AIAA 2007-3953

18th AIAA Computational Fluid Dynamics Conference
Jun 25–28, 2007, Miami, Florida

Towards Aircraft Design Using an Automatic Discrete Adjoint Solver

Charles A. Mader*, Joaquim R.R.A. Martins†
University of Toronto
Toronto, Ontario, Canada, M3H 5T6

Andre C. Marta‡
Stanford University
Stanford, CA 94305, USA

The ADjoint method is applied to a three-dimensional Computational Fluid Dynamics (CFD) solver to generate the sensitivities required for aerodynamic shape optimization. The ADjoint approach selectively uses Automatic Differentiation (AD) to generate the partial derivative terms in the discrete adjoint equations. By selectively applying AD techniques, the computational cost and memory overhead incurred by using AD are significantly reduced, while still allowing for the the accurate treatment of arbitrarily complex governing equations and boundary conditions. Once formulated, the discrete adjoint equations are solved using the Portable, Extensible Toolkit for Scientific computation (PETSc). With this approach, the computed adjoint vector can be used to calculate the total sensitivities required for aerodynamic shape optimization of a complete aircraft configuration. The resulting sensitivities are compared with complex-step derivatives to establish their accuracy. The tools developed are applied to an infinite wing test case to demonstrate the accuracy and efficiency of the method.

I. Introduction

Adjoint methods for sensitivity analysis involving partial differential equations (PDEs) have been known and used for over three decades. They were first applied to solve optimal control problems and thereafter were used to perform sensitivity analysis of linear structural finite-element models. The usefulness of the adjoint method, particularly in the case of design optimization, lies in the fact that it is an extremely efficient approach for computing the sensitivity of one function of interest with respect to many parameters. When using gradient-based optimization algorithms, the efficiency and accuracy of the sensitivity computations has a significant effect on the overall performance of the optimization. Thus, having an efficient and accurate sensitivity analysis is of paramount importance.

The first application of the adjoint method to fluid dynamics is due to Pironneau.[1] The method was then extended by Jameson to perform airfoil shape optimization[2] and since then it has been used to design laminar flow airfoils[3] and to optimize airfoils suitable for multipoint operation.[4] The method has also been extended to three dimensional problems, leading to applications in in aerodynamic[5] and aero-structural[6,7] shape optimization.

However, one of the complications in dealing with adjoint methods is that, as the adjoint equations are typically derived by hand, several assumptions are often made during the formulation of the adjoint

*M.A.Sc. Candidate, AIAA Student Member
†Assistant Professor, AIAA Member
‡Ph.D, AIAA Member

equations. In some cases this is simply to expedite the code development phase, while in others, it is required just to make the derivation of the adjoint possible. For example, the viscous effects of the flow might not be modeled. Furthermore, even when they are modeled,[8] the flow is typically assumed laminar, thus no turbulence model is used. In the latter case, the viscosity and heat transfer ratio are usually assumed to be independent of the flow and are kept constant when deriving the adjoint. This is often true even if the flow solver uses the RANS equations because the derivation of the adjoint typically assumes a constant eddy viscosity, thus ignoring the turbulence equations.

Any simplifications made while deriving the adjoint equations for the flow solver might have a significant impact on the adjoint-based gradients computed, as demonstrated by Dwight.[9] In order to tackle this deficiency, the authors have built an efficient design framework which uses an automated approach – involving AD tools – to develop the adjoint solver. This approach is described by Martins[10] and requires relatively little coding effort. It automatically produces the discrete adjoint equations for any flow solver and has been successfully tested for both the Euler equations[11] and the low magnetic Reynolds number magnetohydrodynamic (MHD) equations[12] using thousands of design variables. A similar approach using complex variables is discussed by Nielsen[13]

The eventual goal of this work is to extend this automated approach to develop the discrete adjoint equations for the RANS equations and to be able to compute gradients with respect to a variety of design variables based on the adjoint solution. The application of the method in the present work is limited to the mesh coordinate sensitivities of the CFD solver using the Euler equations. These sensitivities are then validated against results from the complex step derivative approximation[14] for a simple infinite wing test case.

II. Physical Model

A. Governing Equations

The governing equations used here are the three-dimensional Euler equations which can be written as follows,

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = 0, \tag{1}$$

where x_i are the coordinates in the i^{th} direction, and the state and the fluxes for each cell are

$$w = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \qquad f_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i H \end{bmatrix}. \tag{2}$$

A coordinate transformation to computational coordinates (ξ_1, ξ_2, ξ_3) is used. This transformation is defined by the following metrics,

$$K_{ij} = \left[\frac{\partial X_i}{\partial \xi_j}\right], \qquad J = \det(K), \tag{3}$$

$$K^{-1}_{ij} = \left[\frac{\partial \xi_i}{\partial X_j}\right], \qquad S = JK^{-1}, \tag{4}$$

where S represents the areas of the face of each cell projected on to each of the physical coordinate directions. Details about generalized coordinate transformations can be found in Hoffmann.[15]

The Euler equations in computational coordinates can then be written as,

$$\frac{\partial Jw}{\partial t} + \frac{\partial F_i}{\partial \xi_i} = 0, \tag{5}$$

where the fluxes in the computational cell faces are given by $F_i = S_{ij} f_j$.

