AIAA

A FULLY-COUPLED NEWTON-KRYLOV ALGORITHM FOR AERODYNAMIC OPTIMIZATION

J. Gatsis and D. W. Zingg
Institute for Aerospace Studies
University of Toronto
4925 Dufferin St., Toronto
M3H 5T6, Canada

16th AIAA Computational Fluid Dynamics Conference
June 23–26, 2003/Orlando, FA

A FULLY-COUPLED NEWTON-KRYLOV ALGORITHM FOR AERODYNAMIC OPTIMIZATION

J. Gatsis * and D. W. Zingg †
Institute for Aerospace Studies
University of Toronto
4925 Dufferin St., Toronto
M3H 5T6, Canada

A fully-coupled Newton-Krylov algorithm is presented to solve aerodynamic design optimization problems. The fully-coupled system consists of the discretized flow equations, adjoint equations, and optimality conditions. An inexact Newton method is used to solve the discretized nonlinear system of equations. At each nonlinear iteration, the linear system is solved using the generalized minimal residual (GMRES) method. Grid-sequencing is used to accelerate start-up. A linesearch algorithm also governs the length of the Newton update at start-up. In contrast to a gradient-based optimization method, the flow system is converged only once in the fully-coupled algorithm. Inviscid, two-dimensional airfoil design problems are studied with objective functions that include inverse design and drag minimization. Results show that this algorithm is a fast option for aerodynamic optimization.

Introduction

Computational fluid dynamics plays an essential role in the design optimization of modern aircraft. Its use can dramatically reduce design cycle time. Conventional optimization algorithms are classified as either search-based or gradient-based. While search-based algorithms are able to handle a wide range of problems, they are slow because many flowfield solutions are required and they are not efficient near an optimum solution.[1] Gradient-based algorithms are more efficient near the solution; however these algorithms can also require a substantial number of flow solves. Two popular forms of gradient-based optimizers are based on sensitivity and adjoint approaches. The adjoint method can be implemented in a continuous[2] or discrete[3] fashion.

The equations describing the discrete adjoint formulation include the flow equations, the discrete adjoint equations, and the optimality conditions. These are equivalent to the Karush-Kuhn-Tucker (KKT) or necessary first-order optimality conditions.[4] Golub and Greif[5] describe some advanced techniques for solving these equations. One of the most popular methods for solving the KKT system is based on sequential quadratic programming (SQP). Work by Jou et. al.[6] uses a reduced space form of the classic Lagrange-Newton method. Biros and Ghattas[7] and Feng and Pulliam[8] also investigate the benefits of using SQP. Work by Sung and Kwon[9,10] involves tightly-coupling the system in order to reduce the cost of solving the flow system repeatedly. Ta'asan[11] solves the system by marching each subsystem separately with a pseudo-time parameter and using multigrid.

The objective of this paper is to introduce a new algorithm which solves the equations described in the discrete-adjoint formulation, including the adjoint equations, the flow equations, and the optimality conditions, in a fully-coupled manner. Since the equations are solved simultaneously, only a single flow solve is needed. The discretized system, now a system of nonlinear algebraic equations, is solved using an approximate Newton-Krylov algorithm. A linesearching algorithm is also used on the Newton update. Since the same equations are being solved as the discrete-adjoint method, if the optimum solution is unique, both algorithms yield the same result. Local minima are also the same for both methods.

The scope of this research is currently for two-dimensional, single-element airfoil configurations subject to inviscid flow. However, the algorithm is not restricted by any means to these problems. Furthermore, this research uses a finite-difference approach on structured grids. Successful research in quasi-one-dimensional nozzle shape design optimization has been presented previously.[12]

Governing Equations

Optimization Problem

We first define an objective function $\mathcal{J}(Q, X)$. This objective function can be based on the classic inverse design problem or more general design optimization problems, such as minimizing drag while maintaining

*Ph.D. Candidate, john@oddjob.utias.utoronto.ca.
†Professor, Senior Member AIAA, http://goldfinger.utias.utoronto.ca/dwz

a fixed lift. The variable $\mathcal{Q}$ defines the continuous flowfield state. The design variables for the optimization are given by the vector X, which contains the ordinates of control points representing the airfoil geometry. B-splines are used to represent the shape. The formulation follows the work of Nemec and Zingg.[13]

We wish to minimize $\mathcal{J}(\mathcal{Q}, X)$ subject to the flow equations:

$$\mathcal{R}(\mathcal{Q}, X) = 0. \tag{1}$$

Other constraints may be present and can be treated in a variety of ways. An example is a geometric thickness constraint. Rather than being included in the formulation directly[4] or as penalty terms in the objective function,[14] these constraints are incorporated into the linesearching algorithm.

Flow Equations

For this paper, $\mathcal{R}(\mathcal{Q}, X)$ is given by the steady Euler equations governing compressible inviscid flow, which are given in generalized curvilinear coordinates by

$$\mathcal{R}(\mathcal{Q}, X) = \frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = 0, \tag{2}$$

where

$$\mathcal{Q} = J^{-1} \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}. \tag{3}$$

The flux vectors are

$$\hat{E} = J^{-1} \begin{pmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U \end{pmatrix} \tag{4}$$

$$\hat{F} = J^{-1} \begin{pmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V \end{pmatrix} \tag{5}$$

where

$$U = \xi_x u + \xi_y v, \quad V = \eta_x u + \eta_y v \tag{6}$$

are the contravariant velocities. The variable J represents the metric Jacobian of the transformation. It is given by

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi). \tag{7}$$

The equations are non-dimensionalized by freestream quantities.

The equations are then discretized on a computational grid and artificial dissipation is added giving

$$(\delta_\xi E)_j + (\delta_\eta F)_j + D_j = 0, \qquad j = 1, \ldots, N \tag{8}$$

where the (δ_ξ) and (δ_η) operators indicate second-order central difference approximations to the derivatives in the ξ and η directions, respectively. N is the number of nodes on the computational grid. The boundary conditions and the nonlinear scalar dissipation model follow the method described by Pulliam.[15] The second- and fourth-difference dissipation coefficients are set here to $\kappa_2 = 0$ and $\kappa_4 = 0.01$ respectively. Thus the discretized flow equations and boundary conditions are

$$R(Q, X) = 0 \tag{9}$$

where Q is a vector containing the discretized flow variables.

