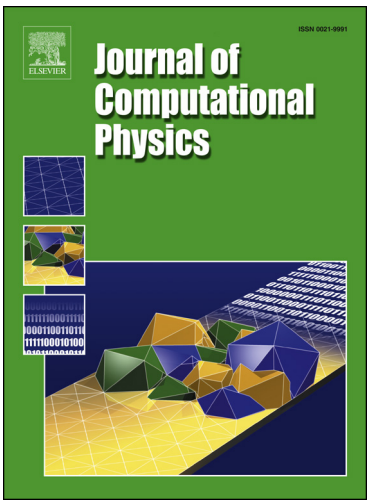
Journal Pre-proof



Nonintrusive Continuum Sensitivity Analysis for Fluid Applications

Mandar D. Kulkarni, Robert A. Canfield, Mayuresh J. Patil

PII: S0021-9991(19)30771-5
DOI: https://doi.org/10.1016/j.jcp.2019.109066
Reference: YJCPH 109066

To appear in: *Journal of Computational Physics*

Received date: 10 October 2018
Revised date: 28 July 2019
Accepted date: 19 October 2019

Please cite this article as: M.D. Kulkarni et al., Nonintrusive Continuum Sensitivity Analysis for Fluid Applications, *J. Comput. Phys.* (2019), 109066, doi: https://doi.org/10.1016/j.jcp.2019.109066.

Highlights

- Efficient and accurate flow sensitivity analysis for optimizing shape of a system.
- Mesh sensitivity not needed to calculate flow sensitivities.
- Nonintusive formulation allows sensitivity analysis with black-box codes.
- Formulation works with unstructured grids and finite volume discretization.

Nonintrusive Continuum Sensitivity Analysis for Fluid Applications

Mandar D. Kulkarni[ψ], Robert A. Canfield[ϕ], Mayuresh J. Patil[ϕ]
[ψ] *Department of Aerospace Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114*
[ϕ] *Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060*

Abstract

Continuum Sensitivity Analysis (CSA) provides an analytic method of computing derivatives for structures, fluids and fluid-structure-interaction problems with respect to shape or value parameters. CSA does not require mesh sensitivity to calculate local shape sensitivity, thereby contributing to its computational efficiency. CSA involves solving linear sensitivity equations with the corresponding sensitivity boundary conditions. Spatial gradients of flow variables are required to construct the sensitivity boundary conditions, the accuracy of which significantly affects the flow shape derivatives. A method called Spatial Gradient Reconstruction (SGR) is used to accurately compute the flow spatial gradients that are further used to set the sensitivity boundary conditions. Here we present a nonintrusive CSA formulation for calculating the material derivatives of flow variables with respect to shape design parameters. Nonintrusive CSA implies no modification to the black-box program that was used for the CFD flow analysis. Moreover, nonintrusive CSA is flow solver agnostic in that knowledge of the solver's discretization is not needed. An example of two-dimensional flow over a NACA0012 airfoil demonstrates the application of nonintrusive CSA to steady flow problems involving Euler (compressible inviscid) equations over unstructured grids with finite volume spatial discretization. Accuracy was studied by investigating convergence for a family of six high-quality grids, ranging from four thousand to four million cells. Factors such as the accuracy of the flow spatial gradients, flow sensitivity boundary conditions, weak implementation of the boundary conditions in the finite volume framework, and the use of an approximate flux Jacobian matrix, all of which affect the accuracy of the material derivatives, are discussed. A novelty of this work is that the sensitivity analysis is done nonintrusively using FLUENT and SU2 software to establish the use of black-box codes for obtaining flow derivatives using the CSA approach.

Keywords: continuum sensitivity, shape derivatives, sensitivity analysis, finite volume, unstructured mesh, nonintrusive

1. Introduction

Sensitivity analysis plays an important role in gradient-based optimization techniques. Convergence of a shape optimization problem depends on the accuracy of derivatives of the performance functions with respect to shape design variables. Sensitivity analysis methods can be broadly categorized as numeric methods (finite difference, complex step), analytic methods (discrete analytic, continuum) or automatic differentiation methods. In general, analytic methods are favored over numeric methods because of their higher accuracy and lower computational cost compared to the finite difference method. Furthermore, analytic methods have the following advantages: (a) there is no need of convergence study for choosing the correct step size (required for the finite difference method), and (b) there is no requirement of the analysis code to handle complex number operations (required for the complex step method). Among the analytic methods, the discrete analytic method involves discretizing the governing equations followed by differentiation. Hence, for shape design variables it suffers from the disadvantage of calculating mesh sensitivity, that is, derivatives of mesh coordinates with respect to shape parameters. Additionally, intimate knowledge of the analysis procedure is required for implementation of the discrete analytic method. Automatic differentiation (AD) requires a line-by-line differentiation of the actual source code, which is automated using a set of scripts that perform operator-overloading and other functions. Hence, the AD method also requires access to the source code and cannot be implemented for black-box software. Continuum Sensitivity Analysis (CSA) offers an accurate and efficient alternative. Furthermore, CSA can be implemented nonintrusively with black-box software, such as FLUENT. Thus, the motivation for using CSA for shape sensitivity is twofold: (a) derivatives are analytic (accurate and more efficient than finite difference) and (b) mesh sensitivity calculation is avoided (a drawback of the discrete analytic shape sensitivity approach).

CSA can be derived in two forms (Liu and Canfield, 2016), the local form, also known as

the boundary velocity sensitivity equation method (Choi and Kim, 2005) or Eulerian sensitivity equation method (Pelletier et al., 2008), and the total form, also known as the domain velocity sensitivity equation method (Choi and Kim, 2005) or Lagrangian sensitivity equation method (Charlot et al., 2012). Although the total form is recommended for best accuracy of shape derivatives, it comes at the expense of having to calculate mesh sensitivity at all points in the domain and deriving spatial derivatives (or finite differences) of cell equations to compute sensitivity loads throughout the domain. In contrast, the local form requires geometric (mesh) sensitivity only on the boundaries that change shape. However, along with the boundary geometric sensitivity, the local form also requires accurate computation of spatial gradients on the boundaries that change shape. The local form was known to give poor flow shape derivative results, compared to the total form, due to inaccurate evaluation of these boundary spatial gradients. Cross and Canfield (2014) demonstrated that accurate spatial gradients can be evaluated for structural applications using the Spatial Gradient Reconstruction (SGR) technique, inspired by Duvigneau and Pelletier (2006), thus resolving one of the challenges that was preventing the use of local form CSA for structural shape derivatives. In the current work, the same SGR technique of Cross and Canfield (2014) has been used to evaluate spatial gradients of flow variables on shape-variable-dependent boundaries. Furthermore, it is shown here that when SGR reinforces the flow tangency boundary condition, the results obtained are more accurate than if SGR is done without reinforcing the boundary condition.

Previous work in the area of Computational Fluid Dynamics (CFD) flow sensitivity that used local CSA was done with the finite difference discretizations (Borggaard and Burns, 1997, 1994) or the finite element (Duvigneau and Pelletier, 2006). Focus of the current work is to apply CSA with SGR to calculate flow derivatives of Euler equations with unstructured finite volume discretization, for which a very limited number of studies are available. Gobal et al. (2015) used CSA with an immersed boundary finite volume scheme to solve for the derivatives of a temperature field. The temperature was governed by the linear thermal

diffusion equation. They used the finite difference method and the Heaviside functions to compute the spatial gradients that appear in the sensitivity equations. However, since the mesh for their problems did not depend on the design variables, the advantage of using their approach for computing flow derivatives on shape dependent boundaries is unclear. Challenges arise when finite volume discretization is used, because boundary conditions are typically imposed with a weak or integral approach (Hirsch, 1990; Palacios et al., 2013). Villa et al. (2012) mention that the weak approach is used by finite-volume CFD solvers, because it imposes a suitable boundary flux, rather than a boundary state as in the strong approach. Villa (2009) studied the drawbacks of a weak imposition of the no-penetration boundary condition on 1-D hyperbolic equation solutions using the finite volume approach and stated that the difference between the computed physical values and the actual values of velocity could be significant. This can be particularly detrimental for solving the sensitivity equations, because it adds a source of error in the imposition of sensitivity boundary conditions. We discuss this issue in detail and highlight areas where improvement in using SGR can be made.

Finally, another contribution of the current work is the nonintrusive, or black-box, implementation of CSA. The discrete analytic method, the complex step method and the automatic differentiation method all require modification to the source code for calculating design derivatives. The operator-overloading approach to performing automatic differentiation is the most convenient, but its implementation is typically 10-35 times slower than the original algorithm (Gay, 2005). Newer advancements on operator-overloading have made the automatic differentiation process faster, but the computational cost is still typically four times or more higher, depending on the application, of the original algorithm (Hogan, 2014; Albring et al., 2016). Although the finite difference method can be used with black-box codes, it is not preferred because of dependence on the finite difference step-size. On the other hand, there can be considerable savings in programming time and computational effort with CSA, because shape derivatives can be computed using the same code that was used for

flow analysis. Another advantage of the present nonintrusive approach may be realized in collaborative design processes, such as design of aerospace systems, wherein multiple design groups and commercial black-box software are involved. In such an environment, CSA can be used across different disciplines such as flow analysis and structural analysis, without the need to have access to the software codes, but by using pre- and post-processing techniques.

The work that most closely resembles a nonintrusive formulation of local CSA for finite volume discretization is by Godfrey and Cliff (2001). Their approach was to obtain the flow solution from a CFD solver and use it to solve the sensitivity equations separately by constructing exact Jacobians. This idea of solving for the shape derivatives separately, that is external to the flow solver, was a key inspiration of the current work. However, the current work differs from the work by Godfrey and Cliff (2001) in several ways. First, Godfrey and Cliff (2001) use the finite difference method to calculate the flow spatial gradients that are used in the sensitivity boundary conditions. While this may be possible for the case of a structured mesh, which they used, the finite difference method cannot be used for unstructured meshes (Becker and Ashcroft, 2014; Duvigneau and Pelletier, 2006). Second, Godfrey and Cliff (2001) focus on understanding the effect of inconsistent turbulence modeling for flow and sensitivity equations. Specifically, they studied the interdependence of the flow and sensitivity solutions, when the sensitivity equation was derived from a turbulence model different from the one that provides the discrete flow data. They report that their sensitivity analysis method provides best results when the turbulence modeling is consistent in the flow solver and sensitivity solver, with the exception of the case when the Baldwin-Lomax flow solution and eddy viscosity are incorporated into the Spalart-Allmaras sensitivity formulation. The focus of this work is on inviscid compressible flows which may be more relevant for high speed flows such as flow over a transport aircraft during cruise. Thirdly, Godfrey and Cliff (2001) report results for a single mesh and compared the CSA results without commenting on the rate of convergence of the computed shape derivatives. In another closely related work, Borggaard and Burns (1994, 1997) discuss how the CSEs can be solved by minimum

modifications to a flow analysis code and state that the same grid and computational scheme can be used to approximate the sensitivities, all in context of finite difference based CFD solvers. However, the issues related to implementing sensitivity boundary conditions in the context of a finite volume based CFD solver are not addressed in their work.

In the current work, we demonstrate a nonintrusive implementation (without modifying the "black-box" analysis source code) of CSA for analyzing fluid systems with a focus on the use of commonly used CFD codes that use finite volume discretization for solving the flow variables and their shape derivatives nonintrusively. In articular, we are interested in finding out how the use of a Jacobian matrix that is inconsistent with the Jacobian matrix used for the flow solution, affects the sensitivity results. Owing to the large size of the Jacobian matrix, the capability of exporting or storing the Jacobian matrix is not available with many flow solvers. For our nonintrusive approach, the Jacobian has to be either exported from the CFD solver (if this capability is available) or constructed outside of the solver. The exact Jacobian for Euler equations is known (Hirsch, 1990) and hence it can be constructed outside of the flow solver; however, it may not be consistent with the Jacobian matrix used for the flow solution. We studied this aspect of the nonintrusive CSA approach. Once the Jacobian matrix was obtained, the linear sensitivity equations were solved using LU factorization with partial pivoting. For larger models, an iterative linear solver could be used.

Details of the proposed nonintrusive CSA approach are given in Section 2. In previous work, CSA was applied to three types of steady flows: (a) potential flow (Liu and Canfield, 2013a) (b) 2-D incompressible viscous flow, which was illustrated with a manufactured solution for a lid-driven cavity example (Kulkarni et al., 2014a), and (c) 1-D compressible inviscid flow, which was illustrated with an example of flow through a convergent-divergent nozzle (Kulkarni et al., 2015). In this paper, an example of a 2-D Euler flow over NACA0012 airfoil is described in Sections 3 and 4. The details about the flow analysis for these examples are given in Table 1

Table 1: Details of flow examples

Flow	Example	Equations	Time discretization	Space discretization	Software	Code verification
2-D Incompressible viscous	Flow in a lid-driven cavity	Navier-Stokes	Explicit	Finite difference	In-house (Matlab)	Grid convergence (MMS)
1-D Compressible inviscid	Flow through convergent-divergent nozzle	Euler	Euler implicit	Finite volume	In-house (Matlab)	Exact solution available
2-D Compressible inviscid	Flow over NACA0012 airfoil	Euler	Euler implicit	Finite volume	SU^2 (C++)	Grid convergence

2. Local Continuum Shape Sensitivity Formulation

2.1. Governing Equations

The partial differential equations governing fluid flow can be written compactly as

$$\boldsymbol{R}\left(\boldsymbol{u}, t; b\right) = \boldsymbol{\mathcal{A}}\left(\boldsymbol{u}, L\left(\boldsymbol{u}\right)\right) - \boldsymbol{f}\left(\boldsymbol{x}, t; b\right) = 0 \quad \text{on } \Omega, \tag{1}$$

with the corresponding boundary conditions (BCs)

$$\boldsymbol{\mathcal{B}}\left(\boldsymbol{u}, L\left(\boldsymbol{u}\right)\right) = \boldsymbol{g}\left(\boldsymbol{x}, t; b\right) \quad \text{on } \Gamma\left(b\right), \tag{2}$$

where the vector of dependent (state) variables $\boldsymbol{u}(\boldsymbol{x}, t; b)$ are functions of the spatial and temporal independent coordinates, $\boldsymbol{x}$ and t, respectively, and depend implicitly on the shape design variable b. The domain Ω and boundary Γ in Cartesian space are shown in Figure 1.

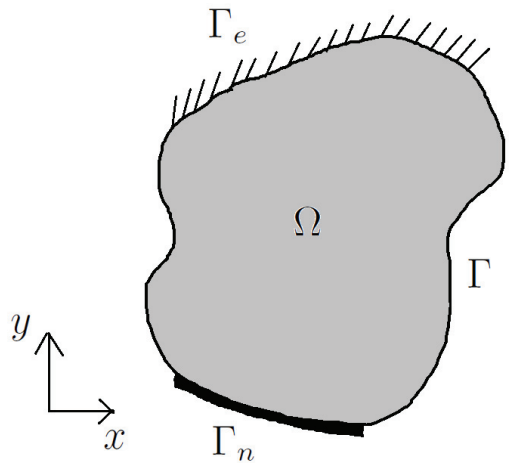


Figure 1: Domain, Ω, with boundary Γ. Γ_e represents boundary for Dirichlet (essential or geometric) boundary conditions and Γ_n represents boundary for Neumann (nonessential or natural) boundary conditions.

Since the design variable b changes the shape of the domain, the boundary can also be represented as $\Gamma\left(b\right)$. The linear differential operator L has terms such as $\left\{\frac{\partial}{\partial t}, \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \ldots\right\}$ that appear in the governing equations or boundary conditions. $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ are algebraic or

integral operators acting on $\boldsymbol{u}$ and $L(\boldsymbol{u})$. In general, the differential operators in the equations (1) and (2) can be nonlinear. The distributed body force applied to the system is given by $\boldsymbol{f}$ in (1). The general BCs in (2) can be either Dirichlet (essential or geometric) such as a prescribed value $\boldsymbol{\mathcal{B}}_e(\boldsymbol{u}) \equiv \boldsymbol{u}|_{\Gamma_e} = \boldsymbol{g}_e$ on the boundary Γ_e, or may involve a differential operator for Neumann (nonessential or natural) BCs such that $\boldsymbol{\mathcal{B}}_n(L(\boldsymbol{u})) = \boldsymbol{g}_n$ on the boundary Γ_n. Since the current examples involve steady-state flow and sensitivity analysis, the time term t in these equations represents pseudo-time for time marching iterative solutions.

