control volume evaluation, where all faces including boundary condition treatment are handled. Given that the adjoint solver is built over a legacy CFD solver, there is a significant development effort involved. Had a new CFD solver been developed from scratch having the adjoint solver already in mind, like by extensively making use of pointers or by using a different code structure with segregated residual and function evaluation kernels, much of this effort would have been considerably reduced because the change from the face-based CFD approach to the cell-based adjoint approach would have been achieved by a simple change in the residual (or function) evaluation main routine.

The AD tool is then applied to the rewritten routines, producing the code that evaluates the local partial derivative matrices $\partial \mathbf{R}/\partial \mathbf{q}$, $\partial Y/\partial \mathbf{q}$, $\partial Y/\partial \mathbf{x}$ and $\partial \mathbf{R}/\partial \mathbf{x}$ that are necessary to compute gradients (7) using the adjoint method (5). These local matrices (cell based) are subsequently assembled in the global matrices according to the connectivity given by the computational stencil and the cell global numbering scheme. The AD tool chosen in this work is Tapenade [17] because it supports Fortran 90, which is a requirement taking into account the programming language used in the flow solver.

The sizes of the global matrices involved in this process are

$$\begin{aligned} &\frac{\partial \mathbf{R}}{\partial \mathbf{q}}(N_q \times N_q), \quad \frac{\partial Y}{\partial \mathbf{q}}(N_Y \times N_q), \\ &\frac{\partial \mathbf{R}}{\partial \mathbf{x}}(N_q \times N_x), \quad \frac{\partial Y}{\partial \mathbf{x}}(N_Y \times N_x), \end{aligned} \tag{8}$$

where N_Y is the number of cost functions, N_x the number of grid coordinates and N_q the size of the state vector. The size of the vector **q** depends on the number of governing equations, N_e, and the number of cells of the computational mesh, N_c, that discretizes the physical domain, according to the relation $N_q = N_e N_c$, which for the solution of a large, three-dimensional problem involving a system of conservation laws, can be very large. The size of the grid coordinates vector **x**, is given by dimensionality of the problem times the number of vertexes corresponding to the computational mesh used, that is, $N_x = 3N_v$ for three-dimensional problems. Although $\partial \mathbf{R}/\partial \mathbf{q}$ and $\partial \mathbf{R}/\partial \mathbf{x}$ can easily be very large matrices, they are extremely sparse, as a consequence of the stencil of dependence illustrated in Fig. 1.

The adjoint linear system of Eq. (5) has to be solved N_Y times because $\boldsymbol{\psi}$ is valid for all grid coordinates **x**, but must be recomputed for each function Y. To solve this large sparse discrete adjoint problem, the Portable, Extensible Toolkit for Scientific Computation (PETSc) [18] is used. The adjoint system of equations is solved using a PETSc built-in Krylov subspace method, more specifically, the Generalized Minimum Residual (GMRES) method [19].

3.3. Constant eddy viscosity approximation

The full RANS adjoint solver described so far makes use of the complete vector of conservative variables and handles the corresponding seven governing equations (2, 3).

The CEV approximation still solves the full RANS flow equations but it assumes that the variation of the turbulent eddy viscosity, μ_T, can be neglected in the derivation of the adjoint equations. Therefore, under the CEV assumption, only five equations (2) are used to derive the adjoint, which significantly reduces the size of the dual problem, as quantified in (8). The benefits being easier implementation, faster run time and reduced memory requirements. The matrix $\partial \mathbf{R}/\partial \mathbf{q}$ is reduced by a factor of $7^2/5^2 = 1.96$, and the vector $\partial Y/\partial \mathbf{q}$ and matrix $\partial \mathbf{R}/\partial \mathbf{x}$ are reduced by a factor of $7/5 = 1.4$.

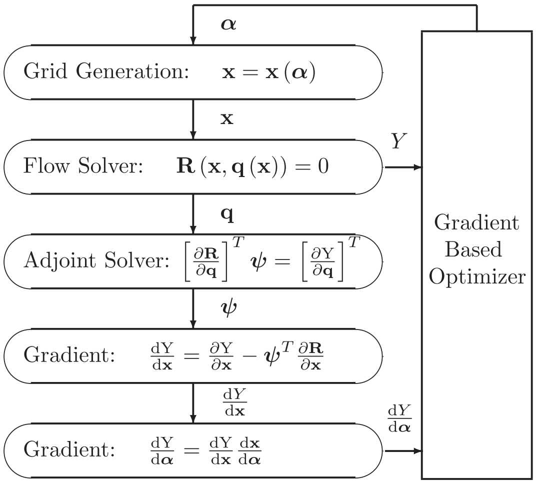

$\boldsymbol{\alpha}$
Grid Generation: $\mathbf{x} = \mathbf{x}(\boldsymbol{\alpha})$
$\mathbf{x}$
Flow Solver: $\mathbf{R}(\mathbf{x}, \mathbf{q}(\mathbf{x})) = 0$
Y
$\mathbf{q}$
Adjoint Solver: $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{q}}\right]^T \boldsymbol{\psi} = \left[\frac{\partial \mathrm{Y}}{\partial \mathbf{q}}\right]^T$
$\boldsymbol{\psi}$
Gradient: $\frac{\mathrm{dY}}{\mathrm{d}\mathbf{x}} = \frac{\partial \mathrm{Y}}{\partial \mathbf{x}} - \boldsymbol{\psi}^T \frac{\partial \mathbf{R}}{\partial \mathbf{x}}$
$\frac{\mathrm{d}Y}{\mathrm{d}\mathbf{x}}$
Gradient: $\frac{\mathrm{dY}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\mathrm{dY}}{\mathrm{d}\mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\boldsymbol{\alpha}}$
$\frac{\mathrm{d}Y}{\mathrm{d}\boldsymbol{\alpha}}$
Gradient Based Optimizer

Fig. 2. Schematic of the adjoint-based optimization algorithm.

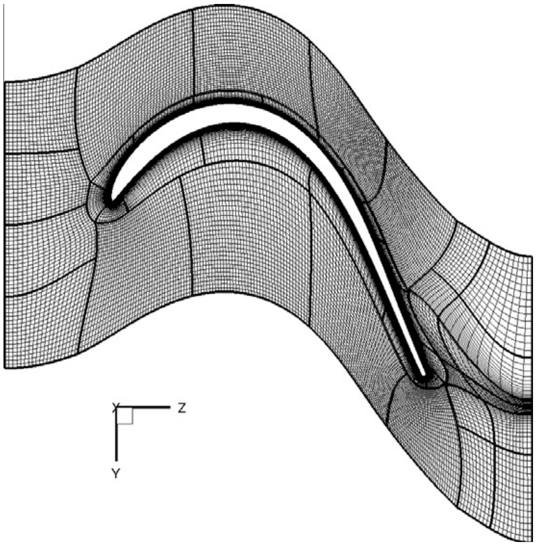


Fig. 3. 2-D vane grid.