In semi-discrete form the Euler equations are,

$$\frac{\mathrm{d}w_{ijk}}{\mathrm{d}t} + \mathcal{R}_{ijk}(w) = 0, \tag{6}$$

where $\mathcal{R}$ is the residual with all of its components (fluxes, boundary conditions, artificial dissipation, etc.). The resulting set of coupled ODEs (6) are marched in time using an explicit five-stage modified Runge–Kutta scheme to steady state.

It must be noted that while the derivation presented in following text is presented only for the Euler equations, because the ADjoint approach is only based on the existence of code that computes the *residual* of the governing equations, the procedure can be extended to the full Reynolds-averaged Navier–Stokes equations without modification. Note that the code for the residual computation is assumed to include the application of the required boundary conditions (however complex they may be) and any artificial dissipation terms that may need to be added for numerical stability.

III. Numerical Model

The NSSUS solver is a new finite-difference, higher-order solver that has been developed at Stanford University under the Advanced Simulation and Computing (ASC) program sponsored by the Department of Energy.[16] It is a generic node-centred, multi-block, multi-processor solver, tested for the Euler equations, and currently being extended to the Reynolds-Averaged Navier–Stokes equations. The finite-difference operators and artificial dissipation terms follow the work by Mattsson[17,18] and the boundary conditions are implemented by means of penalty terms, according to the work by Carpenter.[19,20] Despite being capable of performing computations up to eight-order accuracy, the implementation of the adjoint solver was restricted to second-order for simplicity. The extension to higher order should be straightforward to accomplish.

The internal discretization is straightforward and only requires the first neighbours in each coordinate direction for the inviscid fluxes and the first and second neighbours for the artificial dissipation fluxes, see Figure 1. However, the boundary treatment needs to be explained in more detail. As the finite-difference

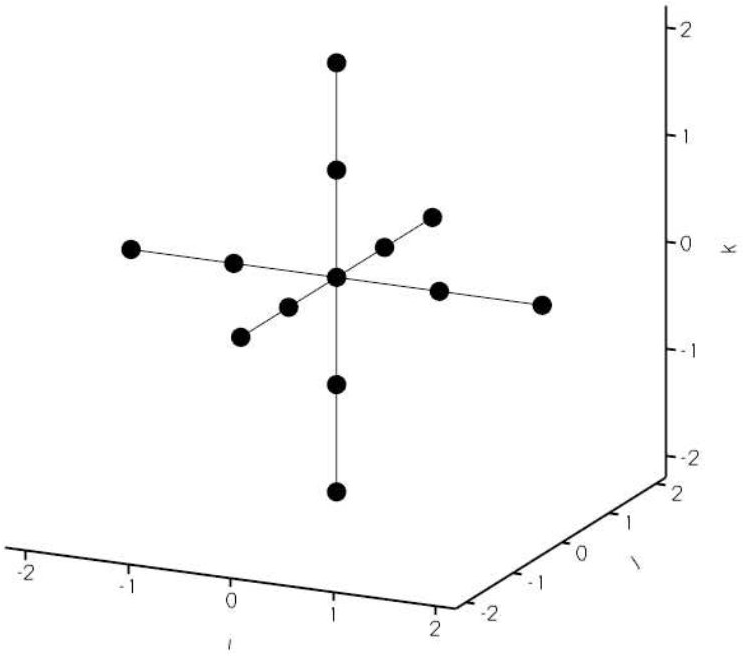


Figure 1. Stencil for the vertex-centred residual computation.

scheme only operates on the nodes of a block, one-sided difference formulae are used near block boundaries (be it a physical or an internal boundary). Consequently, the nodes on the interface of internal boundaries are multiply defined, see Figure 2.

These multiple instances of the same physical node are driven to the same value by means of a penalty term, i.e. an additional term is added to the residual $\mathcal{R}$ which is proportional to the difference between the instances. This reads

$$\mathcal{R}^i_{\text{blockA}} = \mathcal{R}^i_{\text{blockA}} + \tau(w^i_{\text{blockB}} - w^i_{\text{blockA}}) \tag{7}$$

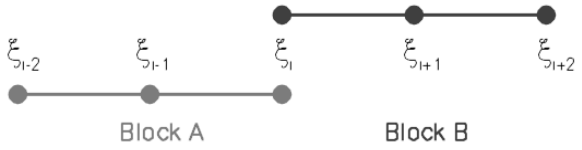


Figure 2. Block-to-block boundary stencil.

with a similar expression existing for $\mathcal{R}^i_{\text{blockB}}$. In equation (7), τ controls the strength of the penalty and is a combination of a user defined parameter and the local flow conditions, see Mattsson[17] for more details. Hence, the residual of nodes on a internal block boundary is a function of its local neighbours in the block and the corresponding instance in the neighbouring block.

The boundary condition (BC) treatment is very similar to the approach described above except that the penalty state used in equation (7) is now determined by the boundary conditions.

