Formulation and Multigrid Solution of the Discrete Adjoint Problem on Unstructured Meshes

Dimitri J. Mavriplis

Department of Mechanical Engineering
University of Wyoming
Laramie, WY USA
mavripl@uwyo.edu

1 Introduction

The use of the adjoint equations is now well established for design-optimization problems in computational fluid dynamics (CFD) for the Euler and Navier-Stokes equations. In the continuous adjoint method approach, the original continuous governing equations are first linearized, and then discretized, while in the discrete adjoint approach, these two steps are performed in reverse order. Because the continuous approach affords more flexibility at the discretization stage, formulations involving reduced memory and cpu overheads have generally been achieved with this approach [1]. On the other hand, the discrete approach reproduces the exact sensitivity derivatives of the original discretization of the governing equations, which provides a verifiable consistency check on the final gradients produced by the adjoint solution [2]. If $f(w^*, D)$ represents a cost functional to be minimized in an optimization process, where D is a design variable, and w^* represents flow-field values which result from the solution of the discretized flow equations, given in residual form as $R(w, D) = 0$, the sensitivity derivative of f with respect to D is given by:

$$\frac{df}{dD} = \frac{\partial f}{\partial w}\frac{\partial w}{\partial D} + \frac{\partial f}{\partial D} \tag{1}$$

where the change in (converged) flow variables with respect to the design variable is given by differentiating the statement $R(w, D) = 0$, which yields:

$$\left[\frac{\partial R}{\partial w}\right]\frac{\partial w}{\partial D} = -\frac{\partial R}{\partial D} \tag{2}$$

Because each sensitivity derivative requires a new solution of equation (2) with a different right-hand side, a more efficient strategy for large numbers of design variables is to use the adjoint-based sensitivity equation:

$$\frac{df}{dD} = -\Lambda^T\frac{\partial R}{\partial D} + \frac{\partial f}{\partial D} \tag{3}$$

where the co-state variables Λ are given by the solution of the adjoint problem [1, 2]:

$$\left[\frac{\partial R}{\partial w}\right]^T \Lambda = \left(\frac{\partial f}{\partial w}\right)^T \tag{4}$$

The discrete adjoint approach described above benefits from the relative simplicity of implementation. In fact, if a linearization of the discrete equations is already available in the form of a Jacobian for use in a fully implicit scheme, the adjoint equations (including boundary conditions) are obtained by simply transposing the Jacobian matrix. Additionally, exact duality preserving iteration strategies can be constructed for the solution of equations (2) and (4), thus ensuring similar convergence rates between governing and adjoint equations [3, 4]. In order to take advantage of these properties, the formulation and solution efficiency of the discrete adjoint equations should be comparable to that achieved in current state-of-the-art continuous adjoint formulations. In this paper, we propose an efficient discrete-adjoint formulation and demonstrate a duality preserving agglomeration multigrid approach for solving the resulting equations.

2 Formulation of Discrete Adjoint

We seek the solution of the steady-state Euler or Navier-Stokes equations in two dimensions, which we denote as: $\mathbf{R}(w) = 0$. If this system is solved using a fully implicit (Newton) scheme, the iteration strategy may be written as: $\left[\frac{\partial R}{\partial w}\right] \Delta w = -\mathbf{R}(w)$. In this case, the adjoint equations are obtained simply by transposing the Jacobian as: $\left[\frac{\partial R}{\partial w}\right]^T \Lambda = \left(\frac{\partial f}{\partial w}\right)^T$, where f represents the cost function to be minimized, and Λ represents the co-state variables. In practice, storage of the exact Jacobian for a second-order accurate discretization involves an extended stencil and is therefore not practical. However, second order-accurate discretizations are most-often constructed in two nearest-neighbor passes, suggesting a similar approach for the construction of the discrete adjoint. For example, the artificial dissipation approach achieves higher-order accuracy by replacing differences of neighboring flow variables $w_k - w_i$ by differences in undivided Laplacians i.e. $q_i = \sum_{k=1}^{neighbors} w_k - w_i$. Therefore, rewriting the Jacobian using the chain rule,

$$\left[\frac{dR}{dw}\right] = \frac{\partial R}{\partial w} + \frac{\partial R}{\partial q}\frac{\partial q}{\partial w} \tag{5}$$

the discrete adjoint can be obtained as

$$\left[\frac{dR}{dw}\right]^T = \left[\frac{\partial R}{\partial w}\right]^T + \left[\frac{\partial q}{\partial w}\right]^T \left[\frac{\partial R}{\partial q}\right]^T \tag{6}$$

The first term represents contributions from convective terms, and in the case of the Navier-Stokes equations, physical dissipation terms (which operate on a nearest neighbor stencil for triangular elements). The last term is equivalent to contributions from a first-order artificial dissipation (applied to the q variables), thus constituting a symmetric matrix (neglecting variations in the interface coefficient matrix, which is evaluated at the Roe state [5]), while the third term represents the linearization (transposed) of the undivided Laplacian, which is a trivial matrix in this case containing unity entries for each edge of the mesh. Assuming this matrix need not be stored due to its simplicity, the evaluation of $\left[\frac{dR}{dw}\right]^T \Lambda$ makes use of existing (edge-based) flow-solver data-structures, and requires the storage of two matrices: $\left[\frac{\partial R}{\partial w}\right]^T$ which is equivalent to a nearest-neighbor stencil-based sparse matrix, and $\left[\frac{\partial R}{\partial q}\right]^T$ of which only half the matrix need be stored, thus resulting in 1.5 times the storage of a nearest neighbor 1st order Jacobian.

For a MUSCL type reconstruction scheme, the linearization can be cast in a similar form, where the residual $\mathbf{R}(w)$ now depends on the flow variables w, as previously, and on the computed gradients of these variables, which take the place of the q variables. The discrete adjoint equations are obtained in a similar fashion, where $\frac{\partial q}{\partial w}$ represents the linearization of the gradient formulation, which is again a matrix which need not be stored, since the matrix entries are grid metrics which are already stored for the gradient computations in the residual evaluation [6].

3 Duality Preserving Iterative Strategy

From equations (1) and (3), we obtain the duality relation:

$$\frac{\partial f}{\partial w}\frac{\partial w}{\partial D} = -\Lambda^T \frac{\partial R}{\partial D} \tag{7}$$

which must hold for the finally converged values of $\frac{\partial w}{\partial D}$ and Λ. However, iterative solution strategies for equations (2) and (4) which preserve relation (7) at each iteration for partially converged values of $\frac{\partial w}{\partial D}$ and Λ have been demonstrated [3, 4]. If equation (2) is solved as:

$$[P]\left(\frac{\partial w}{\partial D}^{n+1} - \frac{\partial w}{\partial D}^{n}\right) = -\frac{\partial R}{\partial D} - \left[\frac{\partial R}{\partial w}\right]\frac{\partial w}{\partial D}^{n} \tag{8}$$

where $[P]$ is the iteration matrix to be inverted (or preconditioner), the corresponding duality preserving iteration strategy for equation (4) is given by:

$$[P]^T\left(\Lambda^{n+1} - \Lambda^n\right) = \frac{\partial f}{\partial w} - \left[\frac{\partial R}{\partial w}\right]^T \Lambda^n \tag{9}$$

The right-hand side of equation (9) requires the evaluation of the adjoint residual, which is performed as a matrix-vector product, using the two-pass construction given by equation (6). A common strategy is to take $[P]$ as the first-order Jacobian of the flow residual. The $[P]$ matrix is then approximately inverted by splitting it into diagonal blocks $[D]$, and off-diagonal blocks $[O]$, and employing a Jacobi or Gauss-Seidel iteration, following:

$$[D]^T \left(\Lambda^{n+1} - \Lambda^n\right)^{k+1} = \frac{\partial f}{\partial w} - \left[\frac{\partial R}{\partial w}\right]^T \Lambda^n - [O]^T \left(\Lambda^{n+1} - \Lambda^n\right)^k \quad (10)$$

in the case of the adjoint problem, where $[D]^T$ refers to the block-diagonals of the $[P]^T$ matrix, $[O]^T$ refers to the off-diagonal blocks, and k denotes the Jacobi or Gauss-Seidel iteration counter. In order to conserve memory space, the individual blocks of the $[P]^T$ matrix need not be stored separately. Rather, they can be quickly reconstructed from the matrices used to evaluate the full second-order adjoint residual as:

$$\left[\frac{\partial R}{\partial w}\right]^T_{first-order} = \left[\frac{\partial R}{\partial w}\right]^T + \left[\frac{\partial R}{\partial q}\right]^T \quad (11)$$

where the matrices on the right-hand side of equation (11) correspond to those defined in equation (6). In practice, a separate copy of the $[D]^T$ block diagonals are factorized and stored in LU form, but the off-diagonal blocks $[O]^T$ are reconstructed as per equation (11) at each iteration. The Gauss-Seidel scheme provides faster convergence due to the ordered sweep across the mesh which employs latest available information. Additionally, both methods enable multiple sweeps before updating the adjoint residual, which incurs an additional expense, since this is evaluated in a two-pass strategy as described above. These Jacobi and Gauss-Seidel schemes may also be viewed as defect-correction approaches [7].

These iterative schemes are then used as the driver for an agglomeration multigrid strategy to accelerate the solution of the adjoint problem. The use of agglomeration multigrid for solving the Euler and Navier-Stokes equations on unstructured meshes is now well established [8]. A duality preserving agglomeration multigrid algorithm has been developed for solving the discrete adjoint equations on unstructured meshes. To preserve exact duality, the order of operations in the multigrid cycle must be reversed when applied to the adjoint equations [4]. For example, a multigrid cycle which employs one pre-smoothing and no post-smoothing iterations must be replaced by a cycle which uses no pre-smoothing and one post-smoothing iteration for the adjoint problem. Furthermore, the restriction and prolongation operators must be exact transposes of one-another.

