NASA-CR-201036

FORMULATION FOR SIMULTANEOUS AERODYNAMIC ANALYSIS AND DESIGN OPTIMIZATION[1]

G. W. Hou,[2] A. C. Taylor, III,[3] and S. V. Mani[4]
Department of Mechanical Engineering
Old Dominion University
Norfolk, VA 23529–0247

P. A. Newman[5]
NASA Langley Research Center
Hampton, VA 23681–0001

Presented at
Second U.S. National Congress on Computational Mechanics
August 16–18, 1993
Washington, D.C.

Submitted to
International Journal for Numerical Methods in Engineering

---

[1] This research is partially funded by NASA Grant No. NAG-1-1265.
[2] Associate Professor, member AIAA and ASME
[3] Assistant Professor, member AIAA
[4] Graduate Research Assistant
[5] Senior Research Scientist

FORMULATION FOR SIMULTANEOUS AERODYNAMIC ANALYSIS AND DESIGN OPTIMIZATION[1]

G. W. Hou,[2] A. C. Taylor, III,[3] and S. V. Mani[4]
Department of Mechanical Engineering
Old Dominion University
Norfolk, VA 23529–0247

P. A. Newman[5]
NASA Langley Research Center
Hampton, VA 23681–0001

ABSTRACT

An efficient approach for simultaneous aerodynamic analysis and design optimization is presented. This approach does not require the performance of many flow analyses at each design optimization step, which can be an expensive procedure. Thus, this approach brings us one step closer to meeting the challenge of incorporating computational fluid dynamic codes into gradient-based optimization techniques for aerodynamic design. An adjoint-variable method is introduced to nullify the effect of the increased number of design variables in the problem formulation. The method has been successfully tested on one-dimensional nozzle flow problems, including a sample problem with a normal shock. Implementations of the above algorithm are also presented that incorporate Newton iterations to secure a high-quality flow solution at the end of the design process. Implementations with iterative flow solvers are possible and will be required for large, multidimensional flow problems.

---

[1] This research is partially funded by NASA Grant No. NAG-1-1265. The paper was presented at the Second U.S. National Congress on Computational Mechanics, August 16–18, 1993, Washington, D.C.
[2] Associate Professor, member AIAA and ASME
[3] Assistant Professor, member AIAA
[4] Graduate Research Assistant
[5] Senior Research Scientist

1.0 NOMENCLATURE

A	Adjoint variables
A	Local cross-sectional area
b	Design variables
e	Specific internal energy
e_o	Specific total energy
F	Quasi-one-dimensional Euler equation flux vector
m	Number of constraints
P	Pressure
Q	State variables (aerodynamic field variables)
R	Nonlinear state equations or residual of flow equations
r	Penalty function
S	Quasi-one-dimensional Euler equation source vector
t	time
u	Flow speed
x	Nozzle axial coordinate
α	Step size (during one-dimensional search)
γ	Ratio of specific heats
Δ	Change in quantity
Φ	Merit function
ρ	Density
Ψ_i	Side-constraint functions
Ψ_o	Objective function
Subscripts	
i	ith side constraint
j	x-coordinate discretization index
o	Objective function
Superscripts	
i	Subiteration index
n, m	Iteration indices

T	Matrix transpose
^	Target values
~	Approximate operator
*	New (or updated) quantity

Abbreviations

ADS	Automated Design Synthesis
CFD	Computational fluid dynamics
CPU	Central processing unit
FD	Finite difference
NR	Newton-Raphson iterations
QA	Quasi-analytical
SAADO	Simultaneous aerodynamic analysis and design optimization

2.0 INTRODUCTION

A typical aerodynamic problem can be highly nonlinear, and its solution can consume a great deal of computer resources in terms of both memory and central processing unit (CPU) time. This demand poses a challenge for incorporating advanced computational fluid dynamics (CFD) software into gradient-based optimization techniques for aerodynamic design. In this study, a strategy is proposed whereby the objective function and constraint violations are reduced; simultaneously, the aerodynamic field equations are solved (i.e., the final steady-state numerical solution of these equations is achieved). With this approach, the aerodynamic field variables are treated as additional design variables, and the aerodynamic residual equations are considered to be additional equality constraints. As a consequence, the steady-state aerodynamic equations are satisfied only at the final optimum solution. With this new approach, overall computational expense is expected to be reduced in comparison with the conventional optimization method (for which many repeated steady-state aerodynamic analyses are performed during the optimization

process). Two quasi-one-dimensional nozzle-flow problems are considered to demonstrate the initial numerical testing of the scheme. In addition, the feasibility of extending the methodology to multidimensional aerodynamic problems is demonstrated.

Other recent studies involve the development of strategies for simultaneous aerodynamic analysis and design optimization. For example, the algorithm developed by Rizk [1, 2] first updated the design variables at each design optimization iteration with a gradient-projection method and then updated the field (state) variables with several flow-solver iterations. The gradients of the objective and side-constraint functions were calculated for the current values of the flow variables by the method of finite differences. The inverse design scheme of Drela [3] employed a streamline formulation of the Euler equations; in addition, the locations of the streamlines that define the shape of the airfoil were taken as the design variables. The optimum design and the corresponding flow solution were simultaneously obtained by solving a global system of equations that included (at each optimization iteration) the linearized flow equations and the necessary conditions of the minimization problem.

Design optimization techniques are generally iterative in nature; the fact that simultaneous analysis and design is efficient only when the method of analysis is also iterative is noted by Haftka [4, 5]. Recent works by Orozco and Ghattas [6–8] and Ta'asan et al. [9, 10] investigated techniques to improve the computational efficiency and storage requirements for simultaneous aerodynamic analysis and design optimization. The former method examined the sparsity of the Jacobian and Hessian matrices; the latter method took advantage of the multigrid method.

Two important concerns exist for the present study: the increased size of the problem (because the aerodynamic field variables are treated as design variables) and the quality of the aerodynamic solution at the end of the design process (measured by the l_2 norm of the aerodynamic residual equations). The first concern is addressed by employing the adjoint-variable method; the resulting equations are similar to those derived in reference [6]. The second concern is addressed by adding flow-solver iterations between the design iterations in the optimization procedure.

3.0 OPTIMIZATION METHODOLOGY

A typical engineering design optimization problem attempts to find the best design **b** that minimizes the objective function, satisfies the constraints, and satisfies a set of equilibrium conditions (i.e., a set of state equations). Mathematically, this optimum design can be stated as

$$\min_{\mathbf{b}} \quad \Psi_o(\mathbf{Q}(\mathbf{b}), \mathbf{b}) \tag{1}$$

which is subject to

$$\Psi_i(\mathbf{Q}(\mathbf{b}), \mathbf{b}) \leq \mathbf{0} \qquad (i = 1, 2, ..., m) \tag{2}$$

where the vectors **Q** and **b** are the aerodynamic field and design variables, respectively, and Ψ_o and Ψ_i are the objective and side-constraint functions, respectively. In an aerodynamic problem, the field variables usually represent the density, velocity components, and pressure, for example. The field variables describe the physical behavior of the system to be designed and are related to the design variables through the nonlinear state equations, which are defined symbolically as

$$\mathbf{R}(\mathbf{Q}(\mathbf{b}), \mathbf{b}) = \mathbf{0} \tag{3}$$

As is characteristic of most realistic aerodynamic problems, the dimension of the vector **b** is of the order of tens to several hundreds; in contrast, the dimensions of the vectors **Q** and **R** are of the order of thousands to even millions. A solution **Q** for Eq. (3) for a given **b** can be not only difficult to find but computationally intensive as well. However, possibly hundreds of such analyses (i.e., repeated solutions of Eq. (3)) are required for the conventional design optimization process. The conventional design optimization process is explained in greater detail in the following section.

3.1 Conventional Approach For Aerodynamic Design Optimization

First, to gain a better understanding of the new algorithm, the general procedure for the conventional approach to a typical aerodynamic design optimization problem is outlined in detail. From an initial design **b**, a typical gradient-based design optimization scheme first solves for the field variables **Q** with the state equations (Eq. (3)) and then evaluates the change in design $\Delta\mathbf{b}$ that reduces the change in the objective function and corrects the constraint violations. This most favorable $\Delta\mathbf{b}$ is obtained by solving a linearized optimization problem

$$\min_{\Delta\mathbf{b}} \quad \Psi_o(\mathbf{Q}(\mathbf{b}),\mathbf{b}) + \left(\frac{d\Psi_o}{d\mathbf{b}}\right)\Delta\mathbf{b} \tag{4}$$

which is subject to

$$\Psi_i(\mathbf{Q}(\mathbf{b}),\mathbf{b}) + \left(\frac{d\Psi_i}{d\mathbf{b}}\right)\Delta\mathbf{b} \le \mathbf{0} \qquad (i = 1,2,...,m) \tag{5}$$

where Eqs. (4) and (5) are linearized approximations of Eqs. (1) and (2), respectively. Note that for simplicity only the first-order terms appear in Eqs. (4) and (5). The derivatives of the

objective function $\frac{d\Psi_o}{db}$ can be evaluated either in terms of the derivatives $\frac{dQ}{db}$ or in terms of the adjoint variables $\mathbf{A}_o$ as

$$\frac{d\Psi_o}{db} = \frac{\partial \Psi_o}{\partial \mathbf{Q}} \frac{d\mathbf{Q}}{db} + \frac{\partial \Psi_o}{\partial \mathbf{b}} \tag{6}$$

or

$$\frac{d\Psi_o}{db} = \mathbf{A}_o^T \frac{\partial \mathbf{R}}{\partial \mathbf{b}} + \frac{\partial \Psi_o}{\partial \mathbf{b}} \tag{7}$$

where the derivatives $\frac{dQ}{db}$ are the solution of the linear sensitivity equation

$$-\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right) \frac{d\mathbf{Q}}{db} = \frac{\partial \mathbf{R}}{\partial \mathbf{b}} \tag{8}$$

This linear sensitivity equation results from the differentiation of Eq. (3) with respect to **b**, and the adjoint variables $\mathbf{A}_o$ are the solution of the linear equation

$$-\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^T \mathbf{A}_o = \left(\frac{\partial \Psi_o}{\partial \mathbf{Q}}\right)^T \tag{9}$$

The derivatives of the side-constraint functions $\frac{d\Psi_i}{db}$ can be evaluated with expressions that are similar to Eqs. (6) through (9).