Fully-Coupled Design Formulation

A Lagrange formulation is employed in deriving the fully-coupled algorithm. Using the objective function, $\mathcal{J}(Q, X)$, and the nonlinear system of flow equations, $R(Q, X)$, a new objective functional is defined as

$$\mathcal{L}(Q, \Phi, X) \equiv R(Q, X)^T \Phi - \mathcal{J}(Q, X), \tag{10}$$

where the variables Φ represent the discrete adjoint variables and are equivalent to Lagrange multipliers.

The stationary values of (10) with respect to Q, Φ, and X yield the governing equations for the discrete adjoint optimization problem. They include the original flow equations (9), the adjoint equations

$$(\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial \mathcal{J}}{\partial Q})^T = 0 \tag{11}$$

and the optimality conditions

$$(\frac{\partial R}{\partial X})^T \Phi - (\frac{\partial \mathcal{J}}{\partial X})^T = 0. \tag{12}$$

The methodology for deriving these equations can be found in Gunzburger.[16]

By combining these three equations, a coupled system of nonlinear equations is created, given by

$$\Gamma(K) = \begin{pmatrix} R \\ (\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial \mathcal{J}}{\partial Q})^T \\ (\frac{\partial R}{\partial X})^T \Phi - (\frac{\partial \mathcal{J}}{\partial X})^T \end{pmatrix} = 0 \tag{13}$$

where the state variables, K, are given by

$$K = \begin{pmatrix} Q \\ \Phi \\ X \end{pmatrix}. \tag{14}$$

We let the length of the discretized flow variables, Q, be n, where $n = 4N$. The length of Φ is also n. The length of the vector of design variables, X, is m, and $m \ll n$.

Objective Functions

For the inverse design problem, a desired pressure distribution is given by $C_{p_j}^\star$ for all nodes, j, on the

airfoil surface. The objective function is then defined as

$$\mathcal{J}(Q) \equiv \frac{1}{2}\sum_j (C_{p_j} - C_{p_j}^\star)^2. \tag{15}$$

This is a special case for $\mathcal{J}$, since it is a function of the flow variables only.

A more general design optimization problem is minimizing drag to a target value, while maintaining a fixed lift. A versatile objective function that can be used for this purpose is given by[17]

$$\mathcal{J}(Q,X) \equiv w_1\mathcal{J}_D + w_2\mathcal{J}_L \tag{16}$$

where

$$\mathcal{J}_D = \begin{cases} \left(1 - \frac{C_D}{C_D^\star}\right)^2 & \text{if } C_D \geq C_D^\star \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$\mathcal{J}_L = \left(1 - \frac{C_L}{C_L^\star}\right)^2 \tag{18}$$

and the weights, w_1 and w_2, are positive fractions that have a sum of unity. For all cases studied, $\mathcal{J}_D$ and $\mathcal{J}_L$ are equally-weighted with $w_1 = w_2 = \frac{1}{2}$.

Numerical Algorithm

An approximate Newton-Krylov algorithm is used to solve (13) for K. The system is solved using a block inversion algorithm which makes use of the generalized minimal residual (GMRES) Krylov subspace method. The startup procedure includes such features as grid sequencing and a linesearching algorithm which also is useful throughout the optimization.

Approximate Newton Method

At iteration n, an approximate Newton update for K_n is determined by solving

$$\tilde{\Theta}(K_n)\Delta K_n = -\Gamma(K_n) \tag{19}$$

for ΔK_n, where $\tilde{\Theta}(K_n)$ is an approximation to the Jacobian of the system. The updated state becomes

$$K_{n+1} = K_n + \alpha_n \Delta K_n. \tag{20}$$

If a full Newton step is taken, then $\alpha_n = 1$. If a linesearch is required, then $\alpha_n \in (0,1)$.

The exact system Jacobian is given by

$$\Theta(K) = \frac{\partial \Gamma(K)}{\partial K} \tag{21}$$

and is represented as a 3x3 matrix of rectangular and square blocks

$$\Theta(K) = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}, \tag{22}$$

where

$$\begin{aligned}
\theta_{11} &= \tfrac{\partial R}{\partial Q} \\
\theta_{12} &= \mathbf{0} \\
\theta_{13} &= \tfrac{\partial R}{\partial X} \\
\theta_{21} &= \tfrac{\partial}{\partial Q}[(\tfrac{\partial R}{\partial Q})^T\Phi - (\tfrac{\partial \mathcal{J}}{\partial Q})^T] \\
\theta_{22} &= (\tfrac{\partial R}{\partial Q})^T \\
\theta_{23} &= \tfrac{\partial}{\partial X}[(\tfrac{\partial R}{\partial Q})^T\Phi - (\tfrac{\partial \mathcal{J}}{\partial Q})^T] \\
\theta_{31} &= \tfrac{\partial}{\partial Q}[(\tfrac{\partial R}{\partial X})^T\Phi - (\tfrac{\partial \mathcal{J}}{\partial X})^T] \\
\theta_{32} &= (\tfrac{\partial R}{\partial X})^T \\
\theta_{33} &= \tfrac{\partial}{\partial X}[(\tfrac{\partial R}{\partial X})^T\Phi - (\tfrac{\partial \mathcal{J}}{\partial X})^T].
\end{aligned} \tag{23}$$

Approximations to System Jacobian

The terms θ_{13}, θ_{23}, and θ_{33} are calculated accurately using finite differences. The cost is minimal, since this requires only m evaluations of $\Gamma(K)$. The term θ_{31} is approximately equal to θ_{23}^T. Thus, in the algorithm θ_{31} is given the value of θ_{23}^T. θ_{32} is already calculated from θ_{13}^T.

θ_{11}, θ_{22}, and θ_{21} are approximated in the system Jacobian. The first two terms are the flow Jacobian and its transpose. Their formulation follows the work of Nemec.[13] Finally, θ_{21} is approximated by dropping the term $\frac{\partial}{\partial Q}[(\frac{\partial R}{\partial Q})^T\Phi]$.