IV. Discrete Adjoint Formulation

The control theory approach has been used extensively in recent years for both aerodynamic shape optimization and aero-structural design[5] and has recently also been proved successful in MHD design[12,21] by the authors. This approach is well known for its capability to effectively handle design problems involving a large number of design variables and a small number of objective functions.

The sensitivities are obtained by solving a system of equations of size equivalent to the governing equations of the flow. When compared to traditional finite-difference methods, the adjoint approach enables large computational savings, at the expense of a more complex implementation.[22] This work employs a discrete adjoint formulation, meaning that the adjoint system of equations is obtained by differentiating the discretized form of the governing equations.

To put this discussion in context, consider a typical optimization problem. Let $\mathbf{U}(\alpha)$ be the set of all flow variables at discrete grid points arising from an approximate solution of the governing equations and α be the set of design variables which influence the flow. Further, let $J(\mathbf{U}(\alpha), \alpha)$ be a scalar function of both α and $\mathbf{U}(\alpha)$ which approximates the desired cost function. Then, in the context of control theory, the design problem can be posed as

$$\begin{aligned} \textit{Minimize} \quad & J(\mathbf{U}(\alpha), \alpha) \\ \textit{w.r.t.} \quad & \alpha \\ \textit{subject to} \quad & \mathcal{R}(\mathbf{U}(\alpha), \alpha) = 0 \\ & C_i(\mathbf{U}(\alpha), \alpha) = 0 \qquad i = 1, ..., m \end{aligned} \tag{8}$$

where $\mathcal{R}(\mathbf{U}(\alpha), \alpha) = 0$ represents the discrete flow equations and boundary conditions that must be satisfied and $C_i(\mathbf{U}(\alpha), \alpha) = 0$ are m additional constraints.

When using a gradient-based optimizer to solve the problem (8), the sensitivity of both the cost function J and the constraints C_i with respect to the design variables are required. If one constructs the following adjoint system of equations

$$\left[\frac{\partial \mathcal{R}}{\partial U}\right]^T \lambda = \left[\frac{\partial J}{\partial U}\right]^T, \tag{9}$$

and solves for the adjoint variables λ, the sensitivity of the cost function is given by

$$\frac{dJ}{d\alpha} = \frac{\partial J}{\partial \alpha} - \lambda^T \frac{\partial \mathcal{R}}{\partial \alpha}. \tag{10}$$

An additional adjoint system has to be solved for each additional constraint function C, which implies computing a new right-hand side for the system (9).

The sensitivity obtained from (10) can then be used to find the search direction of the gradient based optimization algorithm as shown in Figure 3.

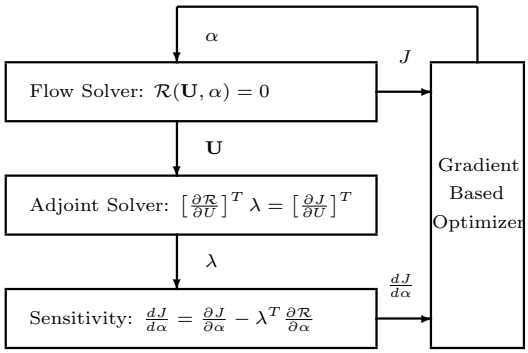


Figure 3. Schematic of the adjoint-based optimization algorithm.

The advantage of the adjoint approach can be seen from equation (10), which is independent of δU. This independence allows the gradient of J with respect to an arbitrarily large vector of design variables α can be determined without the need for additional flow-field evaluations.

A. Discrete Adjoint Solver

The theory of the implementation of the discrete adjoint solver for this work follows that of the authors previous work.[10–12] The discrete adjoint system of equations (9) was constructed by differentiating all the numerical fluxes that comprised the residual $\mathcal{R}_{ijk}$ in the discretized governing equations (6). This differentiation has been automated using an AD tool, namely Tapenade,[23–26] which is a non-commercial tool that supports Fortran 90.

The way the flow solver was coded, the residual is computed with loops over the nodes of each computational block in each processor. To make the implementation more efficient, it was necessary to re-write the flow residual routine such that it computed the residual for a single specified node. This required a considerable amount of copy-and-paste from the original code but this proved to be easy and did not significantly affect development. The re-engineered residual routine became a function with the residual `rAdj` as an output argument and the stencil of flow variables `wAdj` that affected that residual as an input argument

```
subroutine residualAdj(wAdj,rAdj,i,j,k).
```

Note that the general flow state $U(\alpha)$ is written as W in this code. The stencil of flow variables `wAdj` extended two nodes in each direction to allow for second-order discretization as shown in Figure 4.

The boundary condition penalty terms were moved to a separate routine that also had the boundary subface `mm` and the corresponding flow variable that was donor to the penalty state `wDonorAdj` as input parameters

```
subroutine residualPenaltyAdj(wAdj,dwAdj,mm,wDonorAdj,i,j,k)
```

This penalty residual routine was only called when the node `(i,j,k)` was located at a boundary face.