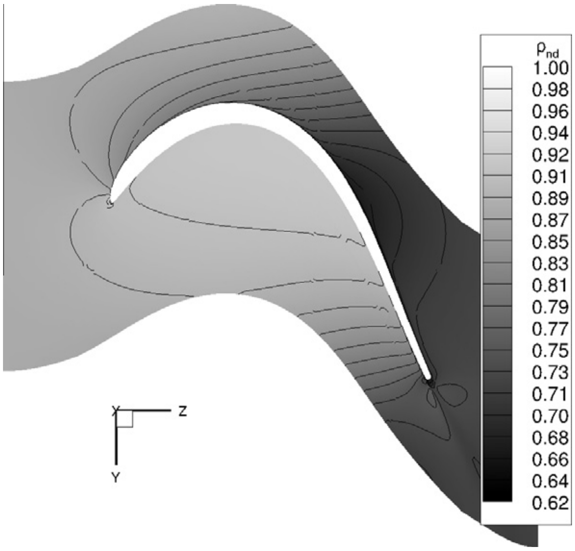


Fig. 4. 2-D vane density distribution.

In the present adjoint solver implementation, a single flag controls whether CEV approximation is to be used. If so, the turbulent equations are neglected in the adjoint and the turbulent eddy viscosity is retrieved from the flow solution and added to the total adjoint viscosity.

3.4. Gradient-based optimization framework

From a design perspective, a turbomachine is geometrically represented by some high-level descriptors, such as stagger, camber angle distribution and thickness distribution. Let $\boldsymbol{\alpha}$ denote the high-level geometric parameters that form the set of design variables. The grid generator receives these parameters as input and it produces a computational mesh in terms of nodes coordinates $\mathbf{x}$.

Given the mesh and the boundary conditions, the flow solver computes the flow solutions $\mathbf{q}$ and, using some post-processing, the cost function values Y can easily be computed. Using the flow solution obtained from running the flow solver, the corresponding cost function gradients with respect to the grid coordinates require an additional solver – the adjoint solver. Once the adjoint solution, $\boldsymbol{\psi}$, is evaluated from (5), the gradients of the functions Y with respect to the grid coordinates are computed by (7), which implies a simple matrix–vector multiplication operation.

Then, it is still necessary to evaluate the gradient of the computational mesh with respect to those high-level geometric parameters, $\mathrm{d}\mathbf{x}/\mathrm{d}\boldsymbol{\alpha}$, according to (6), but because the source code of every tool involved in the grid generation process is not available, it is necessary to use an approximation. In this work, a finite-difference approximation is used to accomplish that, which implies perturbing each design variable individually and re-generating the computational mesh. A sensitivity study was conducted for each high-level geometric parameter to properly select the perturbation step size used in the finite-differences. Since these changes are local to the solid walls, mesh morphing is used to reduce computational cost. The caveat being that one has to guarantee that the mesh topology is maintained during re-meshing. Mesh morphing is also used to modify the grid after an update of the design variables.

Even though this approach makes use of a fast fully automatic grid generator, which is embedded in the optimization framework, the mesh gradients can be computed in parallel in case the number of design variables becomes very large. This is typically possible since computational resources are available at this step: while the flow and adjoint solvers require multi-processor clusters or workstations, the grid generator is run in a single-processor.

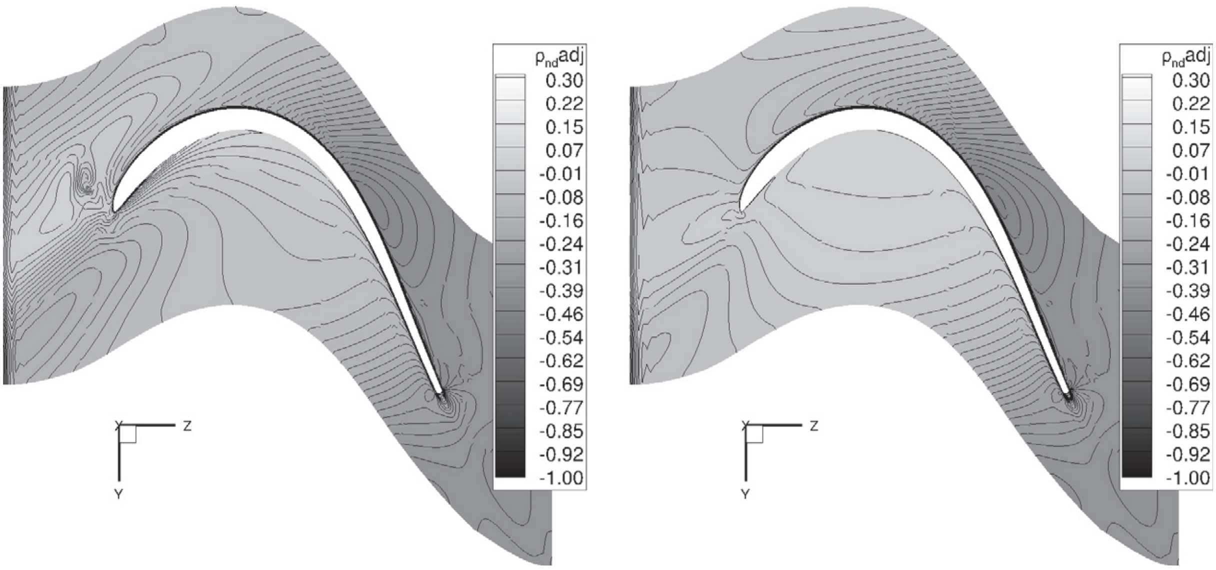
The computed final sensitivity $\mathrm{d}Y/\mathrm{d}\boldsymbol{\alpha}$ is then used by the gradient-based optimizer to find the search direction, along which a step is taken in the design space. The optimizer then loops though the described steps until the optimality criteria are satisfied.

The schematic of such adjoint-based optimization algorithm is illustrated in Fig. 2.

In terms of computational cost, the flow and adjoint solvers are the two main blocks of the process, being the cost of the solution of the adjoint equations approximately the same as that of the solution of the flow governing equations, since both equations are of similar size and complexity.