4 Results

Figures 1 illustrates the unstructured mesh (7884 points) and the convergence of the discrete adjoint problem (c.f. equation (4)) on this mesh using a duality preserving agglomeration multigrid algorithm, compared with the original multigrid algorithm applied to the linearized flow equations (c.f. equation (2)). The freestream Mach number is 0.6 and the incidence is 1.25 degrees for this test case. Five multigrid levels are used, with 4 smoothing iterations on each grid level (two pre-smoothing, and two post-smoothing), using either a Jacobi or Gauss-Seidel iteration for the smoother. Rapid convergence to machine accuracy is observed and the Gauss-Seidel smoother is seen to deliver better overall multigrid convergence than the Jacobi smoother. In both cases, primal and adjoint problems converge at similar rates, which mirror the convergence of the agglomeration algorithm for the non-linear solution of the flow equations. The third part of the figure shows the difference between the sensitivity derivatives calculated by equation (2) and equation (4) for the lift with respect to vertical displacement of a surface grid point (at x=63% chord on the lower surface), as a function of the iteration number, for the duality preserving algorithm, and for a non-duality preserving algorithm where three smoothing passes were performed in the multigrid coarsening phase, and one pass in the refinement phase, for both primal and dual solvers. In this case, the values computed by the two formulations initially differ by approximately 1.e-02, and the difference decreases to machine zero as the respective problems converge, while the difference remains at machine zero for the duration of the iteration procedure for the duality preserving schemes. For this case, the final value of the sensitivity derivative was 0.2530, which is close to the value of 0.2534 computed using a finite-difference approach. Figure 2 depicts

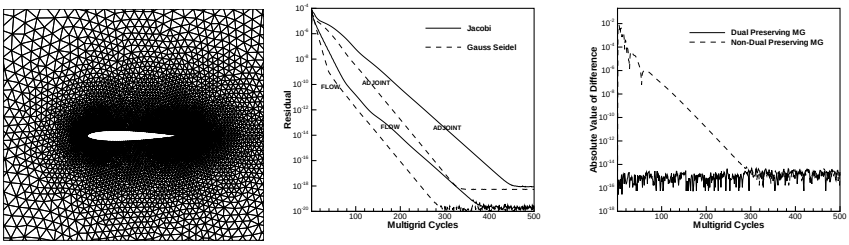

Fig. 1. Unstructured mesh (left), multigrid convergence of primal and adjoint equations (center), convergence of difference between both formulations for dual-preserving and non-preserving multigrid algorithms (right).

a similar experiment for the solution of the laminar Navier-Stokes equations on the same grid, at a Reynolds number of 500 with the same incidence, and a Mach number of 0.8. Using the Jacobi driven multigrid scheme, primal

and adjoint problems both display similar rapid convergence rates. The convergence of the computed lift sensitivity derivative (as defined previously) is depicted in the third part of the figure for both formulations using the non-duality preserving multigrid scheme, showing only minor deviations between formulations, and good final agreement with the value predicted by finite-difference. In future work, this technique will be extended to three dimensions and the effect of turbulence modeling will be incorporated.

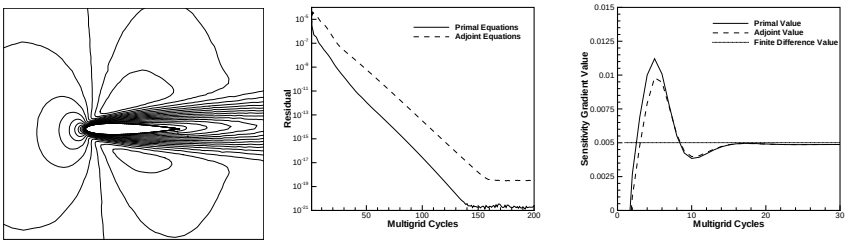

Fig. 2. Viscous flow field (left), convergence of Jacobi multigrid for primal and dual equations (center), convergence of sensitivity derivative for both methods (non-dual preserving algorithm)(right).

References

1. A. Jameson, J. J. Alonso, J. J. Reuther, L. Martinelli, and J. C. Vassberg. Aerodynamic shape optimization techniques based on control theory. AIAA Paper 98-2538, June 1998.
2. E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. AIAA Paper 2001-0596, January 2001.
3. M. B. Giles. Adjoint code developments using the exact discrete approach. AIAA Paper 2001-2596, June 2001.
4. E. J. Nielsen, J. Lu, M. A. Park, and D. L. Darmofal. An exact dual adjoint solution method for turbulent flows on unstructured grids. AIAA Paper 2003-0272, January 2003.
5. C. Hirsch. *Numerical Computation of Internal and External Flows, Volume II: Computational Methods for Inviscid and Viscous Flows.* Wiley, New York, NY, 1988.
6. T. J. Barth and S. W. Linton. An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation. AIAA paper 95-0221, January 1995.
7. D. J. Mavriplis. On convergence acceleration techniques for unstructured meshes. AIAA paper 98-2966, presented at the 29th AIAA Fluid Dynamics Conference, Albuquerque, NM, June 1998.
8. D. J. Mavriplis and S. Pirzadeh. Large-scale parallel unstructured mesh computations for 3D high-lift analysis. *AIAA Journal of Aircraft*, 36(6):987–998, December 1999.