After the most favorable direction of change $\Delta \mathbf{b}$ is found by solving Eqs. (4) and (5) with a proper mathematical linear programming technique, the next procedure is to determine the step size α, which defines the newly updated design $\mathbf{b}^*$ as

$$\mathbf{b}^* = \mathbf{b} + \alpha \Delta \mathbf{b} \tag{10}$$

The step size α serves as a relaxation factor in an iterative scheme for determining $\mathbf{b}^*$; it is determined by solving a nonlinear optimum design problem (similar in form to Eqs. (1)–(3)) in which α is the only design variable. The procedure for determining α is referred to as the

"one-dimensional search" in many design optimization textbooks. The updated values of the field variables $\mathbf{Q}^*(\mathbf{b}^*)$ for the new design $\mathbf{b}^*$ in this procedure is obtained by solving the state equations (one solution for each trial value of α); that is,

$$\mathbf{R}(\mathbf{Q}^*(\mathbf{b}^*), \mathbf{b}^*) = \mathbf{0} \tag{11}$$

Finally, the optimum design is achieved if the Kuhn-Tucker conditions are satisfied; if not, the entire procedure (Eqs. (4) through (11)) is repeated until convergence is achieved.

Generally speaking, the conventional design optimization approach of Eqs. (4) through (11) requires on the order of possibly hundreds of design iterations to converge to the solution of the original design optimization problem (defined previously by Eqs. (1) through (3)). The repeated solution (to steady state) of Eq. (11) is computationally intensive, particularly for three-dimensional nonlinear aerodynamic problems. This computational intensity has motivated several studies (including the present one) with the goal of reducing this computational effort [1–10].

3.2 Proposed Approach For Simultaneous Aerodynamic Analysis and Design Optimization (SAADO)

3.2.1 Derivation of SAADO Equations

The proposed new approach reformulates the design optimization problem presented in Eqs. (1) through (3) as

$$\min_{\mathbf{Q},\mathbf{b}} \quad \Psi_o(\mathbf{Q}, \mathbf{b}) \tag{12}$$

which is subject to

$$\Psi_i(\mathbf{Q}, \mathbf{b}) \leq \mathbf{0} \qquad (i = 1, 2, ..., m) \tag{13}$$

and the additional equality constraints

$$\mathbf{R}(\mathbf{Q},\mathbf{b}) = \mathbf{0} \tag{14}$$

This formulation is referred to as simultaneous analysis and design optimization (SAADO), which (in contrast to the conventional approach) now treats the state (field) variables as part of the set of independent design variables and considers the state equations to be part of the set of side constraints. Because satisfaction of the equality constraints of Eq. (14) is required only at the final optimum solution, the steady-state aerodynamic field equations are not solved at every design optimization iteration. As a result, the excessively large computational burden of the conventional approach may be reduced significantly. However, this advantage is likely to be offset by the large increase in the number of design variables and equality constraint functions unless some remedial procedure is adopted.

The new method begins with a linearized design optimization problem (similar in principle to that of Eqs. (4) and (5)), which is solved for the most favorable change in the design variables $\Delta\mathbf{b}$, as well as in the state (field) variables $\Delta\mathbf{Q}$; that is

$$\min_{\Delta\mathbf{Q},\,\Delta\mathbf{b}} \quad \Psi_o(\mathbf{Q},\mathbf{b}) + \frac{\partial\Psi_o}{\partial\mathbf{Q}}\Delta\mathbf{Q} + \frac{\partial\Psi_o}{\partial\mathbf{b}}\Delta\mathbf{b} \tag{15}$$

which is subject to

$$\Psi_i(\mathbf{Q},\mathbf{b}) + \frac{\partial\Psi_i}{\partial\mathbf{Q}}\Delta\mathbf{Q} + \frac{\partial\Psi_i}{\partial\mathbf{b}}\Delta\mathbf{b} \le \mathbf{0} \qquad (i = 1,2,\ldots,m) \tag{16}$$

and

$$\mathbf{R}(\mathbf{Q},\mathbf{b}) + \frac{\partial\mathbf{R}}{\partial\mathbf{Q}}\Delta\mathbf{Q} + \frac{\partial\mathbf{R}}{\partial\mathbf{b}}\Delta\mathbf{b} = \mathbf{0} \tag{17}$$

Equations (15), (16), and (17) are linearized approximations of Eqs. (12), (13), and (14), respectively. Note in this formulation that **R(Q,b)**, which is the residual of the nonlinear aerodynamic field equations, is not required to be zero until the final optimum design is achieved. In comparison with Eqs. (4) and (5), the mathematical linear programming problem of Eqs. (15)-(17) becomes more computationally difficult to solve because of the dramatic increase in the effective number of design variables and equality-constraint equations. This difficulty is overcome by the introduction of the adjoint-variable method to remove $\Delta\mathbf{Q}$ and Eq. (17) altogether from this linear programming problem.

The derivation of this technique begins with the introduction of an arbitrary vector $\mathbf{A}_o$ to augment the linearized objective function of Eq. (15) by using Eq. (17); the resulting enhanced linearized objective function is expressed as

$$\Psi_o(\mathbf{Q},\mathbf{b}) + \frac{\partial \Psi_o}{\partial \mathbf{Q}}\Delta\mathbf{Q} + \frac{\partial \Psi_o}{\partial \mathbf{b}}\Delta\mathbf{b} + \mathbf{A}_o^T\left[\mathbf{R}(\mathbf{Q},\mathbf{b}) + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\Delta\mathbf{Q} + \frac{\partial \mathbf{R}}{\partial \mathbf{b}}\Delta\mathbf{b}\right] \tag{18}$$

If rearranged, then result is

$$\Psi_o(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial \Psi_o}{\partial \mathbf{Q}} + \mathbf{A}_o^T\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)\Delta\mathbf{Q} + \mathbf{A}_o^T\mathbf{R}(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial \Psi_o}{\partial \mathbf{b}} + \mathbf{A}_o^T\frac{\partial \mathbf{R}}{\partial \mathbf{b}}\right)\Delta\mathbf{b} \tag{19}$$

Then, the adjoint-variable vector $\mathbf{A}_o$ can be specified such that the coefficient of $\Delta\mathbf{Q}$ vanishes, which completely eliminates $\Delta\mathbf{Q}$ from the enhanced objective function of Eq. (19); this result, of course, dramatically reduces the size of the linear programming problem to be solved. Thus, the adjoint equation for $\mathbf{A}_o$ is

$$-\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^T\mathbf{A}_o = \left(\frac{\partial \Psi_o}{\partial \mathbf{Q}}\right)^T \tag{20}$$

and the resulting equation for the enhanced, linearized objective function becomes

$$\Psi_o(\mathbf{Q},\mathbf{b}) + \mathbf{A}_o^T\,\mathbf{R}(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial\Psi_o}{\partial\mathbf{b}} + \mathbf{A}_o^T\frac{\partial\mathbf{R}}{\partial\mathbf{b}}\right)\Delta\mathbf{b} \tag{21}$$

A similar procedure must be applied to reformulate the linearized side-constraint functions of Eq. (16). The final result is

$$\Psi_i(\mathbf{Q},\mathbf{b}) + \mathbf{A}_i^T\mathbf{R}(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial\Psi_i}{\partial\mathbf{b}} + \mathbf{A}_i^T\frac{\partial\mathbf{R}}{\partial\mathbf{b}}\right)\Delta\mathbf{b} \le \mathbf{0} \qquad (i = 1,2,...,m) \tag{22}$$

where $\mathbf{A}_i$ are the solutions to the adjoint equations

$$-\left(\frac{\partial\mathbf{R}}{\partial\mathbf{Q}}\right)^T\mathbf{A}_i = \left(\frac{\partial\Psi_i}{\partial\mathbf{Q}}\right)^T \qquad (i = 1,2,...,m) \tag{23}$$

With $\Delta\mathbf{Q}$ and Eq. (17) eliminated, the linearized optimum design problem of Eqs. (15)-(17) is reformulated as

$$\min_{\Delta\mathbf{b}} \quad \Psi_o(\mathbf{Q},\mathbf{b}) + \mathbf{A}_o^T\mathbf{R}(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial\Psi_o}{\partial\mathbf{b}} + \mathbf{A}_o^T\frac{\partial\mathbf{R}}{\partial\mathbf{b}}\right)\Delta\mathbf{b} \tag{24}$$

which is subject to

$$\Psi_i(\mathbf{Q},\mathbf{b}) + \mathbf{A}_i^T\mathbf{R}(\mathbf{Q},\mathbf{b}) + \left(\frac{\partial\Psi_i}{\partial\mathbf{b}} + \mathbf{A}_i^T\frac{\partial\mathbf{R}}{\partial\mathbf{b}}\right)\Delta\mathbf{b} \le \mathbf{0} \qquad (i = 1,2,...,m) \tag{25}$$

After Eqs. (24) and (25) have been solved for $\Delta\mathbf{b}$ (with a suitable mathematical programming technique), Eq. (17) can be solved to obtain $\Delta\mathbf{Q}$. A one-dimensional search is performed to find the updated values of $\mathbf{Q}^*$ and $\mathbf{b}^*$; the search procedure must solve a nonlinear optimization problem of the form

$$\min_{\alpha} \quad \Psi_o(\mathbf{Q}^*,\mathbf{b}^*) \tag{26}$$

which is subject to

$$\Psi_i(\mathbf{Q}^*, \mathbf{b}^*) \leq 0 \qquad (i = 1, 2, ..., m) \tag{27}$$

and

$$\mathbf{R}(\mathbf{Q}^*, \mathbf{b}^*) = \mathbf{0} \tag{28}$$

where the step size α is the only design variable. Again, note that the equality constraints (Eq. (28)) are not required to be zero until the final optimum design; violations of these equality constraints should simply be progressively reduced during the SAADO procedure. Therefore, the updated $\mathbf{Q}^*$ in this study is defined as $\mathbf{Q}^* = \mathbf{Q} + \Delta\mathbf{Q}^*$, which satisfies the first-order approximation of Eq. (28) as

$$\mathbf{R}(\mathbf{Q}, \mathbf{b}) + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\Delta\mathbf{Q}^* + \frac{\partial \mathbf{R}}{\partial \mathbf{b}}(\alpha\Delta\mathbf{b}) = \mathbf{0} \tag{29}$$

The update for $\mathbf{b}^*$ is given previously in Eq. (10). Note that in the one-dimensional search, Eq. (29) need to be solved for each trial value of α; alternatively, a one-time solution (per design cycle) of the following two equations is performed:

$$-\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\Delta\mathbf{Q}_1 = \frac{\partial \mathbf{R}}{\partial \mathbf{b}}\Delta\mathbf{b} \tag{30}$$

and

$$-\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\Delta\mathbf{Q}_2 = \mathbf{R}(\mathbf{Q}, \mathbf{b}) \tag{31}$$

In addition, $\Delta\mathbf{Q}^* = \alpha\Delta\mathbf{Q}_1 + \Delta\mathbf{Q}_2$ for all α. If more than one trial value of α is needed in the one-dimensional search, Eqs. (30) and (31) clearly provide a more efficient means of finding a proper update of $\Delta\mathbf{Q}^*$ than Eq. (29).