4. Results

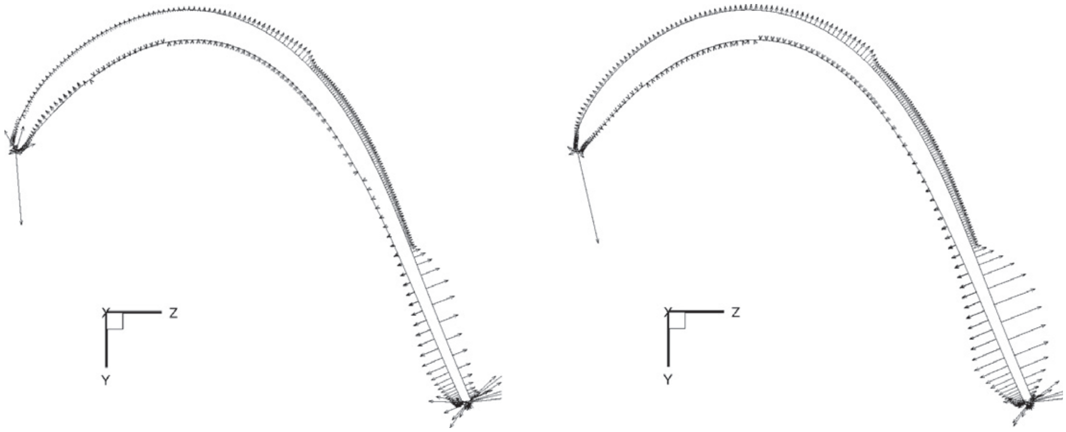
This section includes two testcases, a two-dimensional turbine vane blade and a three-dimensional transonic compressor rotor blade. Each example is comprised of six parts: firstly, the problem is described; secondly, the flow and both adjoint solutions (full



(a) Full RANS adjoint

(b) CEV model adjoint

Fig. 5. 2-D vane adjoint solution of the continuity equation.



(a) Full adjoint

(b) CEV adjoint

Fig. 6. 2-D vane gradient vector plot ($d\eta/d\mathbf{x}$).

RANS and CEV approximation) are evaluated; thirdly, the adjoint-based gradients are computed; then, those gradients are verified against FD derivative approximations; after those steps, an optimization problem is solved using both the full RANS adjoint solver and the CEV approximation; and lastly, a comparison is made in terms of optimal solution found, computational time and memory usage.

4.1. 2-D turbine vane

4.1.1. Description of testcase

The first testcase consists of a two-dimensional turbine vane. This was chosen to validate the discrete adjoint solver developed using the hybrid approach and to perform a preliminary comparative study between the full RANS adjoint and the CEV approximation adjoint solvers.

The geometry is shown in Fig. 3. The mesh has been refined close to the wall to achieve an average unity Y+. The computational domain includes 60 blocks and 25,000 cells and the simulations were run on a workstation using 4 processors.

In this case, the loss coefficient is used as objective function during the optimization,

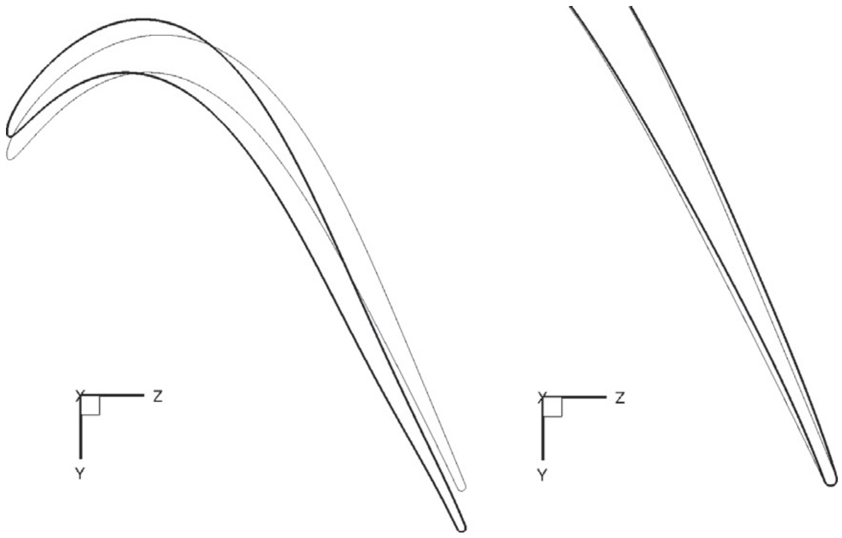
$$\eta = \frac{\left(P^{area}_{ta\,exit} - P^{area}_{ta\,inlet}\right)}{\left(P^{area}_{ta\,inlet} - P^{area}_{s\,inlet}\right)}, \tag{9}$$

where P and T are the area averaged pressure and temperature, and the subscripts ta and s refer to the total absolute and static quantities.

4.1.2. Baseline flow and adjoint solutions

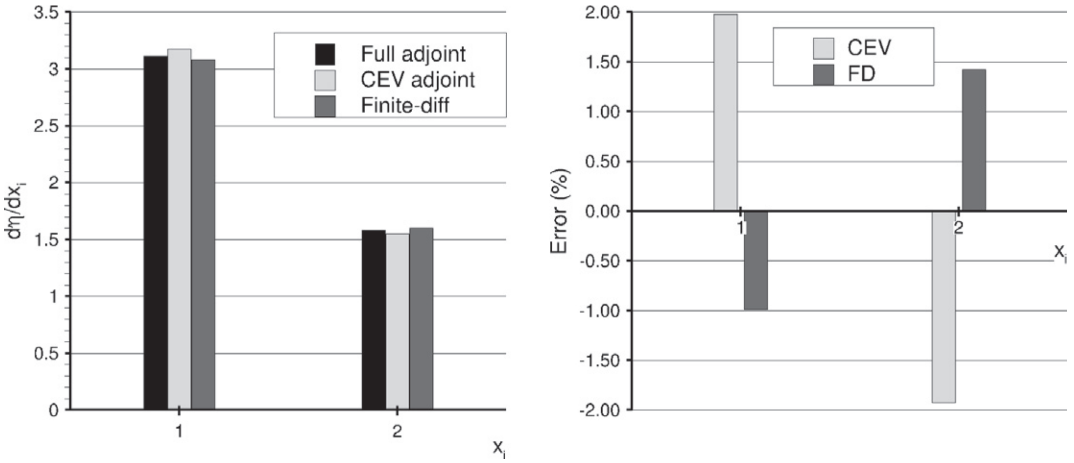
Fig. 4 shows the contours of nondimensional density of the baseline vane blade geometry. The contour plots corresponding to the nondimensional adjoint of the continuity equation are shown in Fig. 5 for the loss coefficient (9), using both the full RANS adjoint and the CEV model adjoint solvers.

As typical for the adjoint solution, both plots show an adjoint flow reversed when compared to the real flow. However, the CEV



(a) Stagger angle (b) Overturning angle

Fig. 7. Changes applied to the vane geometry.



(a) Gradient (b) Error (%)

Fig. 8. 2-D vane comparison of function gradients.

solution misses a great portion of the adjoint field structure, namely at the vane leading edge, where the level of adjoint turbulence is highest.

