Fourth Mississippi State Conference on Differential Equations and
Computational Simulations, Electronic Journal of Differential Equations,
Conference 03, 1999, pp 13–27.
http://ejde.math.swt.edu or http://ejde.math.unt.edu
ftp ejde.math.swt.edu or ejde.math.unt.edu (login: ftp)

Use of discrete sensitivity analysis to transform explicit simulation codes into design optimization codes *

Clarence O. E. Burg

Abstract

Sensitivity analysis is often used in high fidelity numerical optimization to estimate design space derivatives efficiently. Typically, explicit codes are combined with the adjoint formulation of continuous sensitivity analysis, which requires the derivation and solution of the adjoint equations along with appropriate boundary conditions. However, for implicit codes, which already calculate the Jacobian matrix of the discretized governing equations, the discrete approach of sensitivity analysis is relatively easy to implement. Using the complex Taylor's series expansion method to generate derivatives, a highly accurate approximation to the Jacobian matrix can be generated for implicit or explicit codes, allowing uniform application of discrete sensitivity analysis to both implicit and explicit codes.

1 Introduction

High fidelity numerical simulation codes have been developed to solve a wide variety of partial differential equations, including the Euler and Navier-Stokes equations for compressible or incompressible flow, the shallow water equations for flow in the ocean, rivers and channels and the porous media equation for groundwater flow in underground aquifers. The primary goal of these codes has been to simulate the flow for a particular geometry and/or parameter distribution. The codes are used in conjunction with scale models and field measurements for evaluation and analysis of a particular design. Then, the design or the parameters are changed based on the knowledge and experience of the researcher, and a new simulation is run. This trial-and-error approach to design is inefficient and does not take full advantage of the computational tools.

By changing the simulation code into a gradient-based optimization code, the results from the flow simulation are used to direct the search and can efficiently

* *Mathematics Subject Classifications:* 76N25, 49Q12.
Key words: design optimization, sensitivity analysis, adjoint methods, computational fluid dynamics.

Published July 10, 2000.

locate much improved designs. For instance, in the inverse design of airfoils, the designer determines the desired pressure distribution based on certain design criteria and uses an Euler or Navier-Stokes code to find an airfoil whose pressure distribution closely matches the target distribution. Thus, the objective function to be minimized could be a least squares function between the target pressures and the actual pressures for a particular design, and the design parameters would be the shape of the airfoil and the flight conditions, such as Mach number and angle of attack. By varying these parameters, the designer hopes to minimize the objective function. For notation, $\vec{\beta}$ is the vector of design parameters, $F(\vec{\beta})$ is the objective function, $\frac{\partial F}{\partial \beta_i}$ is the design space derivative of F with respect to β_i and $\nabla_{\vec{\beta}} F$ is the design space gradient of F with respect to the vector of design variables $\vec{\beta}$.

These simulation codes can be used to estimate design space derivatives by using one-sided finite differences

$$\frac{\partial F(\vec{\beta})}{\partial \beta_i} \approx \frac{F(\vec{\beta} + e_i \Delta\beta) - F(\vec{\beta})}{\Delta\beta} \tag{1}$$

or central differences

$$\frac{\partial F(\vec{\beta})}{\partial \beta_i} \approx \frac{F(\vec{\beta} + e_i \Delta\beta) - F(\vec{\beta} - e_i \Delta\beta)}{2\Delta\beta} \tag{2}$$

where e_i is a unit vector in the ith direction. These finite difference formulae for estimating the design space gradient are easy to implement. However, for one-sided finite differences, an N additional steady-state simulations are required where N is the number of design parameters, and when central differences are used, an additional $2N$ steady-state simulations are needed. Since a high-fidelity flow solver is being used to calculate the steady-state flow variables, the computational cost of evaluating the objective function is quite large. Thus, the finite difference method is computationally expensive. Furthermore, the accuracy of the method is highly dependent on the size of the perturbation $\Delta\beta$. If the perturbation is too large, the non-linearity of the objective function will dominate; if it is too small, round-off error reduces the accuracy of the derivative. Finally, if the solutions are not strongly converged, then the accuracy of the function values will be limited. In this case, the function values being used in these formulae must be taken from identical locations in the solution process to have any chance of being accurate.

Other methods being investigated to generate numerically exact derivatives include automatic differentiation such as ADIFOR and the complex Taylor's series expansion method. These two methods are relatively easy to implement but are approximately as computationally expensive as finite differences, although current research efforts are being investigated to reduce their computational cost for steady-state problems. Table 1 shows a comparison of these methods in regards to their computational cost, accuracy of derivative and ease of implementation.