2.2. Differentiation of Continuous Equations

The focus of the current work is to obtain the shape design derivatives of flow variables at points in the domain with respect to a limited number of design variables. Hence, we use the direct formulation of CSA as explained next. However, an adjoint formulation of CSA (Kulkarni et al., 2016) is also available, which allows one to calculate derivatives of a limited number of performance measures with respect to many design variables using the same CSA boundary conditions presented here. Consider the problem of obtaining the derivative of the steady-state response $\boldsymbol{u}(\boldsymbol{x}; b)$ with respect to design parameter b at all points in the domain. Following the notation of (Liu and Canfield, 2016), the dependent (state) variables $\boldsymbol{u}(\boldsymbol{x}; b)$ are functions of spatial independent coordinates $\boldsymbol{x} = \{x, y\}$ and depend implicitly on the shape variable b. Since steady response derivatives are of interest in the presented examples, the pseudo-time variable t is suppressed in the further discussion. The boundary velocity (local) formulation of CSA produces CSEs that are posed in terms of the local derivatives of the response, $\boldsymbol{u'} = \partial\boldsymbol{u}/\partial b$. Hence, the CSA solution yields the local derivatives. The total or material derivatives $\dot{\boldsymbol{u}} = D\boldsymbol{u}/Db$ are then obtained by adding the convective term to the local derivatives.

$$\frac{D\boldsymbol{u}}{Db} = \frac{\partial \boldsymbol{u}}{\partial b} + \frac{\partial \boldsymbol{u}}{\partial x}\frac{\partial x}{\partial b} + \frac{\partial \boldsymbol{u}}{\partial y}\frac{\partial y}{\partial b} \quad \Longleftrightarrow \quad \dot{\boldsymbol{u}} = \boldsymbol{u'} + \boldsymbol{\nabla}_{\boldsymbol{x}}(\boldsymbol{u}) \cdot \boldsymbol{\mathcal{V}} \tag{3}$$

The convective term consists of the spatial gradients of the response $\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u} = \partial\boldsymbol{u}/\partial\boldsymbol{x}$, and the geometric sensitivity or design velocity $\boldsymbol{\mathcal{V}}(\boldsymbol{x}) = \partial\boldsymbol{x}/\partial b$, which depends on the parameterization of the computational domain. For value design parameters, the convective term is zero, since the design velocity is zero, and so the material derivative is the same as the local derivative. However, for shape design variables, the design velocity is not zero, and hence there is a need to calculate the convective term wherever necessary.

CSA is based on the philosophy of "differentiate and then discretize" and involves differentiating equations (1) and (2) with respect to b, followed by discretization and solution of the resulting discretized system. Based on the type of differentiation, CSA is categorized as either local form CSA or total form CSA (Liu and Canfield, 2016). The local form CSA involves partial differentiation of equation (1), while the total form CSA involves total differentiation. Due to the advantages of the local-form CSA (Cross and Canfield, 2014) and because total-form CSA may be equivalent to discrete sensitivity (Liu and Canfield, 2013b), the current work focuses on the local-form CSA. The CSEs are obtained by partial differentiation of equation (1) as

$$\frac{\partial \boldsymbol{R}}{\partial b} = \frac{\partial \boldsymbol{\mathcal{A}}(\boldsymbol{u}, L(\boldsymbol{u}))}{\partial \boldsymbol{u}}\boldsymbol{u'} + \frac{\partial \boldsymbol{\mathcal{A}}(\boldsymbol{u}, L(\boldsymbol{u}))}{\partial L}L(\boldsymbol{u'}) - \frac{\partial \boldsymbol{f}(\boldsymbol{x}, t; b)}{\partial b} = 0. \tag{4}$$

Since the material boundary changes due to a change in the shape design parameter, the boundary conditions for CSA are obtained by total or material differentiation of the original boundary conditions and then moving the convective terms to the right side

$$\frac{\partial \boldsymbol{\mathcal{B}}}{\partial \boldsymbol{u}}\boldsymbol{u'} + \frac{\partial \boldsymbol{\mathcal{B}}}{\partial L}L(\boldsymbol{u'}) = \dot{\boldsymbol{g}}(\boldsymbol{x}, t; b) - \boldsymbol{\mathcal{V}}(\boldsymbol{x}) \cdot \left(\frac{\partial \boldsymbol{\mathcal{B}}}{\partial \boldsymbol{u}}\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u} + \frac{\partial \boldsymbol{\mathcal{B}}}{\partial L}L(\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u})\right), \tag{5}$$

where $\dot{\boldsymbol{g}}(\boldsymbol{x}, t; b)$ is the material derivative of the prescribed boundary condition, typically zero for Dirichlet boundary conditions. Nevertheless, even when the boundary condition (2) is homogeneous $\left(\boldsymbol{u}|_{\Gamma_e} = 0\right)$, the CSA boundary condition (5) is in general non-homogeneous due to the convective term: $\boldsymbol{u'}|_{\Gamma_e} = \dot{\boldsymbol{g}}_e - \boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u} \cdot \boldsymbol{\mathcal{V}}(\boldsymbol{x})$, even for $\dot{\boldsymbol{g}}_e = 0$. The commutation

of derivatives on the left side of equation (5) is possible when the derivatives are local. The CSEs (4) with the boundary conditions (5) form a linear system of equations in terms of sensitivity variable $\boldsymbol{u'}$, which can be solved by the same or different numerical method used for solving the analysis problem.

Equations (4) and (5) may be restated in as

$$\frac{\partial \boldsymbol{R}}{\partial b} = \boldsymbol{\mathcal{A}}_b \left(\boldsymbol{u}, L\left(\boldsymbol{u'}\right)\right) - \boldsymbol{f'}\left(\boldsymbol{x}, t; b\right) = 0 \quad \text{on } \Omega, \tag{6}$$

with the corresponding sensitivity BCs

$$\boldsymbol{\mathcal{B}}_b \left(\boldsymbol{u}, L\left(\boldsymbol{u'}\right)\right) = \boldsymbol{g}_b \left(\boldsymbol{x}, t; b\right) \quad \text{on } \Gamma, \tag{7}$$

where $\boldsymbol{g}_b$ is the right-side of equation (5). The linear operator L represents differentiation. Thus, the CSEs are linear in terms of the local sensitivity variables $\boldsymbol{u'}$, whereas flow analysis may be nonlinear in terms of the flow variables $\boldsymbol{u}$. The similarity of equations (6) and (7) to equations (1) and (2) motivates the same solution method for each set of equations with the same mesh for the discretized form. For linear governing equations (1), $\boldsymbol{\mathcal{A}}_b = \boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}_b = \boldsymbol{\mathcal{B}}$. For nonlinear governing equations, the solution $\boldsymbol{u}$ obtained from the solution of equation (1) is needed in equations (6) and (7). Similar to the flow analysis equations, the time term t, suppressed in these equations, represents pseudo-time since the current work involves steady-state sensitivity analysis.

In the direct formulation of CSA, equations (6) and (7) are solved to obtain the local derivatives $\boldsymbol{u'}$ in the domain Ω. This may be followed by adding the convective term, as shown in (3), at the locations of interest in the domain, to obtain the total derivative $\dot{\boldsymbol{u}}$ at those locations. In typical aerodynamic shape optimization applications, it is of interest to obtain the derivative of performance metrics that are obtained by integrating the state variables, for example the lift coefficient C_L or the drag coefficient C_D. The corresponding total derivatives are obtained by chain rule and using the values of $\dot{\boldsymbol{u}}$ given by CSA.

2.3. Discretization of the Flow and Sensitivity Equations

Until this point, the continuous governing equations were differentiated to obtain the CSEs. Thus, there is no approximation involved in deriving the CSA system (6–7). Next, consider a discretization at the mesh level h of the flow equations (1) and a Newton-Raphson implicit scheme that results in a coupled linear system of equations

$$\left[J\left(\{\boldsymbol{u}\}_h^n\right)\right]\{\Delta\boldsymbol{u}\}_h^n = \{\boldsymbol{R}\}_h^n, \tag{8}$$

which has to be solved at each iteration $n = 1, 2, 3, ..., N$ for the updates to the flow variables $\{\Delta\boldsymbol{u}\}_h^n$. This update is used to get the values of flow variables at the next iteration $\{\boldsymbol{u}\}_h^{n+1} = \{\boldsymbol{u}\}_h^n + \{\Delta\boldsymbol{u}\}_h^n$. Here $\left[J\left(\{\boldsymbol{u}\}_h^n\right)\right]$ is the Jacobian or tangent matrix and $\{\boldsymbol{R}\}_h^n$ is the residual of the flow equations at time step n and discretization h. With an Implicit Euler scheme for pseudo-time stepping, the residual at time step $n+1$ is linearized about the time step n yielding the Jacobian matrix

$$\left[J\left(\{\boldsymbol{u}\}_h^n\right)\right] = \sum_i \sum_{j\in\mathcal{N}(i)} -\frac{\partial\{\boldsymbol{R}\}_i^n}{\partial\{\boldsymbol{u}\}_j}. \tag{9}$$

Similarly, the CSEs can be discretized to get a coupled linear system of equations

$$\underbrace{\left[J_{\mathrm{CSE}}\left(\{\boldsymbol{u}\}_h\right)\right]}_{\text{Constant coefficient matrix}} \{\Delta\boldsymbol{u}'\}_h = \underbrace{\{\boldsymbol{R}_{\mathrm{CSE}}\}_h}_{\text{Zero at all interior nodes}}. \tag{10}$$

As a consequence of the CSEs being linear in the sensitivity variables, the steady-state Jacobian matrix $\left[J_{\mathrm{CSE}}\left(\{\boldsymbol{u}\}_h\right)\right]$ in the discretized CSEs (10) is independent of the sensitivity variables $\boldsymbol{u}'$ and only depends on the steady-state flow variables $\boldsymbol{u}$. Borggaard and Burns (1994, 1997) and Liu and Canfield (2013b) showed that if the same discretization used for the analysis is used to discretize the CSEs, then

$$\left[J_{\mathrm{CSE}}\left(\{\boldsymbol{u}\}_h\right)\right] = \left[J\left(\{\boldsymbol{u}\}_h^N\right)\right], \tag{11}$$

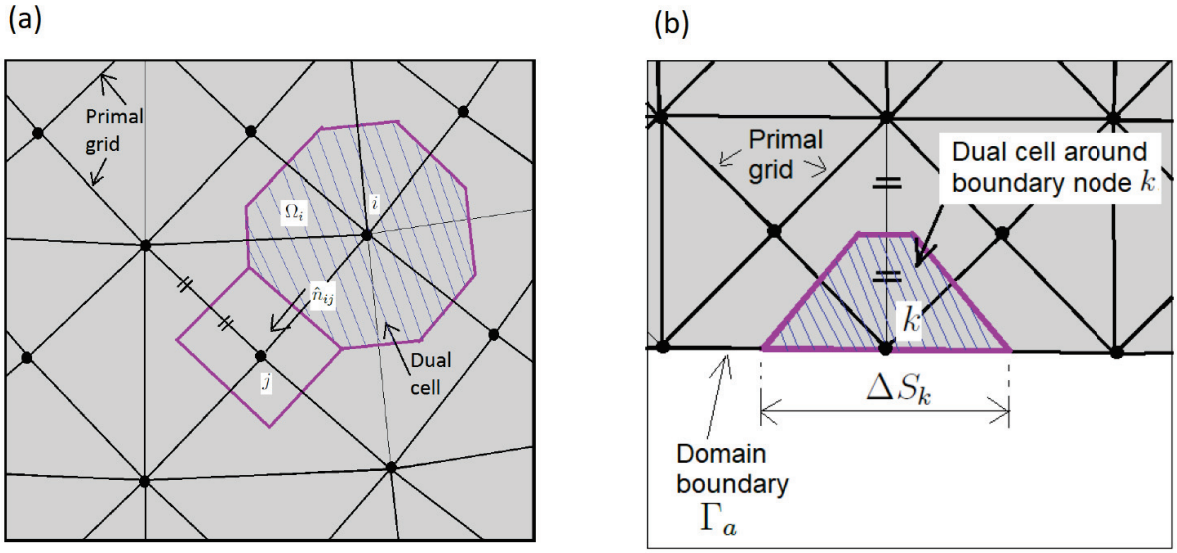


Figure 2: Grid and dual cell (control volume) for vertex-centered finite volume scheme: (a) A dual cell centered around an interior node i, (b) A dual cell centered around a node k on boundary Γ_a.

where N is the last iteration of the flow solver until steady-state convergence. In other words, the coefficient matrix of the discretized CSEs (10), namely $[J_{\text{CSE}}(\{\boldsymbol{u}\}_h)]$, is a constant matrix that can be obtained from the steady-state flow solution $\{\boldsymbol{u}\}_h^N$. As a result, the local shape derivatives are obtained by just a single (one-shot) solution of the linear system (10) for $\{\boldsymbol{u'}\}_h$, assuming $\{\boldsymbol{u'}\}_h = \{\Delta\boldsymbol{u'}\}_h$ with zero initial guess $\left(\{\boldsymbol{u'}\}_h^0 = 0\right)$ without loss of generality. The CSE residual $\{\boldsymbol{R}_{\text{CSE}}\}_h$ is thus zero at all interior nodes, such as node i in Figure 2, and nonzero at boundary nodes, such as node k, based on the boundary conditions. This is described in detail in Section 3.3. Once the local derivatives are computed by solving (10), the total (or material) derivatives can be obtained by adding the convective term according to the discrete form of (3).

2.4. Accurate Boundary Conditions for CSA

Although the boundary conditions (5) do not appear explicitly in the discretized equations (10), they are included in the residual $\{\boldsymbol{R}_{\text{CSE}}\}_h$. In fact, the solution of the local shape design derivatives $\{\boldsymbol{u'}\}$ are mainly driven by CSA boundary conditions that determine the only nonzero terms in $\{\boldsymbol{R}_{\text{CSE}}\}_h$. Hence, it is important to accurately evaluate these while

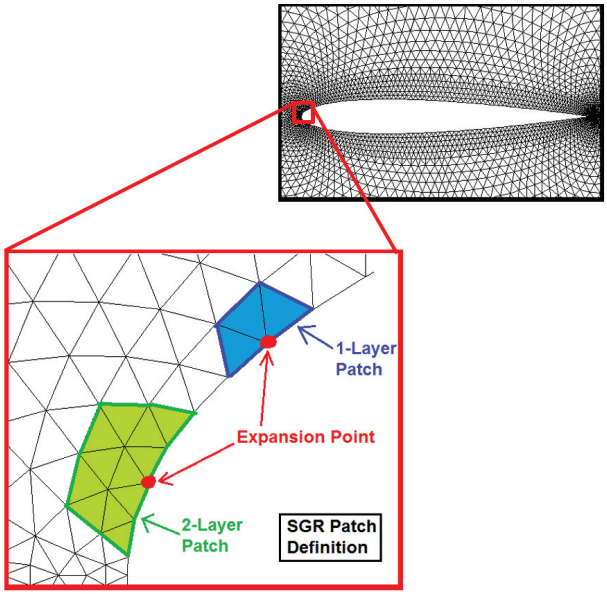


Figure 3: Illustration of SGR patches.

forming the CSA residual $\{\boldsymbol{R}_{\mathrm{CSE}}\}_h$. Duvigneau and Pelletier (2006) have emphasized the effect of CSA boundary conditions on the accuracy of the local shape design derivatives. Once the design variables are defined, all terms on the right side of equation (5) are known except for $\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u}$ in the convective term, which involves spatial gradients of the flow variables. Improvising on the l-patch method by Duvigneau and Pelletier (2006), which itself was based on the super-convergent patch recovery method by Zienkiewicz and Zhu (1992), Cross and Canfield (2014) proposed the Spatial Gradient Reconstruction (SGR) method to approximate the spatial gradient term appearing in the CSA boundary condition (5) for structures. SGR is used in the current research to approximate the spatial gradients $\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u}$ appearing in the CSA boundary conditions.

An SGR patch constitutes a collection of points which are used to obtain the spatial gradients at an expansion point. Examples of 1-layer and 2-layer SGR patches are illustrated in Figure 3. Consider a scalar ϕ, such as pressure or a component of velocity. Then the value of spatial gradients $\frac{\partial \phi}{\partial x}$ and $\frac{\partial \phi}{\partial y}$ at the expansion point (x_0, y_0) can be obtained with least-squares approach by using the values of ϕ at the points in an SGR patch, and the Taylor

series expansion of ϕ at the expansion point

$$\begin{aligned}\phi(x_0+\Delta x,\, y_0+\Delta y) &= \phi(x_0,\, y_0)+\frac{\partial\phi}{\partial x}\Delta x+\frac{\partial\phi}{\partial y}\Delta y \\ &+ \frac{1}{2}\frac{\partial^2\phi}{\partial x^2}\left(\Delta x\right)^2+\frac{1}{2}\frac{\partial^2\phi}{\partial y^2}\left(\Delta y\right)^2+\frac{\partial^2\phi}{\partial x\partial y}\Delta x\Delta y \\ &+ \cdots.\end{aligned} \tag{12}$$

The accuracy of the spatial gradients obtained from SGR can be improved by modifying parameters such as the patch-size and the order of Taylor series. Cross and Canfield (2016) discuss the numerical behavior, accuracy, and convergence of design derivatives when SGR is used to approximate the CSA boundary conditions. They also give guidelines in selecting SGR parameters to achieve accurate results.

Since SGR does not require information of the discretization method used, such as finite element or finite volume or finite difference, it can be used as a post-processing step, following the analysis solution, to calculate the required spatial gradients. This makes CSA amenable for nonintrusive implementation.

2.5. Nonintrusive Implementation

The procedure for nonintrusive implementation of the direct formulation of CSA is illustrated by the flowchart given in Figure 4 and is explained next.