4.1.3. Adjoint-based gradient solutions

The adjoint-based gradient of loss coefficient with respect to the grid coordinates is evaluated using (7). Using the components corresponding to each coordinate,

$$\frac{d\eta}{d\mathbf{x}} = \frac{d\eta}{dx}\mathbf{e}_x + \frac{d\eta}{dy}\mathbf{e}_y + \frac{d\eta}{dz}\mathbf{e}_z, \tag{10}$$

it is possible to plot the gradient vector at the vane wall nodes as shown in Fig. 6.

Each point indicates the direction of increased loss coefficient and its magnitude is the improvement per unit change in the grid node coordinate. Consequently, a designer can easily infer from Fig. 6 how to tune the blade for deceased loss, since those vectors tell him how the surface geometry should change to accomplish it.

As a result of the missing features in the CEV adjoint solution, the adjoint-based gradients are also distinct between the two adjoint turbulence treatment models. Even though the vector gradients show a qualitative match, the gradients computed using the

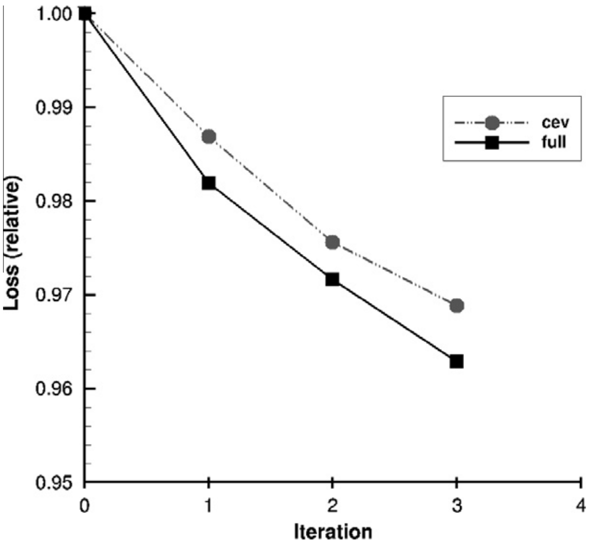


Fig. 9. 2-D vane minimization of loss coefficient.

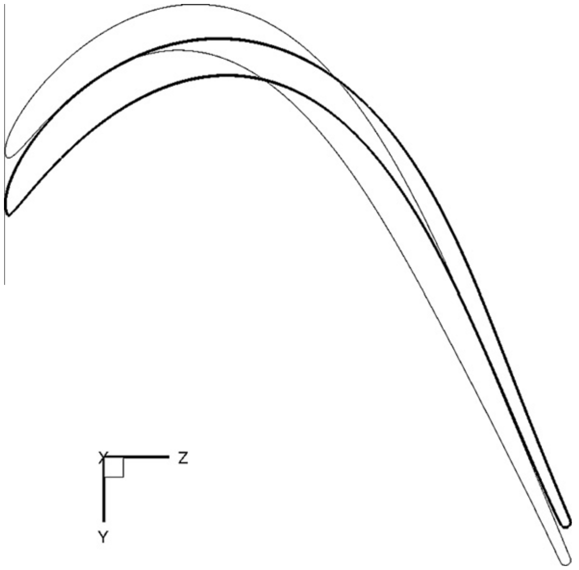


Fig. 10. 2-D vane shape (black: optimized, grey: baseline).

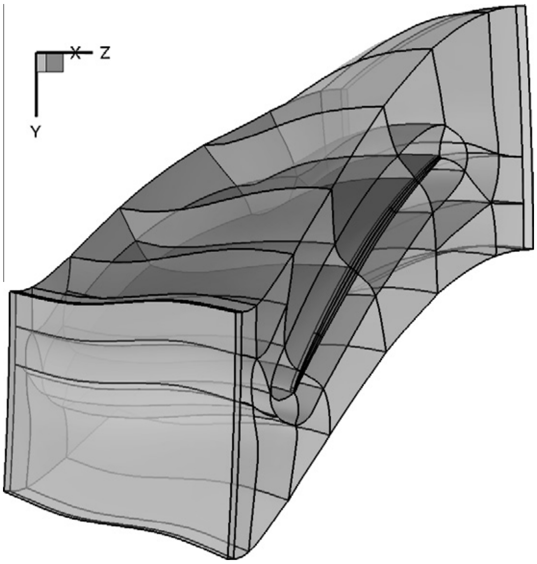


Fig. 12. 3-D rotor multi-block computational mesh.

CEV adjoint are somewhat amplified when compared to the full adjoint gradient values.

4.1.4. Verification of gradients

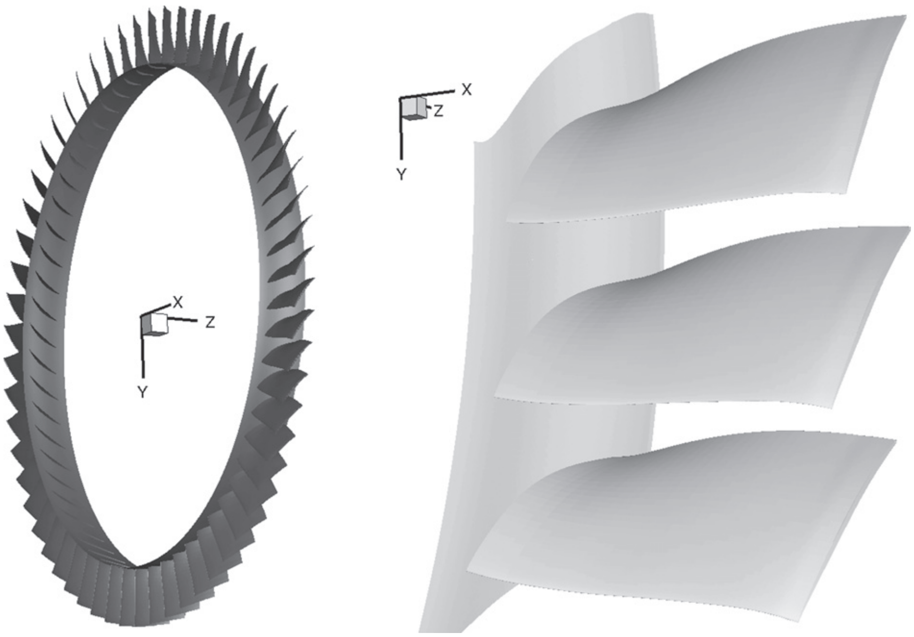
The engineering design variables used in this testcase are the stagger angle and the overturning angle, as illustrated in Fig. 7. The adjoint-based gradients are verified against 2nd-order central-difference derivative approximations.

