JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, October 2005

# A General and Extensible Unstructured Mesh Adjoint Method

Chad E. Burdyshaw* and W. Kyle Anderson†
*Graduate School of Computational Engineering University of Tennessee at Chattanooga, Chattanooga, TN 37403*‡

**An efficient method for computing adjoint based, functional sensitivities has been developed in a manner that minimizes maintenance required to reflect subsequent updates to the function code. This method has been applied to an unstructured mesh, Navier Stokes flow solver, within an object-oriented, polymorphic framework. The use of Complex Taylor Series Expansion (CTSE), within this framework, allows for flexible application in the computation of derivatives.**

## Nomenclature

| | |
|---|---|
| $\alpha$ | angle of attack |
| $\beta$ | design variables |
| $\chi$ | mesh parameters |
| $\Delta t$ | time step |
| $\epsilon$ | machine precision |
| $\Im$ | imaginary component |
| $\lambda$ | adjoint variable vector |
| $\nabla$ | gradient operator |
| $\emptyset$ | order notation |
| $\Re$ | real component |
| $\tilde{I}$ | identity matrix |
| $h$ | perturbation size |
| $i$ | $\sqrt{-1}$ |
| $I_c$ | cost or objective function |
| $L$ | lift |
| $N$ | Newton iteration index |
| $n_{cv}$ | Number of control volumes in the mesh |
| $n_{dv}$ | Number of design variables |
| $n_{st}$ | Number of nodes in a computational stencil |
| $Q$ | flow variables |
| $R^{(1)}$ | 1st order spatial residual |

Received 10 February 2005; revision received 21 July 2005; accepted for publication 22 July 2005. Copyright 2005 by Chad E. Burdyshaw, W. Kyle Anderson. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/04 $10.00 in correspondence with the CCC.

* Research Associate, UT SimCenter at Chattanooga, Design and Optimization Group. AIAA Student Member.

† Professor, UT SimCenter at Chattanooga, Design and Optimization Group. AIAA Associate Fellow.

‡ This work was funded by the University of Tennessee at Chattanooga through the Lupton Renaissance Fund. This support is greatly appreciated.

| | |
|---|---|
| $R^{(2)}$ | 2nd order spatial residual |
| $Vol$ | volume of fluid element |
| $x, y, z$ | Cartesian coordinates |

## I. Introduction

COMPUTATIONAL sensitivity analyses of problems in fluid mechanics have been of interest since the early 1970's. In Hicks, Murman and Vanderplaats[1] 1974 paper, sensitivity information is used in conjunction with a gradient-based optimization method to improve the aerodynamic characteristics of airfoils. The governing equations were based on potential flow assumptions and sensitivity derivatives were obtained using finite-differences. Since this early exploration into design optimization, there have been substantial advances in both the fidelity of the physical models as well as the methodologies used for obtaining sensitivity information.[2–8]

Techniques for obtaining sensitivity derivatives can be characterized as either direct differentiation methods or adjoint methods. In direct methods, derivatives of the flow variables with respect to each design variable are computed initially, and objective or constraint sensitivities are subsequently derived from these quantities. This technique is advantageous when there are many objectives or constraints, but suffers in efficiency when there are many design variables. Examples of direct or forward differentiation methods include Sadrehaghighi et al.,[9] and Burg and Newman.[10]

When the number of design variables substantially exceeds the number of objective and constraint functions, an adjoint method is more suitable. In an adjoint method, the objective function is augmented with the flow equations as a constraint using Lagrange multipliers. This technique can be derived using either a discretization of the differentiated governing equations (continuous), or a differentiation of the discretized governing equations (discrete). For examples of continuous methods see Pirronneau,[11] Anderson,[12] and Jameson,[13] whereas discrete adjoint methods are described in Nielsen and Anderson[14] and Giles et al.[15] In either approach, the end result is that an auxiliary set of linear equations (adjoint equations) are solved for each objective function or constraint, and the derivatives for each design variable are obtained using a matrix-vector multiplication. In addition to its utility in obtaining sensitivity information, the adjoint solution can be used in both error analysis and mesh adaptation.[16–19]

Previous work reported by Nielsen and Anderson,[14,20] describes an adjoint methodology for obtaining derivatives which are discretely consistent with the corresponding flow solver. In the referenced work, the adjoint solver has been obtained by hand differentiation of the baseline flow solver equations. The resulting derivatives are highly accurate, but the effort required to address changes in the flow code via updates and extensions is prohibitive.

The objective of the current study is to develop an adjoint method which yields sensitivity derivatives that are discretely consistent with a numerical pde solver (Navier Stokes equations in this case), and is of a form which facilitates ease of maintenance and extensibility. These attributes are necessary in order to bring the development cycle of sensitivity analysis codes into concurrency with flow solver technology.

There are two technologies with the potential to address these goals. These are automatic differentiation (AD) and complex Taylor series expansion (CTSE).

Automatic differentiation is, as the name implies, a method for differentiating a function solver automatically without the need for hand differentiation. This is usually accomplished by applying a transformation routine to the original function solver code to generate a new differentiated function solver code through automated chain rule applications or operator overloading. A number of AD codes are currently available. Most of these codes implement only direct differentiation, though a small subset of these can implement adjoint methods.[21–23]

As in hand differentiation, AD provides exact discrete derivatives. Its implementation significantly reduces the time and effort needed to write and maintain a hand differentiated code. However, there are a few disadvantages. The application of AD is limited to those function codes written in languages currently supported. Difficulties encountered in implementation must either wait for a new AD code release or rely on modifications to the function code and or the transformed code for resolution. In addition, the efficiency of AD transformed code suffers in comparison to hand differentiation. Finally, anecdotal accounts suggest that applications of AD to a primal code are not as straightforward as one might imagine. However, with continual improvements addressing these disadvantages, it is likely that AD codes will be the obvious choice for sensitivity development in the future.

An alternative to AD transformation can be found through the use of the complex Taylor series expansion. This method which is explained in detail in section (IV.B) is, by itself, simply a way to compute very precise discrete partial derivatives. However, the combination of this tool with polymorphic object code and a few adjoint specific routines, results in a method which has a number of advantages.