Having verified the re-written residual routines by comparing the residual values to the ones computed with the original code, these routines were then fed into Tapenade. Using the reverse mode in Tapenade, the automatically differentiated routines were

```
SUBROUTINE RESIDUALADJ_B(wAdj,wadjb,rAdj,radjb,i,j,k)
```

and

```
SUBROUTINE RESIDUALPENALTYADJ_B(wAdj,wadjb,rAdj,radjb,mm,wdonoradj,wdonoradjb,i,j,k)
```

These AD routines readily compute the entries of $\frac{\partial\mathcal{R}}{\partial W}$ since

$$\texttt{wAdjb}(\texttt{ii},\texttt{jj},\texttt{kk},\texttt{n}) = \frac{\partial\mathcal{R}(i,j,k,m)}{\partial W(i+ii,j+jj,k+kk,n)}\,. \tag{11}$$

This allowed for an easy assembly of the flux Jacobian matrix $\frac{\partial\mathcal{R}}{\partial W}$.

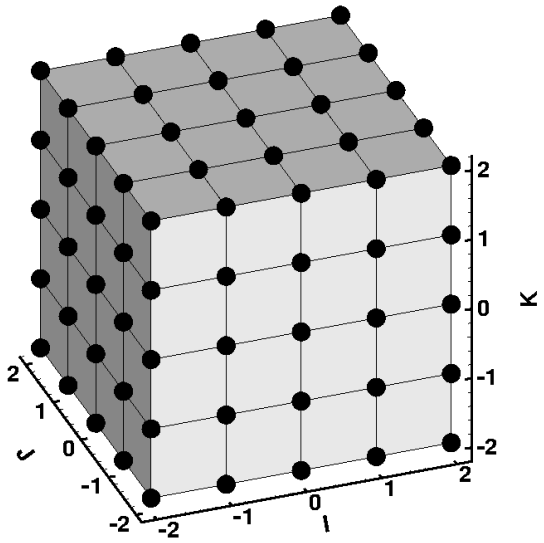


Figure 4. Flow variables stencil `wAdj` used in the AD.

In order to solve the large discrete adjoint matrix problem (9), the Portable, Extensible Toolkit for Scientific Computation (PETSc) [27–29] was used. This allows for MPI implementations, has several linear iterative solvers available and performs extremely well, provided a careful object creation and assembly procedure is followed.

B. Mesh sensitivities

The last term in equation 10, the partial sensitivity of the residuals with respect to the mesh coordinates, $\partial\mathcal{R}/\partial X(i,j,k)$, where $\alpha = X(i,j,k)$, was computed using an extension of the method used to compute the flux jacobian, $\partial\mathcal{R}/\partial W$. The stencil based approach still applies. However in this situation, the metric transformations – equations (3) and (4) – need to be taken into account in the residual computation. Once again the first two layers of adjacent nodes in each of the three coordinate directions are required for each nodal residual. This leads to the same size and shape stencil that was present in the flux Jacobian computation. Therefore, to streamline the code, the stencil based residual routine discussed above was modified to include the metric transformations, making it a function of both the states w and the grid coordinates $X(i,j,k)$. The modified routine was then re-differentiated, simultaneously, with respect to both w and $X(i,j,k)$, allowing for the computation of both sets of derivatives. Once again, the matrix is very sparse, so PETSc data structures are used to store the matrix.

The final partial derivative term required for the total sensitivity equation (10) is the explicit effect of the mesh coordinates on the force and moment coefficients. Unlike the residual routine, which depends on a limited stencil, the force and moment coefficients are a sum over the entire wall surface in the grid. Thus, the small stencil used to compute the flux Jacobian is no longer valid. In this case, to be general, the stencil must become the entire mesh. While this does cause some complications (additional memory requirements for example), it does provide some benefits. One key benefit is that a single reverse mode differentiation will now return all of the sensitivities required for one force or moment coefficient. While this does lead to slightly higher memory costs for this computation, it also leads to a very efficient derivative computation.

With all of the adjoint and partial sensitivity matrices and vectors created as PETSc objects, the adjoint system of equations (9) were solved using a Krylov subspace method, more specifically, the Generalized Minimum Residual (GMRES) method. Finally, the total sensitivity – equation (10) – was computed using the provided matrix-vector operation routines.

V. Results

For this work we are using an infinite wing test case. The mesh for this case, shown in Figure 5, is a single block, C-mesh with 9,471 nodes. The wing section is a standard airfoil section modelled with Euler wall

boundary conditions and has been extended to infinity at either end with symmetry boundary conditions. This case was run at a Mach number of 0.9 and and angle of attack of 5 degrees. A two dimensional cross section of the flow solution is shown in Figure 6.