3.2.2 Implementation of SAADO

Computationally, to assist in the implementation of the line-search procedure of Eqs. (26)–(29), a merit function is constructed using the interior-penalty-function method. This merit function is a combination of the objective, side-constraint, and aerodynamic residual equations and is given by

$$\Phi(\mathbf{Q}, \mathbf{b}, r) = \Psi_o(\mathbf{Q}, \mathbf{b}) + r \sum_{i=1}^{m} 1/\Psi_i\,(\mathbf{Q}, \mathbf{b}) + r^{-1/2}\ \mathbf{R}^T(\mathbf{Q}, \mathbf{b})\ \mathbf{R}\,(\mathbf{Q}, \mathbf{b}) \qquad (32)$$

where r is the penalty-function parameter (which is initially assumed to be unity and is progressively reduced from one design iteration cycle to the next). The value of α, as determined by the line-search process, is designed to ensure that the numerical value of the merit function is reduced when compared with its value from the previous design-iteration cycle (i.e., $\Phi(\mathbf{Q}^*, \mathbf{b}^*, r) < \Phi(\mathbf{Q}, \mathbf{b}, r)$).

At the final optimum design, $\mathbf{Q}^*$ and $\mathbf{b}^*$ are required to satisfy not only the Kuhn-Tucker conditions but must also reduce the residual of the aerodynamic equations to a very small value; that is, $\mathbf{Q}^*$ should represent a high-quality solution of the steady-state aerodynamic field equations. One procedure for meeting this condition is to perform some flow-solver iterations to reduce the residual of the flow equations somewhat (thus, $\mathbf{Q}^*$ is improved and $\mathbf{b}^*$ is held fixed) before the next design-optimization iteration is started. This procedure can also be supplemented by using an appropriate weighting coefficient for the aerodynamic field equations in the merit function, at each cycle in the design optimization process. These and other issues in regard to the implementation of the SAADO algorithm are discussed later in greater detail.

If the steady-state flow equations (Eq. (3)) are linear in **Q**, then a single Newton iteration will always solve these equations exactly; in this case, Eqs. (24) and (25) become identical to Eqs. (4) and (5), respectively. Furthermore, after $\Delta\mathbf{b}$ is found, Eq. (11) can be solved directly for the new $\mathbf{Q}^*$ without resorting to the approximation process described by Eq. (29). Therefore, if the state equations are linear, then the SAADO scheme is exactly equivalent to the conventional optimization method and no computational savings is achieved.

The proposed SAADO procedure can be summarized as follows:

Step 1. Select an initial design **b** and assume an initial value of the field variables **Q** (e.g., the free-stream value).

Step 2. Set up and solve Eqs. (20) and (23) for the adjoint variables $\mathbf{A}_o$ and $\mathbf{A}_i$, i = 1, 2, . . . , m.

Step 3. Set up the linearized optimization problem of Eqs. (24) and (25), and solve for $\Delta\mathbf{b}$ with a linear programming method.

Step 4. Solve Eq. (29) or Eqs. (30) and (31) for $\Delta\mathbf{Q}^*$.

Step 5. Find the step size α with the line-search procedure described by Eqs. (26)–(31), including use of the merit function (Eq. (32)); update the design and field variables.

Step 6. Check all convergence criteria; if the criteria are not satisfied, then return to Step 2 and use a smaller value for the penalty parameter r.

3.3 Extension to Multidimensional Flow Problems

The basic SAADO algorithm is quite general and, in principal, can be applied to any flow problems. However, the successful extension of the SAADO algorithm to those flow problems depends upon the particular CFD solvers used and how they are employed in iteratively solving nonlinear flow equations such as Eq. (3), as well as linear equations such as (Eqs. (20), (23), and, (29)). For one-dimensional and some two-dimensional flow problems, a Newton direct solver can be used to solve these equations on current computers. However, a strict application of Newton's method is not feasible on modern computers when solving these equations for large two- and three-dimensional problems with the Euler or Navier-Stokes equations. An exact construction of the true Jacobian coefficient matrix $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$ and a direct in-core Newton solution strategy applied to the nonlinear flow equation (such as Eq. (3)) at each iteration is impossible on current supercomputers for these problems because of insufficient memory. This same difficulty also applies, of course, to the direct solution of the large linear systems (Eqs. (20), (23) and (29)) associated with the proposed SAADO scheme. However, iterative solution of these large linear systems of equations is practical for multidimensional problems by recasting them in "incremental" or "delta" form. The details of this incremental formulation and a demonstration of the feasibility and benefits for aerodynamic problems are given in references [11–13].

The incremental iterative form, for example, of Eq. (20) is

$$-\left(\frac{\widetilde{\partial \mathbf{R}}}{\partial \mathbf{Q}}\right)^{\mathrm{T}} (\Delta \mathbf{A}_{\mathrm{o}})_{\mathrm{m}} = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)^{\mathrm{T}} (\mathbf{A}_{\mathrm{o}})_{\mathrm{m}} + \left(\frac{\partial \Psi_{\mathrm{o}}}{\partial \mathbf{Q}}\right)^{\mathrm{T}}$$

$$(\mathbf{A}_{\mathrm{o}})_{\mathrm{m+1}} = (\mathbf{A}_{\mathrm{o}})_{\mathrm{m}} + (\Delta \mathbf{A}_{\mathrm{o}})_{\mathrm{m}} \tag{33}$$

where m is the incremental iterative index. An equation similar to Eq. (33) is obtained for Eq. (23). The incremental iterative form of Eq. (29) is

$$-\left(\frac{\widetilde{\partial \mathbf{R}}}{\partial \mathbf{Q}}\right)[\Delta(\Delta \mathbf{Q}^*)]_m = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right)(\Delta \mathbf{Q}^*)_m + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{b}}\right)(\alpha \Delta \mathbf{b}) + \mathbf{R}(\mathbf{Q}, \mathbf{b})$$

$$(\Delta \mathbf{Q}^*)_{m+1} = (\Delta \mathbf{Q}^*)_m + [\Delta(\Delta \mathbf{Q}^*)]_m \tag{34}$$

Note that in Eqs. (33) and (34) the tilde over the Jacobian operator on the left-hand side indicates that this operator can be approximate, whereas those on the right-hand side cannot. Most three-dimensional CFD flow solvers typically do not construct the exact Jacobian matrix operator $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$; the idea here is to use the efficient left-hand-side operator of the existing flow solver and construct derivative terms on the right-hand side via automatic differentiation [14, 15]. Automatic differentiation has been applied to advanced CFD codes [16, 17] to obtain consistent, discrete sensitivity derivatives of aerodynamic output functions with respect to input and modeling parameters. It has not yet been applied, however, in an incremental iterative form, where significant computational gains are expected.

For the present implementation and demonstration of SAADO, a one-dimensional flow problem will be considered. Therefore, a Newton direct-solver method will be used to solve the linear adjoint equations, as well as to iterate on the nonlinear flow problems.

4.0 OPTIMIZATION PROBLEMS AND NUMERICAL RESULTS

Two quasi-one-dimensional sample problems have been selected to demonstrate the SAADO scheme; its performance on these problems will be compared with that of the more conventional "black box" optimization scheme. One of the objectives of this study is to ensure that the

SAADO algorithm is capable of delivering a high-quality flow solution at the end of the design-optimization process. The effect of applying flow-solver type iterations between optimization cycles will be studied to achieve this objective.

The first problem is a fully supersonic flow through a diverging nozzle; the second is a transonic flow through a converging-diverging nozzle, with a normal shock in the diverging section. These sample problems are described subsequently in more detail.

4.1 Problem Statement

In each of the two quasi-one-dimensional sample problems, a "target nozzle" shape is defined through the a priori specification of a target value of each of the design variables; then, a fully converged, steady-state numerical solution of the governing equations is obtained for the target nozzle. Thus, the target flow speed $\hat{u}_j$ is known at each jth grid point of the target-nozzle geometry. Then, an initial nozzle shape (different, of course, from that of the target nozzle) is defined by specification of an initial value for each of the design variables. In the sample problems to be presented, the objective of the optimization procedure is to manipulate the nozzle shape by varying the design variables such that an optimum match is obtained between the flow-speed profile that evolves during the optimization process and the flow-speed profile of the target nozzle. The objective function Ψ_o is defined by using the difference between the (current) evolving flow speed and the target flow speed at each grid point; that is,

$$\Psi_o \equiv \frac{1}{2} \sum_{j=1}^{jmax} (\hat{u}_j - u_j)^2 \tag{35}$$

where u_j is the flow speed that evolves at the jth grid point and jmax is the total number of grid points. No side constraints Ψ_i are present in the sample problems.

The flow speed u_j is obtained by solving the quasi-one-dimensional Euler equations, which are

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{Q})}{\partial x} + \mathbf{S}(\mathbf{Q}) = \mathbf{0} \tag{36}$$

where $\mathbf{Q} = [\rho,\ \rho u,\ \rho e_o]^T, \mathbf{F}(\mathbf{Q}) = [\rho u,\ \rho u^2 + P,\ (\rho e_o + P)u]^T$, and $\mathbf{S}(\mathbf{Q}) = -\frac{dA}{dx}\frac{1}{A}[\rho u,\ \rho u^2,\ (\rho e_o + P)u]^T$. In these equations, ρ is density, u is flow speed, P is pressure, e_o is the specific total energy (i.e., $e_o = e + u^2/2$, where e is the specific internal energy), and A(x) is the local cross-sectional area. The ideal-gas law with a constant ratio of specific heats γ (taken to be 1.4) is used for closure, which implies $P = (\gamma - 1)(\rho e_0 - \rho u^2/2)$. The governing equations are discretized and solved numerically with the upwind flux-vector-splitting method of Van Leer [18], which includes the use of higher order accuracy to approximate the flux terms. A more complete discussion of these numerical procedures is presented in reference [19]. The flow field was discretized with 100 grid points. For steady flow, the discretization of the governing equations, together with the numerical treatment of the boundary conditions, results in a large set of coupled nonlinear algebraic equations with the form of Eq. (3). In this study, the basic procedure for solving the discrete nonlinear flow equations is Newton's root-finding method, or a modified version of the same, which is described subsequently in this section.

4.2 Sample Problem 1: Supersonic Nozzle

The streamwise variation of the cross-sectional area A(x) for the fully supersonic nozzle problem is given by

$$A(x) = b_1 + b_2 \tanh(0.8x - b_3) \qquad (0 \le x \le 10) \tag{37}$$

where b_1, b_2, and b_3 are selected to be the design variables (i.e., $\mathbf{b} \equiv (b_1, \ b_2, \ b_3)^T$). The initial values of these design variables are $b_1 = 1.33, b_2 = 0.30$, and $b_3 = 3.90$; their target values are $b_1 = 1.398$, $b_2 = 0.347$, and $b_3 = 4$. For this fully supersonic flow problem, the Mach number on the inflow boundary is 1.5; ρ, u, and P are specified and held fixed (at the free-stream values) on this boundary and are extrapolated on the outflow boundary. In constructing the merit function (Eq. (32)) for this problem, weighting coefficients were set to 1 (unity) for both the objective function and the root-mean-square value of the residual of the aerodynamic equations. The penalty-function parameter r (Eq. (32)) was updated at every optimization step by halving it.