As in AD, derivatives can be computed with the accuracy of hand differentiation, but with minimal implementation and maintenance costs. Furthermore, this method can be applied to any code which allows complex number data types. Poor efficiencies associated with a complete CTSE conversion are minimized through a naturally targeted application. A disadvantage to this approach is the necessity to compute and store the linearized residual sensitivity matrix. This can result in rather large memory requirements. In addition, to properly apply CTSE, it is necessary to ensure that all functions, including intrinsics, have a complex valued implementation.

This combination of CTSE within a polymorphic, object-oriented pde solver is used to generate adjoint based sensitivities without major modification of the original code.

The purpose of this paper is to present details of this new development in sensitivity code technology.

## II. Flow Solver

The sensitivity methods discussed in this study are applied to an unstructured mesh, three dimensional, fluid flow solver.[24–30] The spatial discretization for this code is based on a node centered scheme in which a control volume is constructed around each of the mesh points in the field. The solution algorithm is based on a backward-Euler time discretization where an approximate solution of the linear system required at each step is obtained using the symmetric Gauss-Seidel point-iterative method described in the references cited above.

The solver is applicable to flow regimes ranging from the incompressible Euler equations, to the compressible, unsteady, turbulent, Reynolds averaged Navier Stokes equations. For computation of high Reynolds number flows, several turbulence models are available, including Spalart-Allmaras,[31] and $q-\omega$[32] models. The solution of the turbulent variables in the flow solver is loosely coupled with the solution of the other flow variables.

The flow solver code has been designed in a modular fashion following an object-oriented model implemented using C++. In addition to the modularity provided by the model, type polymorphism is also utilized through templating, in instances where similar function evaluations are needed for multiple data types. These properties of the solver code are used to great advantage in computing sensitivity information.

## III. Adjoint Solver

The primary objective of the current work is to develop an adjoint solver that is easily maintained and produces sensitivity derivatives which are consistent with the flow solver even as the flow solver code evolves. To this end, the adjoint solver reuses as much of the flow solver code as possible. The following discussion of the flow solver/adjoint sensitivity code will include primarily only those issues that directly affect sensitivity solutions.

### A. Adjoint Formulation

The sensitivity of an objective function ($I_c$) can be obtained by expanding the total derivative of the objective with respect to the design variables ($\beta$). This differentiation includes explicit dependency on the design variables as well as implicit dependency through the mesh ($\chi$) and solution variables ($Q$). The expression for objective sensitivity is presented in equation 1.

Objective Sensitivity:

$$\frac{dI_c}{d\beta} = \frac{\partial I_c}{\partial \beta} + \left[\frac{\partial I_c}{\partial \chi}\right]\frac{\partial \chi}{\partial \beta} + \left[\frac{\partial I_c}{\partial Q}\right]\frac{\partial Q}{\partial \beta} \tag{1}$$

Similarly, the sensitivity of the flux residual is expressed by equation 2.

Residual Sensitivity:

$$\frac{dR}{d\beta} = \frac{\partial R}{\partial \beta} + \left[\frac{\partial R}{\partial \chi}\right]\frac{\partial \chi}{\partial \beta} + \left[\frac{\partial R}{\partial Q}\right]\frac{\partial Q}{\partial \beta} \tag{2}$$

Different combinations of equations (1), and (2), lead to two alternate discrete methods for computing sensitivities. These methods are termed the direct and adjoint methods, each of which have their distinct advantages and disadvantages.

The direct method involves solving for $\frac{\partial Q}{\partial \beta}$ from the residual sensitivity equation (2) and substituting the result into the objective sensitivity equation (1). The solution of the $\frac{\partial Q}{\partial \beta}$ term must be solved for each independent design variable ($\beta$), thus making this technique very expensive for a large number of design variables. The method is advantageous however, when it is necessary to evaluate a large number of objective or constraint functions. For problems where there are few objective functions but many design variables, an adjoint method is more appropriate. The adjoint method is obtained by multiplying equation (2) with a Lagrange multiplier ($\lambda$), and adding it to equation (1), noting that $\frac{dR}{d\beta} = 0$ for a steady state problem.

$$\frac{dI_c}{d\beta} = \left\{ \frac{\partial I_c}{\partial \beta} + \left[ \frac{\partial I_c}{\partial \chi} \right] \frac{\partial \chi}{\partial \beta} \right\} + \lambda^T \left\{ \frac{\partial R}{\partial \beta} + \left[ \frac{\partial R}{\partial \chi} \right] \frac{\partial \chi}{\partial \beta} \right\} + \left\{ \left[ \frac{\partial I_c}{\partial Q} \right] + \lambda^T \left[ \frac{\partial R}{\partial Q} \right] \right\} \frac{\partial Q}{\partial \beta} \tag{3}$$

Since the choice of the Lagrange multiplier is arbitrary, the $\frac{\partial Q}{\partial \beta}$ term can be eliminated by solving the Adjoint Equation (4).

Adjoint Equation:

$$\left\{ \left[ \frac{\partial I_c}{\partial Q} \right] + \left[ \frac{\partial R}{\partial Q} \right]^T \lambda \right\} = 0 \tag{4}$$

Substituting the recently solved adjoint variables back into equation (3) allows the formulation of a complete adjoint objective sensitivity equation (5).