Derivative Method	Cost of Comput.	Accuracy of Derivative	Ease of Implement.
Finite Difference	Expensive	Moderate	Very Easy
Automatic Differentiation	Expensive	Num. Exact	Easy
Complex Taylor's Series Expansion Method	Expensive	Num. Exact	Easy
Continuous Sensitivity Analysis for Explicit Codes	Cheap	Moderate	Very Difficult
Discrete Sensitivity Analysis for Implicit Codes	Cheap	Highly	Moderate
Discrete Sensitivity Analysis for Explicit Codes	Cheap	Moderate	Difficult

Table 1. Costs and Benefits of Various Methods to Generate Design Space Derivatives.

Sensitivity analysis can be used to reduce the computational cost of estimating the design space gradient, by taking advantage of derivative information obtained via differentiation of the governing system of equations. Sensitivity analysis can be divided into two different approaches - the continuous approach and the discrete approach. In the continuous approach, the system of governing differential equations is differentiated to form a separate set of continuous adjoint equations. In the discrete approach, the system of governing equations is discretized first, creating a system of discretized equations $W(Q) = 0$. These discretized equations are differentiated to form a system of discrete adjoint equations. For an overview of sensitivity analysis as applied to complex aerodynamic applications, see Newman, etal [3]. Figure 1 gives a breakdown of the various formulations within sensitivity analysis. The adjoint formulations of both continuous and discrete sensitivity analysis are typically used due to the need to calculate the adjoint variable once per objective function regardless of the number of design variables. Each formulation can be implemented within an explicit or an implicit code; however, traditionally, discrete sensitivity analysis has been used for implicit codes while continuous sensitivity analysis has been used for explicit codes.

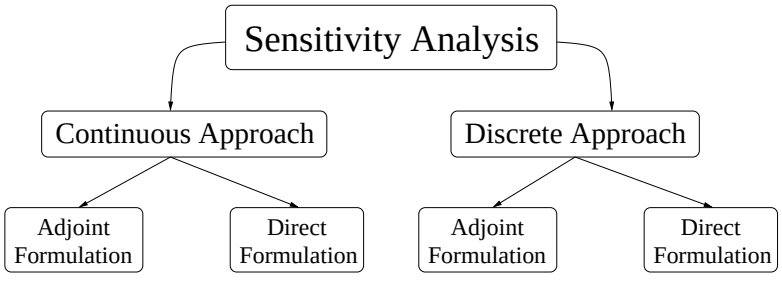

Figure 1. Breakdown of Formulations within Sensitivity Analysis.

For explicit codes, the continuous adjoint equations can be discretized and solved in any consistent fashion; however, Shubin and Frank [6] showed that to achieve

the best agreement between the derivatives produced via the continuous approach with the finite difference derivatives, the same discretization method should be employed to solve the continuous adjoint equations as was used to solve the continuous governing equations. Reuther [5], in his dissertation research, demonstrated these techniques for the two-dimensional Euler equations as applied to airfoil design. Once the adjoint equations are derived and successfully implemented, they are driven to steady-state, which takes approximately twice as long as a flow simulation, according to Reuther. Since the adjoint equations must be solved only once for each objective function regardless of the number of design variables and since the number of objective functions is often quite small in comparison to the number of design variables for single discipline designs, the computational savings of using the adjoint formulation as opposed to finite differences can be quite large.

More recently, Nadarajah and Jameson [2] showed that the discrete approach of sensitivity analysis could be applied to explicit codes by differentiating the discretized equations solved via the explicit code. However, in his derivations, Nadarajah froze certain terms to reduce the complexity of the differentiated equations, which resulted in a loss of accuracy between the finite difference derivative and the derivative calculated via his approach. Furthermore, the computational cost of solving the discrete adjoint equations via an explicit method is quite similar to solving the discrete governing equations. Thus, for explicit codes, both the continuous and discrete approaches of sensitivity analysis can be used, although the complexity of the derivations, the computational cost and the inaccuracy between the derivative results are limitations for these codes.

For implicit codes, both the continuous and discrete approaches can also be employed. However, because implicit codes already calculate the Jacobian matrix of the discretized governing equations with respect to the flow variables, the discrete approach is quite easy to implement; whereas, for the continuous approach, the Jacobian of the discrete adjoint equations would be needed. The focus of the research presented herein is to develop a method for easily generating the Jacobian of the discretized governing equations as implemented within either an implicit or an explicit code, so that discrete sensitivity analysis can be applied uniformly to both solution methodologies.

One final distinction between the continuous and discrete approaches of sensitivity analysis should be emphasized. The derivatives generated by the continuous approach are not "discretely adjoint" to the discretized governing equations. As a result, the derivatives produced by the continuous approach are generally not as accurate as those produced by the discrete approach. The error is a result of the order of discretization and can be summarized by Figure 2. In discrete sensitivity analysis, the discretization error associated with the adjoint variable is consistent with the discretization error generated by solving the discretized governing equations. However, for continuous sensitivity analysis, the discretization error results from solving the discretized adjoint equations, which does not have a direct relationship to the discretization error coming from the governing equations. For discrete simulations, the adjoint variable λ generated by discrete sensitivity analysis will be discretely adjoint to the discretized gov-

erning equations. Since the derivatives are consistent with the discrete solution, these derivatives will be in better agreement with derivatives produced via finite differences or the complex Taylor's series expansion method than those derivatives produced by the continuous approach. However, as the mesh size is reduced, the discretization error will tend towards zero, and the derivatives produced by the two approaches will agree.