In the computational results that follow, the new optimization scheme is compared with the conventional black-box approach, wherein the design optimization module and the flow-analysis module are totally independent. The design optimization software used for this was Automated Design Synthesis (ADS) [20]. The conventional design optimization approach was employed with gradients (derivatives of the objective function with respect to the design variables) calculated by both one-sided finite difference (FD) and quasi-analytical (QA) differentiation of the flow-analysis code. The method of feasible directions was used as the optimizer, and the golden-section method was used for the one-dimensional search for the two cases above that involve the conventional design optimization approach. In addition to the basic SAADO algorithm explained above, two variants of this algorithm were also applied to the present sample problem to study the effect of introducing additional flow-solver iterations between the optimization cycles. The first variation deviates from the basic SAADO algorithm in the following manner: several Newton iterations (three in this study) are performed on the initial guess for the field variables (which in this case is the free stream), and then the actual design-optimization process can be started

and continued as explained in section 3.2.2. This process means that the three Newton iterations are introduced immediately after step 1 (section 3.2.2). This particular procedure is denoted herein as algorithm 1. In the second variation, a few Newton iterations are introduced into the basic SAADO algorithm after $\mathbf{Q}^*$ (the updated field variables) are obtained to improve the quality of the solution for $\mathbf{b}^*$ (the updated design variables) held fixed. This means that a few Newton iterations are introduced immediately after step 5 (section 3.2.2). This procedure is denoted herein as algorithm 2a if one Newton iteration is added; it is denoted algorithm 2b if two Newton iterations are added.

All results presented here were computed on a Cray-2; the flow-analysis code requires nine Newton iterations for a complete, fully converged flow-analysis (steady-state) solution. The design optimization algorithms were applied for 20 cycles each. Figure 1 compares the initial- and target-velocity profiles for the nozzle (although initial values for field variables at interior points are free-stream values). Table 1 compares the algorithms in terms of the final objective function, the root-mean-square value of the residual of the aerodynamic equations at the end of the design process, the total number of equivalent Newton-Raphson (NR) iterations, the overall computational efficiency (i.e., total CPU time), and the final design variables achieved. From the results, algorithm 2b clearly gives the best overall results in terms of the magnitude of the final objective function and the quality of the final flow solution. The other algorithms also give good results because for most engineering purposes, a residual value of 1.0E-4 is considered to be satisfactory. These results show that the number of intermediate Newton iterations and their placement in the SAADO algorithm enables the user to control the residual and, hence, the quality of the flow solution at the end of the design process.

To study the performance of the basic algorithm, the residual and the objective function after each optimization cycle are plotted in Fig. 2. The detailed result shows that a global reduction occurs in the residual as the optimization progresses, but the overall residual at the end of the design process is considered satisfactory for engineering purposes. The effectiveness of using intermediate Newton iterations (algorithms 2a and 2b) in achieving the previously stated objectives is thus reiterated.

Figure 3 shows the residual history during the optimization process using algorithm 2b superimposed on a single standard flow-solver residual history. In this plot, each peak represents the beginning of an optimization cycle; each valley represents the end of a cycle. For the basic algorithm, Fig. 4 shows the residual change due to changes in the first, second, and third design variables; this residual change is substantially different than that in the similar plot shown in Fig. 5, which is generated with algorithm 2b. The boxed points in these figures represent the residual values at the end of an optimization cycle. A decreasing trend can be seen in the peaks in Fig. 5, although it is not clearly distinguishable.

The final velocity profiles obtained by the SADDO algorithms are all in very good agreement with the target-velocity profile. However, note that the sample problem does not impose any constraint on the mass flow or the size of the nozzle and, hence, may not yield a unique solution. That is, the optimization algorithm (depending on the initial design) can converge to any of the local minimum points that are characteristic of the objective function. For example, in Table 1, the final values for design variables **b** are changed in the direction opposite to that which would bring them closer to their target values for all methods. Several different initial design variables were tested with the same unconstrained objective function and SAADO scheme (algorithm 2b).

Each resulted in a different final design; the results are tabulated in Table 2. Figure 6 shows the area profiles for all final designs that were generated, as well as the area profile generated for the target. In all cases, the final flow-speed profile was extremely close to the target profile.

4.3 Sample Problem 2: Transonic Nozzle

The variation of the cross-sectional area A in the streamwise direction x in this sample problem is given by the expression

$$A(x) = 1 - b_1 x + b_2 x^2 \qquad (0 \leq x \leq 1) \tag{38}$$

where b_1 and b_2 are the design variables. The target-design variables are $b_1 = 0.8$ and $b_2 = 0.8$, and their initial values are 0.9 and 1.2, respectively. For this transonic flow problem, the free-stream Mach number is 0.55, and a normal shock is found in the diverging part of the nozzle. On the inflow boundary, the entropy and stagnation enthalpy are fixed at the free-stream values; the flow speed is extrapolated. On the outflow boundary, the density and flow speed are extrapolated; the static pressure is held fixed (i.e., $P/P_\infty = 0.9$ in the present sample). Weighting coefficients of 20 for the objective function and 50 for the residual were set in the merit function (Eq. (32)). The penalty-function parameter is updated in a manner similar to the first sample problem.

With a strict application of Newton's root-finding method to solve the nonlinear flow equations, the method commonly "overshoots" the root and diverges. This overestimation is particularly true during the initial start-up phase of the solution procedure and is most commonly experienced when higher order accurate spatial discretizations are applied to flow fields with shocks. Typically, this problem can be overcome with the use of underrelaxation. In this

study, an underrelaxation factor was included to initiate the flow-solver iterations; however, this initial use of underrelaxation was progressively reduced during the iterative process in proportion to the reduction in the average value of the residual of the flow equations. Thus, the use of underrelaxation was progressively removed during the solution process, such that as the root (i.e., the steady-state flow solution) was approached the strict use of Newton's root-finding method was recovered.

In this second sample problem, without the use of underrelaxation, the strict implementation of the basic SAADO scheme (as outlined in section 3.2) was divergent, which was corrected with modifications to the basic algorithm. First, basic Newton flow-solver iterations with underrelaxation were implemented in a manner as described previously to initially reduce the residual of the flow equations approximately two orders of magnitude (1.0E-2). Then, the modified SAADO algorithm was begun by evaluating the adjoint-variable vector(s) (i.e., by solving Eq. (20) (and Eq. (23) if side constraints were present)) directly without the inclusion of any relaxation factor. After the solution of the linear programming problem (Eqs. (24) and (25)) for $\Delta\mathbf{b}$, Eq. (29) is solved directly for $\Delta\mathbf{Q}^*$ without any relaxation factor. Then, two Newton flow-solver iterations, including the use of an underrelaxation factor that was progressively removed (as described previously), were introduced to reduce the residual immediately after the design changes (i.e., immediately after step 4 of section 3.2.2). After the design and state variables were updated, three Newton-type flow-solver iterations (including the progressively removed underrelaxation factor) were again used to improve the solution, and the new design $\mathbf{b}^*$ was held fixed (i.e., immediately after step 5 of section 3.2.2).

Figure 7 compares the initial-, final-, and target-velocity profiles. The SAADO design optimization was carried out for 78 cycles, and its results are reported in Table 3, including the objective function, residual value, and efficiency of the scheme in terms of CPU time and equivalent flow-solver iterations. The residual history and objective function during the design process are shown in Fig. 8. From these results, the intermediate flow-solver steps appear to provide a useful tool to control the quality of the flow solution at the end of the optimization process. Although not explicitly shown here, note that depending on several different combinations of weighting coefficients and the number of intermediate flow-solver type iterations used the final results can vary slightly in terms of the optimum design and the quality of the final flow solution that is achieved.

In Fig. 9, the residual history during the optimization process is shown superimposed on a single standard flow-solver residual history. Figure 10 shows a sample of these residual values at each design iteration on an expanded scale. The peaks and the valleys represent the beginning and the end, respectively, of each optimization cycle. The decreasing trend in the residual is easily discernible in Fig. 10.

5.0 CONCLUSIONS

An optimization technique for nonlinear aerodynamic problems, simultaneous aerodynamic analysis and design optimization (SAADO), has been investigated in this study. This algorithm enables the simultaneous reduction of the objective function and the correction of the constraint violations; in this strategy, the constraints include the residuals of the aerodynamic equations. This design optimization technique requires that the steady-state aerodynamic equations be

satisfied only at the final optimum solution of the design optimization problem. This requirement obviates the need to obtain the complete the steady-state flow solution accurately at every optimization iteration, which is characteristic of conventional design optimization procedures and accounts for the bulk of central processing unite (CPU) time in the entire design optimization process.

The SAADO algorithm and its variants are successfully implemented on two quasi-one-dimensional nozzle-flow problems, including a sample problem where a discontinuity exists (i.e., a normal shock) in the flow field. A substantial increase occurs in the number of design variables and constraints when the field variables are treated as part of the set of design variables and when the aerodynamic flow equations are considered to be part of the set of constraints; this is nullified by the introduction of the adjoint-variable vector. Furthermore, the appropriate introduction of direct Newton iterations (including the use of an underrelaxation factor in the design optimization process) helps to control the value of the residual of the aerodynamic equations. Depending upon the number of flow-solver iterations used and their placement in the design optimization scheme, a trade-off can be achieved between the values of the residual of the aerodynamic equations (at the end of the design process) and the objective function and the CPU time. In both sample problems, a considerable savings in terms of equivalent Newton iterations (and, hence, overall CPU time) is achieved with the SAADO scheme when compared with the conventional black-box design optimization procedure.

In principle, the SAADO algorithm is extendable to large-scale multidimensional flow problems. However, in these cases, the incremental iterative method should be employed to solve the linear equations that are involved in this new optimization scheme.

6.0 REFERENCES

1. M. H. Rizk, 'The Single-Cycle Scheme: A New Approach to Numerical Optimization,' *AIAA Journal*, **21**, 1640–1647, 1983.

2. M. H. Rizk, 'Optimizing Advanced Propeller Designs by Simultaneously Updating Flow Variables and Design Parameters,' *Journal of Aircraft*, **26**, 515-522, 1989.

3. M. Drela, 'Viscous and Inviscid Inverse Schemes Using Newton's Method,' AIAA Paper 91–18044, 1991.

4. R.T. Haftka, 'Simultaneous Analysis and Design,' *AIAA Journal*, **23**, 1099–1103, 1985.

5. R.T. Haftka, 'Integrated Nonlinear Structural Analysis and Design,' *AIAA Journal*, **27**, 1622–1627, 1989.

6. C. E. Orozco and O. N. Ghattas, 'Jacobian and Hessian Sparsity in Simultaneous and Nested Structural Optimization,' AIAA Paper 91-1093-CP, 1991.