The comparison between the adjoint-based gradients and the full FD derivative approximation is summarized in Fig. 8 for the different design variables. The errors are computed using the full RANS adjoint-based gradients as reference. There is an excellent agreement between the full RANS adjoint-based gradient and the FD derivative approximation, for both design variables used. As expected, following the results shown in Fig. 6, the gradients obtained using the CEV model adjoint solution do not match perfectly. Nevertheless, the mismatch is of the order of only 2% compared to the full RANS adjoint-based gradients.

4.1.5. Adjoint-based gradient optimization

At this point, the adjoint solver implementation is considered successful and ready to be integrated in a gradient-based optimization framework as depicted in Fig. 2.

A sample optimization application, a unconstrained minimization optimization problem using the loss coefficient as the cost function, is performed. Two design variables are used, as described in Section 4.1.4. Separate runs using the full RANS adjoint solver and the CEV approximation adjoint solver are made.



(a) Full wheel

(b) Blade passage

Fig. 11. 3-D rotor geometry.

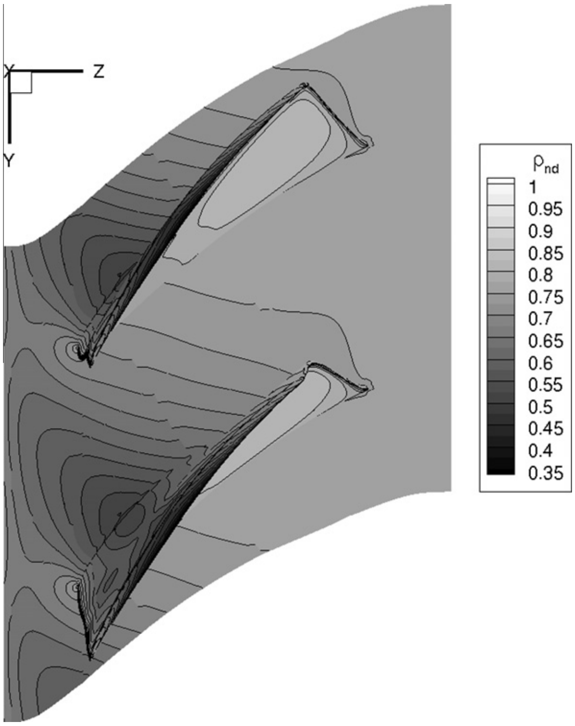


Fig. 13. 3-D blade density distribution.

The relative evolution of the cost function using a gradient-based optimizer, based on the steepest descent method for simplicity, is illustrated in Fig. 9, where the initial loss coefficient value is used as reference. As it can be seen, even though the baseline vane blade corresponded to a tuned geometry, the optimizer is able to improve its performance by 4 points using the full adjoint and 3 points using the CEV adjoint, after three iterations.

More interesting is the fact that, despite the slightly off gradients produced by the CEV adjoint, the optimizer still marches in the right direction, though at a slower convergence rate compared to the full RANS adjoint.

4.1.6. Comparison between full RANS and CEV approximation

There are no significant differences between the optimal shape solutions found by using the full adjoint solver or the CEV adjoint solver. The final vane geometry resulting from the optimization run using the CEV model is shown in Fig. 10.

Given that this was a rather small problem, neither the computational time nor the memory usage are significant for a comparison to be made between the two adjoint turbulence treatment models.

4.2. 3-D compressor rotor blade

4.2.1. Description of testcase

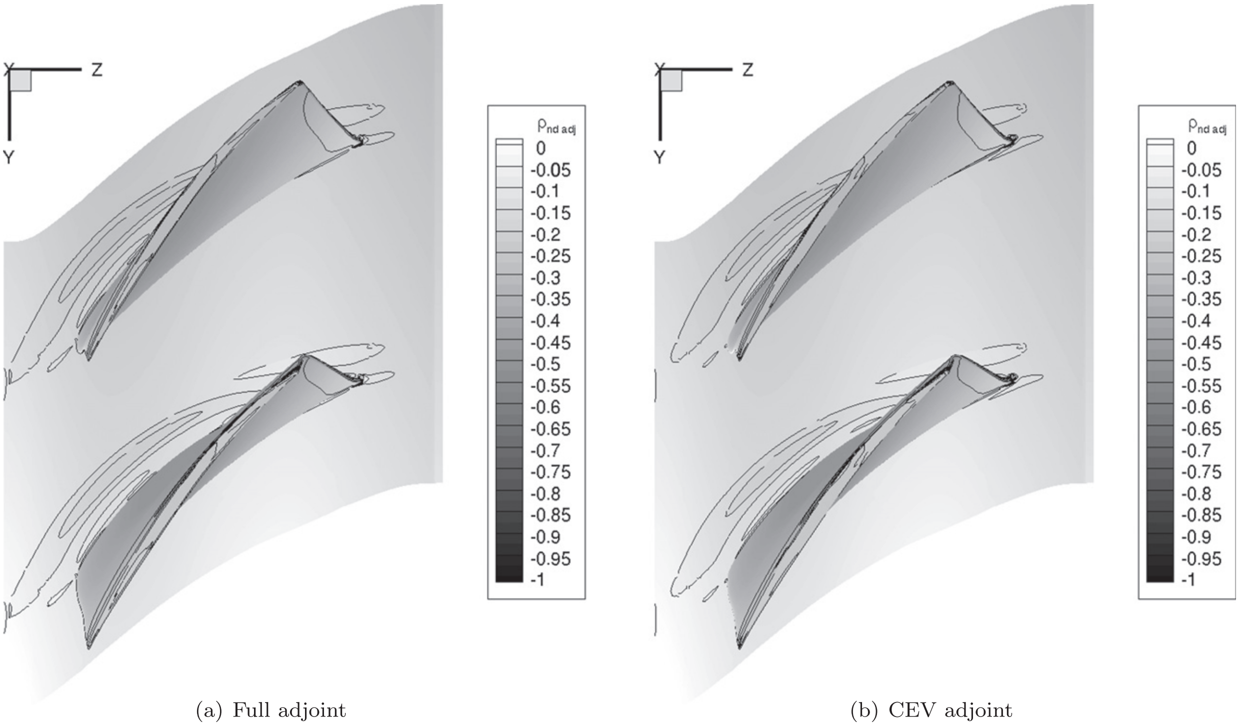
To demonstrate the full capabilities of the adjoint solver developed and to perform the realistic comparative study between the full RANS adjoint and the CEV approximation adjoint solvers, a transonic blade passage of a high-pressure compressor stage is used. The three-dimensional geometry is shown in Fig. 11, where the casing wall has been removed, and the passage passage has been duplicated for visual clarity.

The mesh generated has a OH-grid topology around the blade, reverting to H-grid topology further away from the blade, and a wall refinement leading to an average *Y*+ of 25. A total of 60 blocks are created, as shown in Fig. 12, totaling 1.2 million cells, and the simulations are run on a cluster using 32 processors.