First, the flow analysis solution $\boldsymbol{u}$ is obtained by running the analysis code (for example SU2 or FLUENT). Then, the SGR method is employed to approximate the spatial gradients $\boldsymbol{\nabla_x u}$ that appear in the sensitivity BCs. This is shown in the left branch of the flowchart. Another term required for assembling the CSA boundary conditions (5) is the design velocity $\boldsymbol{\mathcal{V}}(\boldsymbol{x})$. This term can be obtained nonintrusively by using either the analytic equations of the parametric geometry, or by using finite difference approximation. Alternatively, the design velocity could also be calculated using a complex step approximation (Kulkarni et al., 2014b). This is shown in the right branch of the flowchart. With this, the CSA boundary conditions (5) are defined. Next, the linear CSEs (10) have to be solved. This can be done

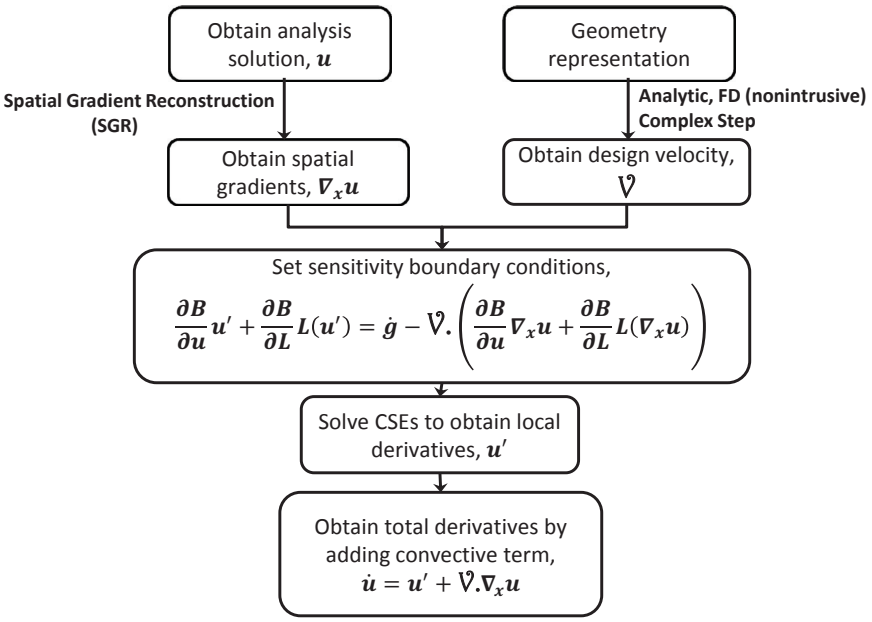


Figure 4: Flowchart for direct formulation of Continuum Sensitivity Analysis

using either of the following two approaches:

1. Solve the CSEs using the same code that solves the flow equations. In this case the flow code would be used to solve the CSEs instead of solving the flow equations (8). Hence, the flow code should be able to perform two tasks: (a) accommodate non-homogeneous boundary conditions such as (5) in order to construct the correct right side $\{\boldsymbol{R}_{\mathrm{CSE}}\}_h$, and (b) use the steady-state Jacobian matrix $\left[J\left(\{\boldsymbol{u}\}_h^N\right)\right]$ as the coefficient matrix to solve the linear system (10). Since this coefficient matrix is a constant matrix for the discretized CSEs, multiple iterations are not required. The solver can be stopped after the first iteration.
2. Solve the CSEs externally. In this case, the Jacobian matrix $\left[J\left(\{\boldsymbol{u}\}_h^N\right)\right]$ should be exported or regenerated from the steady-state solution. At the same time, the right side of the CSEs $\{\boldsymbol{R}_{\mathrm{CSE}}\}_h$ should be generated and the linear system (8) should be solved externally to the flow solver.

The solution of CSEs yields the local derivatives $\boldsymbol{u'}$. In the final step, the convective term is added to these local derivatives to obtain total or material derivatives $\dot{\boldsymbol{u}}$.

With the choice of a zero initial initial guess, $\{\boldsymbol{u'}\}^0 = 0$ for solving equations (10), the residual $\{\boldsymbol{R}_{\text{CSE}}\}_h$ comes solely from boundary sensitivity terms. In other words, the the only nonzero terms in $\{\boldsymbol{R}_{\text{CSE}}\}_h$ are from the nodes on the boundary of the domain, such as the node k shown in Figure 2(b). Details of how the boundary sensitivity terms govern the residual are given in Section 3.3.

Comparing equation (8) with equation (10), we observe the following.

- The Jacobian matrix $[J_{\text{CSE}}(\{\boldsymbol{u}\}_h)]$ appearing on the left side of the discrete CSEs (10) is same as that for the (converged) steady-state primary flow, that is, the Jacobian at the last pseudo-time step of the primary implicit flow solver.

- Taking advantage of the linearity of the CSEs, after discretization the local sensitivities can be obtained by just a single (one-shot) solution of equation (10).

- The right-side of equation (10) is the only term that has to be calculated for solving the linear system, if the Jacobian matrix $\left[J\left(\{\boldsymbol{u}\}_h^N\right)\right]$ at the steady-state flow solution can be output (or stored and reused).

The proposed nonintrusive process can be implemented in two ways, as shown in Figure 5, depending on whether: (a) the flow solver can provide the Jacobian matrix (*e.g.*, SU2 software), or (b) the flow solver cannot provide the Jacobian matrix (*e.g.*, FLUENT software). In case (b), SU2 was used as a tool to reconstruct the Jacobian matrix, following Godfrey and Cliff (2001). Hence, in this case, the Jacobian matrix used for CSA may not be consistent with the primary analysis (FLUENT). Since SU2 is a vertex-centered code and FLUENT is a cell-centered code, FLUENT cell-centered data has to be approximated at the vertices, so that it can be used in SU2 to compute the Jacobian. This approximation was done using FLUENT. Alternatively, SGR may be used for this approximation. In the numerical results presented in Section 4, it is shown that even with these approximations, CSA yields accurate

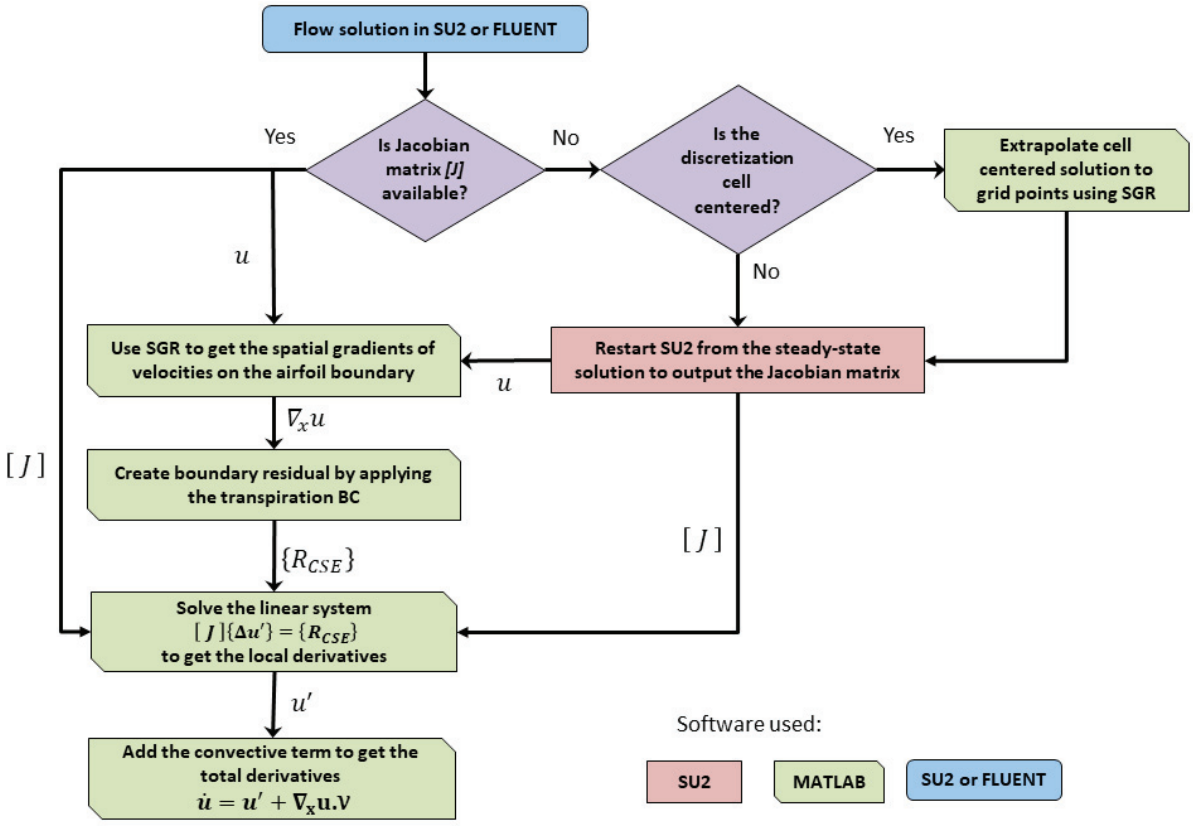


Figure 5: Nonintrusive CSA process when the Jacobian matrix may be output from the flow solver (such as SU2) or may not be output from the flow solver (such as FLUENT).

shape derivatives. The next subsection explains the requirements for a flow analysis code to be used for nonintrusive CSA, based on the procedure aforementioned.

2.6. Requirements of a Flow Analysis Code for Nonintrusive CSA

The advantages of CSA over other design sensitivity analysis methods, such as finite difference, complex step approximation, discrete analytic method, and semi-analytic method are that it can yield more accurate derivatives, it does not suffer from the requirement to choose a finite difference step size, and that it does not require calculation of mesh sensitivity. The latter advantage is mentioned particularly by Borggaard and Burns (1997). Cross and Canfield (2012) have stated the requirements of an analysis code for it to be used as a "black-box" or nonintrusive tool for CSA of a structural system. Similarly, the following can be stated as the requirements of a flow solver so that it can be used for nonintrusive application of CSA.

1. Output response $\boldsymbol{u}$: The solver should output the state variables such as fluid pressure, velocities, and temperature. This corresponds to the step of "Obtain analysis solution" in the flowchart of Figure 4. Although this seems like a trivial requirement, it is stated here because some solvers may have an option of only plotting the flow variables and not reporting the values. The flow variables have to be post-processed to obtain the spatial gradients $\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u}$. As explained in the previous subsection, these spatial gradients are required on the boundaries with nonzero design velocity to form the CSA boundary conditions, and in the final step to calculate the convective term $(\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{u} \cdot \boldsymbol{\mathcal{V}})$ at the locations where the total derivatives $\dot{\boldsymbol{u}}$ are needed.
2. For solving CSEs internally (using the same flow analysis code) there are two requirements. First, the code should be able to handle the non-homogeneous boundary conditions (5), since the boundary conditions for CSA are typically non-homogeneous, unlike the homogeneous boundary conditions for the original analysis. Second, the flow solver should be able to regenerate the Jacobian matrix $\left[J\left(\{u\}_h^N\right)\right]$ from the steady-state solution and further use it to solve (10).

Table 2: Survey of CFD solvers for nonintrusive CSA.

Requirements	FLUENT	ZAERO	ZEUS	FUN3D	OpenFoam (Open source)	SU2 (Open source)
Output response	✓	✓	✓	✓	✓	✓
Apply non-homogeneous BC	×	×	×	?	?	×
Output tangent matrix	×	×	×	?	?	✓

If the CSEs cannot be solved by restarting the flow solver, then they can be solved externally. In this case, the steady-state flow Jacobian matrix $\left[J\left(\{u\}_h^N\right)\right]$ is required. Some flow solvers (for example SU2) can output the full Jacobian matrix. However, if a flow solver (e.g., FLUENT) cannot output the Jacobian matrix, then it can be regenerated from the steady-state solution (using SU2, for example).

If all these requirements are satisfied, a nonintrusive implementation of CSA is possible using the flow solver as a "black-box" tool. The results of a survey of some flow solvers in context of nonintrusive CSA application is shown in Table 2. Columns list the different flow solvers while rows list the requirements that they should satisfy for their use as nonintrusive CSA solvers. OpenFOAM and SU2 are open source software, whereas the others are black-box software. A check mark '✓' indicates that the capability exists, a cross '×' indicates that the capability does not exist while the question mark sign '?' indicates that enough literature is not available to determine that specific requirement. For open source codes, such as SU2 and OpenFOAM, it is in general possible to implement any desired capability. However, the above survey lists capabilities in the most recent release of the software. Based on this survey, it was decided to use SU2 for nonintrusive implementation of compressible inviscid (Euler equation) flow.

3. Sensitivity of Flow Over NACA0012 Airfoil

3.1. Flow Analysis

The Euler flow equations in conservation form are

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} - \boldsymbol{H} = 0 \tag{13}$$

or

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial y} - \boldsymbol{H} = 0 \tag{14}$$

subject to far-field boundary condition at the boundary Γ_f

$$\boldsymbol{u}|_{\Gamma_f} = \boldsymbol{u}_\infty, \tag{15}$$

and flow tangency (or wall) boundary condition at the airfoil boundary Γ_a

$$\boldsymbol{v} \cdot \hat{\boldsymbol{n}}|_{\Gamma_a} = 0, \tag{16}$$

where $\boldsymbol{u}(x,\, y,\, t)$ is the vector of conserved variables, $\boldsymbol{F}(x,\, y,\, t)$ and $\boldsymbol{G}(x,\, y,\, t)$ are the flux vectors in the x and y coordinate directions, $\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{u}}$ and $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{u}}$ are the respective flux Jacobian matrices, and $\boldsymbol{H}$ is the source term, which is zero. The state vector and flux vectors are

$$\boldsymbol{u} = \begin{Bmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho e_t \end{Bmatrix}, \quad \boldsymbol{F} = \begin{Bmatrix} \rho v_x \\ p + \rho v_x^2 \\ \rho v_y v_x \\ \rho v_x h_t \end{Bmatrix}, \quad \boldsymbol{G} = \begin{Bmatrix} \rho v_y \\ \rho v_x v_y \\ p + \rho v_y^2 \\ \rho v_y h_t \end{Bmatrix}. \tag{17}$$

Variables ρ, p, v_x, v_y, $e_t = \left(\frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{v_x^2 + v_y^2}{2}\right)$, and $h_t = e_t + \frac{p}{\rho}$ denote density, pressure, x-component (horizontal) of velocity, y-component (vertical) of velocity, total energy, and total enthalpy in the domain, respectively. The pressure and density can be related to the

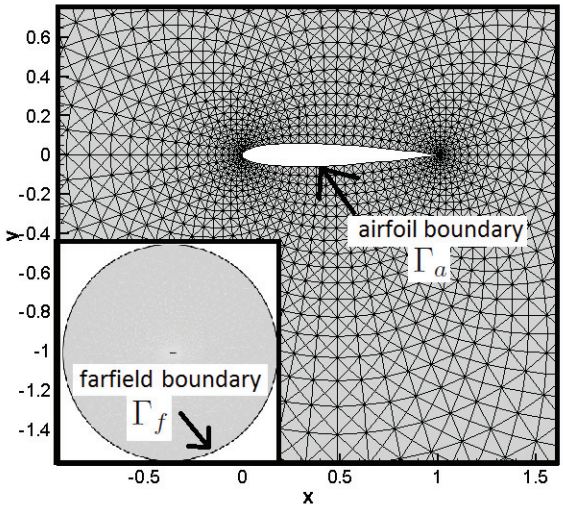


Figure 6: Domain, boundaries and unstructured mesh for flow over an airfoil. The inset figure shows the entire flow field with airfoil at the center.

temperature T by the equation of state $p = \rho \mathcal{R} T$, where $\mathcal{R}$ is the specific gas constant. The far-field boundary condition implies that $\boldsymbol{u}_\infty$ is the prescribed state at the far-field boundary Γ_f. The flow tangency boundary condition implies that the velocity vector $\boldsymbol{v}$ has no component along the unit normal $\hat{\boldsymbol{n}}$ on the airfoil boundary Γ_a. In terms of (x-y) components, the wall boundary condition (16) can be written as

$$v_x n_x + v_y n_y|_{\Gamma_a} = 0, \tag{18}$$

or, in terms of tangential-normal (t-n) components, it can simply be written as

$$v_n|_{\Gamma_a} = 0. \tag{19}$$

The far-field and wall boundaries for flow over an airfoil are shown in Figure 6. When a vertex-centered finite volume discretization is used, the system of nonlinear coupled partial differential equations (13) is approximated by the following semi-discrete system

$$\Omega_i \frac{\partial \{\boldsymbol{u}(t)\}_i}{\partial t} - \{\boldsymbol{R}\}_i = 0, \tag{20}$$

where $\{\boldsymbol{R}\}_i$ is the residual at node i, given by

$$\{\boldsymbol{R}\}_i \equiv -\sum_{j\in\mathcal{N}(i)} \left\{\widetilde{\boldsymbol{F}}\right\}_{ij} \Delta S_{ij}. \tag{21}$$

The grid and associated control volume, sometimes called a dual cell, is shown in Figure 2. $\left\{\widetilde{\boldsymbol{F}}\right\}_{ij}$ is the projected flux for the edge of the control volume that intersects the grid line connecting nodes i and j, ΔS_{ij} is the width of that edge, Ω_i is the area of the control volume centered around node i, and $\mathcal{N}(i)$ consists of all the nodes neighboring node i. The dual cells are constructed by joining the centroids of the triangles in the primary grid. Although these equations can be used for time-accurate solutions, the variable t in the present context represents pseudo-time and is used for the purpose of time marching to reach the steady-state. Any flux approximation scheme, for example Roe's upwind scheme, can be used to approximate the projected flux $\left\{\widetilde{\boldsymbol{F}}\right\}_{ij}$. After assembly, the resulting discrete Euler system of equations (8) can be solved to get the steady-state flow solution.