Adjoint Objective Sensitivity:

$$\frac{dI_c}{d\beta} = \left\{ \frac{\partial I_c}{\partial \beta} + \left[ \frac{\partial I_c}{\partial \chi} \right] \frac{\partial \chi}{\partial \beta} \right\} + \lambda^T \left\{ \frac{\partial R}{\partial \beta} + \left[ \frac{\partial R}{\partial \chi} \right] \frac{\partial \chi}{\partial \beta} \right\} \tag{5}$$

### B. Solution of the Adjoint Equation

The solution of the adjoint equation (4) is obtained in a similar manner as the flow variable solution. Equation (6) presents a form of this sparse matrix problem, which can be solved using a point iterative method. As in the sparse matrix solution of the flow solver, the symmetric Gauss Seidel scheme[25] is used to solve this system of linear equations. The inclusion of a pseudo time derivative term $\left( \frac{Vol}{\Delta t} \tilde{I} \right)$ allows an increase in diagonal dominance by reducing the CFL number, thereby improving the robustness of the adjoint equation solver.[14]

Iterative Adjoint Equation:

$$\left[ \frac{Vol}{\Delta t} \tilde{I} + \frac{\partial R^{(1)}}{\partial Q} \right]^T \Delta\lambda^{N+1} = -\frac{\partial I_c}{\partial Q} - \left[ \frac{\partial R^{(2)}}{\partial Q} \right]^T \lambda^N \tag{6}$$

where $\Delta\lambda^{N+1} = \lambda^{N+1} - \lambda^N$

This iterative form also allows the use of a first order spatial residual linearization ($R^{(1)}$) on the left hand side of the equation. The second order spatial residual ($R^{(2)}$) on the right hand side drives the accuracy of the final solution. The first order residual on the left hand side results in an iteration matrix which generally has a better condition number than the matrix of the second order residual, and thus better convergence properties.

Solution of the adjoint equation requires the partial derivatives of the objective function with respect to the flow variables $\left( \frac{\partial I_c}{\partial Q} \right)$ as well as linearization and transposition of the flux Jacobian $\left( \frac{\partial R}{\partial Q} \right)$. The methodology and associated issues related to computation of these partial derivatives are presented in the next section.

## IV. Partial Derivative Computations

The partial derivatives in equations (1) and (2) necessary to solve the adjoint equations, and to assemble the total sensitivity derivative can be computed in a number of ways. A hand differentiation technique that has been used for developing an adjoint method applicable to turbulent flows on unstructured meshes is described in Nielsen and Anderson.[14,20] This method is tedious, error prone, and is not easily maintained as new functionality is added to the flow solver. Other methods can be derived using Taylor series expansions. Two of these methods which have second order accuracy, namely central finite difference, and complex Taylor series expansion are compared for suitability to this application.

## A. Central Finite Difference

Taylor series expansions with both positive and negative perturbations in real space are given by equations (7) and (8) respectively.

Taylor Series Expansions in Real Space:

$$f(x+h) = f(x) + \frac{h}{1!}\frac{\partial f(x)}{\partial x} + \frac{(h)^2}{2!}\frac{\partial f^2(x)}{\partial x^2} + \frac{(h)^3}{3!}\frac{\partial f^3(x)}{\partial x^3} + \cdots \tag{7}$$

$$f(x-h) = f(x) - \frac{h}{1!}\frac{\partial f(x)}{\partial x} + \frac{(h)^2}{2!}\frac{\partial f^2(x)}{\partial x^2} - \frac{(h)^3}{3!}\frac{\partial f^3(x)}{\partial x^3} + \cdots \tag{8}$$

The central finite difference expression is constructed by subtracting these expansions and truncating at the $h^3$ term, such that the truncation error for the derivative expression (9) is of second order.

Central Finite Difference derivative:

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+h) - f(x-h)}{2h} + \text{Ø}(h^2) \tag{9}$$

This technique requires two function evaluations to compute a second order accurate derivative. Furthermore, when the function values differ by a small degree, the subtraction operation produces a very small number which is then truncated to machine precision. This operation causes a loss of significant digits, and when divided by the perturbation size ($h$), results in an erroneous derivative. This behavior imposes a lower bound on the perturbation size and prevents accurate derivatives in areas of low function sensitivity.

## B. Complex Taylor Series Expansion (CTSE)

The CTSE method[33–35] is developed by performing a Taylor series expansion with a perturbation in imaginary space (10). This requires of course that $f(z)$ be a continuous, complex valued function which is real valued on a real domain.

Taylor Series Expansion in Imaginary Space:

$$f(x+ih) = f(x) + \frac{ih}{1!}\frac{\partial f(x)}{\partial x} + \frac{(ih)^2}{2!}\frac{\partial f^2(x)}{\partial x^2} + \frac{(ih)^3}{3!}\frac{\partial f^3(x)}{\partial x^3} + \cdots \tag{10}$$

The real part of equation (10) corresponds to the unperturbed function value with a truncation error of order $O(h^2)$.

Real Part:

$$\Re[f(x+ih)] = f(x) - \frac{(h)^2}{2!}\frac{\partial f^2(x)}{\partial x^2} + \cdots \tag{11}$$

Similarly, the imaginary part of equation (10), divided by the perturbation size, yields a second order accurate expression (12) for the first derivative.

Imaginary Part:

$$\frac{\Im[f(x+ih)]}{h} = \frac{\partial f(x)}{\partial x} - \frac{(h)^2}{3!}\frac{\partial f^3(x)}{\partial x^3} + \cdots \tag{12}$$

Due to the fact that no subtraction is required with CTSE, it does not suffer from subtractive cancellation error. This allows the perturbation size to be as small as the order of numerical precision, thereby driving the truncation error to machine zero. A typical perturbations size is the square root of machine precision ($\epsilon$). Furthermore, only one function evaluation is necessary to compute a derivative, though there is additional overhead involved with computing in complex arithmetic. This results in an expense comparable to the central difference method.