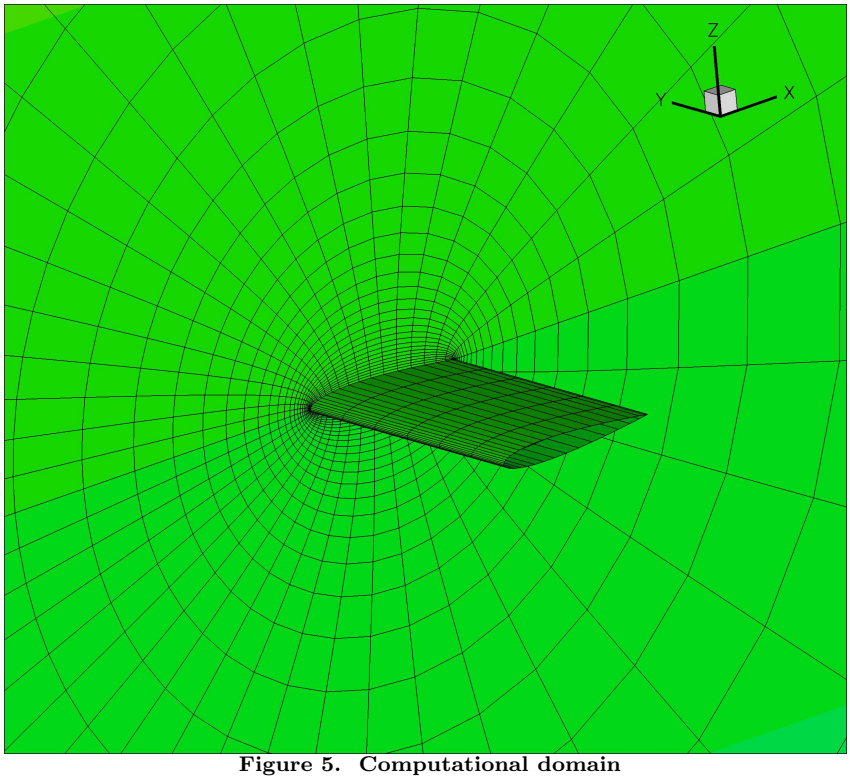


Figure 5. Computational domain

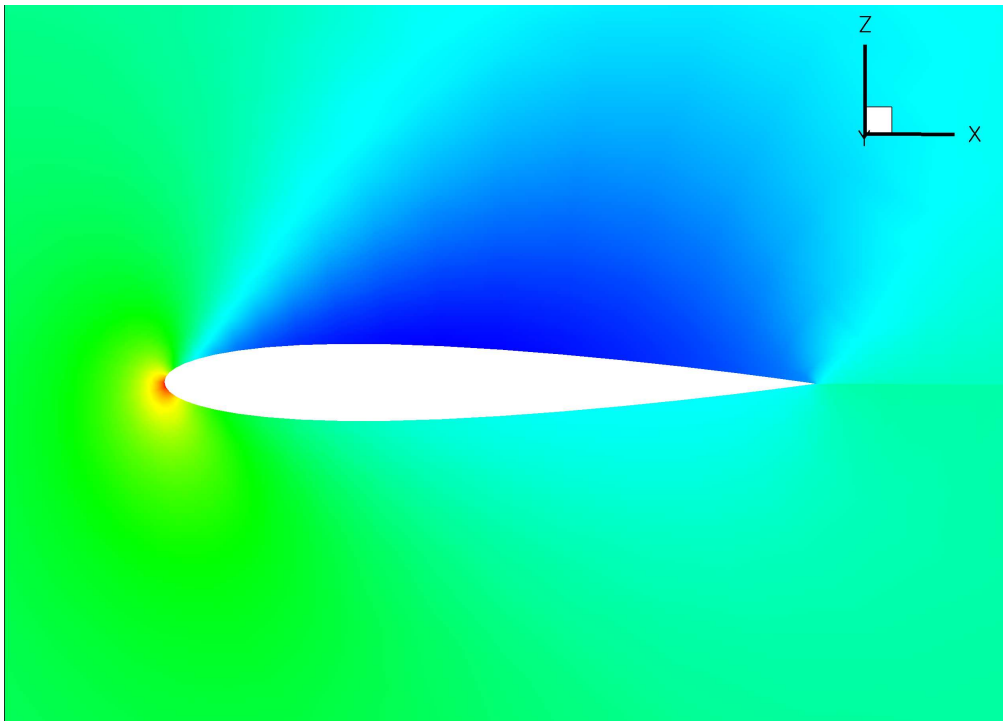


Figure 6. Density solution for a cross section of the wing

Figures 7 and 8 show a comparison of the sensitivities of drag and lift coefficients with repect to the

z-coordinates of the grid points for spanwise and chordwise slices through the wing. The spanwise section shows symmetry, which is expected for an infinite wing while the chordwise section through the wing shows much more variability, which is again expected. Note that both ADjoint sensitivities and complex step sensitivities are shown in the figure. The fact that for each case the lines for complex step and ADjoint are indistiguishable indicates good agreement. The L-2 norm of the error for the spanwise case is $3.40E-9$ for the drag coefficient sensitivities and $1.209E-8$ for the lift coefficient sensitivities. For the chordwise case, the L-2 norm of the error is $1.50E-6$ for the drag coefficient sensitivities and $2.129E-5$ for the lift coefficient sensitivities. These results are consistent with the accuracy requested of the solver, which was $1.0E-8$ for the spanwise case and $1.0E-6$ for the chordwise case. A more detailed accuracy comparison is shown in Table 1.