Linesearching Algorithm

The linesearch algorithm serves two purposes: First, it avoids erratic or large steps in the initial iterations of the optimization and second, it makes efficient steps as the optimizer nears the Newton zone of convergence. Once a Newton update is computed, the linesearch parameter, α_n, is set to 1. If the updated state satisfies all geometric constraints, such as minimum thickness constraints, and also satisfies

$$\| \Gamma(K_n + \alpha_n \Delta K_n) \|_2 \;<\; \beta_n \| \Gamma(K_n) \|_2, \tag{24}$$

then the update is permitted. However, if either condition is violated, then α_n is reduced by half, and so on. If the linesearching parameter reaches a predefined minimum tolerance (e.g. 0.01) then the update is allowed to proceed. The parameter β_n is set to 1 for the entire optimization. It exists to combat more complex problems in the startup procedure.

Once the optimization is within the Newton zone of convergence, then the linesearching algorithm generally returns a value of $\alpha_n = 1$. More sophisticated linesearching algorithms have been explored, such as a backtracking linesearch algorithm.[1] However the halving linesearch algorithm is the most robust for a variety of cases.

Block Inversion Algorithm

The block inversion algorithm is simply a block Gaussian elimination algorithm. It is derived in the appendix. Figure 1 outlines the algorithm in 13 steps. Steps 1 and 2 require the solution of $m+1$ linear systems with the flow Jacobian on the left hand side. Steps 7 and 8 require the solution $m+1$ linear systems with the transpose of the flow Jacobian on the left hand side. These solves are done approximately using GMRES. Step 11 is the only other step where a linear solve is required. Here, the system is solved exactly using LU-factorization, since the system is only size m.

GMRES and Preconditioning

For the systems with the flow Jacobian on the left hand side, a matrix-free GMRES algorithm is used to solve the system. However, a matrix-present GMRES algorithm is necessary for the systems that have the tranpose of the flow Jacobian on the left-hand side.

Without preconditioning, GMRES requires many iterations to converge, and possibly will not converge at all. Incomplete LU (ILU) factorization preconditioning is used to accelerate the convergence and to increase the robustness for the GMRES algorithm. A first-order approximation to the Jacobian is used to generate the preconditioner, with outside dissipation terms in the discretization stencil collapsed onto the main diagonal. This follows the work of Pueyo and Zingg.[18] The level of fill required for the ILU preconditioner is set for each grid individually. The algorithm is converged for 2-3 orders of magnitude.

Reordering

Apart from the flow equations, the adjoint equations, and the optimality conditions, along with their corresponding variables, Q, Φ, and X, which have been ordered in a specific manner to efficiently implement the block inversion algorithm, there is also an important nodal reordering used to accelerate the convergence of the GMRES algorithm. Reverse Cuthill-McKee (RCM)[19] reordering is used to decrease the bandwidth of the linear subsystems in the block inversion algorithm. The root node used for the RCM algorithm lies on the downstream boundary.

Start up

The final element to the fully-coupled algorithm that dramatically increases robustness and speed, is grid sequencing. At startup, the optimization problem is solved on coarser grids, and the results are passed on to finer grids. Three grid levels are used. Flow and adjoint variables are passed to finer grids using a simple averaging strategy. The design variable B-spline control points are passed directly to the next grid and the shape is interpolated for that grid. The coarser grids are generated by removing even-numbered nodes in each dimension from the next highest grid level. This makes the passing of information to finer grids a relatively simple task.

The other important element in the startup of the optimization is the linesearching method. Erratic or large steps in the initial iterations of the optimization are avoided using the robust halving linesearch algorithm.

Results

The CPU times reported in the following sections are obtained on a 667-MHz Alpha 21264 processor (SPECfp 2000 rating of 562 peak). The optimization for all cases is performed using a C-topology grid with 245×41 nodes. Two coarser grids are also used in the grid-sequencing startup that have 123×21 and 62×11 nodes respectively.

The restart value of GMRES is set to 60. For the coarsest grid, the ILU preconditioner requires a level of fill of 6. The finer grids require a fill of 7. GMRES is converged for 2 orders of magnitude on the coarse grid and 3 orders of magnitude on the finer grids. The absolute tolerance for GMRES is set to 10^{-14}. In all cases examined, GMRES does not reach the restart value and averages roughly 10-25 inner iterations.

Three cases are examined. The classic inverse design optimization problem is studied first. To ensure the existence of a unique, attainable, optimum solution, the target pressure distribution is taken from a geometry which is also at the same flight conditions and is interpolated using the same B-spline control point locations. The initial shape is a 15 B-spline control point approximation of the NACA0012 airfoil. Three control points at the leading edge and four control points at the trailing edge of the airfoil are fixed throughout the optimization. The remaining 8 control points are the design variables of the optimization. The target shape is a B-spline representation of the RAE2822 airfoil using the 8 design variables that are free; the leading and trailing edge control points remain fixed to the NACA0012 airfoil's values. The first and second cases are subsonic and transonic inverse design problems. The third case involves drag minimization at fixed lift. Here the initial shape is given by a NACA0012 airfoil. The same 8 design variables are used as in the inverse design cases.

Subsonic Inverse Design

The subsonic inverse design case is the first case used to test the fully-coupled algorithm. The flight conditions are a Mach number $M = 0.3$ and an angle of attack $\alpha = 2°$. The objective is to find an airfoil whose surface pressure distribution matches the pressure distribution of a predefined target airfoil. In this case,

the initial airfoil is the NACA0012 airfoil and the target airfoil is the RAE2822 airfoil. Figure 2(a) shows the initial and final shapes of the optimization. Figure 2(b) shows the initial and final pressure distributions.

The convergence of the flow, adjoint, and optimality conditions are shown in Figures 2(c), 2(d), and 2(e) respectively. The convergence of the flow system is the fastest, while the convergence of the optimality system is the slowest. It is believed that this is due to the fact the flow equations are exactly linearized, while the adjoint and optimality conditions are based on the derivative of the flow equations with respect to the state and design variables, which in some places are approximated. The convergence time is roughly 15 minutes, although practical tolerances are reached much earlier. The number of Newton iterations on the finest grid is nineteen.

The objective function history is given in Figure 2(f). The objective function is driven to machine zero, verifying that the target pressure distribution is reached.

Transonic Inverse Design

This second case is identical to the previous case, except the Mach number is increased to $M = 0.74$ making the flow regime transonic. The initial and final airfoils are shown in Figure 3(a). The initial, target, and final pressure distributions are shown in Figure 3(b). Since the dissipation parameter κ_2 is set to zero for all cases, oscillations exist in the vicinity of the shock.