An illustration of the effects of subtractive cancellation error can be seen from the plots in Fig. 1. These plots compare the error in the computed derivative using both central finite difference and CTSE methods on the function

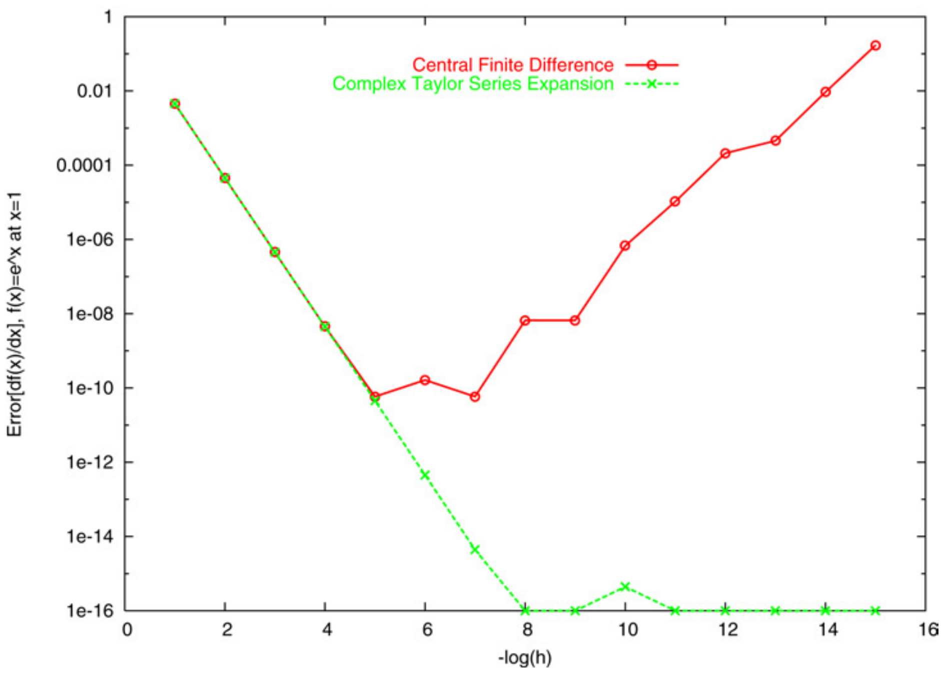

**Fig. 1 Central Finite Difference vs. Complex Taylor Series Expansion.**

$f(z) = e^z$, where complex number $z = a + ib$, and $f$ is strictly real valued on a real domain. Both solutions converge according to the second order truncation error up to a point. As the perturbation size approaches $10^{-6}$, subtractive cancellation error dominates in the central finite difference solution, and the accuracy deviates accordingly. The CTSE solution continues to converge to a minimum error at a perturbation size of $10^{-8}$, which for this case is $\epsilon$.

## V. Sensitivity Computations

Computation of the partial derivatives in equations (1) and (2) can be achieved using the methods described previously. However, there are numerous factors that must be addressed in order to obtain an efficient implementation. A discussion of the methods used to improve efficiency in the current implementation are presented in the following sections.

### A. Efficiency

In the solution of the adjoint equation, a straightforward implementation of the proposed method would be to apply the CTSE method to every function in the flow solver and to apply perturbations sequentially for each of the dependent flow variables in the field. After each perturbation, the residuals throughout the field can be computed and the derivatives extracted from the imaginary part. A similar technique may be employed to evaluate the remaining terms in the sensitivity equations once the adjoint variables are obtained. This method has an obvious disadvantage in that the quantity of extraneous computations performed makes it extremely inefficient.

#### *1. Targeted Differentiation*

One obvious step toward alleviating inefficiencies in the complete differentiation is to apply the CTSE method only to functions necessary to evaluate the terms of interest. This targeted approach saves execution time, by eliminating many extraneous complex computations, but also limits the applicability of the code to problems which involve only targeted functions.

With the use of a fully templated (ie. polymorphic), object oriented code framework, both complete and targeted approaches are easily implemented. The term “polymorphic” refers to an objects ability to alter its form based on the data type that it is asked to work with. For example, a grid object created as type REAL, would modify its variables, member functions, and child objects to operate using type REAL. That same grid object created using type COMPLEX or INTEGER would modify its attributes to perform equivalent operations on its given data type. The catch is, of course, that one must have multiple definitions for all functions and intrinsics used in the object. Derivatives in this framework may be computed by simply passing properly perturbed complex parameters into a function and then reducing the imaginary value of the result.

Complete derivative computation by the method described above is predicated by the assumption that the function properly encapsulates all of its dependencies. These "black box" functions obscure the details of evaluation from the programmer, thereby allowing derivatives to be computed without having to be concerned with implementation details.

### *2. Utilizing Localized Perturbation Effects*

A further method for reducing the number of extraneous computations can be developed by recognizing that, in many instances, the functional data dependence is localized on a reasonably small computational stencil. These stencils describe the local net of data points which contribute to the function result.

Figure 2 shows an example of two first order stencils for a residual computation on a two-dimensional rectangular control volume. The stencil on the left, used in computing $\frac{\partial R}{\partial Q}$, involves only the nearest neighbors which have edge connections to the center node. The metric stencil on the right of Fig. 2 is used to compute $\frac{\partial R}{\partial \chi}$, and is composed of all of the neighbor nodes which form geometric elements in common with the center node. For a second order accurate scheme the corresponding stencil is extended to include all the second nearest nodes as well.

By exploiting the fact that evaluation of the residual over a control volume involves only localized data, multiple columns of the matrix can be computed simultaneously, by perturbing flow variables in non-overlapping regions of dependency.

Figure 3 is an illustration of a set of non-interfering flux stencils for a first order Euler solution.

The stencils denoted by the bold edge connections consist of all of the nodes affected by a perturbation of the node at the center of each stencil. The contributions from the perturbation nodes can be computed simultaneously for all of the nodes in each independent stencil.

By arranging groups of non-interfering stencils into sets of independent stencil lists, sometimes referred to in the literature as "colors", the number of residual calculations required to compute the entire linearized residual is reduced from the total number of control volumes ($n_{cv}$) to the total number of stencil lists ($n_{st}$).

Currently, non-interfering stencil lists are constructed by sequentially traversing the list of mesh nodes and checking each stencil for overlap with stencils that have been previously added to the non-interference list. If overlap is not detected, it is added to the list and the next node is compared. When no additional non-interfering stencils can be added to the current list, a new list is started using the next unlisted node stencil. The procedure continues until all nodes have been included in a non-interfering stencil list.