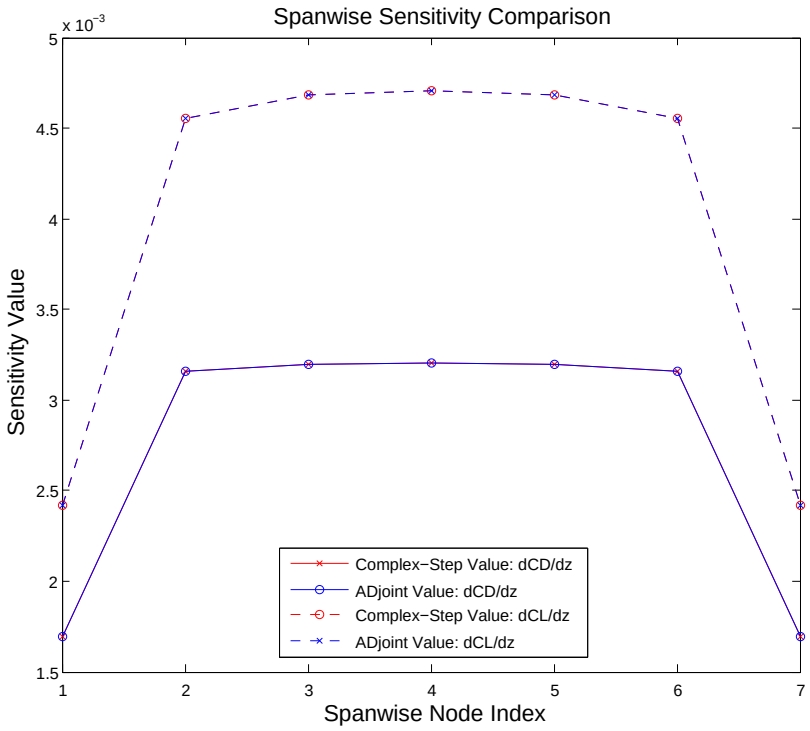


Figure 7. Spanwise comparison of sensitvity values with respect to the mesh z-coordinate

The same sensitivities are shown for the wing surface in Figure 9. Though the sensitivity values span the whole domain, only a section of the wing has been shown. The sensitivities shown here correspond to the lift and drag sensitivities of the wing.

The sensitivities obtained using the discrete adjoint approach were verified using the complex step results.[14] The comparison was made using four sample points on the surface of the wing, two on the top surface and two on the bottom surface, and the results are summarized in Table 1. The sensitivities shown are for a perturbation of the vertical coordinate of the wing surface. The values in Table 1 shows that the agreement between the two different approaches is excellent. The worst case shows 7 digits agreement and the best case shows as many as 11 digits agreement. When considering the fact that the solver itself was only convereged to $1.0E^{-10}$, this is excellent agreement.

In addition to the accuracy verification, timing results were obtained to ensure the efficiency of the

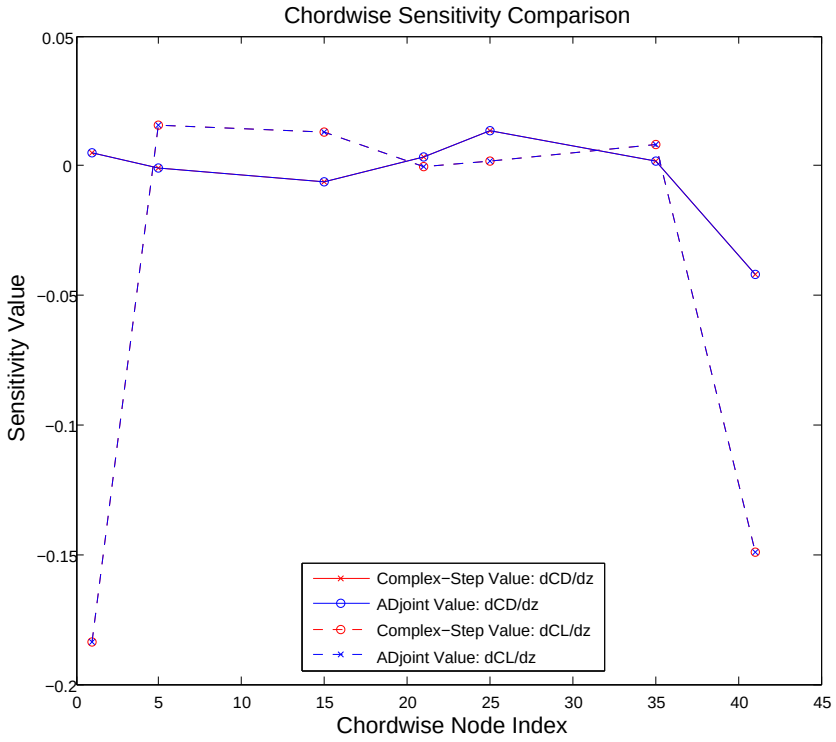
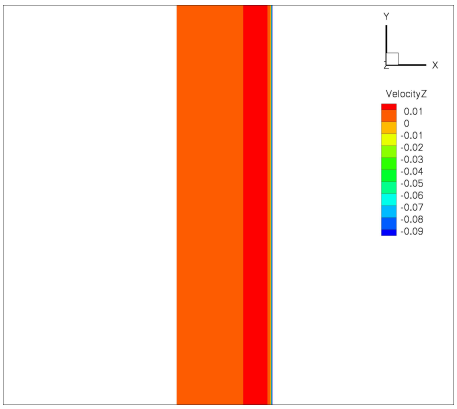
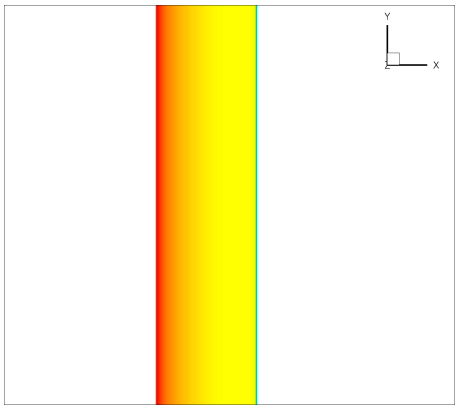