3.2. Sensitivity Analysis

Differentiating the continuous Euler equations (13) with respect to the shape design variable b, we obtain the local-form CSEs,

$$\frac{\partial \boldsymbol{u}'}{\partial t} + \frac{\partial \boldsymbol{F}'}{\partial x} + \frac{\partial \boldsymbol{G}'}{\partial y} = 0, \tag{22}$$

where the sensitivity variables $\boldsymbol{u}'(x,\, y,\, t)$ and sensitivity flux vectors $\boldsymbol{F}'(x,\, y,\, t)$ and $\boldsymbol{G}'(x,\, y,\, t)$ are

$$\boldsymbol{u}' = \left\{\begin{array}{c} \rho' \\ (\rho v_x)' \\ (\rho v_y)' \\ (\rho e_t)' \end{array}\right\}, \quad \boldsymbol{F}' = \left\{\begin{array}{c} \rho' v_x + \rho v_x' \\ p' + (\rho v_x)' v_x + (\rho v_x) v_x' \\ (\rho v_y)' v_x + (\rho v_y) v_x' \\ (\rho h_t)' v_x + (\rho h_t) v_x' \end{array}\right\}, \quad \boldsymbol{G}' = \left\{\begin{array}{c} \rho' v_y + \rho v_y' \\ (\rho v_x)' v_y + (\rho v_x) v_y' \\ p' + (\rho v_y)' v_y + (\rho v_y) v_y' \\ (\rho h_t)' v_y + (\rho h_t) v_y' \end{array}\right\}. \tag{23}$$

The CSEs are linear partial differential equations (that is, linear in terms of sensitivity variables) although the Euler analysis equations (13) are nonlinear partial differential equations.

As explained in Section 2.2, the CSA boundary conditions are obtained by total or material differentiation of the original boundary conditions and then moving the convective terms to the right side. Thus, the far-field CSA boundary condition is

$$\boldsymbol{u}'|_{\Gamma_f} = 0 - \frac{\partial \boldsymbol{u}}{\partial x}\mathcal{V}_x\bigg|_{\Gamma_f} - \frac{\partial \boldsymbol{u}}{\partial y}\mathcal{V}_y\bigg|_{\Gamma_f}. \tag{24}$$

The design velocity $\boldsymbol{\mathcal{V}} = \mathcal{V}_x\hat{\imath} + \mathcal{V}_y\hat{\jmath}$ is zero on the far-field boundary Γ_f, because the design variable only changes shape of the airfoil boundary. Thus, far-field CSA boundary condition (24) is homogeneous

$$\boldsymbol{u}'|_{\Gamma_f} = 0. \tag{25}$$

The CSA boundary condition at the airfoil boundary Γ_a is

$$\begin{aligned} \boldsymbol{v}' \cdot \hat{\boldsymbol{n}}|_{\Gamma_a} &= -\left\{\left(\frac{\partial v_x}{\partial x}\mathcal{V}_x + \frac{\partial v_x}{\partial y}\mathcal{V}_y\right)\hat{\imath} + \left(\frac{\partial v_y}{\partial x}\mathcal{V}_x + \frac{\partial v_y}{\partial y}\mathcal{V}_y\right)\hat{\jmath}\right\} \cdot \hat{\boldsymbol{n}}\bigg|_{\Gamma_a} \\ &\quad - \{v_x\hat{\imath} + v_y\hat{\jmath}\} \cdot \dot{\hat{\boldsymbol{n}}}\big|_{\Gamma_a}, \end{aligned} \tag{26}$$

where $\dot{\hat{\boldsymbol{n}}} = \frac{D\hat{\boldsymbol{n}}}{Db}$ is the material derivative of the unit normal. This boundary condition is a version of the equation (5), obtained by using the flow tangency boundary condition (16). An alternative way of deriving the CSA boundary condition is by using spatial gradients of the velocity components in the tangential-normal (t-n) directions instead of (x-y) directions. Thus, CSA boundary condition at the airfoil boundary Γ_a is

$$\begin{aligned} \boldsymbol{v}' \cdot \hat{\boldsymbol{n}}|_{\Gamma_a} &= -\left\{\left(\frac{\partial v_t}{\partial t}\mathcal{V}_t + \frac{\partial v_t}{\partial n}\mathcal{V}_n\right)\hat{t} + \left(\frac{\partial v_n}{\partial t}\mathcal{V}_t + \frac{\partial v_n}{\partial n}\mathcal{V}_n\right)\hat{n}\right\} \cdot \hat{\boldsymbol{n}}\bigg|_{\Gamma_a} \\ &\quad - \left\{v_t\hat{t} + v_n\hat{n}\right\} \cdot \dot{\hat{\boldsymbol{n}}}\big|_{\Gamma_a}. \end{aligned} \tag{27}$$

However, since the unit vectors $\hat{t}$ and $\hat{n}$ are orthogonal, and $v_n = 0$ on the airfoil according

to flow tangency condition (19), CSA boundary condition (BC) (27) simplifies to

$$\boldsymbol{v'} \cdot \hat{\boldsymbol{n}}|_{\Gamma_a} = -\left\{ \frac{\partial v_n}{\partial t}\mathcal{V}_t + \frac{\partial v_n}{\partial n}\mathcal{V}_n \right\}\Big|_{\Gamma_a} - \left\{ v_t \hat{t} \right\} \cdot \dot{\hat{\boldsymbol{n}}}\Big|_{\Gamma_a}. \tag{28}$$

This form is preferable to (26), because it requires approximating two rather than four spatial gradients. Additionally, this form allows specifying $v_n = 0$ in the $\dot{\hat{\boldsymbol{n}}}$ term. In other words, this form allows specifying the physically correct boundary condition. The CSA BC (28) is called a transpiration boundary condition, because the dependent variable $\boldsymbol{v'}$ (that is, the local derivatives of the velocity) has a nonzero component along the normal direction of the wall boundary Γ_a. More details of this boundary condition are discussed in Section 3.3.

Until this point in the sensitivity analysis, no discretization is involved. Thus, the CSEs (22) with the corresponding boundary conditions (25) and (28) represent continuous equations that govern the local derivatives $\boldsymbol{u'}$ of the flow variables. Derivation of CSA completes the "differentiate" part of the CSA approach: "first differentiate, then discretize." The CSEs can be solved numerically using any discretization scheme. Here we emphasize using the same discretization that was used for the flow analysis. Moreover, this procedure can also be done nonintrusively, if the conditions given in Section 2.6 are met. Using the same spatial discretization scheme (vertex-based finite volume) the residual for the CSEs can be calculated as

$$\{\boldsymbol{R}_{\mathrm{CSE}}\}_i = -\sum_{j \in \mathcal{N}(i)} \left\{ \widetilde{\boldsymbol{F'}} \right\}_{ij} \Delta S_{ij}. \tag{29}$$

This equation is similar to the corresponding equation (21) for the flow analysis. After assembly, the resulting discrete CSEs (10) can be solved to get the local flow derivatives $\{\boldsymbol{u'}\}_h$. The CSA coefficient matrix $[J_{\mathrm{CSE}}(\{\boldsymbol{u}\}_h)]$ is the converged Jacobian matrix of the steady-state flow solution $\{\boldsymbol{u}\}_h^N$, as stated earlier relative to equation (11), and thus is a constant coefficient matrix. As a consequence, the update $\{\Delta \boldsymbol{u'}\}$ can be obtained starting with a zero initial guess $\left(\{\boldsymbol{u'}\}_h^0 = 0\right)$ without loss of generality.. The CSE residual $\{\boldsymbol{R}_{\mathrm{CSE}}\}_h$ is then zero at all interior nodes and is dictated by the CSA boundary conditions as described

next.

3.3. Discrete CSA Boundary Conditions

The CSA boundary conditions (25) and (28), written for the continuous domain, have to be discretized and implemented for the discrete system (10). Although CSA boundary conditions previously were implemented in finite difference (Borggaard and Burns, 1997) and finite element (Duvigneau and Pelletier, 2006) formulation, challenges arise when finite volume discretization is used because boundary conditions are typically imposed with a weak or integral approach (Hirsch, 1990; Palacios et al., 2013). An important contribution of the current work is the way in which these discrete boundary conditions are implemented in the finite volume formulation.

The CSA far-field BC (25) can be recognized as a form of the primary far-field BC (15) where the value of the sensitivity state variables at the boundary Γ_f are $\boldsymbol{u'}|_\infty = 0$. Hence this BC can be treated as was the primary far-field BC. For the following NACA0012 example the design velocity is zero in the far field, so that BC (25) is homogeneous. The CSA transpiration BC (28) however is non-homogeneous, unlike the homogeneous flow tangency BC (16) in the primary analysis. Such transpiration boundary conditions are encountered in simulations involving store ejections or prescribed motion (Zhang et al., 2006; Chen and Zhang, 2013). Consider a dual cell (finite volume) around a node k on the airfoil boundary Γ_a as shown in Figure 2(b). Discretizing BC (28) at node k on boundary Γ_a, we get

$$\{\boldsymbol{v'}\}_k \cdot \{\hat{\boldsymbol{n}}\}_k|_{\Gamma_a} = -\left\{\frac{\partial v_n}{\partial t}\mathcal{V}_t + \frac{\partial v_n}{\partial n}\mathcal{V}_n\right\}_k \cdot \{\hat{\boldsymbol{n}}\}_k\Big|_{\Gamma_a} - \left\{v_t\hat{t}\right\}_k \cdot \left\{\dot{\hat{\boldsymbol{n}}}\right\}_k\Big|_{\Gamma_a}. \tag{30}$$

Define transpiration velocity, g_{b_k}, as the right-side of the above equation (30),

$$g_{b_k} \equiv -\left\{\frac{\partial v_n}{\partial t}\mathcal{V}_t + \frac{\partial v_n}{\partial n}\mathcal{V}_n\right\}_k \cdot \{\hat{\boldsymbol{n}}\}_k - \left\{v_t\hat{t}\right\}_k \cdot \left\{\dot{\hat{\boldsymbol{n}}}\right\}_k \tag{31}$$

so that the discretized boundary condition at node k may be written compactly as

$$\{\boldsymbol{v'}\}_k \cdot \{\hat{\boldsymbol{n}}\}_k|_{\Gamma_a} = g_{b_k}|_{\Gamma_a} \,. \tag{32}$$

In (x-y) coordinates, the left-side of equation (32) can be simplified as $\left\{v'_x n_x + v'_y n_y\right\}_k$. Hence, the value of g_{b_k} from equation (31) is used to substitute the expression $\left\{v'_x n_x + v'_y n_y\right\}_k$ appearing during calculation of the flux at the nodes on the boundary Γ_a, as shown next. In contrast to a dual cell around an interior node i as shown in Figure 2(a), a dual cell (or control volume) around a boundary node k has one of its edges almost coinciding with the domain boundary, as shown in Figure 2(b). For the boundary node k, the width of such a boundary edge is ΔS_k. The flux corresponding to the edge ΔS_k is denoted as $\left\{\widetilde{\boldsymbol{F'}}\right\}_k$, instead of the notation $\left\{\widetilde{\boldsymbol{F'}}\right\}_{ij}$ that represents the flux at an edge of the interior node i in equation (29). Following the weak BC implementation approach (Hirsch, 1990), the projected sensitivity flux at node k on boundary Γ_a is

$$\left\{\widetilde{\boldsymbol{F'}}\right\}_k = (\{\boldsymbol{F'}\}_k \,\hat{\imath} + \{\boldsymbol{G'}\}_k \,\hat{\jmath}) \cdot \{\hat{\boldsymbol{n}}\}_k \,, \tag{33}$$

where $\hat{\boldsymbol{n}} = n_x\hat{\imath} + n_y\hat{\jmath}$. After substituting equation (23) in (33) and regrouping,

$$\begin{aligned}
\left\{\widetilde{\boldsymbol{F'}}\right\}_k &= \left\{\begin{array}{c} \rho' v_x + \rho v_x' \\ p' + (\rho v_x)' v_x + (\rho v_x) v_x' \\ (\rho v_y)' v_x + (\rho v_y) v_x' \\ (\rho h_t)' v_x + (\rho h_t) v_x' \end{array}\right\}_k n_{x_k} + \left\{\begin{array}{c} \rho' v_y + \rho v_y' \\ (\rho v_x)' v_y + (\rho v_x) v_y' \\ p' + (\rho v_y)' v_y + (\rho v_y) v_y' \\ (\rho h_t)' v_y + (\rho h_t) v_y' \end{array}\right\}_k n_{y_k} \\
&= \left\{\begin{array}{c} \rho' (v_x n_x + v_y n_y) \\ p' n_x + (\rho v_x)' (v_x n_x + v_y n_y) \\ p' n_y + (\rho v_y)' (v_x n_x + v_y n_y) \\ (\rho h_t)' (v_x n_x + v_y n_y) \end{array}\right\}_k + \left\{\begin{array}{c} \rho \left(v_x' n_x + v_y' n_y\right) \\ (\rho v_x) \left(v_x' n_x + v_y' n_y\right) \\ (\rho v_y) \left(v_x' n_x + v_y' n_y\right) \\ (\rho h_t) \left(v_x' n_x + v_y' n_y\right) \end{array}\right\}_k \\
&= \left\{\begin{array}{c} \rho \left(g_{b_k}\right) \\ p' n_x + (\rho u) \left(g_{b_k}\right) \\ p' n_y + (\rho v) \left(g_{b_k}\right) \\ (\rho h_t) \left(g_{b_k}\right) \end{array}\right\}_k .
\end{aligned} \tag{34}$$

The subscript k in equation (34) represents the discrete values of the respective terms evaluated at the location of node k on the boundary Γ_a. The term $(v_x n_x + v_y n_y)$ is the component of flow velocity $\boldsymbol{v}$ along the unit normal $\hat{\boldsymbol{n}}$. Hence, according to the flow tangency condition (18), at node k, $\{v_x n_x + v_y n_y\}_k = 0$. Similarly, according to the CSA transpiration BC (32), at node k, $\left\{v_x' n_x + v_y' n_y\right\}_k = g_{b_k}$. Furthermore, since $p_k' = 0$, equation (34) simplifies to

$$\left\{\widetilde{\boldsymbol{F'}}\right\}_k = g_{b_k} \left\{\begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho h_t \end{array}\right\}_k , \tag{35}$$

where the value of g_{b_k} is given by equation (31). The value of the boundary residual at node

k is then calculated as

$$\{\boldsymbol{R}_{\mathrm{CSE}}\}_k = -\left\{\widetilde{\boldsymbol{F'}}\right\}_k \Delta S_k, \tag{36}$$

where ΔS_k is the edge length corresponding to the boundary node k as shown in Figure 2(b). The only nonzero contribution to $\{\boldsymbol{R}_{\mathrm{CSE}}\}_h$ on the right-side of equation (10) is from $\{\boldsymbol{R}_{\mathrm{CSE}}\}_k$, which corresponds to the nodes k on the boundary Γ_a. Thus the CSA transpiration boundary condition (32) solely drives the CSA solutions. Given this, we can list the following terms that contribute significantly to the local derivative CSA solutions.

- Spatial gradients of the velocity components on the boundary Γ_a contribute to g_{b_k} (equation 31), which are calculated nonintrusively from the steady-state flow solution using SGR as described in Section 4.4.
- Design velocity $\boldsymbol{\mathcal{V}}$ on the boundary Γ_a contributes to g_{b_k} (equation 31), which is obtained from the definition of the design variable (39), independent of the flow solution.
- Material derivative of the normal direction vector $\left\{\dot{\hat{\boldsymbol{n}}}\right\}_k = \frac{D\{\hat{\boldsymbol{n}}\}_k}{Db}$ contributes to g_{b_k} (equation 31), which is also obtained from the definition of the design variable (39), independent of the flow solution.
- Tangential component of flow velocities v_t on the boundary Γ_a contributes to g_{b_k} (equation 31), which is obtained from the flow analysis solution.
- Flow density ρ, flow velocity components u and v, and total enthalpy h_t contributes to $\left\{\widetilde{\boldsymbol{F'}}\right\}_k$ in equation (35), which are obtained from the flow analysis solution.
- Element lengths ΔS_k at nodes k on the boundary Γ_a contributes to $\{\boldsymbol{R}_{\mathrm{CSE}}\}_k$ in equation (36), which is obtained from the geometry preprocessor used by the flow solver.

4. Numerical Results

4.1. Geometry, Design Variable, and Grids

The NACA0012 airfoil has been studied widely (Jameson, 1983; Anderson and Bonhaus, 1994). In the current study, we use the NACA geometry used by Vassberg and Jameson (2010). The airfoil geometry is based on the NACA0012 equation

$$y(x) = \pm\frac{0.12}{0.2}\left(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right), \quad (37)$$

$$\frac{\partial y}{\partial x}(x) = \pm\frac{0.12}{0.2}\left(\frac{0.14845}{\sqrt{x}} - 0.1260 - 0.7032x + 0.8529x^2 - 0.4060x^3\right). \quad (38)$$

However, it is extended in chord such that the resulting sharp trailing-edge location is $x_{TE} = 1.0089304115$. The components of unit normal along the airfoil boundary are plotted in Figure 12.