This method for generating non-interfering stencil lists results in an unbalanced set of lists, in which early lists contain many stencils, and later lists may contain as few as a single stencil. The inefficiencies related to these

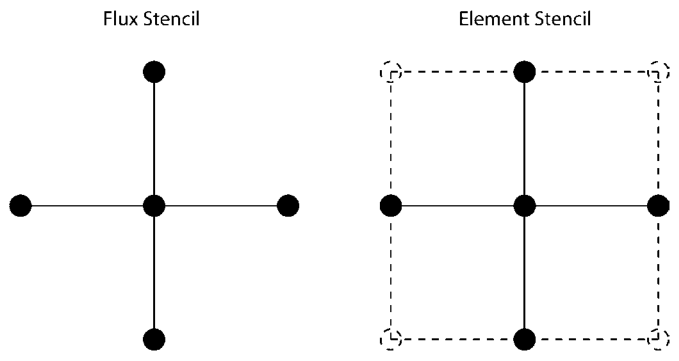

**Fig. 2 First Order Flux and Metric Computational Stencils.**

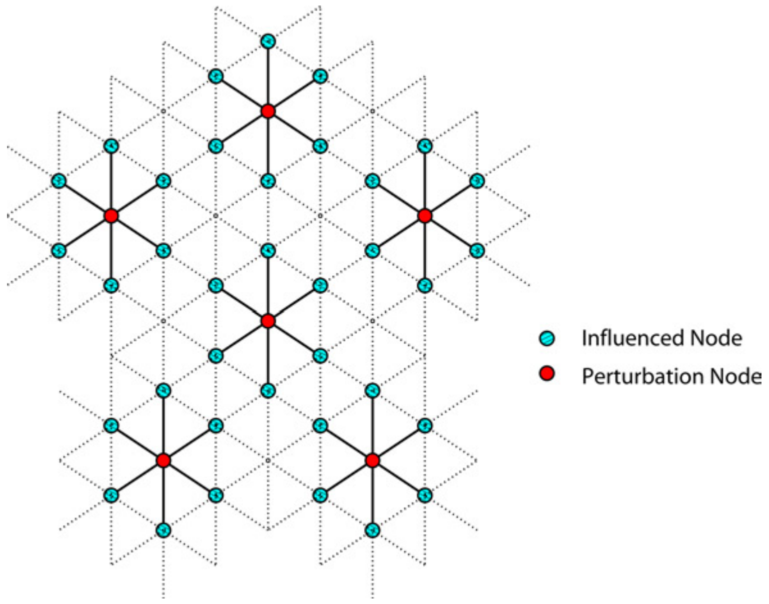

**Fig. 3 1st Order Independent Computational Stencils.**

unbalanced stencil lists have been mitigated by the use of localized function evaluations, which is explained in further detail in following sections.

In Table 1, stencil lists were compiled for three meshes with differing node counts and element compositions. The number of stencil lists required to include the entire domain for both first and second order flux linearizations $\left(\frac{\partial R}{\partial Q}\right)$, and the metric linearization $\left(\frac{\partial R}{\partial \chi}\right)$ are reported. The element ratios of hexahedral to prism volume elements are 2:1 for mesh 1, and 4:5 for mesh 2. These same ratios apply to meshes 1 and 2 for quadrilateral to triangular boundary faces. Mesh 3 is composed almost entirely of hexahedral volume elements and quadrilateral boundary faces.

An examination of the data shows that the number of stencil lists generated, is independent of the number of points in the mesh. The dependency instead relies upon the connectivity of the mesh elements, and the approximation order of the desired linearizations (i.e. the stencil size).

For example, in linearizations with respect to metric parameters, the use of hexahedral elements increases connectivity by including both edge nodes and corner nodes in the stencil. This reduces the number of independent stencils in each list, thereby requiring a greater number of lists to encompass the domain.

### *3. Localized Function Evaluations*

For integral functions such as lift or drag, evaluation of sensitivity with respect to some localized parameter can also be limited to a localized domain. This reduction allows the speed of partial derivative evaluations to be greatly increased by reducing the number of computations performed for each derivative from $(n_{cv} * n_{dv})$ to $(n_{st} * n_{dv})$, where $n_{cv}$ is the total number of control volumes in the mesh, $n_{st}$ is the number of nodes in a local stencil, and $n_{dv}$ is the number of dependent variables.

**Table 1 Number of Stencil Lists Generated For Meshes of Varying Size and Composition.**

| mesh | nodes | 1st order $\frac{\partial R}{\partial Q}$ | 2nd order $\frac{\partial R}{\partial Q}$ | $\frac{\partial R}{\partial \chi}$ |
|---|---|---|---|---|
| mesh 1 | 3480 | 54 | 252 | 535 |
| mesh 2 | 83317 | 62 | 275 | 380 |
| mesh 3 | 103673 | 65 | 262 | 984 |

A caveat is necessary here regarding turbulence models requiring a distance function calculation. In these cases, the computational stencil of a control volume must also include the closest viscous surface point, consequently the stencil is no longer localized. This dependency presents a problem in particular for parallel solutions across spatial domains. Currently, sensitivities which involve a change in the distance function are computed sequentially over the global domain. Thus for the turbulent flow regime, computing sensitivities with respect to large numbers of mesh displacements is prohibitive. However this is a temporary limitation as a method is under development for computing the distance function contributions independent from the localized evaluations.

#### 4. *Loss of Opacity*

Implementing the methods discussed above for increasing efficiency is most easily achieved if all the residual and force routines are coded in a "black box" manner such that the implementation details are opaque for purposes of developing the adjoint code.

If the routines in an existing flow solver are not coded in this manner, it becomes necessary to trace all the dependencies of the function to be localized, and include the effects. This loss of opacity can significantly increase the complexity of the adjoint solver code.

In the current study, several flow solver functions were not coded with the desired encapsulation. Therefore, in order to compute derivatives, it was necessary to ensure that all dependencies not encapsulated within these functions were accounted for. This process proved to be more challenging than originally anticipated. A complete resolution to this problem has yet to present itself, however a great deal of difficulty can be avoided if the desired functions are written such that all dependencies are updated internally.

