4.0 Solution Methods for the Fundamental Equations

4.1 Nonlinear Fluid Analysis

An implicit time advancement algorithm is obtained by applying the backward Euler time-integration scheme to the unsteady term and linearizing the right hand side of the semi-discrete approximation in Eq.(3.5) yielding

$$\left[\frac{V_i}{\Delta t_i}\bar{\bar{I}} + \frac{\partial R_i}{\partial Q}\right]\Delta Q = -R_i \tag{4.1}$$

where R_i has been given previously in Eq.(3.6), and $\partial R_i / \partial Q$ is the inviscid Jacobian matrix. In the present work, the inviscid fluxes are evaluated using the flux vector splitting technique of Van Leer [117], and the Jacobian matrix may then be expressed in terms of the split Van Leer flux Jacobians as

$$\frac{\partial R_i}{\partial Q} = \sum_{j=\kappa(i)} \left(\frac{\partial E^{+}_{i,j}}{\partial Q^{-}_{f,j}}\frac{\partial Q^{-}_{f,j}}{\partial Q} + \frac{\partial E^{-}_{i,j}}{\partial Q^{+}_{f,j}}\frac{\partial Q^{+}_{f,j}}{\partial Q}\right)A_{i,j} \tag{4.2}$$

where the flux Jacobians are evaluated with variables interpolated to the j cell faces as discussed in section 3.1.3, and $\partial Q^{\pm}_{f,j} / \partial Q$ represents the cell center contribution from the interpolation or reconstruction scheme. When higher-order spatial accuracy is desired in Eq.(4.2), the form of the Jacobian becomes extremely complicated and the computational stencil very large. This is due to the upwind biased interpolation scheme that must be used for unstructured grids. The order of accuracy of the aerodynamic analysis, however, is determined from the evaluation of the residual vector, and a first-order Jacobian has been found sufficient to converge the implicit scheme. Since the left hand side operator is not an exact linearization of the residual, the ability to achieve quadratic convergence is now lost. It should be noted that an inexact linearization is not permitted for the aerodynamic sensitivity equation. This is because the underlying equations are linear and no approximations to the higher-order Jacobian matrix are allowed. This will be discussed in a subsequent section. Furthermore, in the current work two methods are available for solving Eq.(4.1) above: an iterative point Gauss-Seidel method and a preconditioned-Generalized Minimal Residual (GMRES) algorithm.

4.1.1 Point Gauss-Seidel

If the coefficient matrix in Eq.(4.1) is denoted as $[A]$, it is possible to write this matrix as the linear combination of the diagonal and off-diagonal matrices as

$$[A]^n = [D]^n + [O]^n \tag{4.3}$$

where n represents the iteration number of the nonlinear flow solution. Following the formulation of Ref. 141, the solution at step n consists of solving the linear systems of equations with a subiterative procedure. In the current work, the subiterative steps are accomplished by first updating all odd-numbered cells with a point-Jacobi iteration as

$$[D]^n \Delta Q_{odd}^m = -\left\{ R^n + [O]^n \Delta Q_{all}^{m-1} \right\} \tag{4.4}$$

where m represents the subiteration number. This step is followed by a point Gauss-Seidel iteration for the even-numbered cells

$$[D]^n \Delta Q_{even}^m = -\left\{ R^n + [O]^n \Delta Q_{odd}^m + [O]^n \Delta Q_{even}^{m-1} \right\} \tag{4.5}$$

where the most recent solution at subiteration m is used for the odd-numbered cells and the solution at a the previous subiteration m-1 is used for the even-numbered cells. This Gauss-Seidel *like* procedure is fully vectorizable. Once again, since an inexact linearization of the residual is used in the coefficient matrix, the ability to achieve quadratic convergence of the nonlinear system is lost, and it is therefore useless to solve the linear systems at each n iteration beyond the truncation error. In practice, it has been found that 15 to 20 subiterations produce the fastest rate of convergence for the nonlinear fluid equations [130].