7. C. E. Orozco and O.N. Ghattas, 'Sparse Approach to Simultaneous Analysis Design of Geometrically Nonlinear Structures,' *AIAA Journal*, **30**, 1877–1885, 1992.

8. C. E. Orozco and O. N. Ghattas, 'Optimal Design of Systems Governed by Nonlinear Partial Differential Equations,' AIAA Paper 92-4836-CP, *Proceedings of the Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, Sept. 21–23, 1992.

9. S. Ta'asan, G. Kuruvila and M.D. Salas, 'Aerodynamic Design and Optimization in One Shot,' AIAA Paper 92-0025, 1992.

10. S. Ta'asan, 'One Shot Methods for Optimal Control of Distributed Parameters Systems I: Finite Dimensional Control,' ICASE Report No. 91-2, Jan. 1991.

11. V. M. Korivi, A. C. Taylor III, P. A. Newman, G. J.-W. Hou and H. E. Jones, 'An Approximately Factored Incremental Strategy for Calculating Consistent Discrete Aerodynamic Sensitivity Derivatives,'AIAA Paper 92-4746-CP, *Proceedings of the Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, Sept. 21–23, 1992.

12. P. A. Newman, G. J.-W. Hou, H. E. Jones, A. C. Taylor III and V. M. Korivi, 'Observations on Computational Methodologies for Use in Large-Scale, Gradient-Based, Multidisciplinary Design,' AIAA Paper 92-4753-CP, *Proceedings of the Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, Sept. 21–23, 1992.

13. S. V. Mani, *Simultaneous Aerodynamic Analysis and Design Optimization*, M.S. Thesis, Old Dominion University, Norfolk, Virginia, Dec. 1993.

14. L. B. Rall, 'Automatic Differentiation: Techniques and Applications,' *Lecture Notes in Computer Science*, **120**, Springer Verlag, Berlin, Germany, 1981.

15. A. Griewank and G. F. Corliss, eds., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, PA, 1991.

16. C. Bischof, G. Corliss, L. Green, A. Griewank, K. Haigler and P. Newman, 'Automatic Differentiation of Advanced CFD Codes for Multidisciplinary Design,' *Computing Systems in Engineering* **3**, 625–637, 1993.

17. L. Green, P. A. Newman and K. Haigler, 'Sensitivity Derivatives for Advanced CFD Algorithms and Viscous Modelling Parameters via Automatic Differentiation,' AIAA Paper 93-3321, 1993.

18. B. Van Leer, 'Flux-Vector Splitting for the Euler Equations,' ICASE Report 82–30, Sept. 1982 (Also *Lecture Notes in Physics* **170**, 507–512, 1982).

19. J. L. Thomas, B. Van Leer and R. W. Walters, 'Implicit Flux-Split Schemes for the Euler Equations,' *AIAA Journal*, **28**, 973–974, 1990.

20. G. N. Vanderplaats, 'ADS—A Fortran Program for Automated Design Synthesis—Version 1.10,' NASA CR-177985, 1985.

List of Figures

Table 1. Optimization Results: Supersonic Nozzle Problem

Schemes	ADS (FD)	ADS (QA)	SAADO basic alg.	SAADO alg. 1	SAADO alg. 2a	SAADO alg. 2b
Objective	8.8E-4	8.9E-4	4.6E-4	1.8E-4	1.4E-4	8.9E-5
Residual	1.0E-12	1.0E-12	3.0E-4	3.0E-4	2.0E-7	1.0E-12
NR	174	102	40	43	60	80
CPU	2.194	1.286	0.394	0.412	0.521	0.694
Final b_1	1.3272	1.3266	1.3032	1.3048	1.3048	1.3028
Final b_2	0.3236	0.3235	0.3196	0.3210	0.3210	0.3226
Final b_3	3.9900	3.9900	3.9692	3.9780	3.9780	3.9852

Table 2. Illustration of Nonuniqueness of Optimum Design Problem

Solution no.	Initial designs b_1, b_2, b_3	Final designs b_1, b_2, b_3	Objective function
1	1.35, 0.30, 3.9	1.3194, 0.3218, 3.9682	1.9903E-4
2	1.37, 0.30, 3.9	1.3355, 0.3246, 3.9774	2.7759E-4
3	1.39, 0.30, 3.9	1.3473, 0.3305, 3.9931	9.7846E-5
4	1.43, 0.30, 3.9	1.3787, 0.3366, 3.9569	2.2020E-4
5	1.43, 0.39, 3.9	1.3983, 0.3500, 3.9476	2.7540E-4

Table 3. Results for Transonic Flow Problem

Results	SAADO	Regular flow solver
Final objective function	4.2834E-4	—
Final residual	1.86E-10	1.0E-12
Equivalent flow-solver iterations	1374	1100
CPU time, sec	14.3	8.43
Design variables b_1, b_2	0.807, 0.809	0.9, 1.2

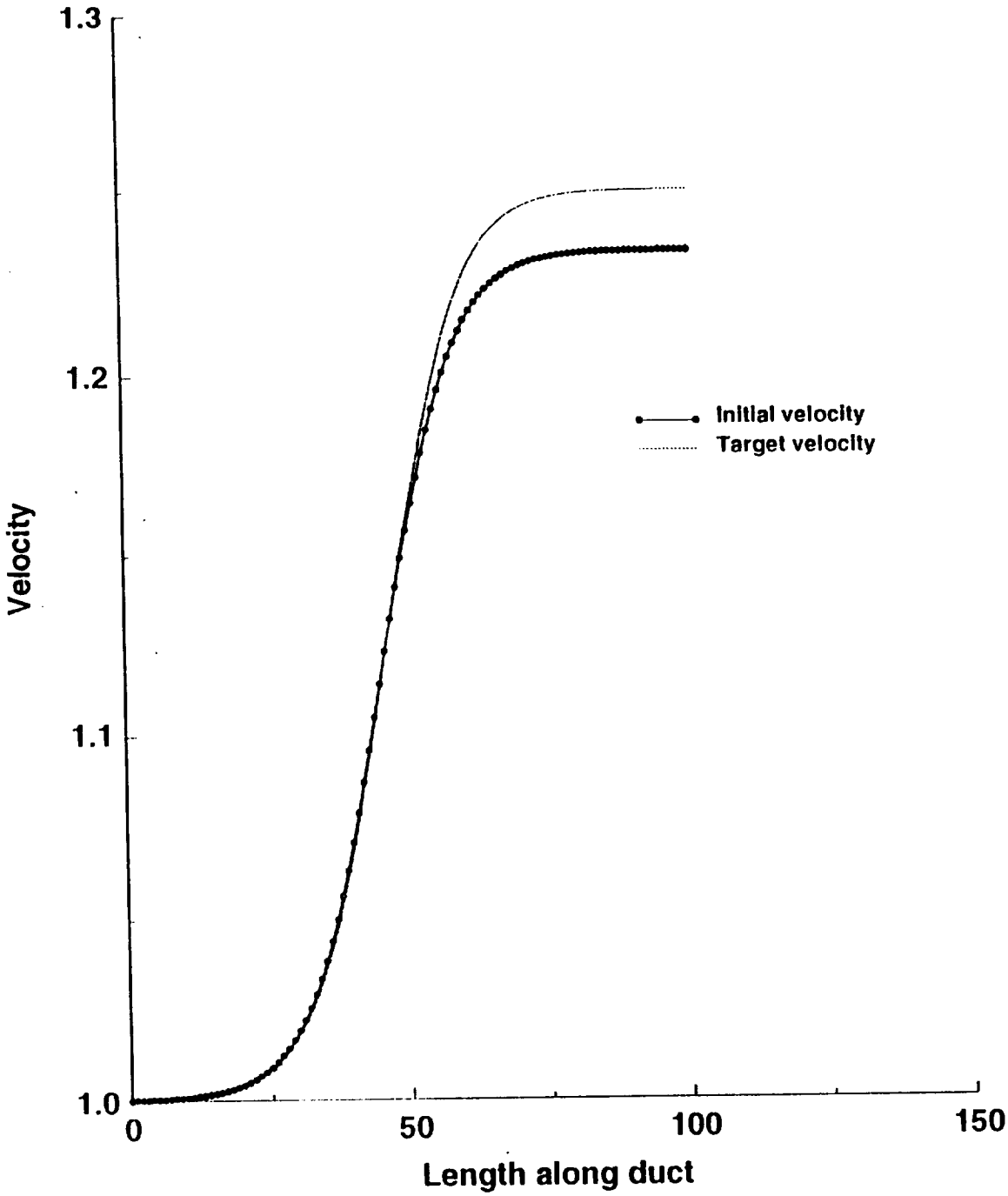

Figure 1. Initial and target velocities (supersonic flow).