That being said, it should also be recognized that encapsulated functions are not necessarily ideal for use in the flow solver. For example, in the flow solution, mesh metrics and flow conditions are generally computed upon initialization, and remain constant for the duration of the solution process. Therefore, the functions that compute these initial values are called outside the solver iteration loop. Segregation of these tasks into initialization functions and time evolution update functions makes sense in terms of flow solver efficiency. The apparent conflict between the flow and adjoint solver regarding efficiency and the form of functions in general will likely result in the increased complexity and reduced maintainability of the adjoint code.

### B. Storage of Residual Linearizations

The form of the iterative solution algorithm for the adjoint equation requires a residual linearization matrix on both sides of the equality (6). Due to the formulation of the method, the matrix on the left hand side of the equality may be a less accurate approximation of the true linearization of the residual which is required on the right hand side. In the present study, the linearization on the left hand side is identical to that used in the flow solver so that data structures can be reused. The matrix on the right hand side however, determines the accuracy of the adjoint vector and therefore, must be a precise linearization of the full second order accurate residual whose computation is dependent on both nearest and second nearest neighboring nodes. To accommodate the extended stencil requires a deviation from the data structure used in the flow solution algorithm. While the flow solver uses an edge based scheme to store flux Jacobians on the right and left hand sides of a flux face, storage of the level two contributions are accounted for using a compressed row storage structure.[36]

## VI. Results

### A. Test Cases

#### 1. *Incompressible and Compressible Viscous Flow Over a Coarse NACA0012 Airfoil Section*

The primary test case used for verification of the adjoint algorithm is a very coarse NACA 0012 wing section. This wing section is a three-dimensional mesh constructed through an extrusion of a two-dimensional NACA 0012 airfoil geometry. Though the results for this mesh are not physically accurate, the small size is useful for validation purposes.

Laminar cases were run for incompressible and compressible flow regimes in which sensitivities were calculated using both forward mode CTSE and the adjoint method. For both cases the flow angle was set at 5°, with a chord based Reynolds number of $1.0E^6$. For the compressible case, the Mach number was 0.3. Additionally, an incompressible turbulent solution was computed in which the one-equation Spalart-Allmaras model was used.

**Table 2 Incompressible 2nd Order Laminar: Sensitivity Comparison on Coarse NACA0012 Airfoil Mesh.**

| Term | Forward CTSE | Adjoint | $\mathrm{Error}_{rel}$ |
|---|---|---|---|
| $dL/d\alpha$ | $1.07797088826512E^{0}$ | $1.07797088826523E^{0}$ | $1.054E^{-14}$ |
| $dL/d\chi_{y,2528}$ | $-1.32325310797162E^{0}$ | $-1.32325310797183E^{0}$ | $1.269E^{-13}$ |
| $dL/d\chi_{y,2537}$ | $4.6557836415831E^{0}$ | $4.6557836415915E^{0}$ | $1.806E^{-12}$ |
| $dL/d\chi_{y,2541}$ | $-1.61913550267185E^{0}$ | $-1.61913550267235E^{0}$ | $3.069E^{-13}$ |

The results of the sensitivity comparisons are presented in Tables 2, 3, and 4 which include the sensitivity of the lift with respect to the angle of attack ($\alpha$), and y-coordinate perturbations of three surface points. The locations of the sampled surface sensitivities are shown in Fig. 4, with points at the leading and trailing edges and one on the upper surface at the location of maximum thickness. The computed sensitivities show excellent agreement between CTSE and adjoint methods for all three cases.

*2. Incompressible Turbulent Flow Over a Refined NACA0012 Airfoil Section*

The second case is a refined NACA0012 airfoil mesh constructed similar to the coarse mesh by extruding a two-dimensional NACA0012 airfoil geometry. The wall spacings near the viscous surface have a normalized distance of $1.0E^{-5}$ which is appropriate for a laminar boundary layer. The airfoil portion of the mesh is shown in Fig. 6.

An incompressible turbulent flow solution was computed on this mesh with a chord based Reynolds number of $1.0E^{6}$, and a inflow angle of $5.0^{o}$. The Spalart-Allmaras turbulence model was also used here to compute the eddy viscosity. Sensitivities were computed using both forward CTSE and the adjoint method and are presented in Table 5. The design variables compared in Table 5 are: angle of attack ($\alpha$), and four surface node perturbations in the y-coordinate. The locations of the sample surface points are at the leading and trailing edges and at the top and bottom surface points at the location of maximum thickness. These points are labeled in Fig. 5.

**Table 3 Compressible 2nd Order Laminar: Sensitivity Comparison on Coarse, NACA0012 Airfoil Mesh.**

| Term | Forward CTSE | Adjoint | $\mathrm{Error}_{rel}$ |
|---|---|---|---|
| $dL/d\alpha$ | $4.8290302548364E^{-2}$ | $4.8290302528E^{-2}$ | $4.141E^{-10}$ |
| $dL/d\chi_{y,2528}$ | $-4.7797862766137E^{-2}$ | $-4.77978627641E^{-2}$ | $2.092E^{-10}$ |
| $dL/d\chi_{y,2537}$ | $1.7976576836922E^{-1}$ | $1.797657683707E^{-1}$ | $4.325E^{-12}$ |
| $dL/d\chi_{y,2541}$ | $-5.6316689338406E^{-2}$ | $-5.63166893346E^{-2}$ | $1.775E^{-10}$ |

**Table 4 Incompressible 2nd Order Turbulent: Sensitivity Comparison on Coarse NACA0012 Airfoil Mesh.**

| Term | Forward CTSE | Adjoint | $\mathrm{Error}_{rel}$ |
|---|---|---|---|
| $dL/d\alpha$ | $1.057700178815E^{0}$ | $1.057700178814E^{0}$ | $1.615E^{-12}$ |
| $dL/d\chi_{y,2528}$ | $-1.3289159690006E^{0}$ | $-1.3289159690007E^{0}$ | $1.279E^{-13}$ |
| $dL/d\chi_{y,2537}$ | $4.66574361639E^{0}$ | $4.66574361640E^{0}$ | $1.799E^{-12}$ |
| $dL/d\chi_{y,2541}$ | $-1.6221132170704E^{0}$ | $-1.6221132170709E^{0}$ | $3.082E^{-13}$ |