4.1.2 Preconditioned-GMRES

The GMRES algorithm developed by Saad and Schultz [142], which has its genesis in the conjugate gradient methods, is an efficient technique for solving sparse nonsymmetric linear systems. For a more detailed discussion on the use of GMRES in CFD, the reader is directed to the literature [122,141,143-146]. Following Ref. 142, at each iteration n of the nonlinear flow analysis an approximation to the solution of the linear system may be denoted as ΔQ_o^n, and the subiterative solution may be advanced as

$$\Delta Q^m = \Delta Q_o^n + y^m \tag{4.6}$$

where GMRES(m) finds the best solution for y^m over the Krylov subspace $<v_1, [A]v_1, [A^2]v_1, \ldots, [A^{m-1}]v_1>$ by solving the minimization problem

$$\|r^m\| = \min_y \|v_1 + [A]y\| \tag{4.7a}$$

with

$$v_1 = [A]\Delta Q_o^n + R^n, \qquad r^m = [A]\Delta Q^m + R^n \tag{4.7b}$$

The GMRES algorithm forms an orthogonal basis $v_1, v_2, \ldots, v_{\tilde{m}}$ (referred to as search directions) that span the Krylov subspace by a modified Gram-Schmidt method. As the number of search directions increases, so do the memory requirements of this algorithm. Hence a restarted algorithm, denoted as GMRES(m,k), is utilized which discards the m

search directions and recomputes them every k restart cycles. Once again, a detailed explanation of this algorithm may be found in reference 142.

An important requirement, especially for the solution of the nonlinear fluid equations, is the selection of the preconditioner used in conjunction with GMRES. Preconditioning has been found to effectively cluster the eigenvalues of the coefficient matrix $[A]$ around a single value [145]. This clustering allows GMRES to more efficiently eliminate the errors associated with each eigenvalue and, thus, reduced the number of iterations required to solve the linear systems or even cause a divergent system to converge. Hence, the selection of a good preconditioner is vital. In the present work, an incomplete LU factorization with zero fill-in beyond the original sparsity pattern, referred to as ILU(0), is used. Details of this preconditioner may be found in references 145, 147, and 148.

4.1.3 Convergence Acceleration Techniques

For steady-state calculations, the governing equations are integrated from an arbitrary initial condition to a time-asymptotic state. Thus, when a steady-state solution is desired, it is typical to employ first order time accurate schemes and use non-time-like maneuvers in an attempt to accelerate the algorithm. The time step term, Δt_i, in Eq.(4.1) may therefore be viewed as a relaxation parameter and used as a means of conditioning the coefficient matrix. In the current work, two methods are used to specify the time step term: local time stepping [128,149] and switch-evolution relaxation [9,145].

Local time stepping is used for computations performed with the point Gauss-Seidel algorithm. For this method, the time step may be expressed as

$$\Delta t_i = CFL \frac{V_i}{\left(|u_i| + a_i\right) P_i^x + \left(|v_i| + a_i\right) P_i^y + \left(|w_i| + a_i\right) P_i^z} \tag{4.8}$$

where *CFL* is the Courant-Friedrichs-Lewy number, and P_i^x, P_i^y, P_i^z are the projected face areas in the *x*-, *y*-, and *z*-directions

$$P_i^x = \sum_{j=\kappa(i)} (\eta_x A)_{i,j} , \quad P_i^y = \sum_{j=\kappa(i)} (\eta_y A)_{i,j} , \quad P_i^z = \sum_{j=\kappa(i)} (\eta_z A)_{i,j} \tag{4.9}$$

where the summation is over the *j* faces of cell *i*. The *CFL* number is normally linearly ramped from 25 to 200 over 75 to 100 iterations, after which it remains 200.