The inlet boundary conditions are absolute tangential velocity fixed and pressure extrapolated from the interior. The exit static pressure is held fixed. All solid walls are considered impermeable with no-slip condition. The remaining faces are either block-to-block interfaces or periodic.

In this case, the isentropic efficiency is used as cost function,

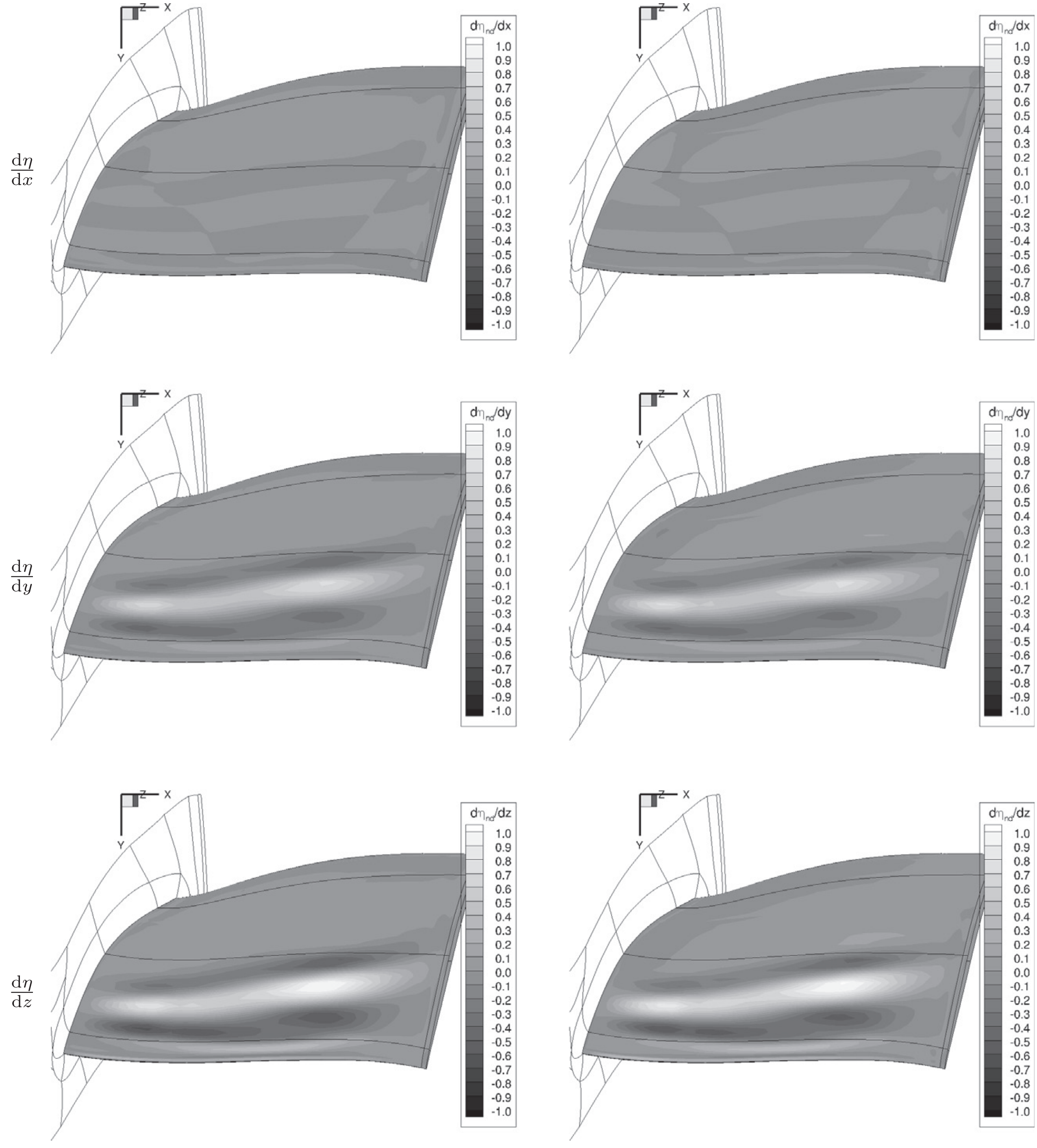
$$\eta = \frac{\left(P^{enth}_{ta\,exit}/P^{enth}_{ta\,inlet}\right)^{(\gamma-1)/\gamma} - 1}{\left(T^{mass}_{ta\,exit}/T^{mass}_{ta\,inlet}\right) - 1}, \tag{11}$$



(a) Full adjoint

(b) CEV adjoint

Fig. 14. 3-D blade adjoint solution of the continuity equation.



(a) Full adjoint (b) CEV adjoint

Fig. 15. 3-D blade gradient of efficiency w.r.t. node coordinates.

where the superscripts *enth*, *mass* and *area* indicate an enthalpy, mass and area averaged quantities at the inlet or exit sections.

4.2.2. Baseline flow and adjoint solutions

The contours of nondimensional density on the hub and blade surface planes corresponding to the baseline blade geometry are shown in Fig. 13. The corresponding adjoint solution of the continuity equation for isentropic efficiency is shown in Fig. 14.

Similarly to Section 4.1.2, the plot shows an adjoint flow some how reverse of the real flow. But in this testcase, the difference of the adjoint solutions obtained using the two adjoint turbulence treatments is not significant. A very slight decrease of the adjoint wake extension towards the leading edge occurs when switching to the CEV approximation model.

4.2.3. Adjoint-based gradient solutions

The contour of the gradient of efficiency with respect to the grid coordinate components evaluated using (7) is illustrated in Fig. 15 for the blade surface nodes.

The large values at the blade leading edge reveal how sensitive the machine performance is relative to this region. In contrast to the two-dimensional testcase in Section 4.1.3, the comparison