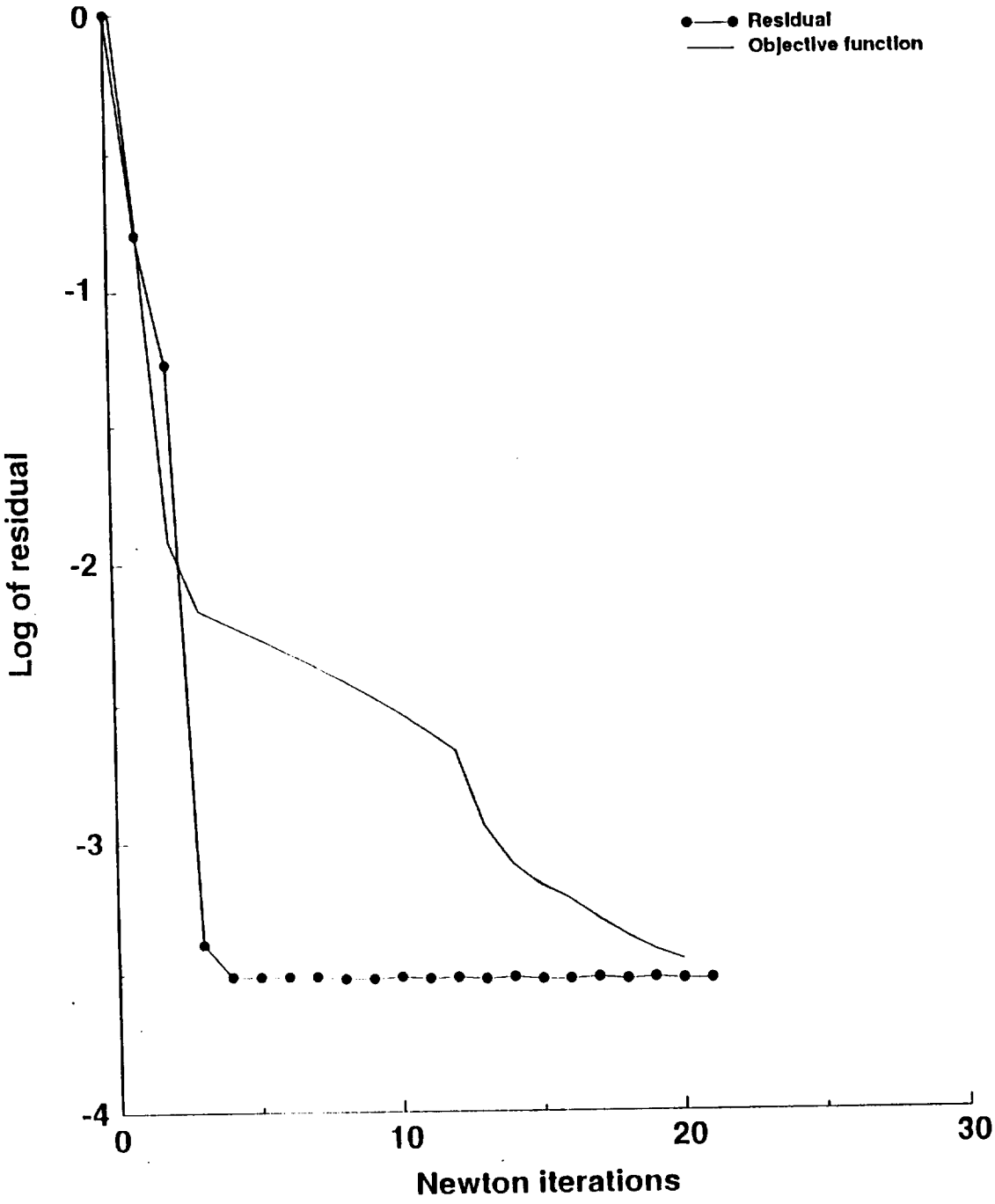

Figure 2. Residual and objective function (basic algorithm).

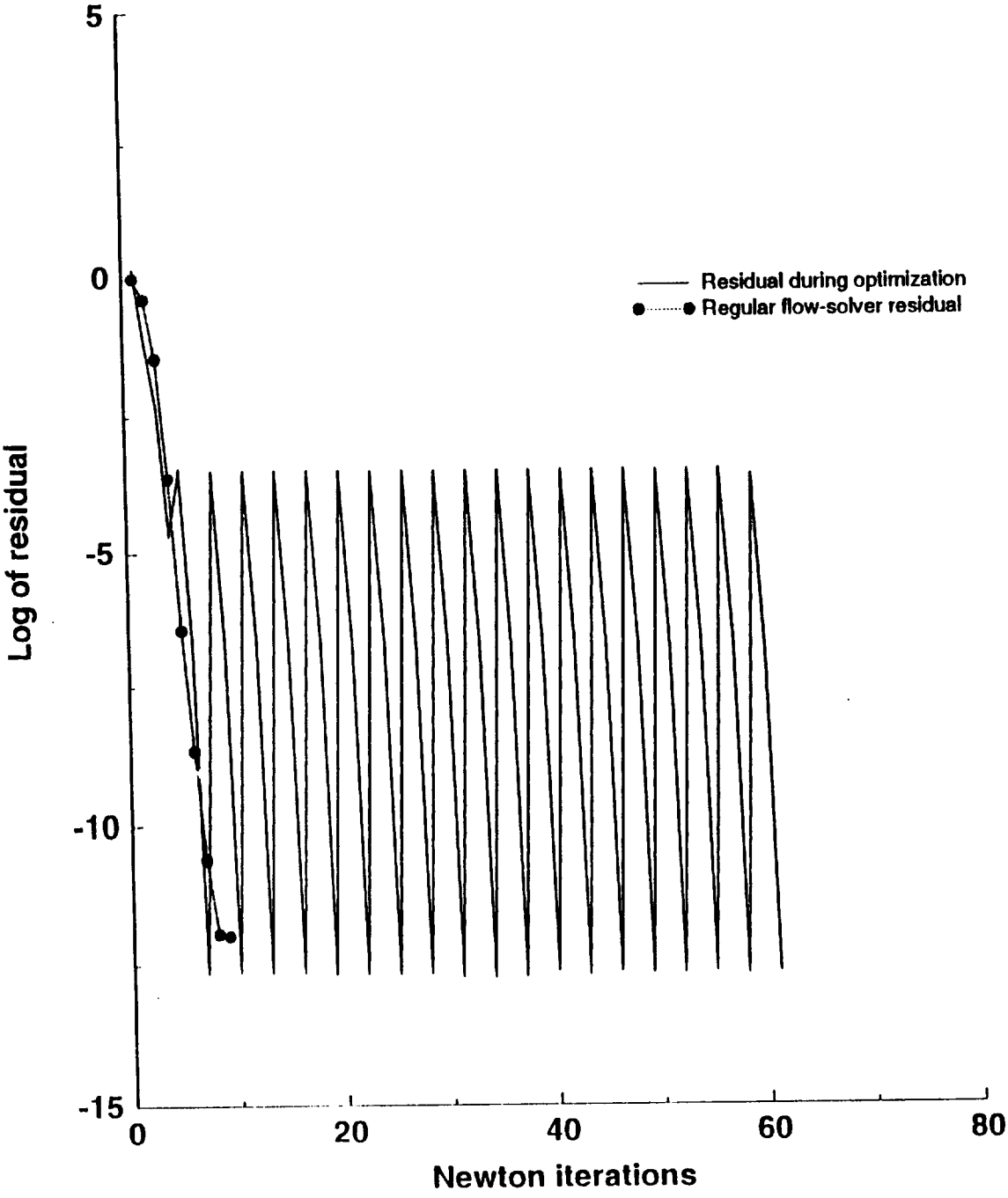

Figure 3. Residual history of SAADO algorithm 2b and flow-analysis code (supersonic flow).

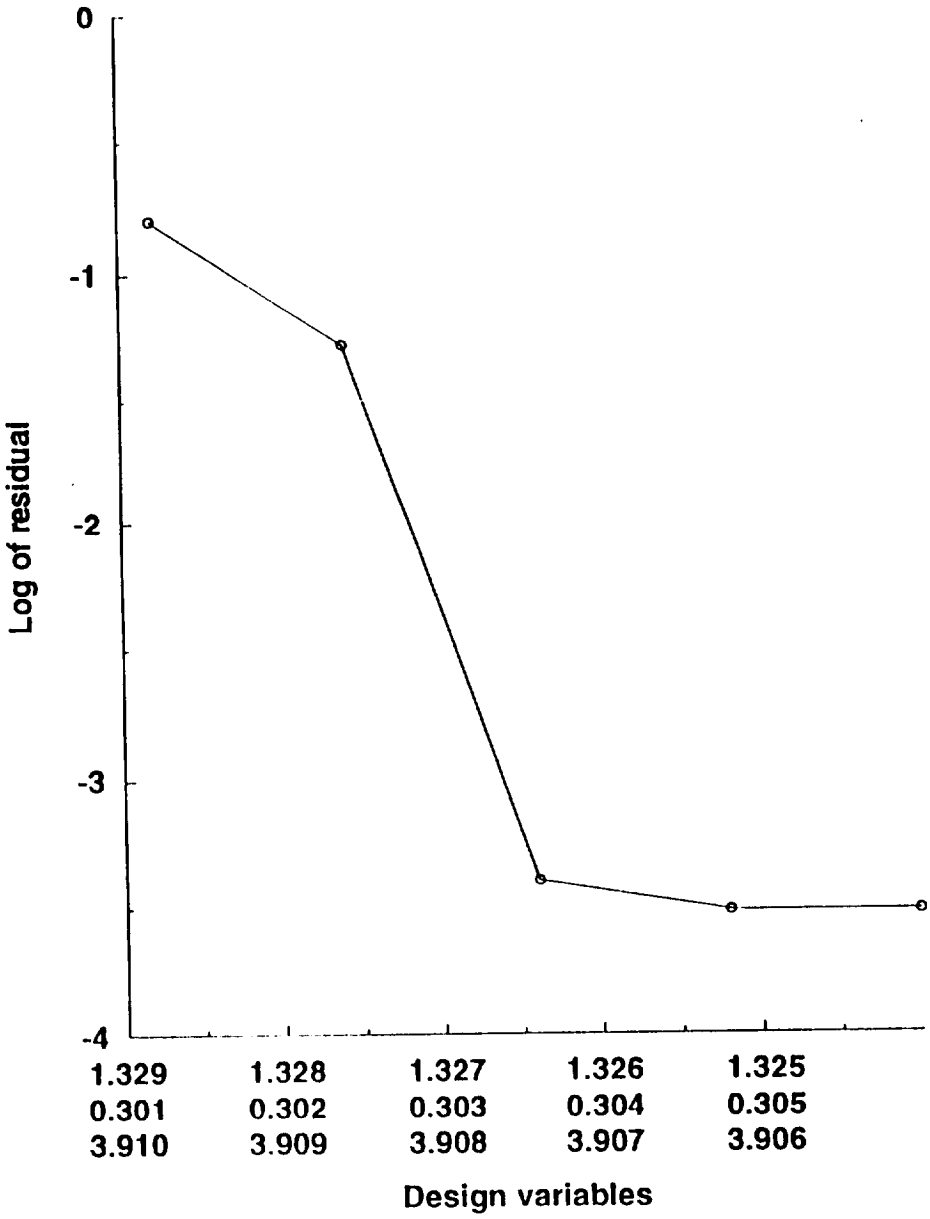

Figure 4. Trend of residual with changes in design (basic algorithm).