For the computations which utilize the GMRES algorithm, the convergence characteristics of this preconditioned iterative scheme may also be improved by judicious specification of the time-step term. Moreover, with this fully implicit scheme, much higher *CFL* numbers are permissible and, therefore, a different technique is used. This technique, referred to as switch-evolution relaxation (SER), varies the time step term inversely proportional to the L_2-norm of the residual until a threshold is reached

$$\Delta t_i = min(\omega_{res}, \omega_{max}) \tag{4.10a}$$

with

$$\omega_{res} = 1.0 / \|L_2\| \tag{4.10b}$$

where ω_{max} is the limiting value of the time step term and taken as 1,800 for the computations herein. The specification of a threshold is necessary when the Jacobian in the coefficient matrix is not a consistent and exact linearization of the residual vector. Since the approximation used for the left-hand-side coefficient matrix of Eq.(4.1) is the spatially first-order accurate Jacobian, switch-evolution relaxation must be used.

4.2 Linear Structural Analysis

The finite-element structural analysis program used in the present work has been documented in reference 150. Since the stiffness matrix for linear static structural analysis is symmetric and positive-definite, a Choleski factorization is used to solve the system of equations. This direct method utilizes a variable-band width storage scheme and takes advantage of column heights to reduce the number of operations in the Choleski factorization [152]. Further details of the solution algorithm may be found in reference 152. The solution to this system of equations produces the vector of nodal displacements. From this deformation field, element stresses can be computed. Derivations of the element stresses for the constant strain triangle membrane element and for truss members may be found in virtually any finite-element text [132-134].

4.3 Static Aeroelastic Analysis

An integrated static aeroelastic analysis procedure, which adopts a domain-decomposition approach, has been developed in the present work. Since a domain-decomposition approach is used, the disciplines must be coupled at the boundary interfaces. This coupling, referred to as aerodynamic-structural geometric coordination, requires the interdisciplinary transfer of loads from the aerodynamic analysis to the structural finite-element model; the resulting structural deformations must then be represented on the aerodynamic mesh. Furthermore, the rate or frequency at which the discipline analyses are interacted is controlled via the introduction of interaction analysis control parameters. Proper specification of these parameters permits extremely efficient static aeroelastic analyses to be performed. The methods used in the current work to perform the aerodynamic-structural geometric coordination and for the interaction analysis control are discussed below.

4.3.1 Aerodynamic-Structural Geometric Interaction

To resolve the nonlinear fluid flow around an arbitrary object, both the surface and the volume around that surface must be discretized. For the structural analysis, the discretization encompasses the surface of the object and the volume interior to the surface. In practice, the nonlinear aerodynamic analysis requires a higher degree of resolution than linear structural analysis. Therefore, coordination between the fluid and the structure becomes an important concern in the aeroelastic analysis of a flexible body. This discrepancy in resolution is shown in Fig. 4.1, which illustrates the structural finite-element and unstructured CFD surface meshes for a transport wing.

In performing aeroelastic analysis to determine the static equilibrium position of a wing, structural properties may be lumped into sectional quantities, and a reduced resolution model (e.g., classical beam theory [153]) can be used. When element stresses are required for constraints and when structural optimization is performed, a more detailed model is necessary. Because multidisciplinary analysis and optimization is the ultimate focus of this work, a detailed model is used, even though only the static equilibrium position is sought.

The interaction between the fluid and the structure is accomplished by lumping the aerodynamic forces at the surface structural nodes, using a bilinear interpolation method, and applying them directly to the jig shape. An alternative to this load-lumping procedure, which has been discussed in Chapter 1, is to introduce a virtual surface between the aerodynamic and structural models. During the course of this study a method for transferring interface information via a virtual surface, based on a Bezier surface formulation, has been developed. Since the computations performed in the current research only seek the static equilibrium position, the load-lumping procedure is used exclusively. Guruswamy and Byun [77], however, have shown that the virtual surface method more accurately conserves the virtual work done during the course of an unsteady, dynamic calculation; but utilize the load-lumping procedure for static analyses.