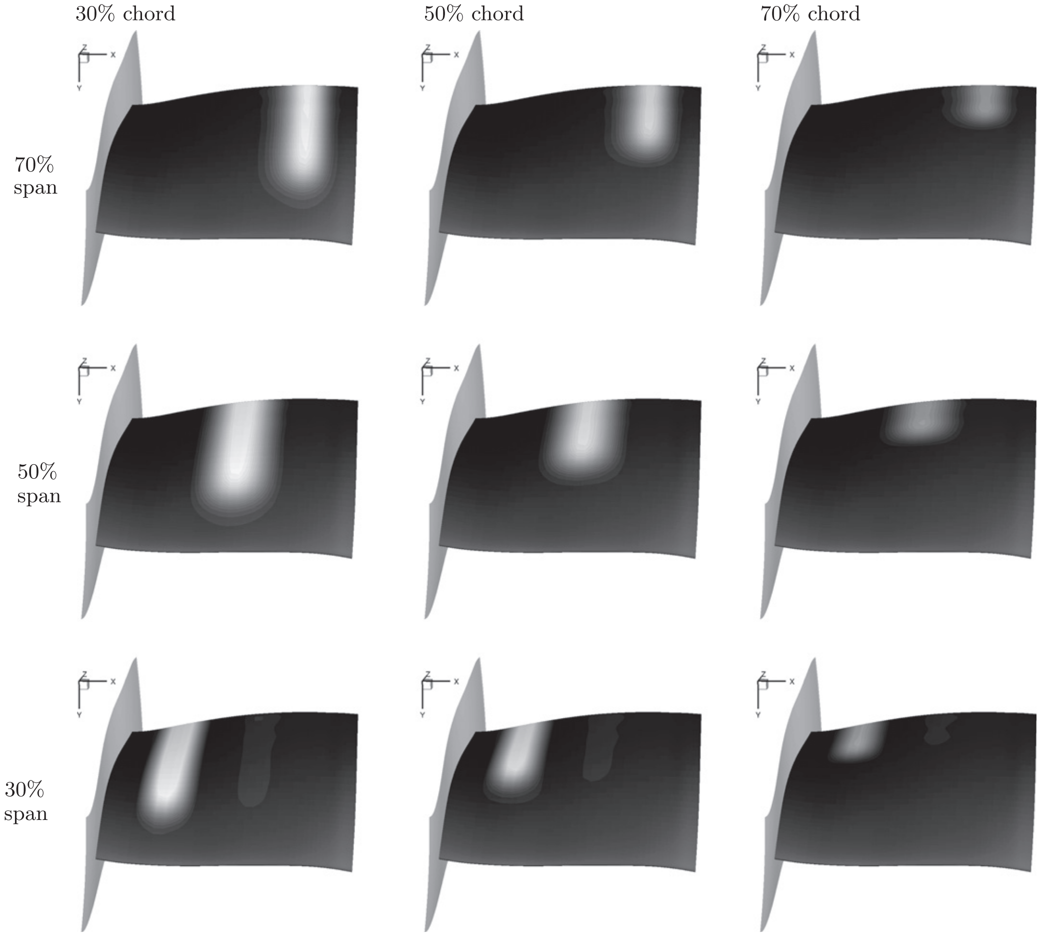
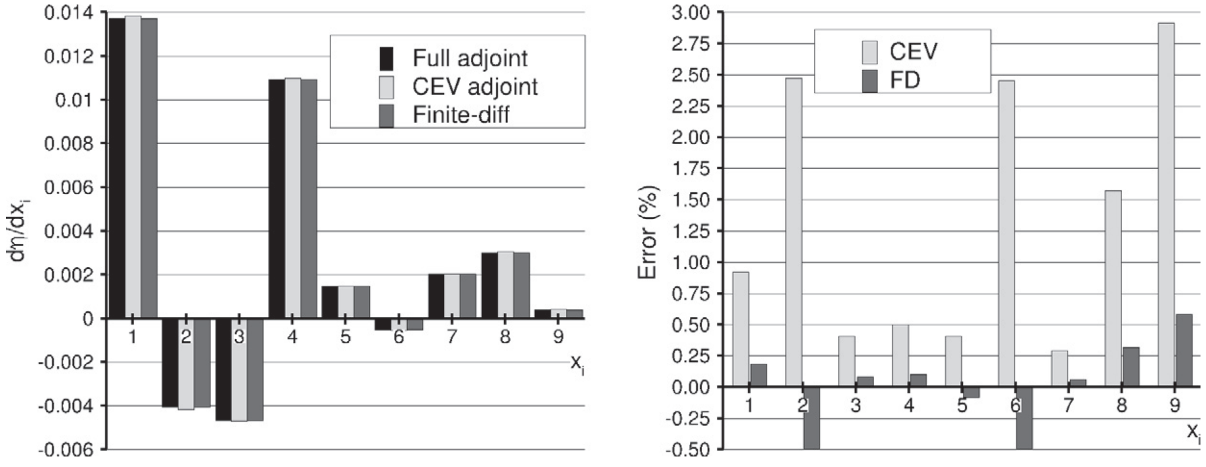


Fig. 16. 3-D blade: Hicks–Henne bumps applied to the blade camber line angle.



(a) Gradient

(b) Error (%)

Fig. 17. Comparison of function gradients.

between the full adjoint and the CEV approximation reveals little difference between them. This is in line with the negligible difference in the adjoint solutions seen in Fig. 14.

4.2.4. Verification of gradients

Similarly to the two-dimensional testcase in Section 4.1.4, the adjoint-based gradients are verified against FD derivative approximations. However, to minimize the necessary number of evaluations of the flow solver, the 1st-order forward-difference formula for the first derivative is used.

Hicks–Henne bump functions [20] are used to test the integration with the grid generation module and compute higher-level gradients of the form $dY/d\boldsymbol{\alpha}$. Fig. 16 shows the individual perturbation produced by applying nine bumps on the blade camber line

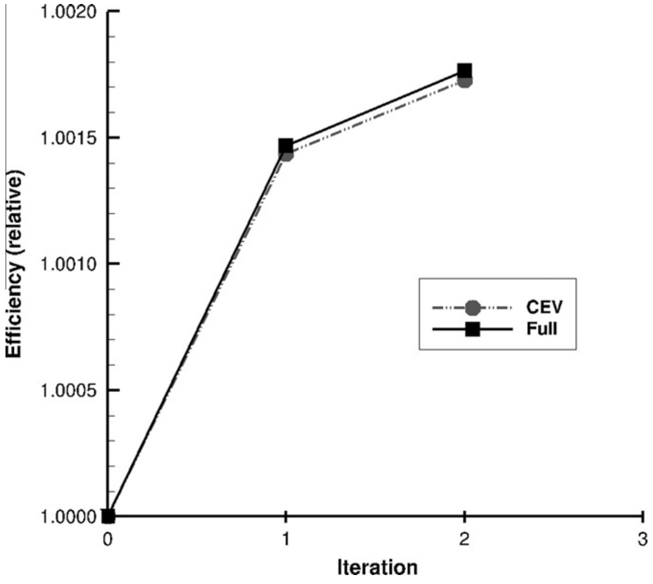


Fig. 18. 3-D blade maximization of efficiency.

Table 1
3-D blade comparison of optima design variables.

Bump #	Full adjoint	CEV adjoint
1	2.3700	2.3167
2	0.4506	0.4521
3	0.5683	0.5666
4	1.0734	1.0753
5	1.2685	1.2522
6	1.4305	1.4301
7	0.7134	0.6860
8	0.7064	0.6963
9	0.7512	0.7343

Table 2
3-D blade comparison of runtime and memory use.

	Flow solver (reference)	Full adjoint	CEV adjoint	CEV-full difference (%)
Runtime	1.00	1.66	1.04	−37.7
Memory	1.00	22.83	14.38	−37.0

angle, evenly distributed along the blade chord (30%, 50%, 70%) and span (30%, 50%, 70%).