Figure 8. Chordwise comparison of sensitvity values with respect to the mesh z-coordinate



(a) Z coordinate sensitivity $J = C_L$: top view



(b) Z coordinate sensitivity $J = C_D$: top view

Figure 9. Sensitivity with respect to the mesh z-coordinates.

Node Index	Coefficient	ADjoint	Complex Step
10,1,3	C_L	1.03737827*205E-002*	1.03737827*112E-2*
	C_D	-4.2921761*724E-003*	-4.2921761*667E-3*
	C_{Mx}	-3.44421800*547E-002*	-3.44421800*306E-2*
	C_{My}	-3.9976406*323E-002*	-3.9976406*273E-2*
	C_{Mz}	-1.59291147*470E-002*	-1.59291147*259E-2*
10,1,5	C_L	1.03737827*228E-002*	1.03737827*112E-2*
	C_D	-4.2921761*722E-003*	-4.2921761*667E-3*
	C_{Mx}	-4.5239521*327E-002*	-4.5239521*281E-2*
	C_{My}	-3.9976406*328E-002*	-3.9976406*273E-2*
	C_{Mz}	-2.551081*401E-002*	-2.551081*398E-2*
30,1,3	C_L	1.70505774*790E-003*	1.70505774*414E-3*
	C_D	6.21953590*692E-003*	6.21953590*701E-3*
	C_{Mx}	-7.1419627*827E-003*	-7.1419627*7110E-3*
	C_{My}	-1.825756264624*0E-002*	-1.825756264*434E-2*
	C_{Mz}	1.8512382304*326E-002*	1.8512382304*184E-2*
30,1,5	C_L	1.70505774*756E-003*	1.70505774*414E-3*
	C_D	6.21953590*7229E-003*	6.21953590*7011E-3*
	C_{Mx}	-1.07832034*589E-002*	-1.07832034*431E-2*
	C_{My}	-1.82575626*4730E-002*	-1.82575626*443E-2*
	C_{Mz}	2.986569888*9194E-002*	2.986569888*9621E-2*

Table 1. CFD mesh coordinate sensitivity verification: $\frac{dJ}{dX(i,j,k)}$.

approach. Table 2 shows the computational time breakdown of the ADjoint method for this test case. These timing results clearly demonstrate the efficiency of the method. The solution of the adjoint system take less than 3% of the computational time of the flow solution, while the computation of the partial derivative terms requires only 5%. Overall the total cost of obtaining the adjoint sensitivities is less than $1/10^{th}$ the cost of a flow solution. Putting this in context: to get the flow solver to converge, starting from the baseline solution every time a design variable was perturbed, took over $1\frac{1}{2}$ minutes. Extrapolating to include three degrees of freedom on all nodes, giving 28,413 design variables, it would have taken a month to obtain the same results that took mere seconds to compute with the adjoint method described.

VI. Conclusion

In this work we applied the ADjoint method to the NSSUS flow solver to generate the mesh coordinate sensitivities required to perform aerodynamic shape optimization. We validated the resulting sensitivities

	Infinite Wing
No. Nodes	9471
Flow solution	241.57
ADjoint	17.39
Breakdown:	
Setup PETSc Variables	0.03
Compute flux Jacobian	5.43
Compute grid partial	5.73
Compute RHS	0.00
Solve the adjoint equations	6.16
Compute the total sensitivity	0.04

Table 2. ADjoint computational cost breakdown (times in seconds)

by comparing them to complex step result and achieved a 7-11 digit agreement. The implementation has also been shown to be very efficient, with the total ADjoint solution taking only a small fraction of the time required for the flow solver. Finally, we showed some overall sensitivity distributions for an infinite wing case, as an example of the results that the ADjoint implementation on NSSUS can generate. These results also form the basis of the sensitivities required for aerodynamic shape optimization.

Acknowledgments

The first two authors are grateful for the funding provided by the Canada Research Chairs program and the Natural Sciences and Engineering Research Council. The third author is grateful for the Air Force Office of Scientific Research contract No. FA9550-04-C-0105 under tasks monitored by John Schmisseur.

References

[1]Pironneau, O., "On optimum design in fluid mechanics," *Journal of Fluid Mechanics*, Vol. 64, 1974, pp. 97–110.

[2]Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, No. 3, sep 1988, pp. 233–260.

[3]Driver, J. and Zingg, D. W., "Optimized Natural-Laminar-Flow Airfoils," *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2006, AIAA 2006-0247.