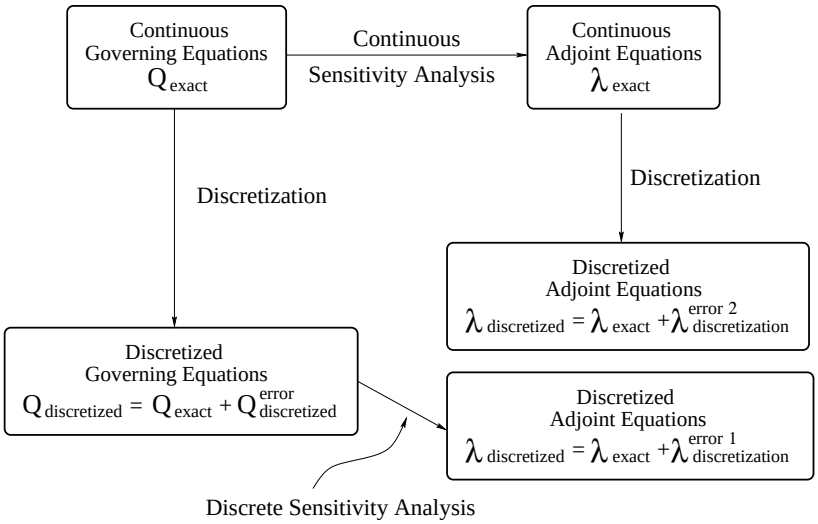

Figure 2: Effects of Discretization Error on Accuracy of Adjoint Variable.

For discrete sensitivity analysis, one of the primary factors affecting the accuracy of the design space derivatives is the accuracy of the Jacobian matrix. To obtain each term in this matrix by hand-differentiation is also quite challenging and error-prone. However, the complex Taylor's series expansion method can be used to overcome the difficulties associated with determining the Jacobian matrix, by using complex arithmetic to generate highly accurate derivatives without the need for hand-differentiation. Since the steady-state discretized equations are the same for explicit and implicit codes, the complex Taylor's series expansion method can be used to generate highly accurate Jacobian matrix for both explicit and implicit codes. Using this method, highly accurate Jacobian matrices can be generated for explicit and implicit codes, and discrete sensitivity analysis can be uniformly applied to both solution methodologies.

Sensitivity analysis is presented in detail in the next section. The complex Taylor's series expansion method is discussed in section 3. In section 4, transformation of the Euler simulation code is explained, and in section 5, the resulting optimization code is discussed and is applied to an inverse design problem.

2 Sensitivity Analysis

The objective function F is explicitly a function of the flow variables Q and possibly of the grid χ or the design variables $\vec{\beta}$, so $F(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta})$. Thus,

the design space derivative of F with respect to the design variable β_i can be expressed as

$$\frac{dF}{d\beta_i} = \frac{\partial F}{\partial Q}\frac{\partial Q}{\partial \beta_i} + \frac{\partial F}{\partial \chi}\frac{\partial \chi}{\partial \beta_i} + \frac{\partial F}{\partial \beta_i} \tag{3}$$

Since F is explicitly dependent on Q, χ and β, the terms $\frac{\partial F}{\partial Q}$, $\frac{\partial F}{\partial \chi}$ and $\frac{\partial F}{\partial \beta_i}$ can be calculated easily by hand. The term $\frac{\partial \chi}{\partial \beta_i}$ can be estimated via finite differencing the results of the grid generation code. Unfortunately, the term $\frac{\partial Q}{\partial \beta_i}$ can not be estimated via finite differences without an additional steady-state simulation. ($\frac{dF}{d\beta_i}$ is actually a partial derivative because F depends on several design variables; however, due to the limitations of notation, it is written as the total derivative, being the sum of the explicit and implicit dependencies of F on the design variable β_i.)