The convergence of the flow, adjoint, and optimality conditions are shown in Figures 3(c), 3(d), and 3(e) respectively. The optimization time for this case is roughly 10 minutes, although practical results are obtained much earlier. The objective function history is shown in Figure 3(f). The target pressure distribution is matched correctly. Thirteen Newton iterations are needed on the finest grid.

Drag Reduction at Fixed Lift

The third case involves drag minimization while maintaining a fixed lift. The flight conditions are the same as the transonic inverse design case with $M = 0.74$ and $\alpha = 2°$. The initial shape is given by the NACA0012 airfoil. In this case, the final design can not be anticipated before the optimization is executed, as opposed to the inverse design cases tested earlier. The initial NACA0012 airfoil at the given flight conditions has lift and drag coefficients of $C_L = 0.4670$ and $C_D = 0.0281$ respectively. The target lift and drag coefficients are given by $C_L^\star = 0.4670$ and $C_D^\star = 0.0099$ respectively. The reduction in drag is 65%.

Figure 4(a) shows the initial and final airfoils. The corresponding initial and final pressure distributions are given in Figure 4(b). The shock is not eliminated. However the change in the pressure distribution is enough to dramatically reduce drag, while maintaining the initial lift.

The convergence of the flow, adjoint, and optimality conditions are shown in Figures 4(c), 4(d), and 4(e). The optimization time is approximately 42 minutes; practical results are available much earlier. Finally, the objective function history is given in Figure 4(f). This case requires sixty-eight Newton iterations.

Conclusions

A fully-coupled algorithm for aerodynamic design optimization has been presented. The algorithm is governed by the same equations as the discrete adjoint method; however it only requires a single flow solve. An approximate Newton-Krylov algorithm is used to solve the coupled system of equations, which includes the discrete flow equations, the discrete adjoint equations, and the optimality conditions. The algorithm provides a fast means for obtaining an optimum solution for airfoil shape design problems for Euler flows on two-dimensional structured grids and can be applied to more complex problems. One concern with respect to the fully-coupled approach is that many optimization problems involve multiple operating points. This could lead to very large systems.

References

[1]Pierre, D. A., *Optimization Theory with Applications*, Dover, 1986.

[2]Jameson, A., "Aerodynamic Design via Control Theory," NASA CR-181749, Institute for Computer Applications in Science and Engineering, 1988.

[3]Baysal, O. and Eleshaky, M. E., "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations," *J. Fluids Engineering*, Vol. 113, No. 4, 1991, pp. 681–688.

[4]Nocedal, J. and Wright, *Numerical Optimization*, Springer Verlag, 1999.

[5]Golub, G.H. and Greif, C., "Techniques for Solving General KKT Systems," Tech. rep., SCCM-00-05, 2000.

[6]Jou, W.H., Huffman, W.P., Young, D.P., Melvin, R.G., Bieterman, M.B., Hilmes, C.L., and Johnson, F.T., *Practical Considerations in Aerodynamic Design Optimization*, No. AIAA-95-1730-C, 1995.

[7]Biros, G. and Ghattas, O., "Parallel Newton-Krylov Methods for PDE-Constrained Optimization," *The SCXY Conference Series*, November 1999.

[8]Feng, D. and Pulliam, T.H., "Aerodynamic Design Optimization Via Reduced Hessian SQP With Solution Refining," Tech. rep., Research Institute for Advanced Computer Science, 1995.

[9]Sung, C. and Kwon, J. H., "Efficient Aerodynamic Design Method Using a Tightly Coupled Algorithm," *AIAA Journal*, Vol. 40, No. 9, September 2002, pp. 1839–1845.

[10]Sung, C. and Kwon, J. H., "Aerodynamic Design Optimization Using the Navier-Stokes and Adjoint Equations," *AIAA Journal*, May 2001.

[11]Ta-asan, S., *Trends in Aerodynamic Design and Optimization: A Mathematical Viewpoint*, No. AIAA-95-1731-CP, San Diego, CA, June 1995.

[12]Gatsis, J. and Zingg, D.W., "A Fully-Coupled Algorithm for Aerodynamic Design Optimization," *48th Annual Conference, 8th Aerodynamics Symposium*, Canadian Aeronautics and Space Institute, Toronto, Canada, May 2001.

[13]Nemec, M. and Zingg, D.W., "Newton-Krylov Algorithm for Aerodynamic Design Using the Navier-Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, June 2002, pp. 1146–1154.

[14]Nemec, M. and Zingg, D.W., "Towards Efficient Aerodynamic Shape Optimization Based on the Navier-Stokes Equations," AIAA Paper 2001-2532, June 2001.

[15]Pulliam, T. H., "Efficient Solution Methods for the Navier-Stokes Equations," Lecture Notes For The Von Karman Institute For Fluid Dynamics Lecture Series: *Numerical Techniques For Viscous Flow Computation In Turbomachinery Bladings*, Jan. 1986.

[16]Gunzburger, M.D., "Inverse Design and Optimization Methods - Introduction into Mathematical Aspects of Flow Control and Optimization," von Karman Institute for Fluid Dynamics Lecture Series 1997-05, April 1997.

[17]Nemec, M. and Zingg, D.W., and Pulliam, T.H., "Multi-Point and Multi-Objective Aerodynamic Shape Optimization," AIAA Paper 2002-5548, September 2002.

[18]Pueyo, A. and Zingg, D.W., "An Efficient Newton-GMRES Solver for Aerodynamic Computations," AIAA Paper 97-1955, 1997.

[19]Cuthill, E. and McKee, J., "Reducing the Bandwidth of Sparse Symmetric Matrices," *24th National Conference of the Association for Computing Machinery*, No. ACM P-69, Brandon Press, New York, 1969.

Appendix

Block Inversion Algorithm

We wish to solve the block linear system

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \Delta\vec{Q} \\ \Delta\vec{\Phi} \\ \Delta\vec{X} \end{pmatrix} = \begin{pmatrix} \vec{R} \\ \vec{S} \\ \vec{T} \end{pmatrix} \tag{25}$$

where bold face indicates matrices. Generally this linear system may be written as

$$\mathbf{J}\Delta\vec{K} = \vec{\Gamma}. \tag{26}$$

Assuming the system is of the form seen in the Jacobian of the fully-coupled algorithm for aerodynamic design optimization, the following simplifications may be done:

$$\mathbf{E} = \mathbf{A}^T \tag{27}$$
$$\mathbf{H} = \mathbf{C}^T \tag{28}$$
$$\mathbf{B} = \mathbf{0} \tag{29}$$

where $\mathbf{0}$ is a zero matrix of the same dimensions as $\mathbf{B}$.