The gradient of isentropic efficiency, η, with respect to the bump amplitude, a, are computed using the adjoint solver. Fig. 17 summarizes these results, together with the comparison using full FD derivative approximation. There is an excellent agreement between the full RANS adjoint-based gradient and the FD derivative approximation, for the different bump locations. On the other hand, adjoint-based gradients using the CEV model differs from the full RANS adjoint-based gradients by as much as 3%.

4.2.5. Adjoint-based gradient optimization

An unconstrained maximization optimization problem is run, using isentropic efficiency as the cost function, and the previously described nine Hicks–Henne bumps on the camberline angle used as design variables, as shown in Fig. 16.

Fig. 18 shows the optimization history using the same optimizer as in Section 4.1.5, having set the initial efficiency value as reference. Again, it was possible to improve the blade performance, even though it corresponded to a highly hand-tuned geometry, with a net gain of about 0.16 pts after two iterations.

In contrast to the 2-D vane testcase in Section 4.1.5, the final maximum blade efficiency obtained by the gradient-based optimizer, using either the full adjoint solver or the CEV adjoint solver, is virtually the same.

4.2.6. Comparison between full RANS and CEV approximation

Comparing the final design variable values at the optimal solution, condensed in Table 1 in nondimensional form, it can be seen that the optimum blade shape is nearly identical whether using the full RANS adjoint or the CEV adjoint solvers. A value greater than one means an increase of the camber line angle, whereas a smaller than one means a reduction. It should be noted that the design variables are not bounded, that is to say, there are no geometric constraints set in the optimization problem.

In terms of computational time and memory usage, the CEV adjoint solver proves to be both faster and less memory intensive compared to the full adjoint solver. For this particular problem setup, the CEV adjoint solver is nearly 38% faster than its full RANS adjoint counterpart, while requiring less than 37% memory, as summarized in Table 2.

It should be noted that the full adjoint matrix $\partial\mathbf{R}/\partial\mathbf{q}$ is stored during the GMRES solve, thus the significantly higher memory needs of the adjoint solver when compared to the flow solver. Had the adjoint linear system been solved using a matrix-free method, the adjoint memory required would have been comparable to the flow solver, at a penalty of higher runtime.

5. Conclusions

The two most relevant approaches for the development of adjoint solvers for the RANS equations were analyzed. As a direct consequence of the assumption of frozen turbulence in the adjoint, it was shown that some features in the adjoint solution were lost when using the CEV model. Therefore, the CEV model leads to an error in the adjoint-based gradients of the order of a few percent (typically 2–3% for the testcases shown), when compared to the exact full RANS adjoint-based gradients. Despite the slightly off gradients produced by the CEV adjoint, the optimizer still marches in the right direction, though at a slower convergence rate compared to the full RANS adjoint. As such, the impact on the optimal solution found by an optimization design framework is almost negligible. Moreover, the CEV adjoint solver proved to be up to 38% faster and require less 37% memory compared to the full RANS adjoint solver, for the testcases presented.

The CEV approximation in RANS adjoint solvers seems to be a valid approach to be used in optimization engineering problems using high-fidelity CFD models. This is extremely relevant since industry has the need to handle larger and more complex design problems, where the use of RANS equations to model the flow physics are absolutely necessary. This way, the code developers can significantly reduce the implementation effort and the designers can effectively reduce the hardware needs.

Acknowledgements

The authors would like to thank Daniel Wilkin, from GE Aviation, for his support and feedback. Moreover, the authors are thankful to General Electric for giving permission to publish this paper.

References

[1] Nocedal J, Wright SJ. Numerical optimization. Springer; 1999.
[2] Griewank A. Evaluating derivatives. Philadelphia: SIAM; 2000.
[3] Pironneau O. On optimum design in fluid mechanics. J Fluid Mech 1974;64:97–110.

[4] Jameson A. Aerodynamic design via control theory. J Sci Comput 1988;3(3):233–60.

[5] Jameson A, Pierce NA, Martinelli L. Optimum aerodynamic design using the Navier–Stokes equations. Theoretical and computational fluid dynamics, vol. 10. Springer-Verlag GmbH; 1998. p. 213–37.

[6] Reuther JJ, Alonso JJ, Jameson A, Rimlinger MJ, Saunders D. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. J Aircraft 1999;36(1):51–60.

[7] Martins JRRA, Alonso JJ, Reuther JJ. High-fidelity aerostructural design optimization of a supersonic business jet. J Aircraft 2004;41(3):523–30.

[8] Marta AC, Alonso JJ. Toward optimally seeded airflow on hypersonic vehicles using control theory. Comput Fluids 2010;39(9):1562–74.

[9] Luo J, Xiong J, Liu F, McBean I. Three-dimensional aerodynamic design optimization of a turbine blade by using an adjoint method. J Turbomach 2011;133(1):011026.

[10] He L, Wang DX. Concurrent blade aerodynamic–aero-elastic design optimization using adjoint method. J Turbomach 2011;133(1):011021.

[11] Wang DX, He L. Adjoint aerodynamic design optimization for blades in multistage turbomachines—part I: methodology and verification. J Turbomach 2010;132(2):021011.

[12] Corral R, Gisbert F. Profiled end wall design using an adjoint Navier–Stokes solver. J Turbomach 2008;130(2):021011.

[13] Marta AC, Mader CA, Martins JRRA, van der Weide E, Alonso JJ. A methodology for the development of discrete adjoint solvers using automatic differentiation tools. Int J Computat Fluid Dynam 2007;21(9–10):307–27.

[14] Wilcox DC. Reassessment of the scale-determining equation for advanced turbulence models. AIAA J 1988;26(11):1299–310.

[15] Giles MB, Pierce NA. An introduction to the adjoint approach to design. In: Flow. Turbulence and combustion, vol. 65. Kluwer Academic Publishers; 2000. p. 393–415.

[16] Cusdin P, Müller J-D. On the performance of discrete adjoint CFD codes using automatic differentiation. Int J Numer Methods Fluids 2005;47(6–7):939–45.

[17] Hascoët L, Pascual V. Extension of TAPENADE towards Fortran 95. In: Bücker HM, Corliss G, Hovland P, Naumann U, Norris B, editors. Automatic differentiation: applications, theory, and implementations. Lecture notes in computational science and engineering, vol. 50. Springer; 2006. p. 171–9.

[18] Balay S, Buschelman K, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, et al. PETSc users manual, Tech rep ANL-95/11. Revision 2.3.0. Argonne National Laboratory; 2004.

[19] Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7(3):856–69.

[20] Hicks RM, Henne PA. Wing design by numerical optimization. AIAA J 1978;15(7):407–12.