The system of governing equations can be expressed as $W(Q(\vec{\beta}), \chi(\vec{\beta}), \vec{\beta}) = 0$. Thus,

$$\frac{dW}{d\beta_i} = \frac{\partial W}{\partial Q}\frac{\partial Q}{\partial \beta_i} + \frac{\partial W}{\partial \chi}\frac{\partial \chi}{\partial \beta_i} + \frac{\partial W}{\partial \beta_i} = 0 \tag{4}$$

Again, since W is explicitly dependent on Q, χ and β, the associated terms $\frac{\partial W}{\partial Q}$, $\frac{\partial W}{\partial \chi}$ and $\frac{\partial W}{\partial \beta_i}$ can be calculated efficiently. The term $\frac{\partial \chi}{\partial \beta_i}$ is already known via finite differences. Hence, the only unknown term in this equation is $\frac{\partial Q}{\partial \beta_i}$. By multiplying equation (4) by the adjoint variable λ and adding to equation (3), we get

$$\frac{\partial F}{\partial \beta_i} = \left(\frac{\partial F}{\partial Q} + \lambda^T \frac{\partial W}{\partial Q}\right)\frac{\partial Q}{\partial \beta_i} + \left(\frac{\partial F}{\partial \chi} + \lambda^T \frac{\partial W}{\partial \chi}\right)\frac{\partial \chi}{\partial \beta_i} + \left(\frac{\partial F}{\partial \beta_i} + \lambda^T \frac{\partial W}{\partial \beta_i}\right) \tag{5}$$

By choosing λ such that

$$\frac{\partial F}{\partial Q} + \lambda^T \frac{\partial W}{\partial Q} = 0 \tag{6}$$

or

$$\frac{\partial W}{\partial Q}^T \lambda = -\frac{\partial F}{\partial Q}^T \tag{7}$$

the need to calculate the term $\frac{\partial Q}{\partial \beta_i}$ is removed, and the design space derivative can be calculated without the need of any additional steady-state simulations. Equation (7) does not depend on the design variables; hence λ is the same for all the design variables, and equation (7) must only be solved once, regardless of the number of design variables. However, equation (7) is dependent on the objective function F; thus, this equation must be solved for each function for which a derivative is needed.

In the discrete approach, the terms F and W represent the discretized objective function and governing equations, whereas in the continuous approach, these terms are the continuous objective function and governing differential equations. For the objective function used herein, F is only explicitly dependent

on the flow variables Q; hence, after solving for λ, the design space derivative is

$$\frac{\partial F}{\partial \beta_i} = \lambda^T \left(\frac{\partial W}{\partial \chi} \frac{\partial \chi}{\partial \beta_i} + \frac{\partial W}{\partial \beta_i} \right) = \lambda^T \frac{dW}{d\beta_i}\bigg|_{\text{Q fixed}} \tag{8}$$

To study the differences in implementation between the two approaches, a review of the differences between implicit and explicit simulation codes is appropriate. For explicit codes, the flow variables are updated via an equation similar to

$$Q^{n+1} = Q^n + f_{explicit}(Q^n, dt^n) \tag{9}$$

whereas for implicit codes, the flow variables are updated via

$$Q^{n+1} = Q^n + f_{implicit}(Q^{n+1}, Q^n, dt^n) \tag{10}$$

where Q^{n+1} is the vector of flow variables at time level $n+1$, Q^n is the vector of flow variables at time level n, which are known, and $f_{explicit}$ and $f_{implicit}$ are functions that define the discretized spatial derivative. If higher order discretizations of the temporal derivatives are used or if fractional step algorithms are used, equations (9) and (10) are modified to account for these changes. For explicit codes, the function $f_{explicit}$ depends only on known values, whereas for implicit codes, the unknowns Q^{n+1} are included within the update function $f_{implicit}$. For implicit codes, equation (10) can be recast as

$$W(Q^{n+1}, Q^n, dt^n) = Q^{n+1} - Q^n - f_{implicit}(Q^{n+1}, Q^n, dt^n) = 0 \tag{11}$$

and can be solved iteratively via

$$\frac{\partial W}{\partial Q^{n+1}} \Delta Q^m = -W(Q^{n,m}, Q^n, dt^n) \tag{12}$$

where $Q^{n,m+1} = Q^{n,m} + \Delta Q^m$. This equation is solved iteratively until the residual vector $W(Q^{n,m}, Q^n, dt^n)$ is sufficiently small, at which point $Q^{n+1} = Q^{n,m}$. The matrix $\frac{\partial W}{\partial Q^{n+1}}$ is called the Jacobian matrix of the residual vector with respect to the flow variables. Since equation (12) is solved iteratively in delta form, the Jacobian matrix does not need to be exact for the flow solver as long as W is driven to zero.

At steady-state, $Q^{n+1} = Q^n$, so $f_{explicit} = 0$ and $f_{implicit} = 0$. Furthermore, after using the identity that $Q^{n+1} = Q^n$, the Jacobian matrices can be expressed as

$$\frac{\partial W}{\partial Q^{n+1}} = -\frac{\partial f_{implicit}}{\partial Q^{n+1}} \tag{13}$$

and

$$\frac{\partial W}{\partial Q^n} = -\frac{\partial f_{explicit}}{\partial Q^n} \tag{14}$$

Typically, $\frac{\partial f_{explicit}}{\partial Q^n}$ is not calculated in explicit codes because it is not needed for the flow solver. However, if this matrix were available, discrete sensitivity

analysis could be applied to explicit codes in the same way as it is applied to implicit codes.