The system is separated into three equations and, at the moment, only simplification (29) is applied. The resulting equations are

$$\mathbf{A}\Delta\vec{Q} + \mathbf{C}\Delta\vec{X} = \vec{R} \tag{30}$$
$$\mathbf{D}\Delta\vec{Q} + \mathbf{E}\Delta\vec{\Phi} + \mathbf{F}\Delta\vec{X} = \vec{S} \tag{31}$$
$$\mathbf{G}\Delta\vec{Q} + \mathbf{H}\Delta\vec{\Phi} + \mathbf{I}\Delta\vec{X} = \vec{T}. \tag{32}$$

Equations (30) and (31) are of equal size and are much larger then (32).

Intermediate variables are introduced along the way and are denoted by the ($'$) symbol. Isolating $\Delta\vec{Q}$ in (30) we have

$$\Delta\vec{Q} = \vec{R}' - \mathbf{C}'\Delta\vec{X} \tag{33}$$

where

$$\vec{R}' = \mathbf{A}^{-1}\vec{R} \tag{34}$$

and

$$\mathbf{C}' = \mathbf{A}^{-1}\mathbf{C}. \tag{35}$$

Substituting $\Delta\vec{Q}$ into (31) gives

$$\mathbf{E}\Delta\vec{\Phi} + \mathbf{F}'\Delta\vec{X} = \vec{S}' \tag{36}$$

where

$$\mathbf{F}' = \mathbf{F} - \mathbf{D}\mathbf{C}' \tag{37}$$

and

$$\vec{S}' = \vec{S} - \mathbf{D}\vec{R}'. \tag{38}$$

Substituting $\Delta\vec{Q}$ into (32) gives

$$\mathbf{H}\Delta\vec{\Phi} + \mathbf{I}'\Delta\vec{X} = \vec{T}' \tag{39}$$

where

$$\mathbf{I}' = \mathbf{I} - \mathbf{G}\mathbf{C}' \tag{40}$$

and

$$\vec{T}' = \vec{T} - \mathbf{G}\vec{R}'. \tag{41}$$

Isolating $\Delta\vec{\Phi}$ in (36) we have

$$\Delta\vec{\Phi} = \vec{S}'' - \mathbf{F}''\Delta\vec{X} \tag{42}$$

where

$$\vec{S}'' = \mathbf{E}^{-1}\vec{S}' \tag{43}$$

and

$$\mathbf{F}'' = \mathbf{E}^{-1}\mathbf{F}'. \tag{44}$$

Substituting $\Delta\vec{\Phi}$ into (39) gives

$$\mathbf{I}''\Delta\vec{X} = \vec{T}'' \tag{45}$$

where

$$\mathbf{I}'' = \mathbf{I}' - \mathbf{H}\mathbf{F}'' \tag{46}$$

and

$$\vec{T}'' = \vec{T}' - \mathbf{H}\vec{S}''. \tag{47}$$

Finally, (45) may be solved for $\Delta\vec{X}$ using a direct LU solve, since it is very small when compared to the overall system size. Once $\Delta\vec{X}$ is found, it may be substituted into (42) and (33) to find $\Delta\vec{\Phi}$ and $\Delta\vec{Q}$ respectively.

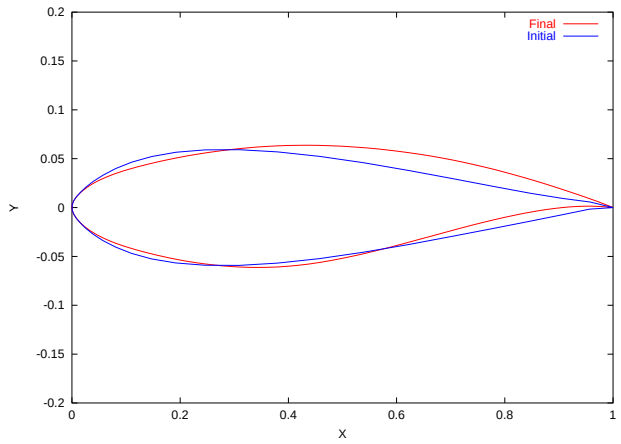
Although simplification (29) has been applied, this inversion process is likely more expensive than (26). However, once the simplifications (27) and (28) are also applied, especially (27), this inversion process

becomes more advantageous. In the block inversion process, three matrices need to be inverted: $\mathbf{A}$, $\mathbf{E}$, and $\mathbf{I}''$. The structure of $\mathbf{A}$ and $\mathbf{E}$ is such that an approximately-factored algorithm or Krylov solver such as GMRES can be used to solve the system. The reduction in cost is significant. Furthermore, since (27) is true, $\mathbf{E}^{-1} = (\mathbf{A}^{-1})^T$, and only one inversion process is required.

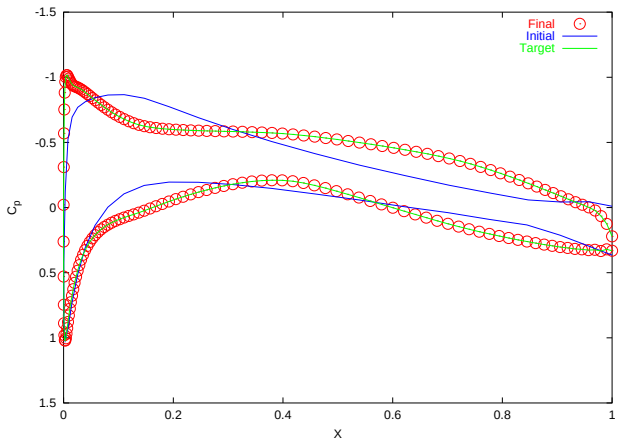
The block inversion process is summarized as follows:

1. Solve $\mathbf{A}\vec{R}' = \vec{R}$ for $\vec{R}'$
2. Solve $\mathbf{AC}' = \mathbf{C}$ for $\mathbf{C}'$
3. $\mathbf{F}' = \mathbf{F} - \mathbf{DC}'$
4. $\vec{S}' = \vec{S} - \mathbf{D}\vec{R}'$
5. $\mathbf{I}' = \mathbf{I} - \mathbf{GC}'$
6. $\vec{T}' = \vec{T} - \mathbf{G}\vec{R}'$
7. Solve $\mathbf{A}^T\vec{S}'' = \vec{S}'$ for $\vec{S}''$
8. Solve $\mathbf{A}^T\mathbf{F}'' = \mathbf{F}'$ for $\mathbf{F}''$
9. $\mathbf{I}'' = \mathbf{I}' - \mathbf{HF}''$
10. $\vec{T}'' = \vec{T}' - \mathbf{H}\vec{S}''$
11. Solve $\mathbf{I}''\Delta\vec{X} = \vec{T}''$ for $\Delta\vec{X}$
12. $\Delta\vec{\Phi} = \vec{S}'' - \mathbf{F}''\Delta\vec{X}$
13. $\Delta\vec{Q} = \vec{R}' - \mathbf{C}'\Delta\vec{X}$

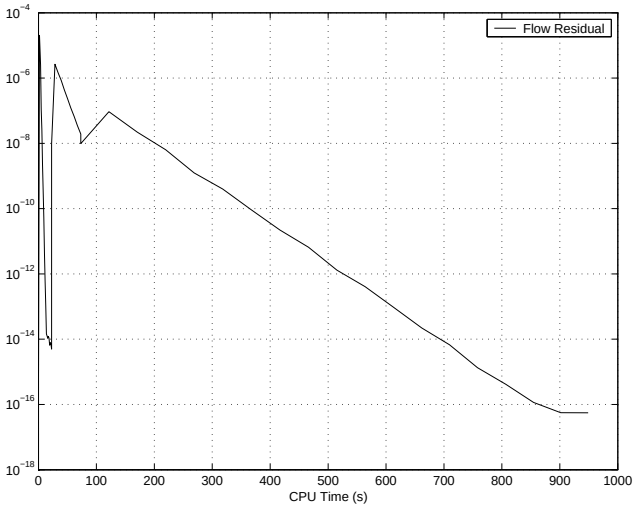
Fig. 1 Block Inversion Algorithm



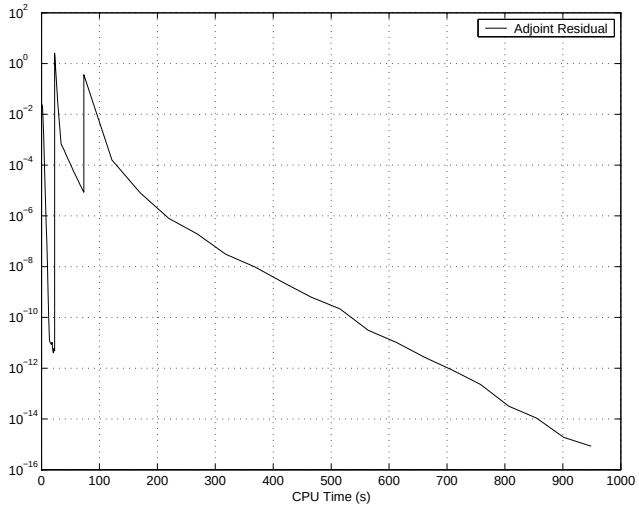
(a) Initial and final airfoils.



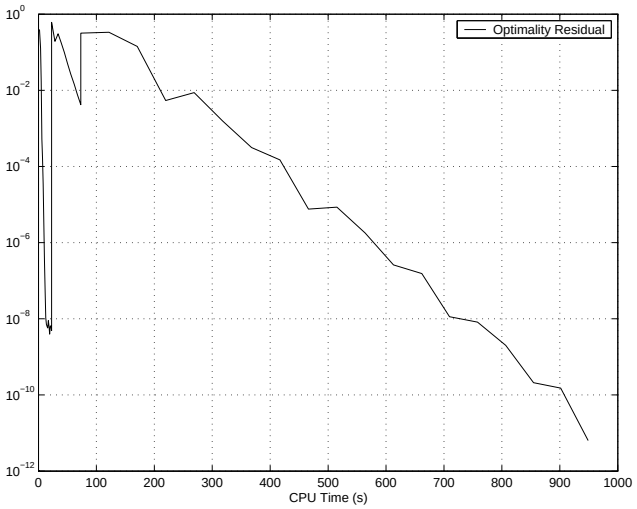
(b) Initial, target, and final pressure distributions.



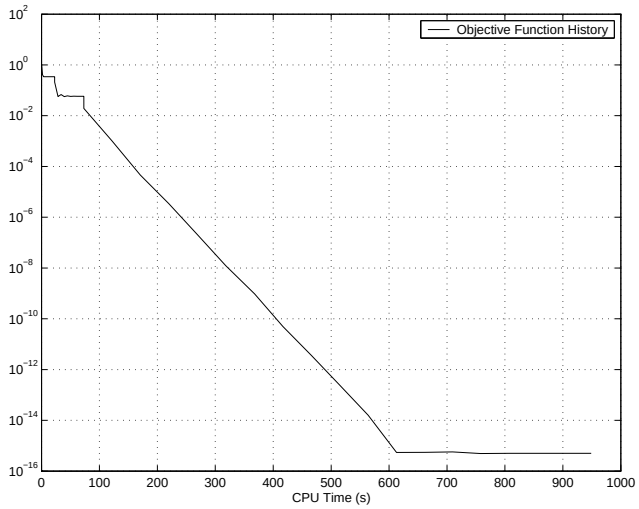
(c) Flow residual convergence.



(d) Adjoint residual convergence.

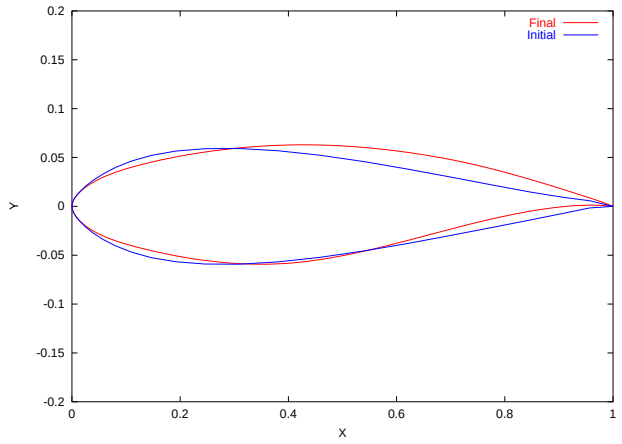


(e) Optimality residual convergence.