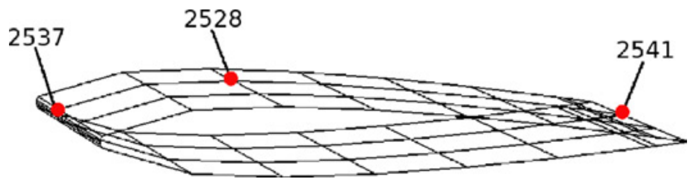

**Fig. 4 Sample Locations of Sensitivity Derivatives on Coarse Airfoil Mesh.**

**Table 5 Incompressible 2nd Order Turbulent: Sensitivity Comparison on Refined NACA0012 Airfoil Mesh.**

| Term | Forward CTSE | Adjoint | $\text{Error}_{rel}$ |
|---|---|---|---|
| $dL/d\alpha$ | $4.41121612E^{-1}$ | $4.41121607E^{-1}$ | $1.360E^{-9}$ |
| $dL/d\chi_{y,860}te$ | $-1.2401370E^{+1}$ | $-1.2401731E^{+1}$ | $2.911E^{-5}$ |
| $dL/d\chi_{y,928}le$ | $-8.6023914E^{-5}$ | $-8.6023926E^{-5}$ | $-1.411E^{-7}$ |
| $dL/d\chi_{y,9716}btm$ | $5.2893985E^{-2}$ | $5.28939654E^{-2}$ | $3.809E^{-7}$ |
| $dL/d\chi_{y,9771}top$ | $5.0922894E^{-2}$ | $5.09228677E^{-2}$ | $5.243E^{-7}$ |

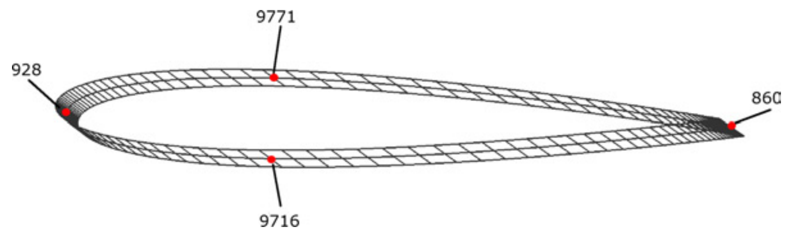

**Fig. 5 Sample Locations of Sensitivity Derivatives on Refined Airfoil Mesh.**

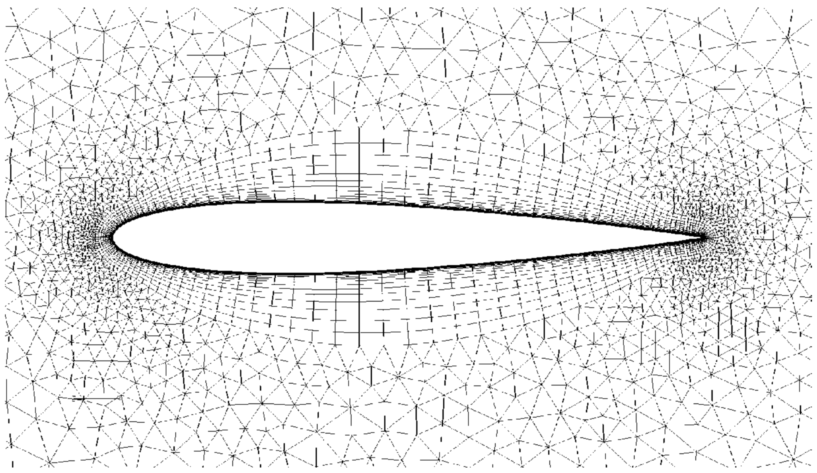

**Fig. 6 Closeup of Airfoil Section for Refined NACA0012 Mesh.**

## VII. Summary and Discussion

A method for computing adjoint solutions has been developed and implemented in an effort to reduce the maintenance requirements necessary to remain consistent with the capabilities of an evolving flow code. This method has been implemented and evaluated for accuracy, efficiency, and ease of implementation. The accuracy of the adjoint computed sensitivities are shown to be consistent with CTSE sensitivities for suitably converged flow solutions. A number of issues related to efficiency have been identified and solutions incorporated into the present implementation.

Targeted differentiation of routines, in conjunction with a means for computing function derivatives using simultaneous perturbations on localized computational stencils comprise the primary contributions to increased efficiency. Although efficiency is substantially improved using these techniques, the implementation introduces a degree of complexity which hinders maintenance and extensibility. Efficiency and maintenance design requirements for the adjoint code may conflict with that of the flow solver. Resolution of this issue requires a cooperative approach to development, and a significant familiarity with the flow code on the part of the adjoint developer.

## References

[1]Hicks, R., Murman, E. M., and Vanderplaats, G. N., "An Assessment of Airfoil Design by Numerical Optimization," Tech. Rep. NASA TM X-3092, 1974.

[2]Baysal, O. and Eleshaky, M. E., "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations," *Journal of Fluids Engineering*, Vol. 113, 1991, pp. 681–688.

[3]Hou, G. J., Marjou, V., Taylor, A. C., Korivi, V. M., and Perry, A., "Transonic Turbulent Airfoil Design Optimization with Automatic Differentiation in Incremental Iterative Forms," No. AIAA-1995-1962 in Collection of Technical Papers Pt.1 (A95-36501 09-34), June 1995, pp. 512–526.

[4]Newman, J. C., Taylor, A. C., and Barnwell, R. W., "Aerodynamic Shape Sensitivity Analysis and Design Optimization Using the Euler Equations on Unstructured Grids," No. AIAA-1997-2275, June 1997.

[5]Sherman, L. L., Taylor, A. C., Green, L. L., Newman, P. A., Hou, G. J., and Korivi, V. M., "First and Second Order Aerodynamic Sensitivity Derivatives via Automatic Differentiation with Incremental Iterative Methods," No. AIAA-1994-4262, Sept 1994, pp. 87–120.

[6]Frank, P. D. and Shubin, G. R., "A Comparison of Optimization Based Approaches for a Model Computational Aerodynamics Problem," *Journal of Computational Physics*, Vol. 98, 1992, pp. 74–89.