Vassberg and Jameson (2010) prepared high-quality quadrilateral grids specifically to perform a grid convergence study, based on the outcome of the *4th Drag Prediction Workshop* AIAA (2009). These grids were developed using the Karman-Treffetz transformation and are based on the standard O-mesh topology. Each quadrilateral cell of the mesh has an aspect ratio of one, and the intersecting grid lines are essentially orthogonal at each vertex within the mesh. The far-field boundary is approximately 150 chord lengths away from the airfoil. The grids used in the current study are obtained by dividing each quadrilateral element of the Vassberg and Jameson (2010) grids into four triangular elements, that is by joining the quadrilateral vertices to the quadrilateral centroid. The airfoil mesh thus obtained is shown in Figure 7. The reason to change the quadrilaterals into triangles was to demonstrate the capability of the current CSA formulation for unstructured meshes. Kulkarni (2016) demonstrated similar results for an example with a quadrilateral mesh. Additionally, the nodes near the top surface of the airfoil at the maximum thickness location are positioned in a peculiar way to allow the use of finite difference method, only at one location on the airfoil. The details about this are explained in Section 4.4. The resulting mesh is an

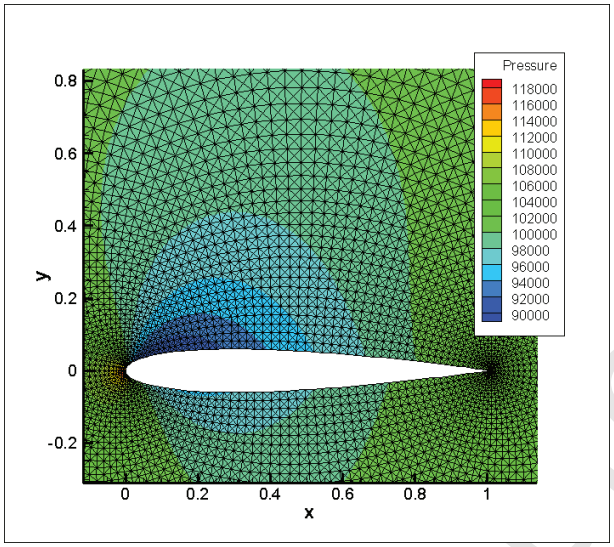


Figure 7: Airfoil mesh with 256×256 triangular cells. Color contours show pressure distribution in N/m^2 on the airfoil, $M = 0.5$, $\alpha = 1.25°$.

unstructured mesh with triangular elements. The six meshes used in the current study have triangular cells ranging from 64×64 (approximately 4 thousand) cells for the coarsest mesh to 2048×2048 (approximately 4 million) cells for the finest mesh. These meshes are identified with the parameter NC which represents the square root of the number of cells in the mesh. The refinement of grids is uniform and consistent such that with each refinement the edge length halves and the number of cells quadruples. Hence, the six mesh levels are: $NC = 64$, 128, 256, 512, 1024, 2048. The corresponding values of $1/NC$ are used as abscissa in the grid convergence studies.

4.2. Flow Analysis

The reference values and other details used for flow analysis are shown in Table 3. The flow analysis was done using SU2. The density, flow velocity components and coefficient of pressure, C_p on the airfoil are plotted against x-coordinate locations in Figure 8. The pressure field is shown in Figure 7. The Mach number contours near the airfoil are shown in Figure 9. These results match closely to those presented by Vassberg and Jameson (2010) for the same test case. Results such as these are output at each of the six mesh levels. The true or exact solution for the coefficients of lift and drag, C_L and C_D, respectively,

Table 3: Details of the flow analysis

Parameter	Value
Mach number, M	0.5
Angle of attack,α	1.25°
Reference chord length, c	1.0 m
Reference moment center, X_{ref}	0.25 m
Free stream pressure, p_∞	101325 N/m^2
Free stream temperature, T_∞	288.15 K
Flux scheme	Roe's II order upwind
Convergence criteria	$\log_{10}$ of L_2 norm of continuity residual less than -10

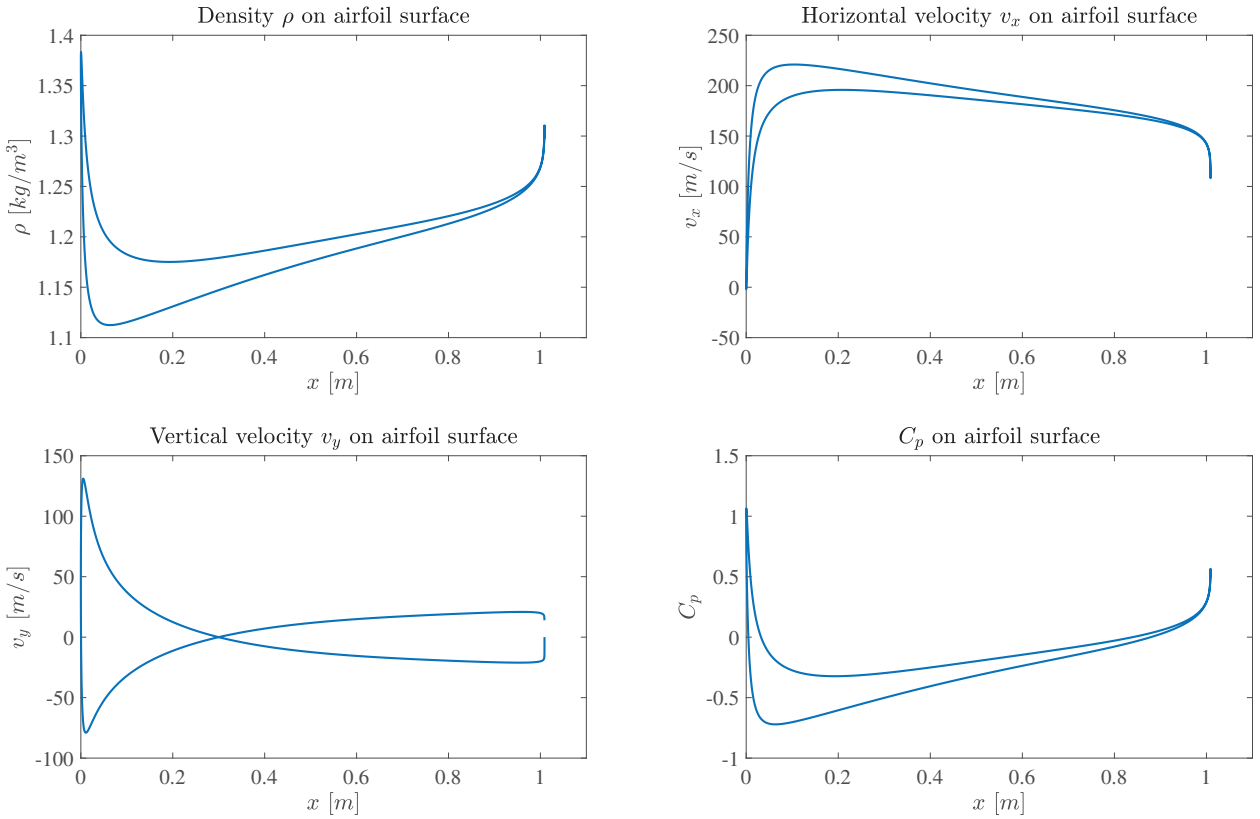


Figure 8: Flow solution on NACA0012 airfoil, $M = 0.5$, $\alpha = 1.25°$, grid 1024×1024.

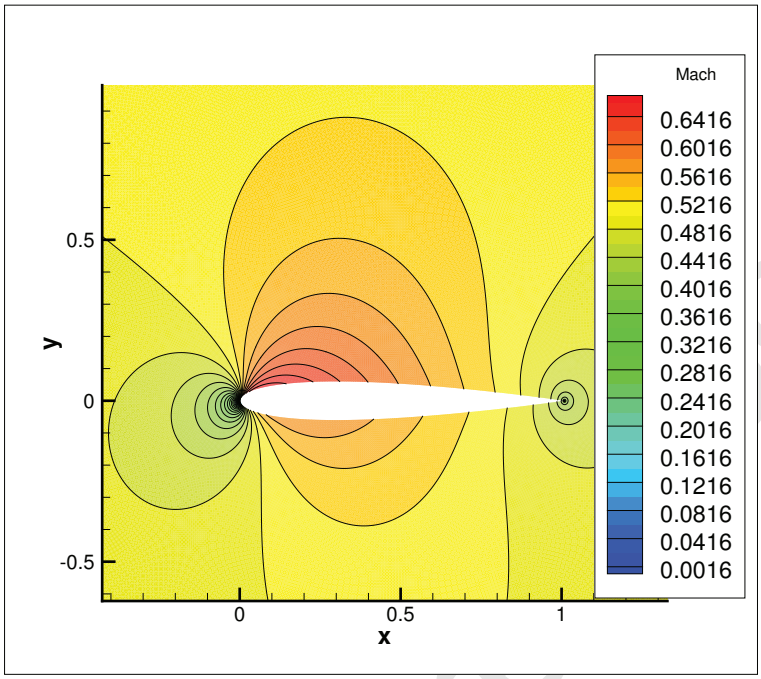


Figure 9: Mach number contours for flow solution on NACA0012 airfoil, $M = 0.5$, $\alpha = 1.25°$, grid 1024×1024, $M_{min} = 0.0016$, $M_{max} = 0.6616$, $\Delta M = 0.02$.

are not known for Euler flow analysis of NACA0012 airfoil. Hence, the best approximation of the continuum values, C_L^* and C_D^*, and the rate of convergence, p, are calculated using Richardson extrapolation of the values of these coefficients on the set of finest, medium and coarse grids, exactly as outlined by Vassberg and Jameson (2010) and Choudhary and Roy (2018). The grid convergence results for C_L and C_D are shown in Figure 10. The ordinate represents $\log_{10}$ of the error in the C_L and C_D values with the corresponding continuum values C_L^* and C_D^*, respectively. The abscissa, $\log_{10}(1/NC)$, is representative of the edge length for the six mesh levels.

The rate of convergence of lift ($p = 3.117$) and drag ($p = 3.557$) for SU2 is higher than for the FLO82 result reported by Vassberg and Jameson (2010). This super-convergence is not well understood. The extrapolated values of lift coefficient, C_L^*, obtained from both solvers are quite close, but there is a mismatch in extrapolated values of drag, C_D^*. The super-convergent lift and drag from SU2, and the mismatch in the C_D^* values may be attributed to the errors in functional evaluations. According to Vassberg and Jameson (2010), even if a

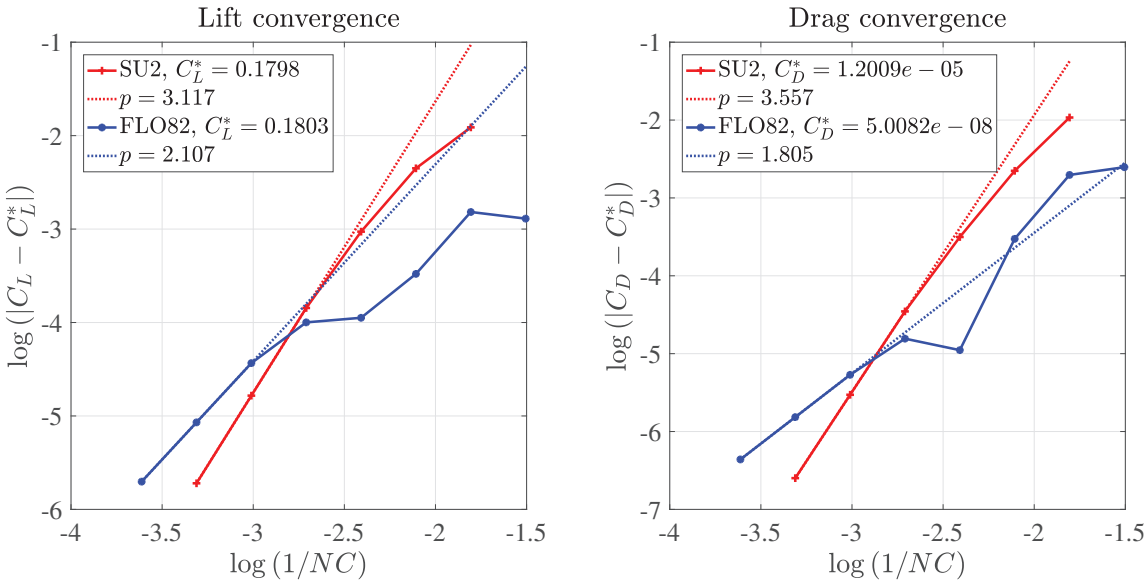


Figure 10: Lift and Drag convergence, $M = 0.5$, $\alpha = 1.25°$.

spatially second-order-accurate flux scheme is used for discretizing the Euler equations, errors in functional evaluations may cause the estimated order of accuracy to vary significantly from the value of two.

4.3. Shape Material Derivatives

The design variable is chosen to be the magnitude of a single Hicks-Henne bump function (Hicks and Henne, 1978) that perturbs the top surface of the airfoil at the mid-chord location, $x_H = 0.5$. Hence the modified top surface of the airfoil for a perturbation Δb is given by

$$y^{\text{top,new}} = y^{\text{top,old}} + \Delta b \sin^3\left(\pi\left(\frac{x}{c}\right)^e\right), \; e = \frac{\log(0.5)}{\log(x_H/c)}, \tag{39}$$

where c is the reference chord length of the airfoil and $x_H = 0.5$ is the location at which the bump is chosen to be located. The old and new airfoil shapes for $\Delta b = 0.01$ are shown in Figure 11. The bottom surface of the airfoil is not altered because of this particular choice of the design variable. Also the top surface gets displaced only in the y (vertical) direction. In general, a shape variable may perturb the boundary in any direction. The airfoil geometry and the design variable defined by the Hicks-Henne bump function (39) are used to derive

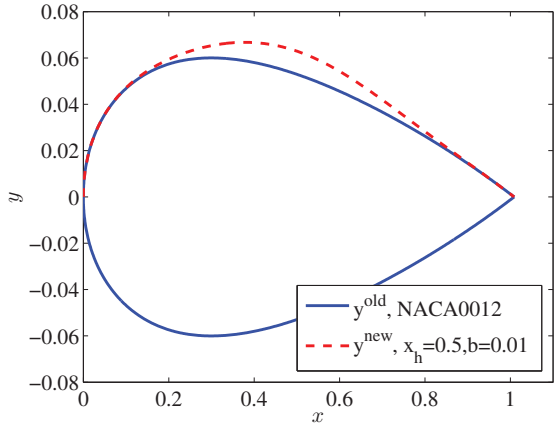


Figure 11: Change in shape of NACA0012 airfoil with design variable b. Note that the axes are scaled to magnify the shape perturbation.

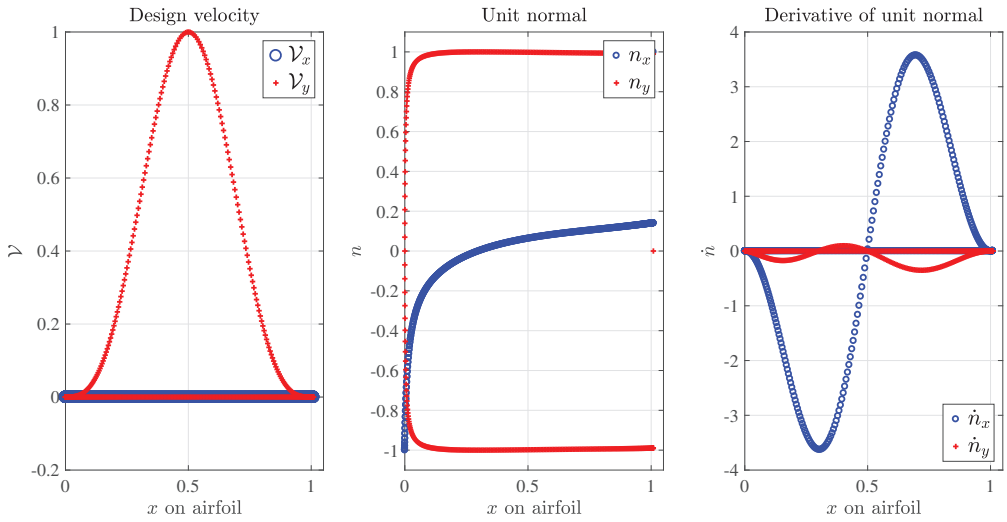


Figure 12: Design velocity, unit normal and derivative of unit normal on the airfoil surface.

the analytic (geometrically exact) design velocity and derivative of the unit normal, plotted in Figure 12. Shape derivatives were computed with CSA for two flow solvers, SU2 and FLUENT, without distinction between their formulations. CSA was agnostic to the vertex- or cell-centered formulation, aside from reconstructing flow variable values at nodes for the cell-centered scheme (FLUENT).