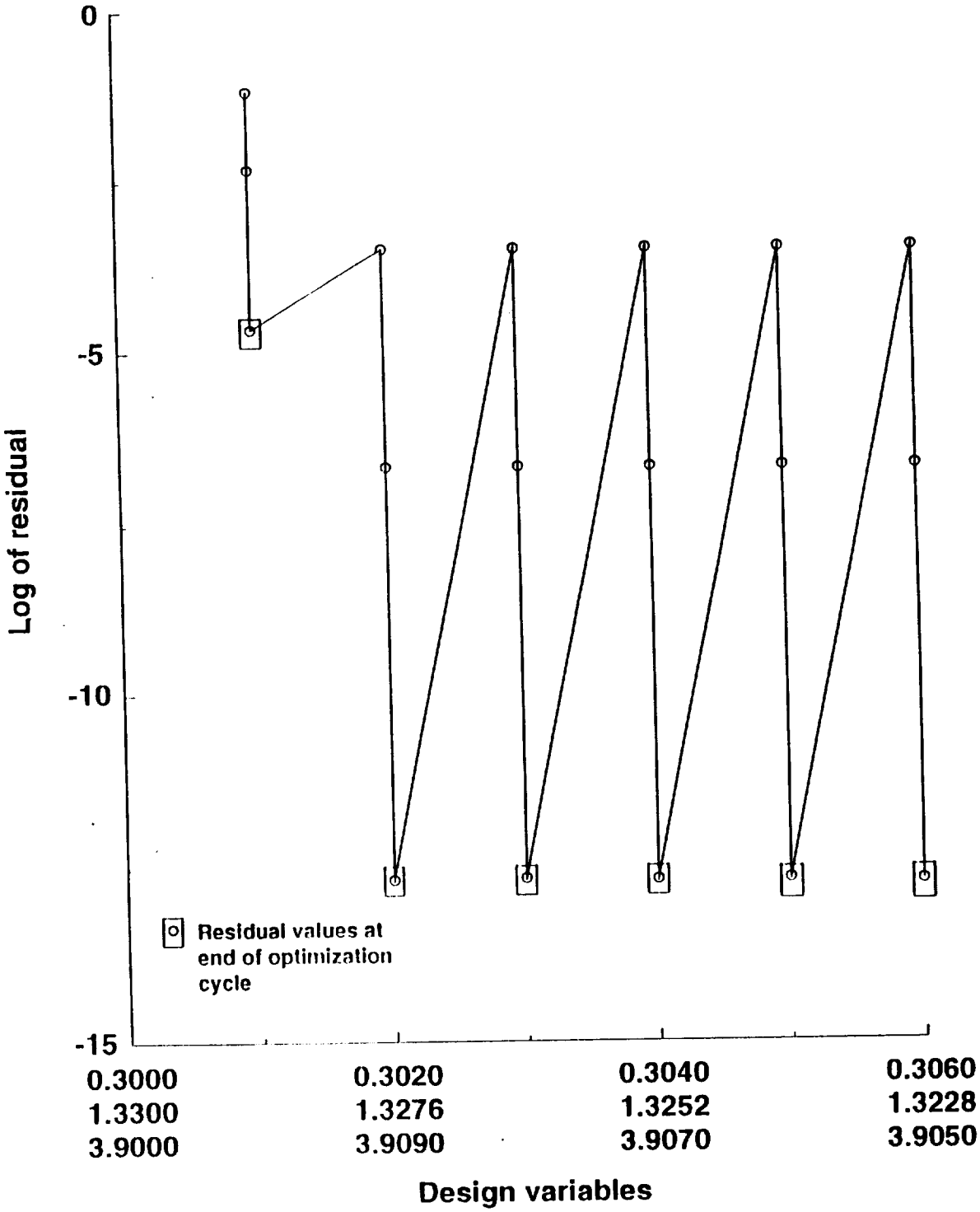

Figure 5. Trend of residual with changes in design (SAADO algorithm 2b).

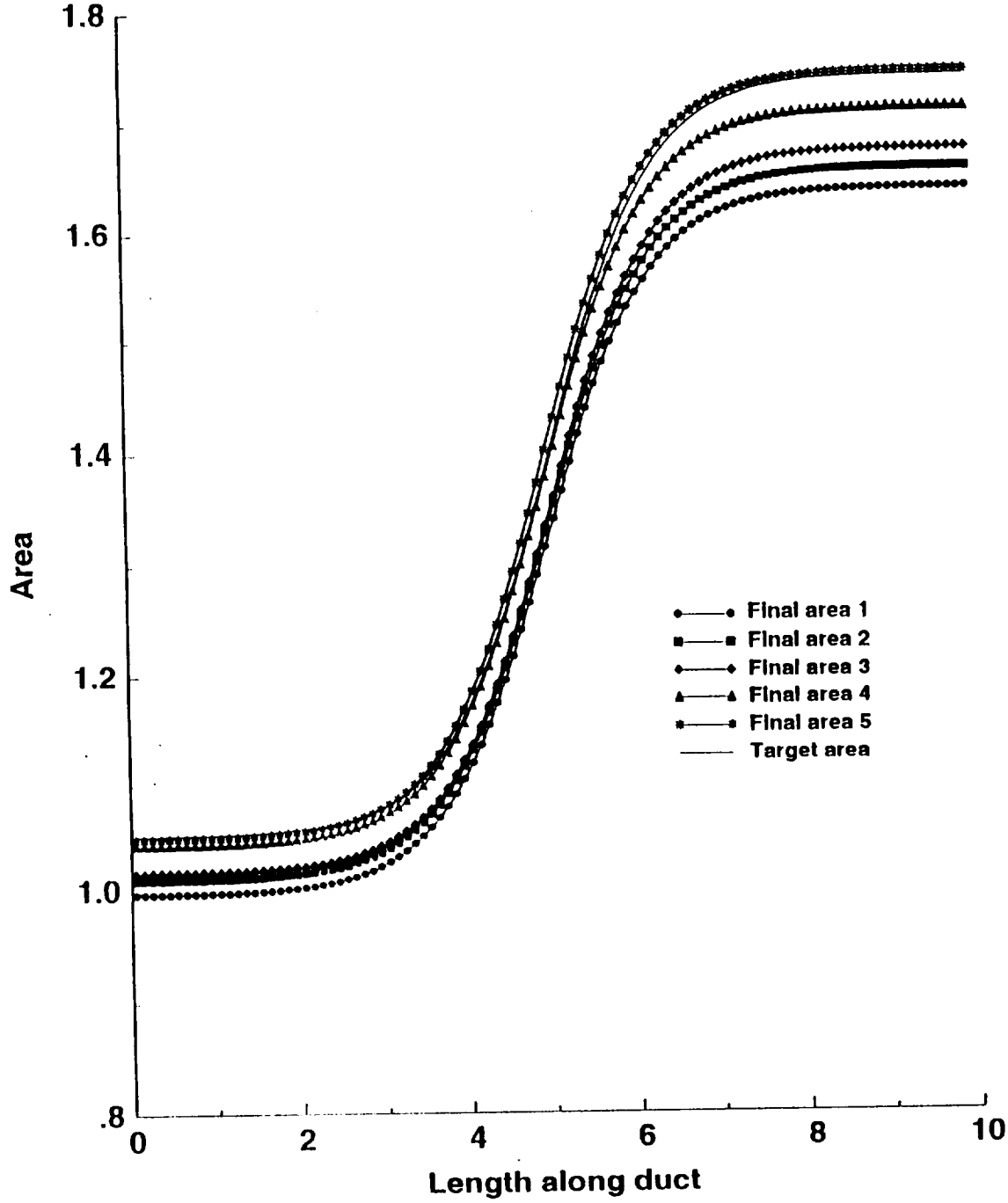

Figure 6. Area profiles of different final designs generated by SAADO algorithm 2b.

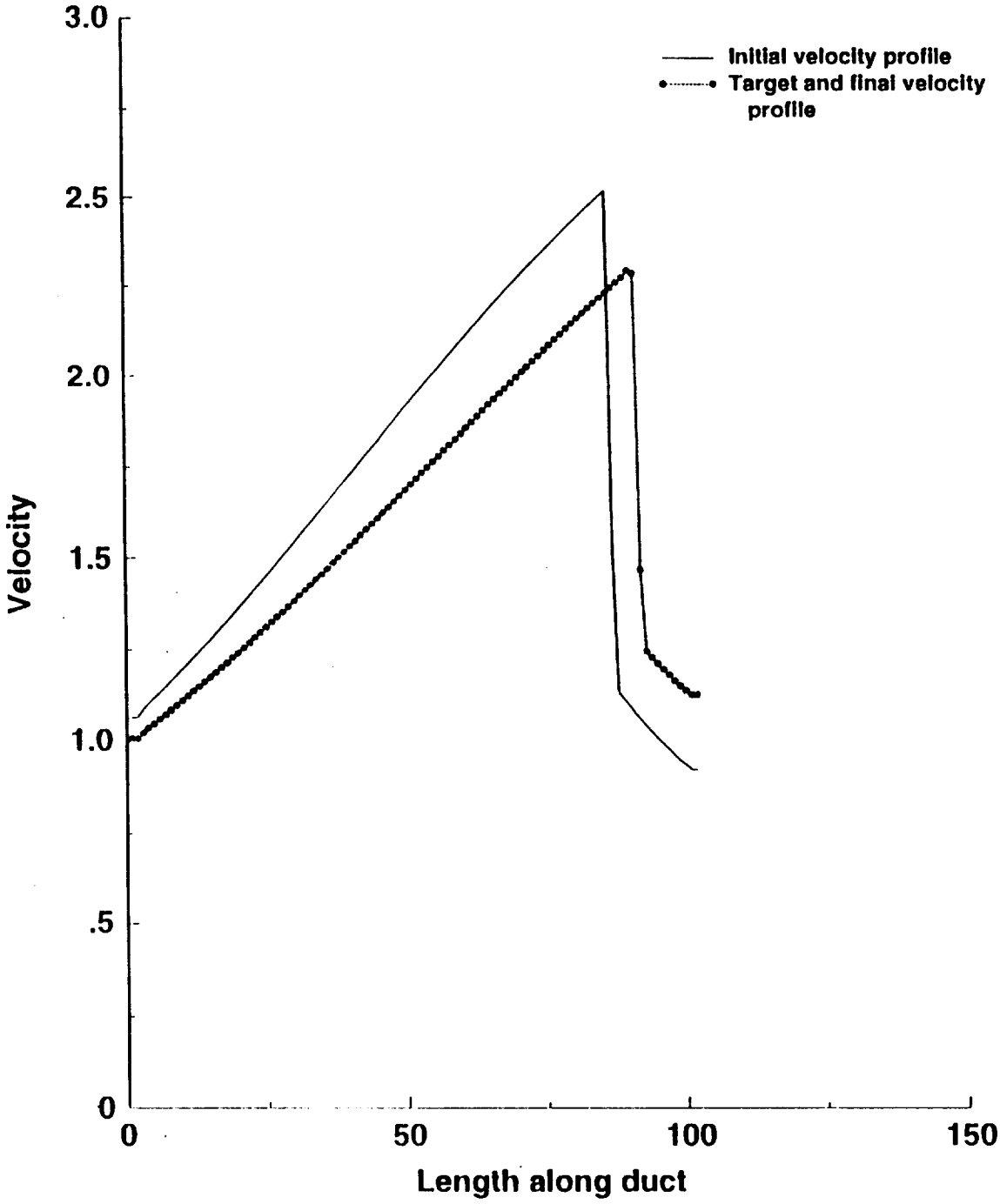

Figure 7. Initial-, final- and target-velocity profiles (transonic flow).

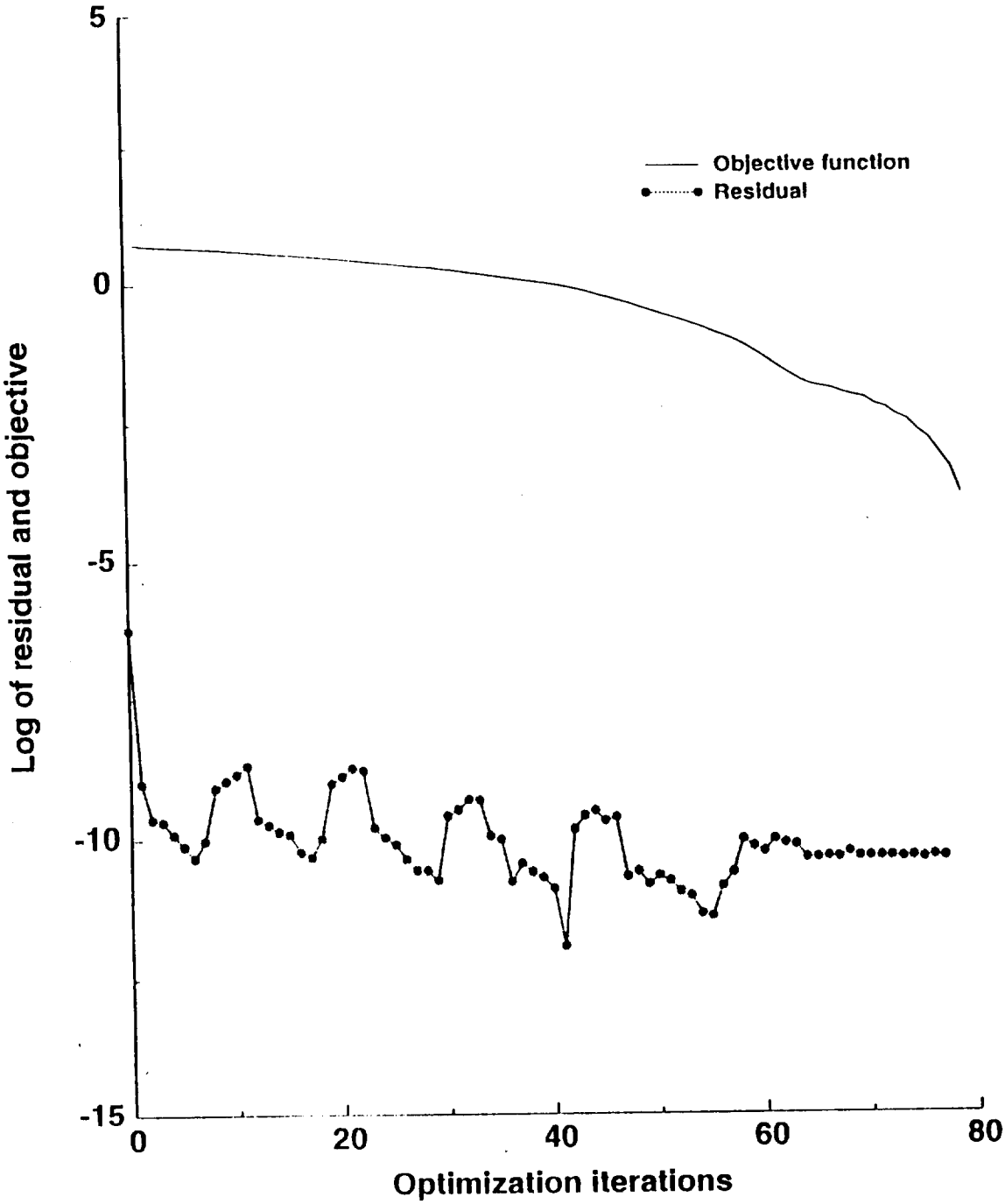

Figure 8. Residual and objective function (transonic flow).

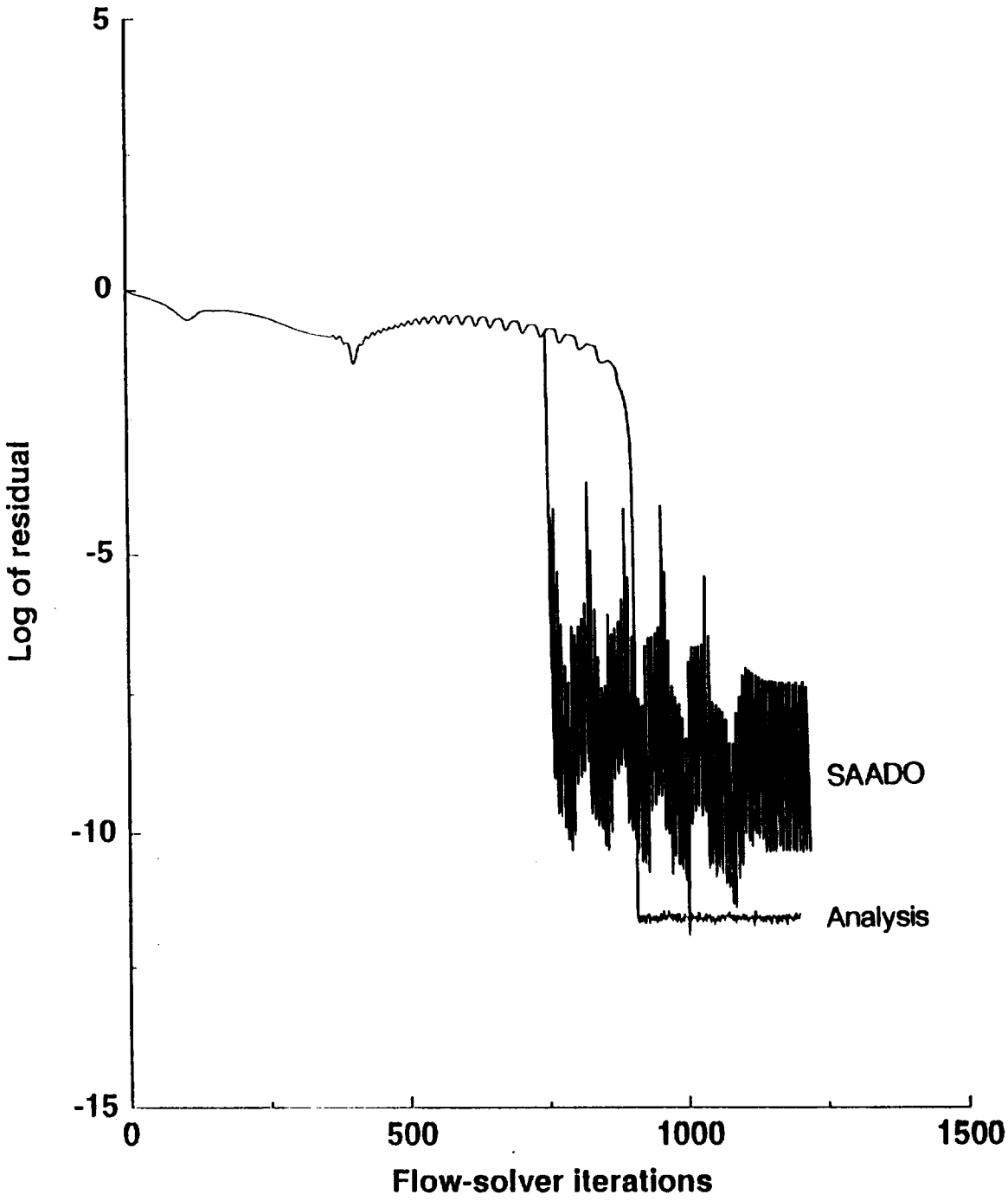

Figure 9. Residual history of SAADO algorithm and single flow solver (transonic flow).

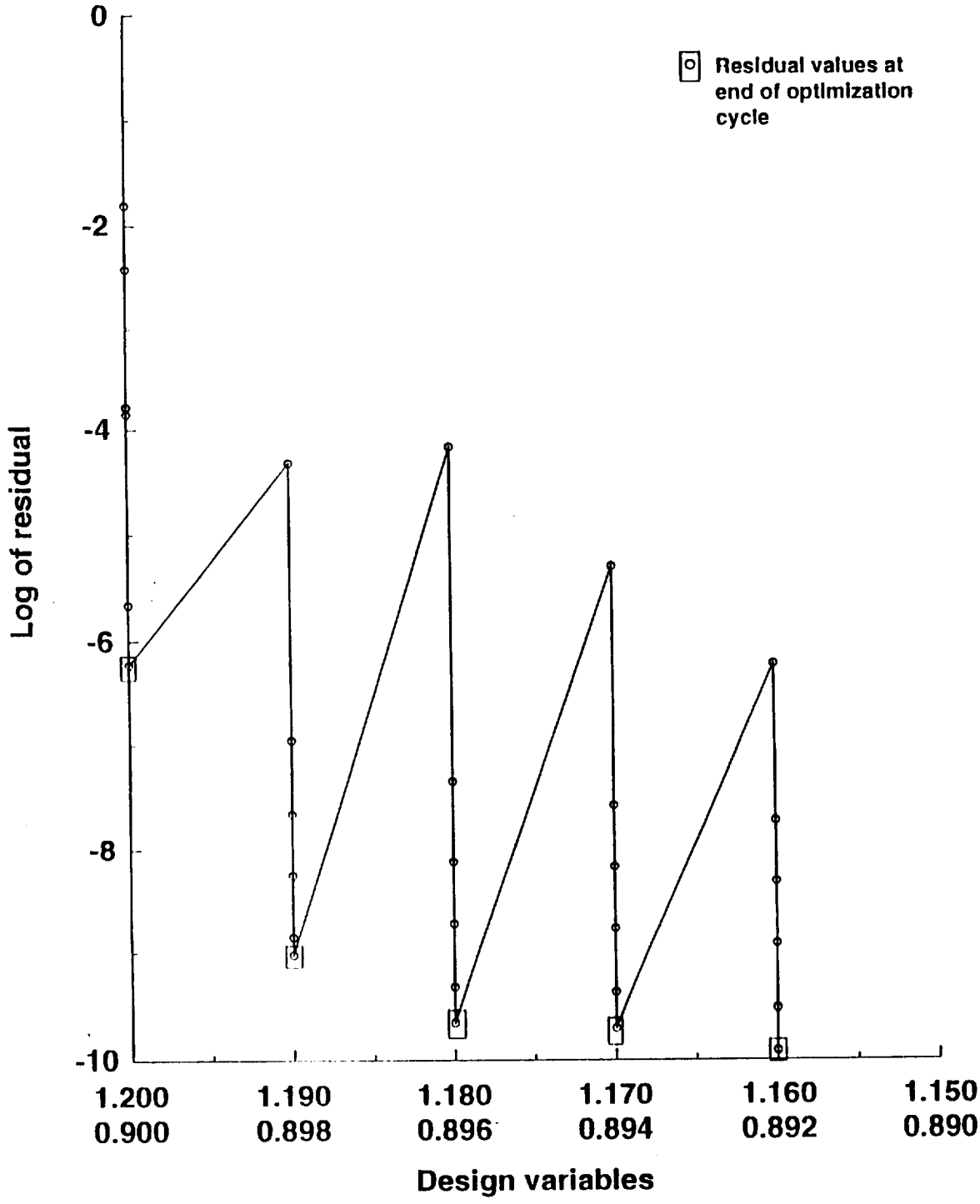

Figure 10. Trend of residual with changes in design (transonic flow).