After the static structural equilibrium equations have been solved and the deformations have been determined, the corresponding aerodynamic surface mesh must be updated. Because the structural deformations at each node are three dimensional, changes in cross-sectional properties (e.g., camber or thickness) of the wing depicted in Fig. 4.1 are possible. These cross-sectional deviations tend to be relatively small, as verified by the initial results for the wing used in the present study. Thus, it is assumed herein that the airfoil sections of the wing do not change. As a result, the in-plane deformations of the sections are limited to rigid-body translation and rotation (i.e., wing-section canting has been neglected here). Furthermore, for this wing these section deformations can be modeled approximately with simply an equivalent vertical translation and a twist angle (i.e., deformations that are consistent with beam theory). The translation and the twist at each section are determined by minimizing (in a least-squares sense) the discrepancy between this approximate form and the deformations predicted by the structural analysis. These approximated structural deformations are applied directly to the aerodynamic surface by using a simple polynomial regression. For the aeroelastic wing computations presented in the current work, a third-order polynomial regression is used for both the vertical translation and the twist to smoothly vary the aerodynamic surface mesh between the structural stations in the spanwise direction.

Once the surface is moved to conform to the structural deformations, the volume grid for the fluid solver must also be adapted to reflect these changes. A discussion of the mesh movement strategy used to update the interior volume grid is deferred to section 5.3.

4.3.2 Interaction Analysis Control

Interaction analysis control parameters are defined in order to study the convergence of the coupled system. Two methods are explored in the current work; both methods are based on the aerodynamic analysis because it has many more degrees of freedom than the structural analysis and involves the solution of nonlinear equations. The first method is based on the residual reduction of the aerodynamic analysis; the second specifies a fixed number of aerodynamic analysis iterations before the structural analysis is performed. In method 1 the structural analysis is interacted when the CFD residual criterion $\mathrm{Log}(R/R0) \leq N_s \quad \mathrm{Log}(F_n)$ is satisfied. Here, N_s represents the number of accumulated structural interactions, F_n determines the rate of interaction, and R0 is the initial CFD residual. The second method simply interacts the two systems at a prescribed constant rate after every I_{fs} CFD iterations. Note that the two methods become identical at the extremes. That is, methods 1 and 2 produce the same level of interaction when F_n and I_{fs} are unity (i.e., the disciplines interact every iteration) and as F_n approaches zero and I_{fs} becomes large (i.e., the disciplines interact after the aerodynamic analysis is essentially converged). The disciplines are not interacted when the root mean square (rms) of nodal displacements falls below a prespecified tolerance.

The ability to trim the wing with the angle of attack has also been developed and is accomplished by incorporating a feedback loop in the analysis. The particular feedback loop used is adopted from USM3D [153], where lifting-line theory is used to estimate $d\alpha/dC_L$. In turn, this derivative is used to compute the required change in angle of attack to achieve the specified lift. Because this derivative is approximate, the feedback loop is iterated until the computed lift coefficient and the specified lift coefficient are within a desired tolerance. The trim loop iteration is controlled by the same aerodynamic and structural interaction control parameters discussed above; thus, the angle of attack and structural deformations are updated simultaneously. However, this frequent angle-of-attack update must be under-relaxed or limited for the flexible-wing computations because of the large departure from the target lift in the early stages of the interaction. For the present results, the angle-of-attack update increment was limited.

4.4 Aerodynamic Sensitivity Analysis

The solution of Eq.(3.24c) poses the difficulty of solving an extremely large linear system for each design variable. Solving these systems, however, is made more tractable when these equations are decomposed and recast into what has been termed the incremental iterative form [15,44,139,140]. In this form, the sensitivity of the interior cells may be iteratively solved as

$$\begin{aligned}\tilde{A}\,\Delta^n\left(\frac{\partial Q_o}{\partial \beta_k}\right) &= -\left(\frac{\partial R}{\partial \beta_k} + \frac{\partial R}{\partial X}\frac{\partial X}{\partial \beta_k} + \frac{\partial R}{\partial Q_o}\left(\frac{\partial Q_o}{\partial \beta_k}\right)^n + \frac{\partial R}{\partial Q_b}\left(\frac{\partial Q_b}{\partial \beta_k}\right)^n\right) \\ &= -\left(\frac{\partial R}{\partial \beta_k} + \frac{\partial R}{\partial X}\frac{\partial X}{\partial \beta_k} + \frac{\partial R}{\partial Q}\left(\frac{\partial Q}{\partial \beta_k}\right)^n\right)\end{aligned} \qquad (4.11a)$$