The derivatives of pressure on the airfoil surface were integrated on the airfoil to obtain derivatives of lift and drag coefficients, $\dot{C}_L$ and $\dot{C}_D$, respectively. Rates of convergence of about 1.0 were obtained for both $\dot{C}_L$ and $\dot{C}_D$, as shown in Figure 13. The results obtained

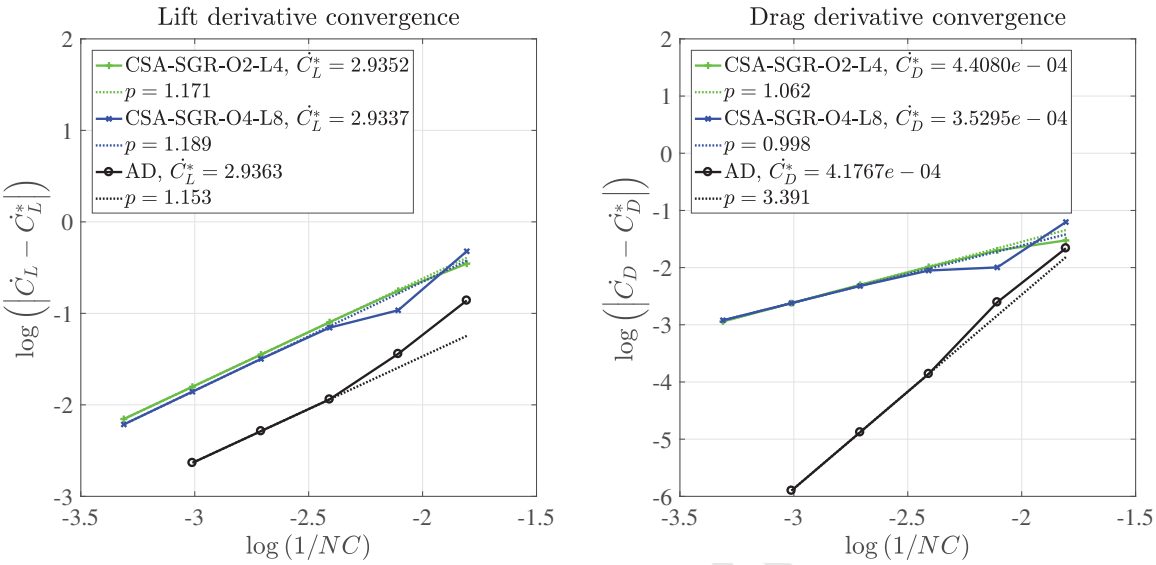


Figure 13: Rates of convergence of the material derivatives of lift and drag coefficients obtained using CSA SGR-O2-L4, CSA SGR-O4-L8, and Automatic Differentiation (AD) method implemented in SU2. left: $\dot{C}_L$, right: $\dot{C}_D$.

from the Automatic Differentiation (AD) method implemented in SU2 are also shown in the figure as a reference. The results for continuum values of the derivatives, $\dot{C}_L^*$ and $\dot{C}_D^*$ from all methods are very close. The rates of convergence of 1.2 for CSA lift derivatives match the rate from AD. The super-convergent rate of 3.39 for drag derivative obtained from AD is not understood. CSA-SGR results are shown for two choices of patches: second-order reconstruction on four layers (SGR-O2-L4) and fourth-order reconstruction on eight layers (SGR-O4-L8), described more fully in a Section 4.4. Although the two cases, SGR-O2-L4 and SGR-O4-L8, differ in terms of spatial gradient accuracy at the expansion point, there is not a considerable difference in the resulting material derivatives. Some possible reasons are explored for why the rate of convergence of the lift and drag material derivatives are close to 1.0, although the rates of convergence of lift and drag are close to 3.0, and the rates of convergence of spatial gradients at the chosen expansion point are close to 2.0. Approximation errors may be introduced at each of the steps shown in Figure 4 while proceeding from the flow solution $\boldsymbol{u}$ to the material derivatives $\dot{\boldsymbol{u}}$. Numerical errors in computing the spatial gradients arise because they have to be reconstructed from the flow variables. We shall show

next that SGR with BC enforcement yields the best spatial gradients. Still, there may be a loss in accuracy based on the variation of the accuracy of the flow variables at individual points in the domain, as discussed in Section 4.4. Also, the weak enforcement of the sensitivity BC (Figure 21) clearly introduces errors in the CSA solution. In the last step, the convective term is added to the local derivative term to obtain the material derivatives, thus accumulating errors in both of these terms. Due to all these factors, it is possible that the requirement for satisfying the theoretical rate of convergence of the lift and drag derivatives is not met (Thomas et al., 2008; Diskin and Thomas, 2010), resulting in convergence rate of only 1.0 for $\dot{C}_L$ and $\dot{C}_D$.

The CSA material derivatives of flow variables on the airfoil surface are plotted in Figure 14. As a reference, the material derivatives were also computed using the forward finite difference method with a step size, $\Delta_b = 0.001$. The CSA derivatives match the finite difference very well. This particular result was obtained using nonintrusive CSA on an unstructured flow domain with SU2 software. Also, for the results shown in Figures 13 and 14, SGR was used with BC enforcement.

To leverage the nonintrusive formulation of CSA, the same problem was solved using the commercial solver FLUENT, which does not allow the output of the flow Jacobian (Kulkarni, 2016). The results of this study are presented in Figure 15. The approach shown in Figure 5 was followed. The mesh used for this study was a quadrilateral mesh with 128×128 cells. The graphs labelled "CSA (Fluent)" represent the CSA derivatives obtained from using FLUENT as the primary solver. Since the FLUENT solver cannot output the Jacobian matrix, it was reconstructed by running SU2 with the converged flow solution from FLUENT. The graphs labelled "CSA (SU2)" represent the CSA derivatives obtained from using SU2 as the primary solver, but for the same mesh. In both the CSA results, SGR was used to approximate the spatial gradients without enforcing the flow tangency BC. The corresponding results for the material derivatives obtained using the forward finite difference method ($\Delta_b = 0.001$) with FLUENT and SU2 solvers are labelled as "FD (Fluent)" and "FD (SU2)," respectively. All

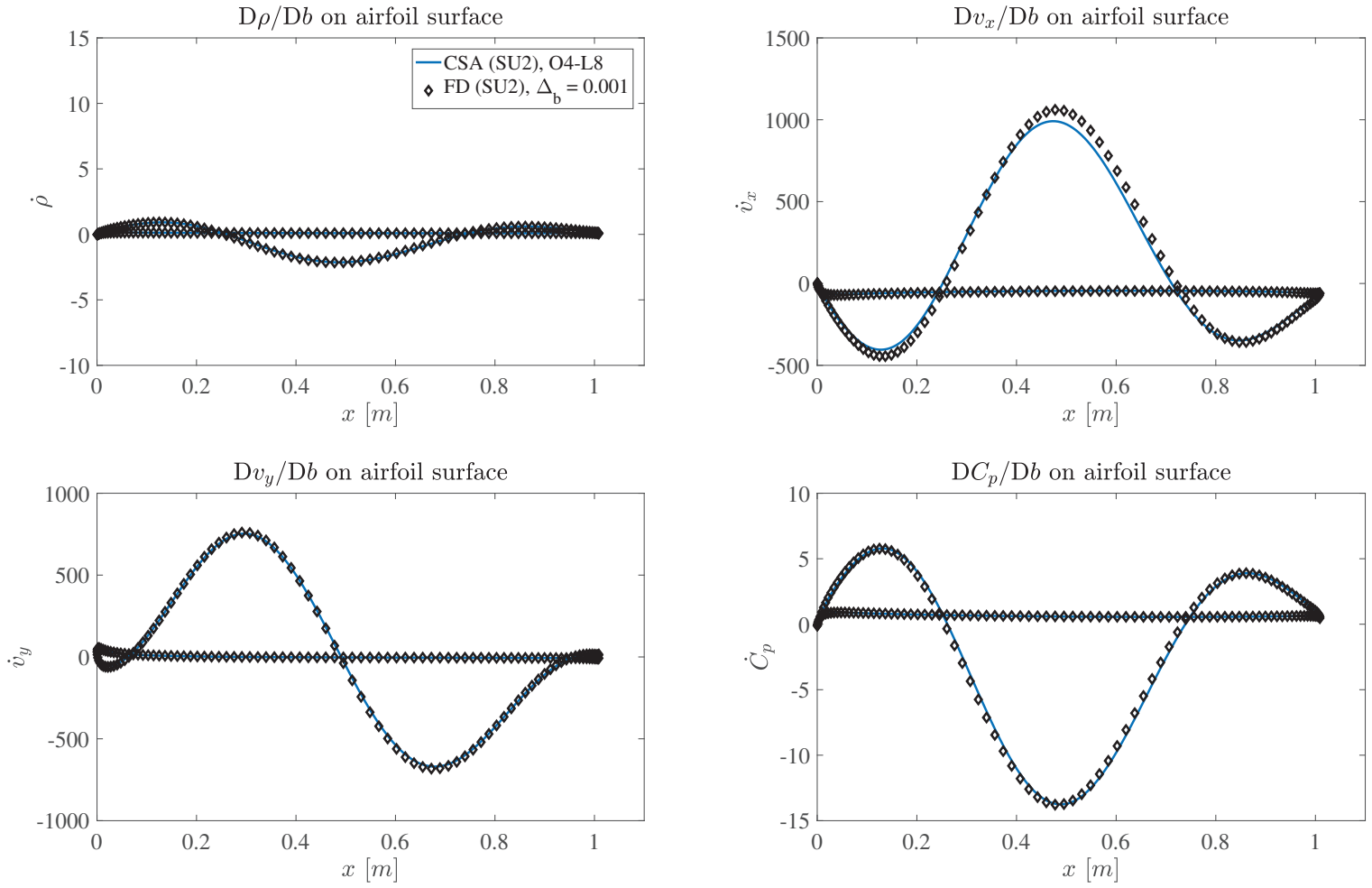


Figure 14: Material derivatives of density, velocities and coefficient of pressure on the airfoil surface obtained using CSA with SGR O4-L8, and finite difference step size $\Delta_b = 0.001$, for the 4×512×512 mesh.

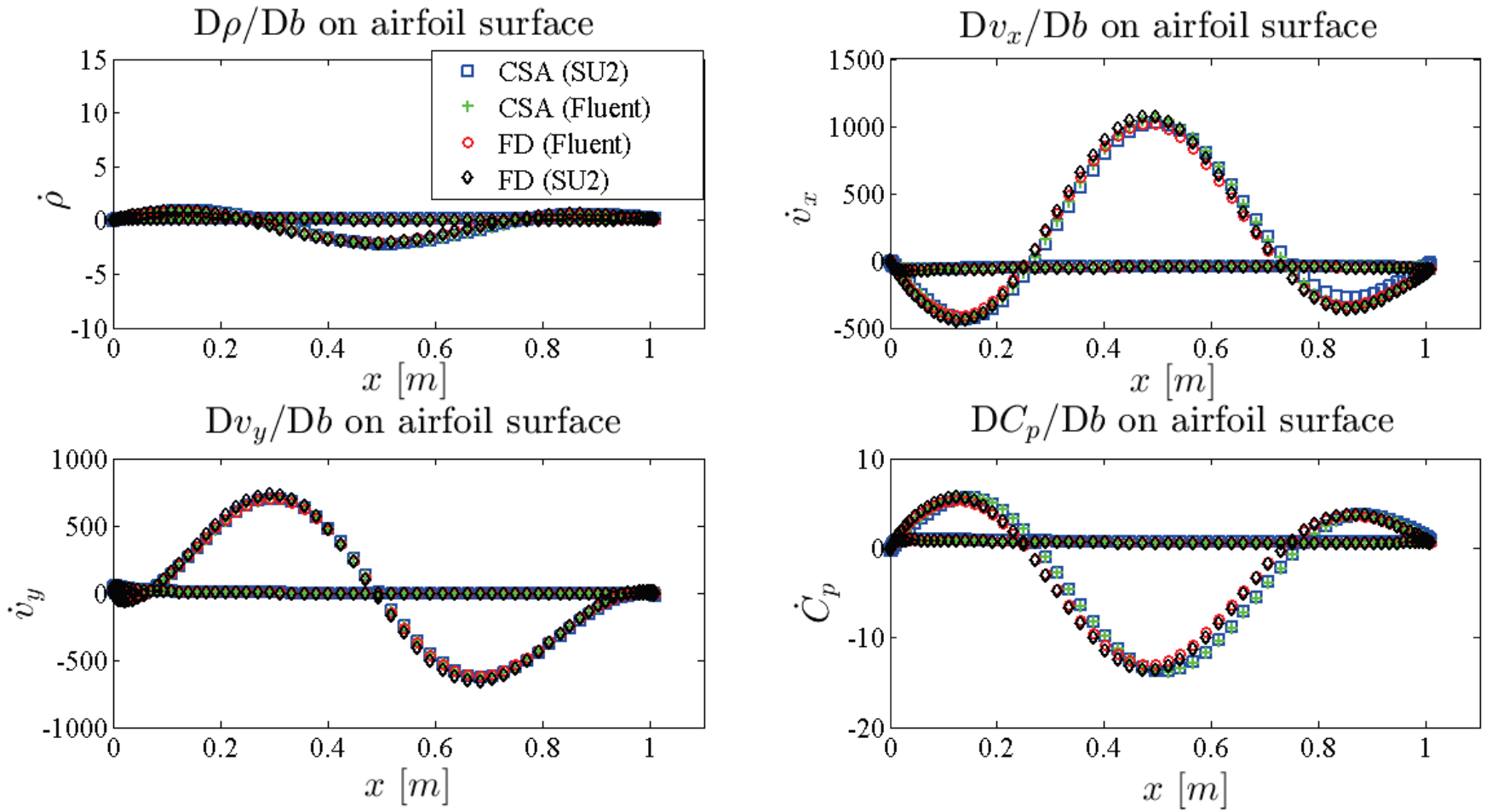


Figure 15: Material derivatives of density, velocities and coefficient of pressure on the airfoil surface obtained using CSA and finite difference, with FLUENT software, which does not allow the output of flow Jacobian.

these results match very well with each other, thus confirming that the nonintrusive CSA approach gives expected results, even if the flow Jacobian cannot be output from the flow solver and is independently reconstructed. This is a major contribution of the presented CSA formulation.

Finally, an advantage of the direct formulation of CSA is that flow material derivatives are obtained at all locations in the domain Ω with a single linear matrix solution, which requires a fraction of the computational cost of the primary analysis. This was done nonintrusively, that is without modifying the analysis code (FLUENT or SU2). An adjoint formulation of CSA (Kulkarni et al., 2016) is also possible, wherein the material derivatives of flow performance measures could be computed efficiently with respect to multiple shape design variables.

Details of how the flow analysis results were post-processed and used to compute sensitivity BCs for CSA are explained next. Results are plotted to gain insight on sources of inaccuracy that may contribute to the observed rates of convergence for CSA.

4.4. Spatial Gradients of Velocity and the CSA Transpiration Boundary Condition

The integrated pressure field converges at a high rate (specifically, C_L converges at a rate of 3.12 and C_D converges at a rate of 3.56), and at each mesh level the L_2 norm of the continuity residual of the steady state solution is less than 10^{-10}. However, this does not guarantee the same level of accuracy or the rates of convergence of flow variables at individual points in the domain. SU2 software uses weak enforcement of the flow tangency boundary condition. As a result of this, the normal velocity, v_n, is not strictly zero on the airfoil boundary. The percentage error in the normal velocity on the airfoil boundary, defined as

$$\epsilon_{v_n} = \frac{v_n}{\max\left(v_t\right)} \times 100, \tag{40}$$

is plotted for the grid 1024×1024 in Figure 16. Although this error decreases with mesh refinement, it is clear that there is significant error in the normal velocity, especially near the

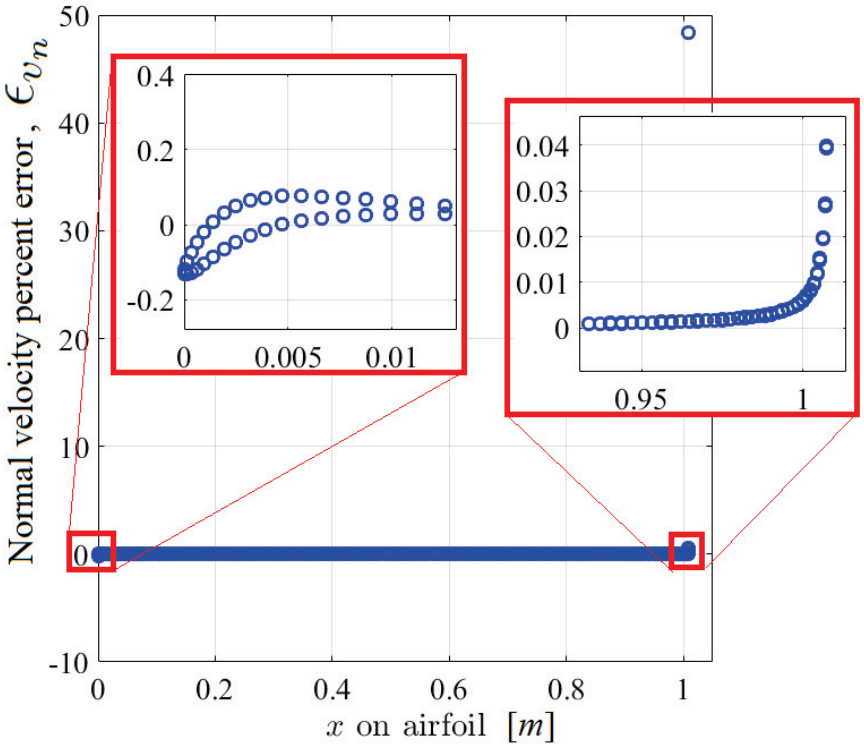


Figure 16: Error in normal velocity, $M = 0.5$, $\alpha = 1.25°$, grid 1024×1024.

leading and trailing edges. This may contribute to the errors in estimated order of accuracy of lift and drag coefficients. As explained in Section 3.3, the accuracy of the local flow derivatives, $\boldsymbol{u}'$, is highly dependent on these boundary terms. SGR is used in an appropriate way to get the most physically correct value of the spatial gradients, as described next.

According to the flowchart, Figure 4, results of the flow analysis and design velocity are used to calculate CSA BCs. Specifically, for the NACA0012 problem considered here, the CSA far-field BC (25) is homogeneous, whereas the CSA transpiration BC (32) is non-homogeneous. The right side of the transpiration BC, defined by g_b, requires spatial gradients of velocities on the airfoil boundary which are calculated using SGR. An SGR patch constitutes a collection of points which are used to get the spatial gradients at an expansion point. Examples of SGR patches are illustrated in Figure 3.