[4]Nemec, M. and Zingg, D. W., "Multipoint and Multi-Objective Aerodynamic Shape Optimization," *AIAA Journal*, Vol. 42, No. 6, June 2004, pp. 1057–1065.

[5]Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60.

[6]Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "High-Fidelity Aerostructural Design Optimization of a Supersonic Business Jet," *Journal of Aircraft*, Vol. 41, No. 3, 2004, pp. 523–530.

[7]Maute, K., Nikbay, M., and Farhat, C., "Coupled Analytical Sensitivity Analysis and Optimization of Three-Dimensional Nonlinear Aeroelastic Systems," *AIAA Journal*, Vol. 39, No. 11, 2001, pp. 2051–2061.

[8]Jameson, A., Pierce, N. A., and Martinelli, L., "Optimum Aerodynamic Design using the Navier Stokes Equations," *Theoretical and Computational Fluid Dynamics*, Vol. 10, Springer-Verlag GmbH, Jan. 1998, pp. 213–237.

[9]Dwight, R. P. and Brezillon, J., "Effect of Various Approximations of the Discrete Adjoint on Gradient-Based Optimization," *AIAA Paper* 2006-0690, Jan. 2006, Proceedings of the 44^{th} AIAA Aerospace Sciences Meeting & Exhibit, Reno, NV.

[10]Martins, J., Alonso, J., and van der Weide, E., "An Automated Approach for Developing Discrete Adjoint Solvers," *AIAA Paper* 2006-1608, May 2006, Proceedings of the 47^{th} AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Newport, RI.

[11]Martins, J., Mader, C. A., and Alonso, J., "ADjoint: An Approach for Rapid Development of Discrete Adjoint Solvers," *AIAA Paper* 2006-7121, Sept. 2006, Proceedings of the 11^{th} AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA.

[12]Marta, A. C. and Alonso, J. J., "High-Speed MHD Flow Control Using Adjoint-Based Sensitivities," *AIAA Paper* 2006-8009, Nov. 2006, Proceedings of the 14^{th} AIAA/AHI International Space Planes and Hypersonic Systems and Technologies Conference, Canberra, Australia.

[13]Nielsen, E. J. and Kleb, W. L., "Efficient Construction of Discrete Adjoint Operators on Unstructured Grids Using Complex Variables," *AIAA Journal*, Vol. 44, No. 4, 2006, pp. 827–836.

[14]Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, No. 3, 2003, pp. 245–262.

[15]Hoffmann, K. A., *Fundamental Equations of Fluid Mechanics*, Engineering Education System, 1996.

[16]ASC, "Advanced Simulation and Computing," http://www.llnl.gov/asc, 2007.

[17]Mattsson, K. and Nordström, J., "Summation by Parts Operators for Finite Difference Approximations of Second Derivatives," *Journal of Computational Physics*, Vol. 199, No. 2, Sept. 2004, pp. 503–540.

[18]Mattsson, K., Svärd, M., and Nordström, J., "Stable and Accurate Artificial Dissipation," *Journal of Scientific Computing*, Vol. 21, No. 1, Aug. 2004, pp. 57–79.

[19]Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-Stable Boundary Conditions for Finite-Difference Schemes Solving Hyperbolic Systems: Methodology and Application to High-Order Compact Schemes," *Journal of Computational Physics*, Vol. 111, No. 2, April 1994, pp. 220–236.

[20]Carpenter, M. H., Nordström, J., and Gottlieb, D., "A Stable and Conservative Interface Treatment of Arbitrary Spatial Accuracy," *Journal of Computational Physics*, Vol. 148, No. 2, Jan. 1999, pp. 341–365.

[21]Marta, A. C. and Alonso, J. J., "Discrete Adjoint Formulation for the Ideal MHD Equations," *AIAA Paper* 2006-3345, June 2006, Proceedings of the 3^{rd} AIAA Flow Control Conference, San Francisco, CA.

[22]Martins, J. R. R. A., Alonso, J. J., and Reuther, J. J., "Complete Configuration Aero-Structural Optimization Using a Coupled Sensitivity Analysis Method," *AIAA Paper* 2002-5402, Sept. 2002, Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA.

[23]Dervieux, A., Hascoet, L., Pascual, V., Koobus, B., and Vazquez, M., "TAPENADE Web page," 2005, http://www-sop.inria.fr/tropics/tapenade.html.

[24]Hascoët, L. and Pascual, V., "TAPENADE 2.1 User's Guide," Technical report 300, INRIA, 2004.

[25]Hascoët, L., "TAPENADE: A tool for Automatic Differentiation of programs," *Proceedings of 4^{th} European Congress on Computational Methods, ECCOMAS'2004, Jyvaskyla, Finland*, 2004.

[26]Pascual, V. and Hascoët, L., "Extension of TAPENADE towards Fortran 95," *Automatic Differentiation: Applications, Theory, and Tools*, edited by H. M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, Lecture Notes in Computational Science and Engineering, Springer, 2005.

[27]Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Web page," 2001, http://www.mcs.anl.gov/petsc.

[28]Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, 2004.

[29]Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, Birkhäuser Press, 1997, pp. 163–202.