Since the Jacobian matrix is used in implicit codes, it is relatively straightforward to convert implicit flow solvers into optimization codes using discrete sensitivity analysis, although the accuracy of the resulting design space derivatives will be limited by the accuracy of the Jacobian matrix. By using the complex Taylor's series expansion method, the exact Jacobian can be generated regardless of the complexity of the turbulence models or spatial discretization methods, and highly accurate design space derivatives can be achieved.

To avoid the difficulties associated with the continuous approach of sensitivity analysis, the complex Taylor's series expansion method can be applied to explicit codes at steady-state to calculate the Jacobian matrix $\frac{\partial f_{explicit}}{\partial Q^n}$. Hence, discrete sensitivity analysis can be used in conjunction with explicit codes to generate the design space derivatives efficiently without the need for deriving and implementing the adjoint equations as well as driving these equations to steady-state.

3 Complex Taylor's Series Expansion Method

The complex Taylor's series expansion method has been recently used by Squire and Trapp [7] to calculate the derivatives of real-valued functions and by Newman etal [4] to estimate derivatives for an aero-structural design problem. This simple method can be applied to any numerical code that yields real-valued functional information and is based on the Taylor's series expansion of $F(x + i\Delta x)$ or

$$F(x + i\Delta x) = F(x) + iF'(x)\Delta x - \frac{F''(x)\Delta x^2}{2!} - i\frac{F^{(3)}(x)\Delta x^3}{3!} + O(\Delta x^4) \quad (15)$$

Since $F(x)$ is real valued, the Taylor's series alternates between real and imaginary terms and can be decomposed as

$$\begin{aligned} & F(x + i\Delta x) \quad (16) \\ = \quad & \left(F(x) - \frac{F''(x)\Delta x^2}{2!} + O(\Delta x^4), F'(x)\Delta x - \frac{F^{(3)}(x)\Delta x^3}{3!} + O(\Delta x^5) \right) \end{aligned}$$

Hence,

$$Im(F(x + i\Delta x)) = F'(x)\Delta x - \frac{F^{(3)}(x)\Delta x^3}{3!} + O(\Delta x^5) \quad (17)$$

or

$$F'(x) = \frac{Im(F(x + i\Delta x))}{\Delta x} + \frac{F^{(3)}(x)\Delta x^2}{3!} + O(\Delta x^4) \quad (18)$$

This method is second order accurate as is the central finite difference equations, but as there is no subtraction error, there are no round-off errors, and Δx can

be as small as the computer will allow. Hence, choosing Δx such that the higher order terms are smaller than machine precision, we have

$$F'(x) = \frac{Im(F(x + i\Delta x))}{\Delta x} \tag{19}$$

Thus, numerically exact Jacobians can be estimated via the application of the complex Taylor's series expansion method to the residual vector $W(Q^n, \Delta t^n)$. For a two-dimensional, structured grid, using a first-order spatial discretization, the equations at an interior node (i, j) will be dependent on the values at (i, j), $(i-1, j)$, $(i+1, j)$, $(i, j-1)$ and $(i, j+1)$, as shown in Figure 3. For the two-dimensional Euler equations, there are four differential equations to be solved and four flow variables to be determined at each node. Hence, there are four discretized equations associated with each node, and each equation can be dependent on as many as twenty different flow variables.

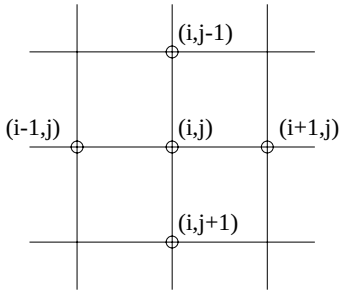

Figure 3. Grid Dependencies for Two-Dimensional, First-Order Scheme.

Repeated application of the complex Taylor's series expansion method to each discretized equation represented in the residual vector W will generate the values of the twenty derivative terms associated with a row in the Jacobian matrix, without any need for hand-differentiation. Hence, any turbulence model or spatial discretization scheme can be used, regardless of its complexity, as long as the derivatives exist. Furthermore, for unstructured grids, or for higher-order spatial discretizations, the only added complexity is the connectivity stencil. In other words, to generate the exact Jacobian matrix, the dependencies of the vector of discretized equations with the flow variables must be known, but the knowledge of the particulars of these dependencies is unnecessary as the complex Taylor's series expansion method will handle these complexities.