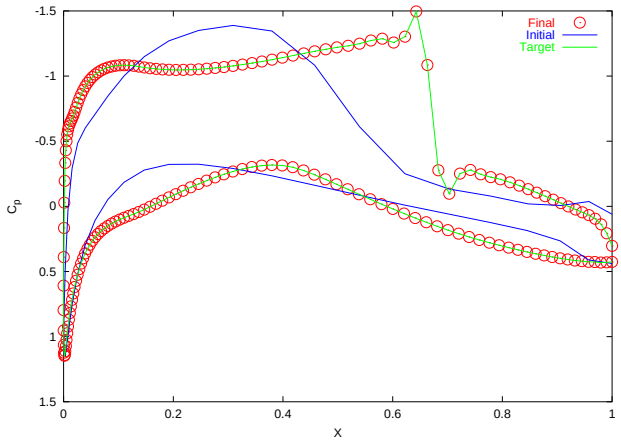


(f) Objective function history.

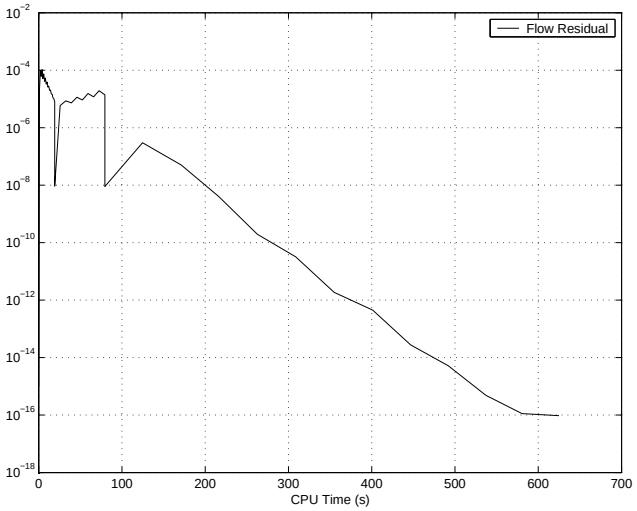
Fig. 2 Subsonic inverse design case.



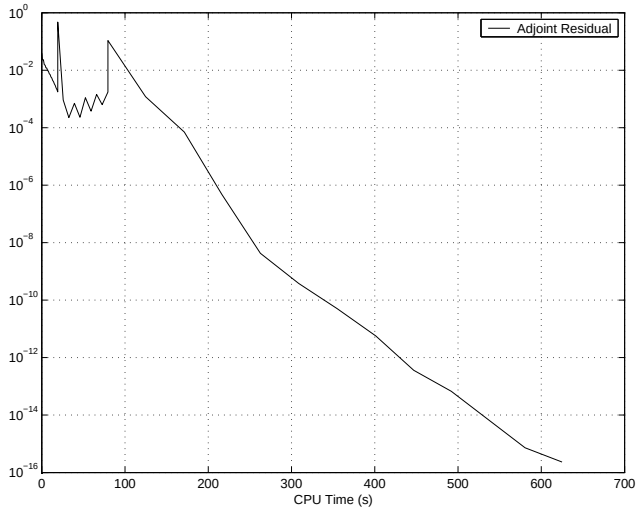
(a) Initial and final airfoils.



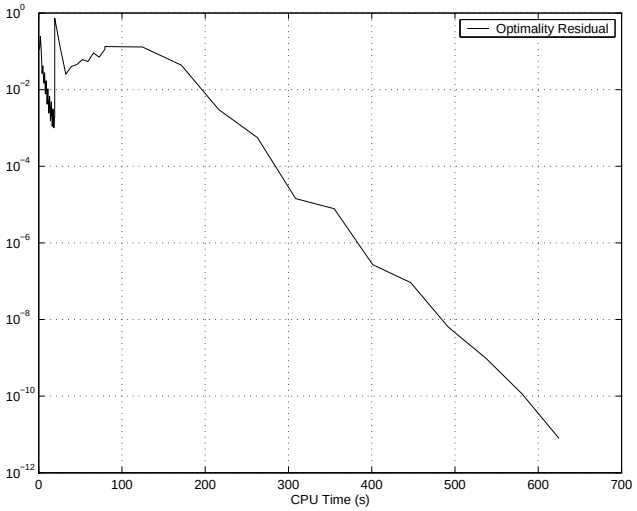
(b) Initial, target, and final pressure distributions.



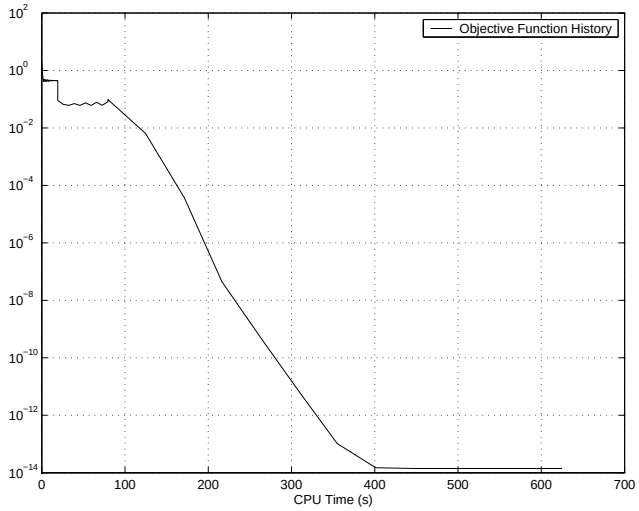
(c) Flow residual convergence.



(d) Adjoint residual convergence.