The right side of the transpiration boundary condition can be written in two forms, one with the spatial gradients and velocity components in the (x-y) coordinate directions

(equation 26) and another with the components in the tangential-normal (t-n) directions (equation 28). The discretized version of the (t-n) spatial gradient BC is given by

$$\{\boldsymbol{v'}\}_k \cdot \{\hat{\boldsymbol{n}}\}_k|_{\Gamma_a} = -\left\{ \underbrace{\frac{\partial v_n}{\partial t}}_{From\,SGR} \mathcal{V}_t + \underbrace{\frac{\partial v_n}{\partial n}}_{From\,SGR} \mathcal{V}_n \right\}_k \cdot \{\hat{\boldsymbol{n}}\}_k \Bigg|_{\Gamma_a} - \left\{ \underbrace{v_t}_{From\,SGR} \hat{t} \right\}_k \cdot \left\{\dot{\hat{\boldsymbol{n}}}\right\}_k \Bigg|_{\Gamma_a} . \tag{41}$$

The advantage of using this form, rather than the (x-y) spatial gradient form, is that the flow tangency condition (19) can be imposed during SGR. In other words, the Taylor series (12) of normal velocity v_n at the expansion point k on the airfoil has the constant term zero, since the normal velocity at the airfoil boundary points is zero, $v_n\,(x_k,\, y_k) = 0$

$$\begin{aligned} v_n(x_k^*,\, y_k^*) \;=\; & \underbrace{0}_{v_n(x_k,\, y_k)=0} + \frac{\partial v_n}{\partial t}\Delta t + \frac{\partial v_n}{\partial n}\Delta n \\ + \;& \frac{1}{2}\frac{\partial^2 v_n}{\partial t^2}(\Delta t)^2 + \frac{1}{2}\frac{\partial^2 v_n}{\partial n^2}(\Delta n)^2 + \frac{\partial^2 v_n}{\partial t \partial n}\Delta t \Delta n \\ + \;& \cdots \end{aligned} \tag{42}$$

where $(x_k^*\hat{\imath} + y_k^*\hat{\jmath}) = (x_k\hat{\imath} + y_k\hat{\jmath}) + \left(\Delta\, t\hat{t} + \Delta n\, \hat{n}\right)$. Imposing this constraint is easier when the velocity components are expressed in the (t-n) directions. Another advantage of this form is that the spatial gradients of the tangential velocity, $\frac{\partial v_t}{\partial t}$ and $\frac{\partial v_t}{\partial n}$, are not required (as explained in Section 3.2). Finally, the third advantage is that it allows specifying $v_n\,(x_k,\, y_k) = 0$ also in the $\dot{\hat{\boldsymbol{n}}}$ term in equation (27).

Tangential velocity, v_t, appears in the last term of equation (41). The value of this velocity component obtained from SU2 directly is suspect, since the value of normal velocity, v_n, is not strictly zero on the airfoil boundary, as illustrated in Figure 16. Thus, the use of v_t obtained directly from SU2 might lead to erroneous CSA results. A way to avoid this is by

reconstructing $v_t\,(x_k,\,y_k)$ using SGR based on the Taylor series at the expansion point k

$$\begin{aligned} v_t(x_k^*,\,y_k^*) &= v_t\,(x_k,\,y_k) + \frac{\partial v_t}{\partial t}\Delta t + \frac{\partial v_t}{\partial n}\Delta n \\ &+ \frac{1}{2}\frac{\partial^2 v_t}{\partial t^2}\,(\Delta t)^2 + \frac{1}{2}\frac{\partial^2 v_t}{\partial n^2}\,(\Delta n)^2 + \frac{\partial^2 v_t}{\partial t \partial n}\Delta t \Delta n \\ &+ \cdots \end{aligned} \tag{43}$$

where $(x_k^*\hat{\imath} + y_k^*\hat{\jmath}) = (x_k\hat{\imath} + y_k\hat{\jmath}) + \left(\Delta\, t\hat{t} + \Delta n\,\hat{n}\right)$. The reconstructed value of $v_t\,(x_k,\,y_k)$ obtained in this way is expected to be more accurate than that obtained directly from SU2.

Accuracy of spatial gradients obtained using SGR depends on factors such as SGR patch size, Taylor series (TS) order, and specification of known terms (for example, specifying zero constant term in the Taylor series expansion of v_n (42)). In order to judge the accuracy and rate of convergence of the spatial gradients, SGR patches with different number of layers (L) and orders (O) of Taylor series were analysed. The grid point on the top surface of the airfoil at the maximum thickness location was chosen as the expansion point $(x_0,\,y_0) = (0.2913,\,0.0600)$, as shown in Figure 17. Values of $\frac{\partial v_x}{\partial y}$, $\frac{\partial v_y}{\partial y}$, $\frac{\partial v_n}{\partial t}$ and $\frac{\partial v_n}{\partial n}$ at this point were calculated using SGR and finite difference (FD) methods. Typically, it is not possible to obtain finite difference gradients in an unstructured mesh. However, the six meshes created for this example were modified such that four points directly above the chosen expansion point have the same x coordinate, $x = 0.2913$, as shown in Figure 17. Hence, it was possible to obtain FD y-spatial gradients, $\frac{\partial v_x}{\partial y}$ and $\frac{\partial v_y}{\partial y}$, at the expansion point up to the fourth-order.

The rates of convergence of the velocity components at the chosen expansion point are given in Table 4. The rates of convergence for various orders of reconstruction of the spatial gradients are summarized in Table 5. Finite difference spatial gradients up to the fourth-order are plotted in Figure 18, and SGR spatial gradients for the fourth-order Taylor series (SGR-O4) are shown in Figures 19–20.

Due to the special arrangement of grid points above the expansion point, it was possible

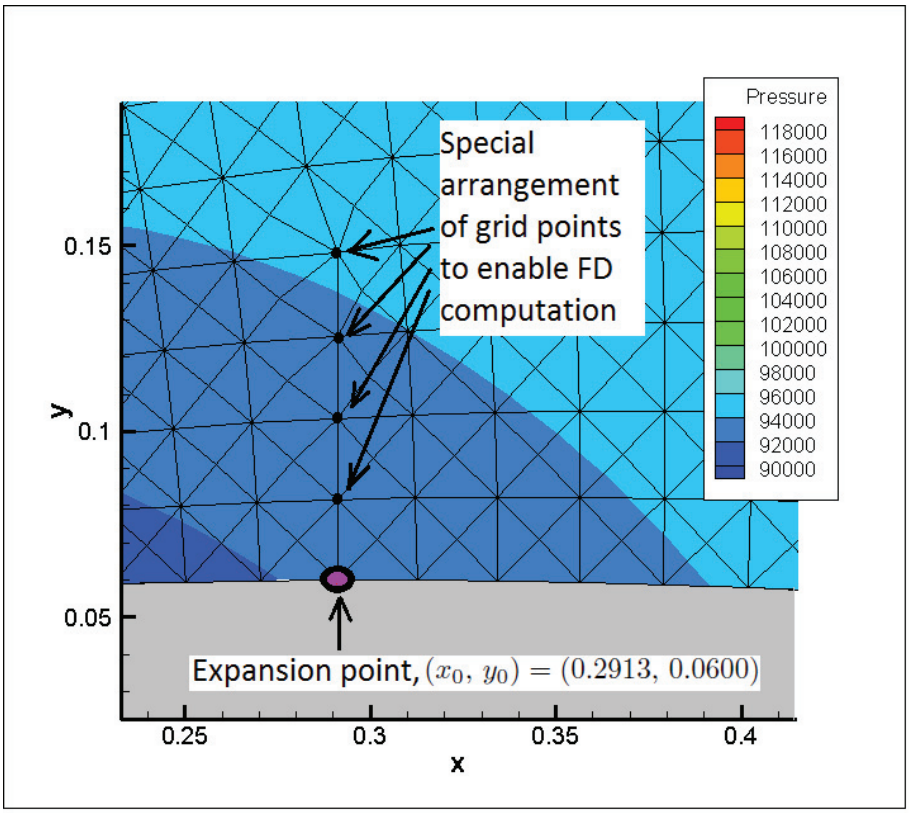


Figure 17: Expansion point on the top surface of the airfoil at the maximum thickness location, grid 256×256, $M = 0.5$, $\alpha = 1.25°$.

Table 4: Rates of convergence of velocity components at the expansion point$(x_0, y_0) = (0.2913, 0.0600)$.

Velocity components	$v_x\,(x_0, y_0)$ $v_x^*\,(x_0, y_0) = 210.31$ m/s	$v_y\,(x_0, y_0)$ $v_y^*\,(x_0, y_0) = 0.83$ m/s
Rate of convergence	2.46	1.70

Table 5: Rates of convergence of spatial gradients of the velocity components at the expansion point$(x_0, y_0) = (0.2913, 0.0600)$. Here O stands for Taylor series order and L stands for patch layers. Maximum rates are shown in red. Outliers are shown in blue. Data that is not monotonic is indicated by parenthesis as (Not mon.)

Spatial gradients	$\frac{\partial v_x}{\partial y}$	$\frac{\partial v_y}{\partial y}$	$\frac{\partial v_n}{\partial t}$	$\frac{\partial v_n}{\partial n}$
	Without enforcement of $v_n\,(x_o,\, y_o) = 0$		**With** enforcement of $v_n\,(x_o,\, y_o) = 0$	
SGR TS order 2 (O2)	L2 > L4 > L6 > L8 1.63 1.50 1.32 1.21	L6 > L8 (L4 & L2) 2.60 2.37 (Not mon.)	L8 = L6 > L4 > L2 1.94 1.94 1.89 1.38	L4 > L6 > L8 (L2) 3.56 2.46 2.33 (Not mon.)
SGR TS order 3 (O3)	L4 > L2 > L6 > L8 1.69 1.62 1.57 1.42	L6 > L8 > L4 > L2 1.47 1.17 0.77 0.54	L8 > L2 > L4 > L6 11.80 2.55 2.48 1.99	L6 > L8 > L4 > L2 1.44 1.10 0.52 0.42
SGR TS order 4 (O4)	L6 > L4 = L8 > L2 1.78 1.65 1.65 1.56	L8 > L6 (L4) 1.93 0.94 (Not mon.)	L8 > L2 > L6 > L4 2.87 2.80 2.79 2.74	L8 > L6 (L4) 2.00 0.61 (Not mon.)
Finite Difference (FD)	O3 > O4 > O2 1.69 1.67 1.64	O4 > O3 > O2 1.44 1.13 0.71	–	–

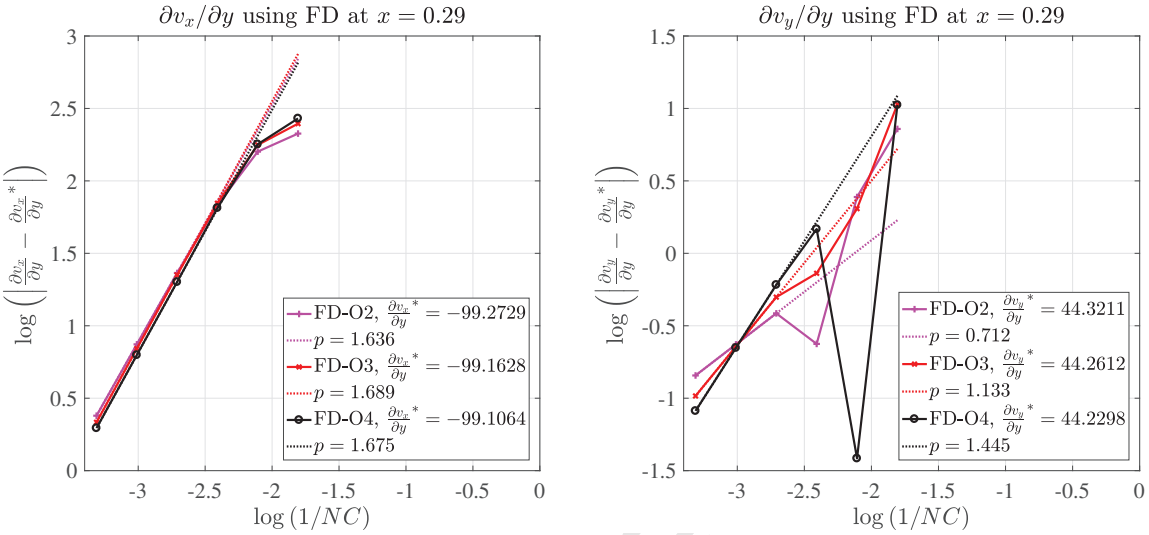


Figure 18: Rates of convergence of the spatial gradients $\frac{\partial v_x}{\partial y}$ and $\frac{\partial v_y}{\partial y}$ at the expansion point $(x_0, y_0) = (0.2913, 0.0600)$ obtained using finite difference (FD), left: $\frac{\partial v_x}{\partial y}$, right: $\frac{\partial v_y}{\partial y}$.

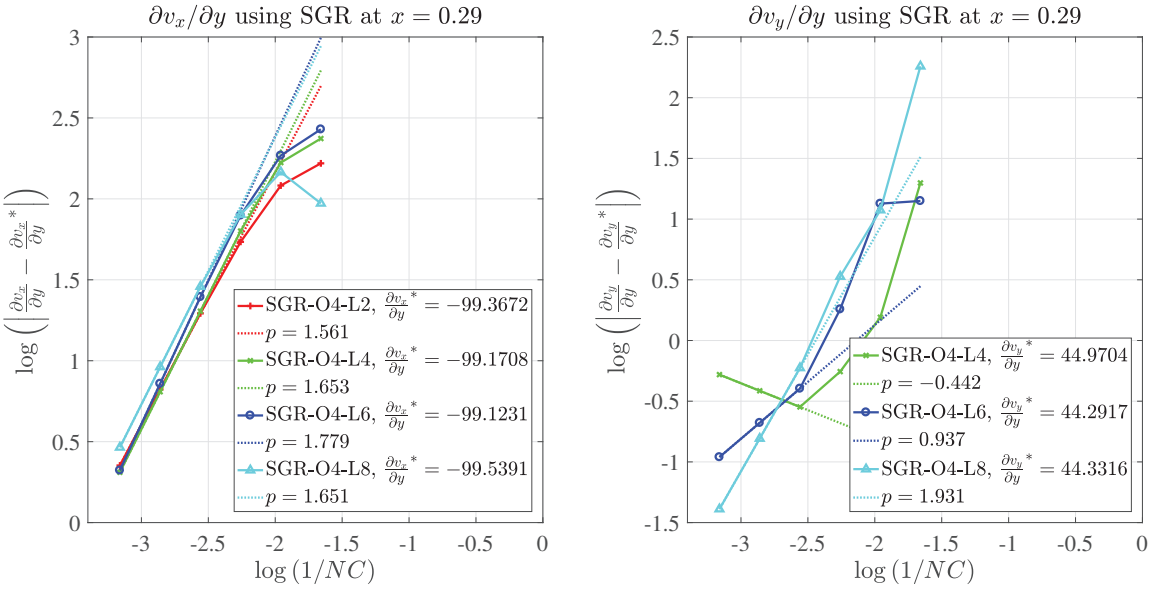


Figure 19: Rates of convergence of the spatial gradients $\frac{\partial v_x}{\partial y}$ and $\frac{\partial v_y}{\partial y}$ at the expansion point $(x_0, y_0) = (0.2913, 0.0600)$ obtained using SGR TS O4, left: $\frac{\partial v_x}{\partial y}$, right: $\frac{\partial v_y}{\partial y}$.

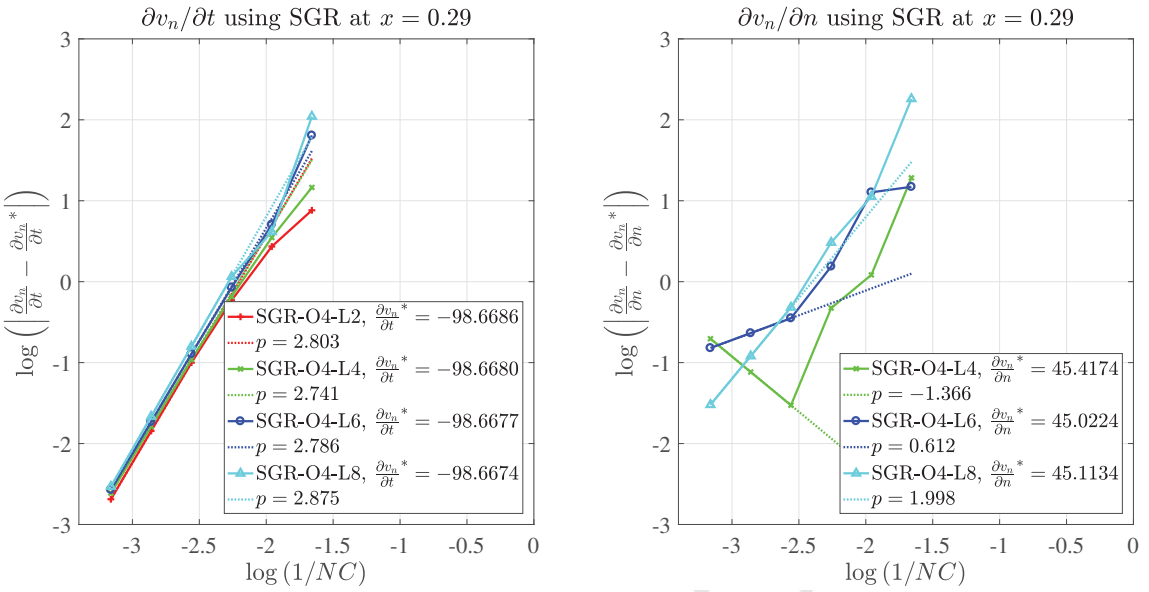


Figure 20: Rates of convergence of the spatial gradients $\frac{\partial v_n}{\partial t}$ and $\frac{\partial v_n}{\partial n}$ at the expansion point $(x_0, y_0) = (0.2913, 0.0600)$ obtained using SGR TS O4 , left: $\frac{\partial v_n}{\partial t}$, right: $\frac{\partial v_n}{\partial n}$.

to obtain the values of spatial gradients $\frac{\partial v_x}{\partial y}$ and $\frac{\partial v_y}{\partial y}$ at the expansion point using FD. The theoretical rates of convergence were not achieved. Figure 18 indicates that the best rate of convergence for $\frac{\partial v_x}{\partial y}$ is 1.69 (FD-O3) and for $\frac{\partial v_y}{\partial y}$ is 1.44 (FD-O4). Possible reasons for this discrepancy could be less accurate or lower-order rates of convergence of velocity components at the FD data points. For example, at the expansion point, the horizontal velocity component v_x converges at the rate of 2.46 and the vertical velocity component v_y converges at the rate of 1.70 (Table 4). Although the SU2 code is run using a second-order spatial flux scheme, the reason for a sub-second-order rate of convergence of v_y at the expansion point is not clear. One possible source of error is the weak enforcement of flow tangency boundary condition that results in nonzero normal flow velocity at the airfoil boundary (as shown in Figure 16). These results suggest that inaccuracies in the flow analysis will be carried forward in the CSA process.