4 Application to Euler Code

The explicit flow simulation code that has been transformed into a design optimization code solves the two-dimensional Euler equations for flow around the NACA64A006 airfoil, using a structured C-grid. A wide variety of solution methods are available within the code, including implicit methods and fractional step methods; however, these subroutines have been deleted, leaving only

the explicit implementation of the first order Flux-Vector-Split method of Steger and Warming [8]. Since knowledge of the implementation of this method is not used in the transformation of the simulation code, a discussion of this method will not be presented here - the key element is that this method is explicit, and the stencil is the same as shown in Figure 3.

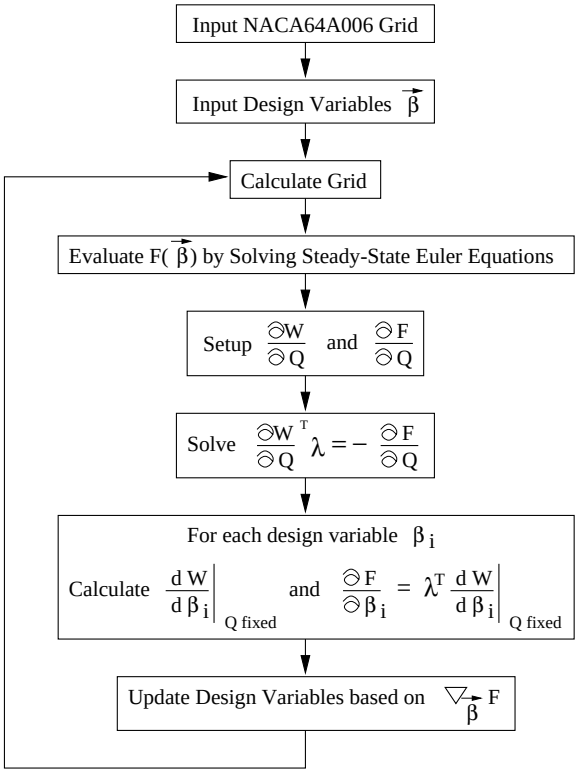

Figure 4. Flow-Chart for Algorithm to Estimate Derivatives

The algorithm for using discrete sensitivity analysis to estimate the design space derivatives is presented in Figure 4. The grid is determined by perturbing the shape of the NACA64A006 airfoil and propagating the perturbation into the two-dimensional grid. The shape of the airfoil was controlled by a B-spline curve with 14 control points. Ten of these control points were determined by the ten design variables, with the additional control points controlling the smoothness at the ends of the curve. Given a set of design variables, the B-spline curve was determined for each node on the surface of the airfoil. The value of the curve at each node determined the displacement in the y-direction that the new airfoil was from the NACA64A006 airfoil. Hence, if the B-spline curve's value at node 98 was +0.0012, then the y-location of the surface at node 98 for the new airfoil was the value for the NACA airfoil plus 0.0012. Once the displacement values were determined for the surface of the airfoil, these values were propagated into the grid by solving Laplace's equation for the NACA grid where the boundary

conditions were the displacement values. Once the values were propagated into the interior of the grid, the y-values for the new grid were determined by adding the value at a node to its original value.

The complex Taylor's series expansion method is used to generate the Jacobian matrix $\frac{\partial W}{\partial Q}$. The vector $\frac{\partial F}{\partial Q}$ is calculated via hand differentiation of the function F which is explicitly a function of the flow variables Q. The vector $\frac{dW}{d\beta_i}|_{\text{Q fixed}}$ is calculated via central differences or

$$\frac{dW}{d\beta_i} = \frac{W(Q(\vec{\beta}), \chi(\vec{\beta} + e_i\Delta\beta_i), \vec{\beta} + e_i\Delta\beta_i) - W(Q(\vec{\beta}), \chi(\vec{\beta} - e_i\Delta\beta_i), \vec{\beta} - e_i\Delta\beta_i)}{2\Delta\beta_i} \tag{20}$$

This calculation requires the formation of the grids associated with $\chi(\vec{\beta}+e_i\Delta\beta_i)$ and $\chi(\vec{\beta} - e_i\Delta\beta_i)$.

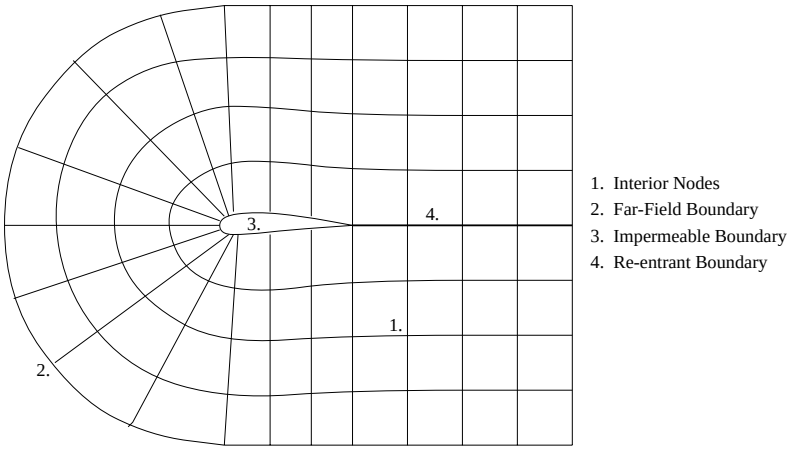

Figure 5. Typical C-grid around an airfoil.

To generate the Jacobian matrix, the nodal dependencies of the residual vector must be analyzed. The NACA64A006 grid is a C-grid, similar to the grid shown in Figure 5. For the interior nodes, the nodal connectivity stencil is the same as shown in Figure 3, resulting in a banded, block structure for the Jacobian matrix. For the far-field boundary and impermeable boundary, the connectivity stencil is a subset of the stencil for the interior nodes, so the banded, block structure of the Jacobian matrix is unchanged at these boundaries. However, for the re-entrant boundary, the nodal connectivity reaches to the other side of the grid, destroying the banded, block structure of the Jacobian. Thus, to solve the matrix equation (7), an iterative solver is employed. Unfortunately, this solver does not converge rapidly, and the solution is not highly converged.

Finally, to update the design variables, the BFGS update method has been employed. This method gradually builds an approximation to the Hessian matrix by updating the approximate Hessian matrix with gradient and design variable information. One additional function evaluation is performed in the search direction to determine a more appropriate step size. See Gill, Murray and Wright [1] for more details on this method.

5 Results

The design problem is an inverse design where the pressure distribution for the target design is specified. The design variables for the target design are (-0.001, -0.003, 0.001, 0.005, 0.01, 0.01, 0.02, 0.03, 0.02, 0.01). The steady-state solution for this design is approximated, and the pressure distribution along the surface of the airfoil are stored in Cp^{target}. The mach number is 0.8700, which results in transonic flow across the airfoil, and the angle of attack for the simulation is +3.2 degrees. The objective function is

$$F(\vec{\beta}^n) = \sum_{i=itl+1}^{itu} \left(Cp_i(\vec{\beta}^n) - Cp_i^{target} \right)^2 \tag{21}$$

where itl and itu specify the initial and final nodes for the surface of the airfoil in the C grid. The initial set of design variables is a vector of 0.0's, which represents the unmodified NACA airfoil. To analyze the accuracy of the design space derivatives, the derivatives for this set of design variables are calculated via the complex Taylor's series expansion method applied to the objective function. Hence, these derivatives are numerically exact, although the computational cost is quite large. These exact derivatives are compared with the derivatives generated by discrete sensitivity analysis and by one-sided finite differences and are presented in Table 2.

Design Variables	Complex Taylor's Series Expansion	Finite Differences	Adjoint Form. of Discrete Sensitivity Analysis
1	-58.918524793680	-58.918108	-58.925014473517
2	2.1553606507097	2.155659	2.1527299069564
3	9.3080663602748	9.308617	9.3059170047628
4	-4.4977859312973	-4.496398	-4.5001198714489
5	3.1437212541549	3.146781	3.1419623720023
6	587.33976898291	587.343377	587.35094264658
7	-457.04698505623	-457.044045	-457.04649298139
8	-431.92433510382	-431.923165	-431.98154907862
9	-60.885888964239	-60.885054	-60.878678137755
10	-128.31651602328	-128.315028	-128.20253621961
Comp. Cost	25870.7 sec.	4136.0 sec.	159.9 sec.

Table 2. Comparison of Design Space Derivatives.

Due to the errors introduced into the design space derivatives by the inexact solution of equation (7), the derivatives produced by discrete sensitivity analysis are not quite as accurate as those produced by finite differences, although they agree with the exact derivatives to at least three significant digits. However, the computational cost of such calculations, given in the bottom row, show the tremendous savings of discrete sensitivity analysis over the other two methods. In Table 3, the computational cost for each component in a design iteration using discrete sensitivity analysis is given. Since there are 10 design variables, the cost

of estimating the design space gradient via finite differences is approximately 4100 seconds, which is substantially more than the cost associated with discrete sensitivity analysis. Furthermore, as the number of design variables increases, the computational cost of using finite differences scales with the number of design variables, whereas the cost of discrete sensitivity analysis grows marginally, because the majority of the cost lies in determining the adjoint vector, which must only be calculated once, regardless of the number of design variables.

Component	Cost
Generating 1 Steady-State Solution	413.8 sec.
Generating $\nabla_{\vec{\beta}} F$ Via Discrete Sensitivity Analysis	159.9 sec.
Updating Design Variables (includes an additional solution)	413.6 sec.
Total Cost of Design Iteration Via Discrete Sensitivity Analysis	987.3 sec.

Table 3. Computational Costs for Components of Design Process.