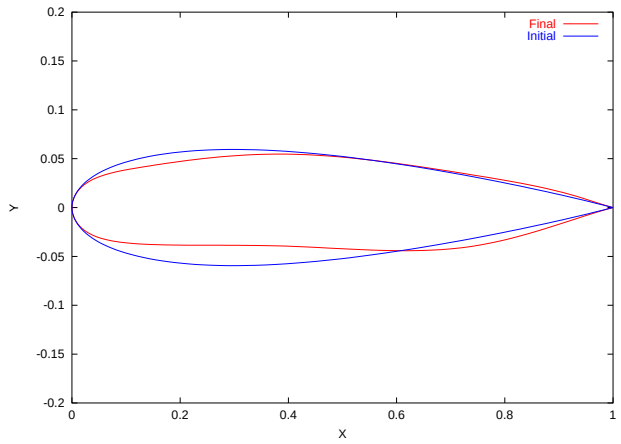


(e) Optimality residual convergence.

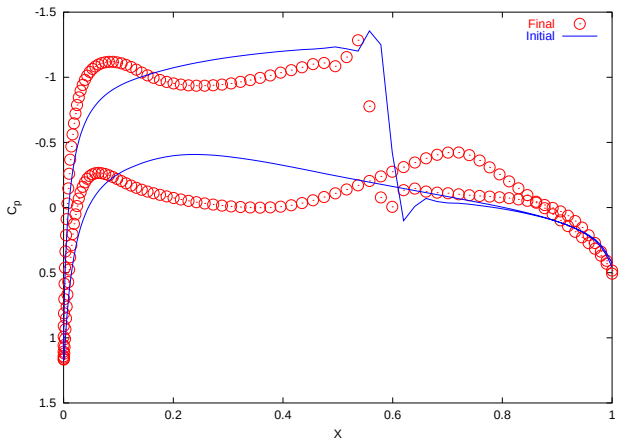


(f) Objective function history.

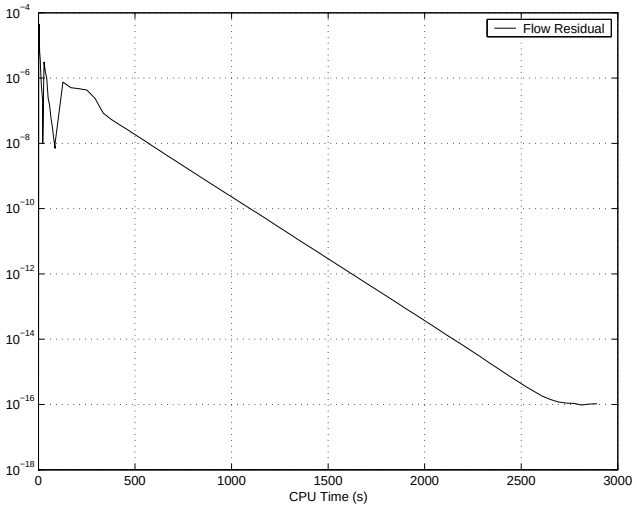
Fig. 3 Transonic inverse design case.



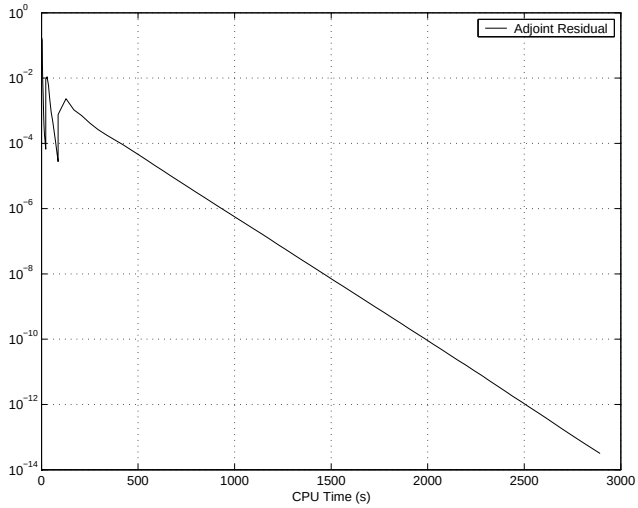
(a) Initial and final airfoils.



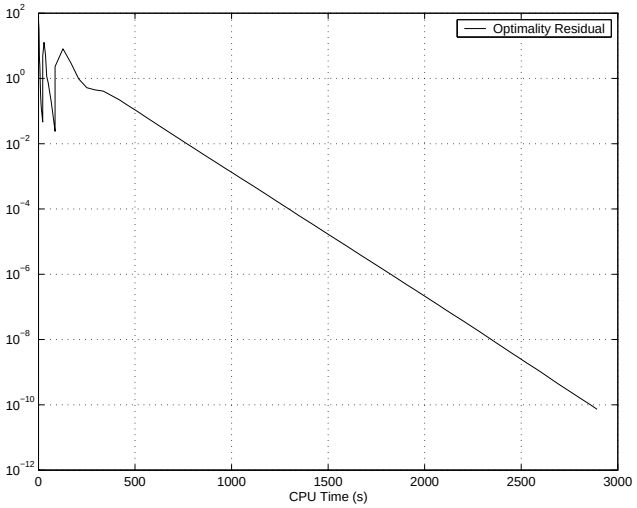
(b) Initial and final pressure distributions.



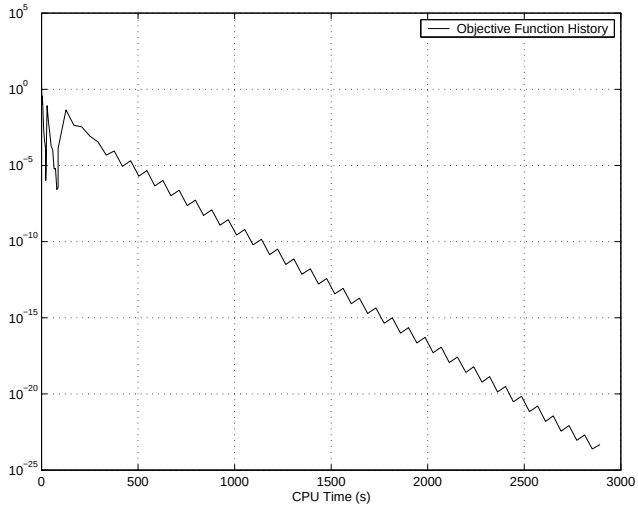
(c) Flow residual convergence.



(d) Adjoint residual convergence.



(e) Optimality residual convergence.



(f) Objective function history.

Fig. 4 Drag reduction at fixed lift case.