$$\left(\frac{\partial Q_o}{\partial \beta_k}\right)^{n+1} = \left(\frac{\partial Q_o}{\partial \beta_k}\right)^{n} + \Delta^n\left(\frac{\partial Q_o}{\partial \beta_k}\right) \qquad (4.11b)$$

followed by the update of the boundary sensitivity

$$\left(\frac{\partial Q_b}{\partial \beta_k}\right)^{n+1} = -\left[\frac{\partial B}{\partial Q_b}\right]^{-1}\left\{\frac{\partial B}{\partial \beta_k} + \frac{\partial B}{\partial X}\frac{\partial X}{\partial \beta_k} + \frac{\partial B}{\partial Q_o}\left(\frac{\partial Q_o}{\partial \beta_k}\right)^{n+1}\right\} \qquad (4.12)$$

In the above, $\tilde{A}$ may now be any convenient approximation to the higher-order Jacobian which converges the linear system. This is because Eq.(4.11a) is now cast in *delta* form, with the physics contained in the right-hand-side vector. It has been found that the first-order Jacobian works well for use in the coefficient matrix of Eq.(4.11a), and is therefore used in the present work. The memory requirements for the first-order Jabobian, unlike that for the higher-order Jacobian, can be computed exactly for any unstructured mesh to be $5\times25\times$*ncell*. Comparing with the example cited in section 3.3.1 for the memory needs of the higher-order Jacobian for a 350,000 cell mesh, the first-order Jacobian only requires 43.75 mega-words of storage. This is approximately a factor of 10 reduction in memory.

Two particularly attractive features of the incremental iterative strategy are that (*i*) a more diagonally dominant matrix may be used to drive the solution of the linear systems (as opposed to the sometimes ill-conditioned higher-order Jacobian), and (*ii*) the higher-order Jacobian now resides on the right-hand-side of the equations and may be dealt with in an explicit manner. When in this form, only the k-vectors resulting from the matrix-vector product of $(\partial R/\partial Q)\ (\partial Q/\partial \beta_k)$ are of concern. Hence, CPU time and memory efficient methods for constructing the exact matrix-vector product can be exploited. To this end, Barth and Linton [26] have developed a new technique which permits the construction of the higher-order Jacobian-vector product using slightly less memory than that which would be required to evaluate the first-order Jacobian. This is accomplished by avoiding the need to assemble the full Jacobian prior to multiplication. With the details omitted (and the reader directed to Ref. 26 for the proof and further explanation of this method), this technique, applied to the desired matrix-vector product in Eq.(4.11a), may be symbolically written (using the notation of Eq.(4.2)) as

$$\frac{\partial R_i}{\partial Q}\frac{\partial Q}{\partial \beta_k} = \sum_{j=\kappa(i)} \left(\frac{\partial E^+_{i,j}}{\partial Q^-_{f,j}} \left(\frac{\partial Q}{\partial \beta_k}\right)^-_{f,j} + \frac{\partial E^-_{i,j}}{\partial Q^+_{f,j}} \left(\frac{\partial Q}{\partial \beta_k}\right)^+_{f,j} \right) A_{i,j} \tag{4.13}$$

where $(\partial Q/\partial \beta_k)^-_{f,j}$ and $(\partial Q/\partial \beta_k)^+_{f,j}$ are the vectors reconstructed from $\partial Q/\partial \beta_k$ using the same upwind interpolation scheme employed in the CFD analysis. That is, instead of using Q to obtain the solution at the cell interfaces, the sensitivity of the state vector $\partial Q/\partial \beta_k$ is used in the interpolation of equation 3.15. The resulting vector from this matrix-vector product is then scattered to the adjacent cells in the same manner as used for the nonlinear flow-residual calculation.

It should be noted that this method only requires the storage of the 5x5 flux Jacobians and the reconstructed vectors at the cell faces. Since this product is to be used in the sensitivity analysis, the memory which was utilized to compute the flux Jacobians for the first-order Jacobian and that used to reconstruct the CFD state vector may be reused. Thus, spatially first- or higher-order accurate sensitivity analysis may be performed with *virtually* the same memory as the CFD analysis. The only additional memory is due to the storage of the grid and metric sensitivity terms and the derivative $\partial R/\partial\beta_k$ <u>or</u> $(\partial R/\partial X)\,(\partial X/\partial\beta_k)$. (Note that for geometric design variables $\partial R/\partial\beta_k$ is zero, and for non-geometric design variables $(\partial R/\partial X)\,(\partial X/\partial\beta_k)$ is zero). Another attribute of this method is that the matrix-vector product computation only requires a fraction of the CPU time originally needed to assemble the full higher-order Jacobian; hence, the benefits are two-fold. The use of Barth and Linton's technique within the incremental iterative method has significant ramifications in that it makes large-scale optimizations of practical three-dimensional configurations possible [50].

4.5 Solution Methodologies

The linear systems resulting from the nonlinear aerodynamic analysis (i.e., Eq.(4.1)), and those from the aerodynamic shape sensitivity analysis (i.e., Eq.(3.25) or Eq.(4.11a)), may be solved using the same techniques. Within the optimization process, it is evident that the aerodynamic analysis not only consumes more CPU time (than the shape sensitivity analysis) to converge the nonlinear systems, it also is needed more frequently. Thus, solution algorithms which have high convergence rates are imperative when multiple analyses are to be performed. To this end, it has been found that a fully implicit iterative solver, such as preconditioned-GMRES [142], often out performs conventional explicit as well as classical iterative solvers [122,154,155]. However, the selection of the preconditioner used in conjunction with GMRES essentially governs the performance and memory required by this algorithm.

For two-dimensional, and relatively small three-dimensional (in terms of mesh size), configurations the GMRES algorithm may be utilized for both the aerodynamic and linear sensitivity analyses. With an incomplete LU factorization (ILU(0)) as a left preconditioner, 20 search directions with 3 restart cycles, i.e., GMRES(20,3), were found sufficient to converge the nonlinear fluid equations. For computations that permitted the use of GMRES, the pre-eliminated form of the sensitivity equation, given in Eq.(3.25), was solved. This linear system used an ILU(0) as a right preconditioner, required 20 search directions, and converged solutions were obtained in approximately 20 restart cycles for two-dimensional and 30 restart cycles for three-dimensional configurations. Due to the memory requirements associated with this algorithm and the storage of the spatially higher-order accurate Jacobian, it was not possible to utilize it for large three-dimensional configurations.

For large three-dimensional configurations the Gauss-Seidel iterative method is used to solve the fluid and sensitivity equations. The Gauss-Seidel method is also used for the aeroelastic analyses performed. The best performance of this algorithm, in terms of CPU time required to solve the nonlinear flow equations, was found through numerical experiments to be with 20 subiterations. Furthermore, when it was necessary to resort to the Gauss-Seidel algorithm due to memory considerations, the incremental iterative form of the sensitivity equation, given in Eq.(4.11a), was utilized. For this system of equations it was found that 10 subiterations were sufficient to converge the iterative scheme.

As a final note, it has been observed that the *ordering* of the cells in a grid has an affect on the rate of convergence of iterative solvers. With this in mind, many researchers [130,154,156] currently working with unstructured grids (which usually have a random ordering) have adopted renumbering algorithms such as Cuthill-McKee (CM) [157] or reverse-CM [158]. These algorithms attempt to reorder a given mesh such that the bandwidth of the coefficient matrix is minimized. In the present work, reordering is accomplished, during the initial preprocessing of the grid, using the Gibbs-Poole-Stockmeyer [159] algorithm.