Figure 6 shows the convergence history of the objective function values for the BFGS update method, using one additional function evaluation. The method is slow at improving the design variables for the first fifteen design iterations; however, after the twentieth design iteration, the objective function decreases greatly, stabilizing near $2.3E-10$. By using the BFGS update method, superlinear convergence results have been obtained, which reduces the number of design iterations and hence the total computational cost of the design process. The design variables corresponding to iteration #23 are in agreement with the target set of design variables to at least seven decimal places. Hence, one can conclude that the design space derivatives are accurate enough to drive the design variables quite close to the target.

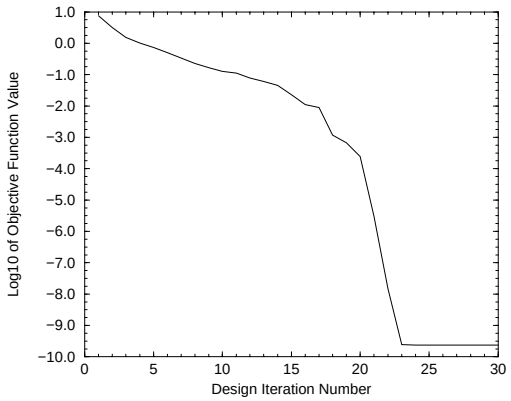

Figure 6. Optimization History for BFGS Update Method.

6 Conclusions

Discrete sensitivity analysis in conjunction with the complex Taylor's series expansion method has been used to transform an explicit, two-dimensional Euler solver into a design optimization code. As has been derived in many other papers, the formulation of discrete sensitivity analysis presented herein requires the Jacobian of the discretized system of equations, which is not generated or used in the explicit code. The complex Taylor's series expansion method is used to generate the exact Jacobian matrix, once the steady-state solution has been obtained. Thus, there is no need to derive or solve the continuous adjoint equations as is typically done when applying the continuous approach of sensitivity analysis to explicit codes. The only information needed about the discretized equations is the connectivity stencil, showing the dependence of the discretized equations on the nodes within the grid. As a result, the complex Taylor's series expansion method should be easily applicable to complicated turbulence models and higher-order schemes.

This method has been applied to an explicit code that solves the two-dimensional Euler equations for flow around an airfoil. The resulting design space derivatives have been compared with the numerically exact derivatives, agreeing to at least three significant digits. These design space derivatives are also accurate enough to drive the design variables towards the target set of design variables, with the converged set of design variables agreeing with the target set to at least seven decimal places. Furthermore, the computational cost of estimating these derivatives for the discrete sensitivity analysis is substantially less than for finite differences.

The explicit code is only first-order in space and does not use fractional steps. More research is necessary to address the difficulties of using higher-order spatial discretizations, which increase the connectivity stencil and hence the computational cost of solving equation (7). Furthermore, more study is necessary to apply discrete sensitivity analysis to fractional step methods, such as a predictor-corrector method.

Acknowledgments. The author would like to thank Dr. Mark Janus for providing the explicit Euler code used in this study. Furthermore, the author would like to thank Dr. David Huddleston for introducing the author to the concept of sensitivity analysis and Dr. David Whitfield for introducing the author to the complex Taylor's series expansion method.

References

[1] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press Ltd., San Diego, 1981.

[2] Nadarajah, S., and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization", AIAA Paper 2000-0667.

[3] Newman, J. C. III, Taylor, A. C. III, Barnwell, R. W., Newman, P. A., and Hou, G. J. W., "Overview of Sensitivity Analysis and Shape Optimization for Complex Aerodynamic Configurations", *AIAA J. of Aircraft*, Vol. 36, No.1, Jan.-Feb., 1999., pp. 87-96.

[4] Newman, J. C. III, Anderson, K. W., and Whitfield, D. L., "Multidisciplinary Sensitivity Derivatives Using Complex Variables", Mississippi State University Publication, MSSU-EIRS-ERC-98-08, July, 1998.

[5] Reuther, J. J., "Aerodynamic Shape Optimization Using Control Theory", Dissertation, University of California, Davis, 1996.

[6] Shubin, G. R., and Frank, P. D., "A Comparison of Two Closely-Related Approaches to Aerodynamic Design Optimization", Third International Conf. on Inverse Design Concepts and Opt. in Eng. Sciences (ICIDES-III), Washington, D. C., Oct. 23-25, 1991.

[7] Squire, W., and Trapp, G., "Using Complex Variables to Estimate Derivatives of Real Functions", *Soc. Ind. Appl. Math.*, Vol. 40, No.1, pp.110-112, March, 1998.

[8] Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gas Dynamic Equations with Applications to Finite-Difference Methods", *J. Comp. Phys.*, Vol. 40, 1981, pp. 263-293.

Clarence O. E. Burg
Research Engineer, Computational Simulation and Design Center
Engineering Research Center
Mississippi State University, Mississippi State, MS, USA
email: burg@erc.msstate.edu