[7]Cabuk, H. and Modi, V., "Optimum Plane Diffusers in Laminar Flow," *Journal of Fluid Mechanics*, Vol. 237, 1992, pp. 373–393.

[8]Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D., "Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation," No. AIAA-1996-0094, Jan 1996.

[9]Sadrehaghighi, I., Smith, R. E., and Tiwari, S. N., "Grid Sensitivity and Aerodynamic Optimization of Geometric Airfoils," *Journal of Aircraft*, Vol. 32, No. 6, 1995, pp. 1234–1239.

[10]Burg, C. O. E. and Newman, J. C., "Computationally Efficient, Numerically Exact Design Space Derivatives via the Complex Taylors Series Expansion Method," *Computers and Fluids*, Vol. 32, Sept 2003, pp. 373–383.

[11]Pironneau, O., "Optimum Profiles in Stokes Flow," *Journal of Fluid Mechanics*, Vol. 59, 1973, pp. 117–128.

[12]Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," No. AIAA-1997-0643, Jan 1997.

[13]Nadarajah, S. K. and Jameson, A., "A Comparison of the Continuous and Discrete Adjoint Approach to Automatic Aerodynamic Optimization," No. AIAA-2000-0667, Jan 2000.

[14]Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization On Unstructured Meshes," No. AIAA-2001-0596, Jan 2001.

[15]Giles, M. B., Duta, M. C., and Muller, J. D., "Adjoint Code Developments Using The Exact Discrete Approach," No. AIAA-2001-2596, June 2001.

[16]Vendetti, D. A., *Grid Adaptation For Functional Outputs of Compressible Flow Simulations*, Ph.D. thesis, Massachusetts Institute of Technology, 2002.

[17]Vendetti, D. A. and Darmofal, D. L., "Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Two-Dimensional Inviscid Flows," *Journal of Computational Physics*, Vol. 176, 2002, pp. 40–66.

[18]Müller, J. D. and Giles, M. B., "Solution Adaptive Mesh Refinement Using Adjoint Error Analysis," No. AIAA-2001-2550, June 2001.

[19]Park, M. A., "Adjoint-Based Three-Dimensional Error Prediction and Grid Adaption," No. AIAA-2002-3286, June 2002.

[20]Nielsen, E. J., Lu, J., Park, M. A., and Darmofal, D. L., "An Exact Dual Adjoint Solution Method for Turbulent Flows On Unstructured Grids," No. AIAA-2003-0272, Jan 2003.

[21]Cusdin, P. and Müller, J.-D., "Generating Efficient Code With Automatic Differentiation," Tech. rep., ECCOMAS, 2004.

[22]Araya-Polo, M. and Hascoët, L., "Data Flow Algorithms In The Tapenade Tool For Automatic Differentiation," Tech. rep., INRIA Sophia-Antipolis, TROPICS team, 2004 Route des lucioles, BP 93, 06902 Valbonne, France, 2004.

[23]Giering, R., Kaminski, T., and Slawig, T., "Generating efficient derivative code with TAF: Adjoint and tangent linear Euler flow around an airfoil," Tech. rep., FastOpt, Martinistr 21, 20251 Hamburg Germany, 2004, submitted to Elsevier Science.

[24]Sheng, C., Hyams, D. G., Sreenivas, K., Gaither, J. A., Marcum, D. L., Whitfield, D. L., and Anderson, W. K., "Three-Dimensional Incompressible Navier-Stokes Flow Computations About Complete Configurations Using Multiblock Unstructured Grid Approach," No. AIAA-1999-0778, Jan 1999.

[25]Hyams, D. G., *An Investigation Of Parallel Implicit Solution Algorithms For Incompressible Flows On Unstructured Topologies*, Ph.D. thesis, Mississippi State University, May 2000.

[26]Sreenivas, K., Hyams, D. G., Mitchell, B., Taylor, L. K., Briley, W. R., and Whitfield, D. L., "Physics Based Simulations of Reynolds Number Effects in Vortex Intensive Incompressible Flows," *Symposium of Advanced Flow Management*, Applied Vehicle Technology Panel Meeting, Norway, May 2001.

[27]Sreenivas, K., Hyams, D., Mitchell, B., Taylor, L., Marcum, D., and Whitfield, D., "Computation of Vortex Intensive Incompressible Flow Fields," No. AIAA-2002-3306, June 2002.

[28]Burg, C., Sreenivas, K., Hyams, D. G., and Mitchell, B., "Unstructured Nonlinear Free Surface Flow Solutions: Validation and Verification," No. AIAA-2002-2977, June 2002.

[29]Blades, E., Sreenivas, K., and Hyams, D. G., "Arbitrary Overlapping Interfaces for Unsteady Unstructured Parallel Flow Simulations," No. AIAA-2003-0276, January 2003.

[30]Sreenivas, K., Cash, A. N., Hyams, D. G., and Taylor, L. K., "Computational Study of Propulsor-Hull Interactions," No. AIAA-2003-1262, January 2003.

[31]Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aerospatiale*, Vol. 1, 1994, pp. 5–21.

[32]Coakley, T. J. and Hsieh, T., "A Comparison Between Implicit and Hybrid Methods for the Calculation of Steady and Unsteady Inlet Flows," No. AIAA-85-1125, July 1985.

[33]Lyness, J. N. and Moler, C. B., "Numerical Differentiation Of Analytic Functions," *SIAM Journal of Numerical Analysis*, Vol. 4, 1967, pp. 202–210.

[34]Squire, W. and Trapp, G., "Using Complex Variables To Estimate Derivatives Of Real Functions," *SIAM Review*, Vol. 10, No. 1, March 1998, pp. 100–112.

[35]Newman, J. C., Anderson, W. K., and Whitfield, D. L., "Multidisciplinary Sensitivity Derivatives Using Complex Variables," Tech. Rep. MSSU-COE-ERC-98-08, Mississippi State University, July, 1998.

[36]Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A., *Sourcebook of Parallel Computing*, Elsevier Science, 2003.