On the other hand, when SGR is used to obtain the same spatial gradients, slightly better rates of convergence are obtained (Figure 19 and Table 5). The best rate of convergence for $\frac{\partial v_x}{\partial y}$ with SGR is 1.78 with O4-L6, while the best rate of convergence for $\frac{\partial v_y}{\partial y}$ with SGR

is 2.60 with O2-L6 (Table 5). However, the SGR for $\frac{\partial v_x}{\partial y}$ and $\frac{\partial v_y}{\partial y}$ did not enforce the flow tangency boundary condition. In some cases, such as O4-L4, the SGR gradients obtained are not monotonic. This may be attributed to errors in the velocity data at the grids points of the SGR patch.

SGR yielded better results when the flow tangency BC was enforced (Figure 20 and Table 5). The best rate of convergence for $\frac{\partial v_n}{\partial t}$ with SGR is 2.88 with O4-L8, while the best rate of convergence for $\frac{\partial v_n}{\partial n}$ with SGR is 3.56 with O2-L4 (Table 5). In general, SGR with BC enforcement yields more accurate spatial gradients that have rates of convergence of about 2. This is attributed to BC enforcement that imposes the inherent physics of the problem, thus reducing numerical errors. Still, a clear trend in the rate or convergence with the number of layers or the order of Taylor series is not evident.

Spatial derivatives obtained from SGR were used to calculate the transpiration velocity g_b at the airfoil boundary. Based on the results of spatial derivatives mentioned above, two cases were selected: Case 1 SGR-O2-L4, which gave the best rate of convergence with a lower order Taylor series (convergence rate of 1.89 for $\frac{\partial v_n}{\partial t}$ and convergence rate of 3.56 for $\frac{\partial v_n}{\partial n}$), and Case 2 SGR-O4-L8 which gave the best rate of convergence with a higher-order Taylor series (convergence rate of 2.87 for $\frac{\partial v_n}{\partial t}$ and convergence rate of 2.00 for $\frac{\partial v_n}{\partial n}$). Although this choice was made by observing the rates of convergence of spatial gradients at one specific expansion point $(x_0, y_0) = (0.2913, 0.0600)$, the transpiration boundary condition (32) requires spatial gradients at all points on the airfoil boundary Γ_a. Hence the accuracy of the spatial gradients at all the boundary points affects the accuracy of the material derivatives. Material derivatives of flow variable with respect to the chosen shape design variable for Case 1 and Case 2 were shown and discussed in Section 4.3.

4.5. Sensitivity Boundary Conditions

The accuracy of spatial gradients directly affects the accuracy of the transpiration BC. This boundary condition is imposed weakly in the finite volume method (Hirsch, 1990) as explained in Section 3.3. As a result, the CSA transpiration velocity, g_{b_k} (equations 30-32),

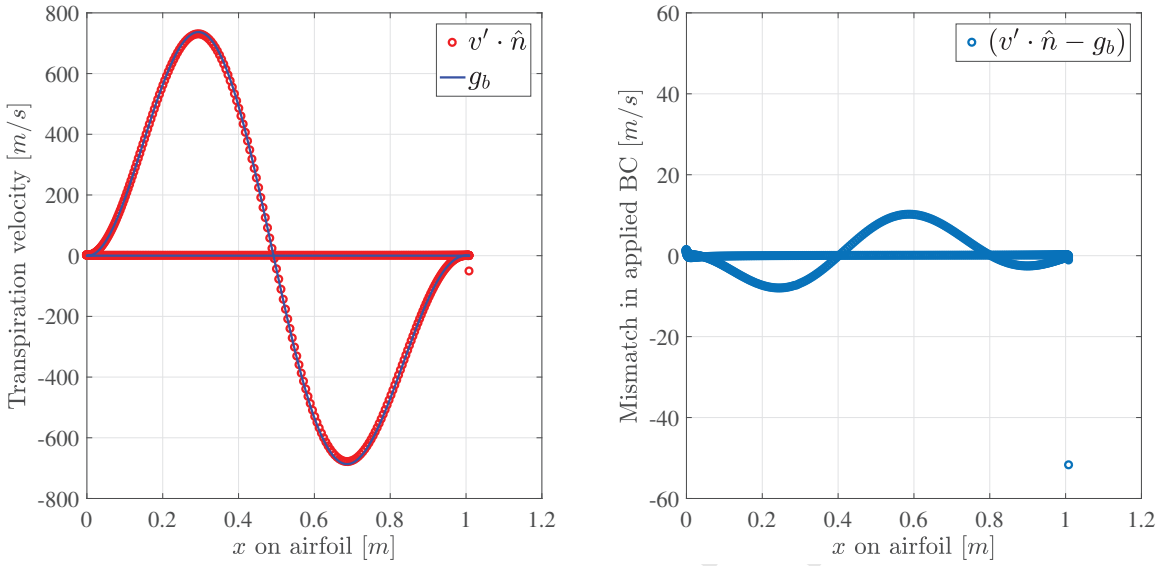


Figure 21: Transpiration velocity on the airfoil calculated using SGR O4-L8 for the 4×512×512 mesh. The right figure shows the mismatch in the applied and the realized transpiration BC.

is applied exactly only in the continuum limit, that is, the accuracy with which the BC is imposed increases with increase in the number of cells. The value of g_b that was used to impose the BC and the value of the term $(\boldsymbol{v'} \cdot \hat{\boldsymbol{n}})$ from the CSA solution are shown in Figure 21. If the transpiration BC was applied strongly, these two terms would be the same. However, it is clear from the mismatch seen in the figure that the transpiration BC is not applied accurately. The reason for this is the weak BC enforcement.

5. Conclusion

A direct formulation of CSA was used for calculating the material derivatives of flow parameters with respect to a shape design parameter. The focus of this work was compressible flow over an airfoil obtained through the solution of Euler equations. Most of the previous work in the area of shape derivatives of compressible flows using CSA has been done using finite difference or finite element discretizations. The challenges that arise in implementing sensitivity boundary conditions when a finite volume discretization is used are highlighted. One known issue is that the accuracy of sensitivity boundary conditions is compromised due to errors in approximating the spatial gradients of flow variables. Use of Spatial Gradient

Reconstruction (SGR) in the tangential-normal coordinate system was shown to improve the accuracy of spatial gradients, and consequently enhanced the accuracy of the material derivatives. To illustrate this, an example of flow over a NACA0012 airfoil was presented that highlighted the effect of the accuracy of the sensitivity boundary conditions on the derivatives of lift and drag. The spatial gradients of flow velocities, calculated using SGR, contribute significantly to the transpiration sensitivity boundary condition and thus affect the accuracy of material derivatives of the flow variables. Weak enforcement of boundary conditions, typical of finite volume formulations, leads to errors in the solution to the sensitivity equations. This may contribute to convergence rates of flow derivatives obtained using CSA being lower than the convergence rate of the flow analysis itself. Nevertheless, accuracy of CSA was comparable to that calculated using the automatic differentiation method and the finite difference method.

Another contribution is that the current sensitivity analysis method works well even if the domain is discretized using an unstructured grid. For unstructured domains it is impossible to use finite difference method for approximating the flow spatial gradients. This justifies the use of SGR, because it handles unstructured meshes as easily as structured meshes.

Finally, the current method is a nonintrusive sensitivity analysis method. That is, it may be used without modifying the flow analysis source code. Consequently, it can be implemented in "black-box" fashion with commercial software. In particular, CSA was demonstrated for commonly used CFD codes, FLUENT and SU2, which use finite volume discretization. For the NACA0012 airfoil example, CSA derivatives were calculated using the flow solution from both the SU2 and FLUENT solvers and a Jacobian matrix exported from the SU2 solver. The CSA algorithm was the same whether the flow solution originated from SU2 or FLUENT; that is, CSA was agnostic to the particular formulation used to solve the Euler flow equations. The CSA results match well with the corresponding finite difference results.

Acknowledgements

The material in this dissertation is based on research sponsored by Air Force Research Laboratory (AFRL) under agreement number FA8650-09-2-3938 and by Air Force Office of Scientific Research (AFOSR) under agreement number FA9550-16-1-0125DEF. Authors gratefully acknowledge the support of AFRL Senior Aerospace Engineers Dr. Raymond Kolonay, Dr. Philip Beran and Dr. Edward Alyanak, AFOSR Program Officer Dr. Jean-Luc Cambier, and director of the Virginia Tech Collaborative Center for Multidisciplinary Sciences Dr. Rakesh Kapania.

Bibliography

AIAA, June 2009. 4th aiaa cfd drag prediction workshop. San Antonio, TX. 30

Albring, T., Sagebaum, M., Gauger, N. R., 2016. A Consistent and Robust Discrete Adjoint Solver for the SU2 Framework - Validation and Application. Springer International Publishing, Cham, pp. 77–86.
URL http://dx.doi.org/10.1007/978-3-319-27279-5_7 4

Anderson, W. K., Bonhaus, D. L., 1994. An implicit upwind algorithm for computing turbulent flows on unstructured grids. Computers and Fluids 23 (1), 1–21. 30

Becker, K., Ashcroft, G., January 2014. A comparative study of gradient reconstruction methods for unstructured meshes with application to turbomachinery flows. In: 52nd AIAA Aerospace Sciences Meeting, National Harbor, MD, USA. 5

Borggaard, J., Burns, J., 1994. A sensitivity equation approach to shape optimization in fluid flows. Nasa contractor report 191598, Langley Research Center. 3, 5, 12

Borggaard, J., Burns, J., 1997. A pde sensitivity equation method for optimal aerodynamic design. Journal of Computational Physics. 136, 366–384. 3, 5, 12, 19, 26

Charlot, L., Etienne, S., Pelletier, D., 2012. A continuous lagrangian sensitivity equation method for incompressible flow. Journal of Computational Physics 231, 5989–6011. 3

Chen, P. C., Zhang, Z., April 2013. Store ejection loads analysis using an euler solver. In: 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. AIAA, Boston, MA. 26

Choi, K. K., Kim, N.-H., 2005. Structural sensitivity analysis and optimization. Springer Science+Business media, New York. 3

Choudhary, A., Roy, C. J., 2018. Verification and Validation for Multiphase Flows. Springer Singapore, Singapore, pp. 1–37.
URL https://doi.org/10.1007/978-981-4585-86-6_24-1 33

Cross, D. M., Canfield, R. A., September, 17-19 2012. Continuum shape sensitivity with spatial gradient reconstruction of nonlinear aeroelastic gust response. In: 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. No. AIAA 2012-55597. Indianapolis, Indiana. 19

Cross, D. M., Canfield, R. A., December 2014. Local continuum shape sensitivity with spatial gradient reconstruction. Structural and Multidisciplinary Optimization 50 (6), 975–1000. 3, 10, 14

Cross, D. M., Canfield, R. A., 2016. Convergence study of local continuum sensitivity method using spatial gradient reconstruction. AIAA Journal.
URL http://dx.doi.org/10.2514/1.J053800 15

Diskin, B., Thomas, J. L., Mar. 2010. Notes on accuracy of finite-volume discretization schemes on irregular grids. Appl. Numer. Math. 60 (3), 224–226.
URL http://dx.doi.org/10.1016/j.apnum.2009.12.001 37

Duvigneau, R., Pelletier, D., 2006. On accurate boundary conditions for a shape sensitivity equation method. International Journal for Numerical Methods in Fluids. 50 (2), 147–164. 3, 5, 14, 26

Gay, D. M., 2005. Semiautomatic differentiation for efficient gradient computations. In: Bücker, H. M., Corliss, G., Hovland, P., Naumann, U., Norris, B. (Eds.), Automatic Differentiation: Applications, Theory, and Implementations. Lecture Notes in Computational Science and Engineering. Springer, pp. 147–158. 4

Gobal, K., Grandhi, R. V., Kolonay, R. M., 2015. Continuum sensitivity analysis for struc-

tural shape design variables using finite-volume method. AIAA Journal 53 (2), 347–355. 3

Godfrey, A. G., Cliff, E. M., January 8-11 2001. Sensitivity equations for turbulent flows. In: 39th AIAA Aerospace Sciences Meeting & Exhibit. No. 2001-1060. Reno, NV. 5, 17

Hicks, R. M., Henne, P. A., 1978. Wing design by numerical optimization. Journal of Aircraft 15 (7), 407–412.
URL http://dx.doi.org/10.2514/3.58379 34

Hirsch, C., 1990. Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows. John Wiley and Sons. 4, 6, 26, 27, 48

Hogan, R. J., Jul. 2014. Fast reverse-mode automatic differentiation using expression templates in c++. ACM Transactions on Mathematical Software 40 (4), 26:1–26:16.
URL http://doi.acm.org/10.1145/2560359 4

Jameson, A., 1983. Solution of the euler equations for two dimensional transonic flow by a multigrid method. Applied Mathematics and Computation 13 (3), 327 – 355.
URL http://www.sciencedirect.com/science/article/pii/009630038390019X 30

Kulkarni, M. D., 2016. Continuum sensitivity analysis using boundary velocity formulation for shape derivatives. Ph.D. thesis.
URL http://hdl.handle.net/10919/73057 30, 37

Kulkarni, M. D., Canfield, R. A., Patil, M. J., June 2014a. Nonintrusive continuum sensitivity analysis for aerodynamic shape optimization. In: 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. No. AIAA 2014-2043. AIAA, Atlanta, GA.
URL http://dx.doi.org/10.2514/6.2014-2043 6

Kulkarni, M. D., Canfield, R. A., Patil, M. J., June 2015. Nonintrusive continuum sensitivity analysis for aerodynamic shape optimization. In: 16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. No. AIAA 2014-3237. AIAA, Dallas, TX.
URL http://dx.doi.org/10.2514/6.2015-3237 6

Kulkarni, M. D., Canfield, R. A., Patil, M. J., Alyahak, E. J., January 2014b. Integration of geometric sensitivity and spatial gradient reconstruction for aeroelastic shape optimization. In: 10th AIAA Multidisciplinary Design Optimization Conference. No. AIAA 2014-0470. AIAA, National Harbor, MD.
URL http://dx.doi.org/10.2514/6.2014-0470 15

Kulkarni, M. D., Cross, D. M., Canfield, R. A., 2016. Discrete adjoint formulation for continuum sensitivity analysis. AIAA Journal 54 (2), 758–766.
URL http://arc.aiaa.org/doi/10.2514/1.J053827 9, 40

Liu, S., Canfield, R. A., 2013a. Boundary velocity method for continuum shape sensitivity of nonlinear fluid structure interaction problems. Journal of Fluids and Structures 40, 284 – 301.
URL http://www.sciencedirect.com/science/article/pii/S0889974613001199 6

Liu, S., Canfield, R. A., 2013b. Equivalence of continuum and discrete analytic sensitivity methods for nonlinear differential equations. Structural and Multidisciplinary Optimization 48 (6), 1173–1188.
URL http://dx.doi.org/10.1007/s00158-013-0951-4 10, 12

Liu, S., Canfield, R. A., 2016. Two forms of continuum shape sensitivity method for fluid structure interaction problems. Journal of Fluids and Structures 62, 46 – 64.
URL http://www.sciencedirect.com/science/article/pii/S0889974616000104 2, 9, 10

Palacios, F., Michael R. Colonno, M. R., Aranake, A. C., Campos, A., Copeland, S. R.,

Economon, T. D., Lonkar, A. K., Lukaczyk, T. W., Taylor, T. W. R., J., A. J., January, 7-10 2013. Stanford university unstructured (su2): An open-source integrated computational environment for multi-physics simulation and design. In: 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition. No. AIAA 2013-0287. Grapevine, Texas. 4, 26

Pelletier, D., Hay, A., Etienne, S., Borggaard, J., 2008. The sensitivity equation method in fluid mechanics. European Journal of Computational Mechanics/Revue Europeenne de Mecanique Numerique 17 (1-2), 31–61. 3

Thomas, J. L., Diskin, B., Rumsey, C. L., 2008. Towards verification of unstructured-grid solvers. AIAA journal 46 (12), 3070–3079. 37

Vassberg, J. C., Jameson, A., August 2010. In pursuit of grid convergence for two-dimensional euler solutions. Journal of Aircraft 47 (4), 1152–1166. 30, 31, 33

Villa, A., 2009. Convergence of weakly imposed boundary conditions: The one-dimensional hyperbolic case. SIAM Journal on Scientific Computing 31 (4), 3116–3127.
URL http://dx.doi.org/10.1137/080720516 4

Villa, A., Barbieri, L., Malgesini, R., 2012. Ghost cell boundary conditions for the euler equations and their relationships with feedback control. Communications in Applied and Industrial Mathematics 3 (1).
URL http://openjournal.simai.eu/index.php/caim/article/view/394 4

Zhang, Z., Liuy, F., Schusterz, D. M., 2006. An effcient euler method on non-moving cartesian grids with boundary-layer correction for wing flutter simulations. In: Aerospace Sciences Meeting. Reno, Nevada. 26

Zienkiewicz, O. C., Zhu, J. Z., 1992. The superconvergent patch recovery (spr) and adaptive finite element refinement. Computer Methods in Applied Mechanics and Engineering 101 (1), 207–224. 14

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: