A Fully-Coupled Algorithm for Aerodynamic Design Optimization

by

John Gatsis

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

A Fully-Coupled Algorithm for
Aerodynamic Design Optimization

John Gatsis
Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto
2001

A fully-coupled algorithm is presented to solve aerodynamic design optimization problems. The discretized flow, adjoint, and optimality condition equations are solved as a single system of nonlinear equations using an inexact Newton method with linesearching. Quasi-1D, inverse, nozzle design and inviscid, 2D, inverse, airfoil shape design cases are solved, however the method is applicable to more general design cases.

While the system is approximately twice as large as the system describing the flow solution, fewer iterations are required to find the optimum. The most important advantage of this method is that only a single flow solve is needed to perform the optimization, as opposed to the discrete adjoint method which requires several. Many improvements have been made to the algorithm, both in the system formation and solution method. The result is an algorithm that is reliable, robust, accurate, and fast in providing optimum solutions for the cases considered.

Acknowledgements

First, and foremost, I would like to thank my supervisor, Professor David Zingg for his wisdom, motivation, and patience. The two years in your group have been among the most memorable ones of my academic and personal life.

Very seldom is one as fortunate as I am to be surrounded by such talented people. Their help and inspiration will be passed on to future students that will enter the CFD group once they have moved on. I would like to thank the following individuals who have made significant contributions to this research:

- Luis Manzano for his encouragement and his support. You truly made me feel at home during my first few months in the CFD group.
- Marian Nemec and Peterson Wong for their expertise and collaboration in optimization. I appreciated the many times you answered my questions and discussed the challenges I was facing. Marian, your knowledge about the optimization and CFD in general is impressive. Peterson, your programming knowledge is outstanding.
- Jason Lassaline for answering all of my many computing questions. You are truly a computer guru.
- Anton de Ruiter for testing my mathematical skill, verifying my many derivations and lending me that excellent optimization book.
- The rest of the CFD group for their friendship, including Stan, Todd, Mike, Ian, Peter, Edward, and Gary.

I would like to thank my parents, Peter and Angela, my sister, Anna, my brother, Joseph, and my grandmother, Katerina, for making many sacrifices over the years for me. Finally, I'd like to thank the Government of Canada for granting me an NSERC fellowship to continue my post-graduate academic studies at the Institute for Aerospace Studies.

JOHN GATSIS

University of Toronto
October 3, 2001

Contents

List of Figures

List of Tables

List of Symbols

Cp	pressure distribution (coefficient of pressure)
Cp^*	target pressure distribution
D_i	discretized artificial dissipation
$\hat{E}$	curvilinear convective flux vector
$\hat{F}$	curvilinear convective flux vector
H	Hessian matrix
J	metric Jacobian, objective function
K	fully-coupled system state vector
L	Lagrange functional
M	Mach number
$\hat{Q}$	continuous state vector
Q	discretized state vector
R	flow equations
R_{flow1D}	1D flow equations
R_{flow2D}	2D flow equations
$\hat{S}$	continuous area distribution
S	discretized area distribution
U	tangential contravariant velocity
V	normal contravariant velocity
V_i	state selection perturbation vector
X	design variables
e	total energy
$\hat{f}$	1D horizontal dimension flux vector
f_i	discretized 1D flux vector
$\hat{g}$	1D source term
g_i	discretized 1D source term
h_{ij}	objective function Hessian component
k_i	fully-coupled system state vector component
n	nodes on 1D grid, iteration number
p	pressure
q_i	state Q, conservative element i

u	horizontal Cartesian velocity
$u^\star$	target velocity distribution
v	vertical Cartesian velocity
v_j	state selection perturbation vector component
x	horizontal Cartesian coordinate

$\Gamma(K)$	fully-coupled system right hand side
Δ	difference
$\Theta(K)$	fully-coupled system Jacobian
Φ	adjoint variables

α_n	linesearch update parameter
β_n	linesearch parameter
δ_j	central difference operator
δ_{ij}	Kronecker delta function
ϵ	small number
ϵ_{mz}	machine zero
η	normal curvilinear coordinate
θ_{ij}	fully-coupled system Jacobian block component
ξ	tangential curvilinear coordinate
ρ	density

BLA	Backtracking Linesearch Algorithm
GMRES	Generalized Minimum Residual Algorithm
HLA	Halving Linesearch Algorithm
KKT	Karush-Kuhn-Tucker
NACA	National Advisory Commitee for Aeronautics
RAE	Royal Aircraft Establishment
RCM	Reverse Cuthill-McKee

List of Tables

List of Symbols

Cp	pressure distribution (coefficient of pressure)
Cp^*	target pressure distribution
D_i	discretized artificial dissipation
$\hat{E}$	curvilinear convective flux vector
$\hat{F}$	curvilinear convective flux vector
H	Hessian matrix
J	metric Jacobian, objective function
K	fully-coupled system state vector
L	Lagrange functional
M	Mach number
$\hat{Q}$	continuous state vector
Q	discretized state vector
R	flow equations
R_{flow1D}	1D flow equations
R_{flow2D}	2D flow equations
$\hat{S}$	continuous area distribution
S	discretized area distribution
U	tangential contravariant velocity
V	normal contravariant velocity
V_i	state selection perturbation vector
X	design variables
e	total energy
$\hat{f}$	1D horizontal dimension flux vector
f_i	discretized 1D flux vector
$\hat{g}$	1D source term
g_i	discretized 1D source term
h_{ij}	objective function Hessian component
k_i	fully-coupled system state vector component
n	nodes on 1D grid, iteration number
p	pressure
q_i	state Q, conservative element i

u	horizontal Cartesian velocity
u^*	target velocity distribution
v	vertical Cartesian velocity
v_j	state selection perturbation vector component
x	horizontal Cartesian coordinate

$\Gamma(K)$	fully-coupled system right hand side
Δ	difference
$\Theta(K)$	fully-coupled system Jacobian
Φ	adjoint variables

α_n	linesearch update parameter
β_n	linesearch parameter
δ_j	central difference operator
δ_{ij}	Kronecker delta function
ϵ	small number
ϵ_{mz}	machine zero
η	normal curvilinear coordinate
θ_{ij}	fully-coupled system Jacobian block component
ξ	tangential curvilinear coordinate
ρ	density

BLA	Backtracking Linesearch Algorithm
GMRES	Generalized Minimum Residual Algorithm
HLA	Halving Linesearch Algorithm
KKT	Karush-Kuhn-Tucker
NACA	National Advisory Commitee for Aeronautics
RAE	Royal Aircraft Establishment
RCM	Reverse Cuthill-McKee

Chapter 1

Introduction

The optimization of modern aircraft design is an important issue for aerospace companies. An inefficient process could translate into a significant loss of profit due to a lengthy design cycle. An important component in this design cycle is the optimization of aerodynamic bodies. With the use of computers, designers are able to simulate the physical problem and iterate the design process, while using fewer wind tunnel tests and models.

It is for this reason that aerodynamic design optimization is currently an important area of research in the field of computational fluid dynamics. Today's optimizers vary in type and capability, depending on what is required of them. Typically, the objectives in the design of aerodynamic bodies include such aspects as drag reduction and maximizing lift for a given set of freestream conditions. Furthermore, many constraints are also present in realistic optimization problems. Optimizers must be efficient and robust in order to solve such problems.

1.1 Background

Currently, the two main classes of optimizers are search-based and gradient-based. Search-based optimizers, such as genetic algorithms, are able to handle a broad range of optimization problems, including global optimization problems, and are usually easy to implement. However they are generally slow in terms of the number of flowfield evaluations required. Furthermore, they are not efficient near an optimum solution [1]. While gradient-based optimizers are efficient near an optimum solution, they still require many flowfield evaluations and are not able to handle as broad a range of optimization problems as search-

based optimizers. Other algorithms exist which contain elements from both of these classes of optimizers or try to solve the equations describing the optimization problem in a different manner. Ta'asan [2] chooses to solve the governing equations by marching each of them separately with a pseudo-time parameter. The method resembles a gradient-based approach, except the order in which the governing equations are solved is changed. Feng and Pulliam [3] combine some of the governing equations into a new system and solve this system with a combined gradient-based and linesearch approach. Jou et. al. [4] use a similar approach called a reduced space form of the classic Lagrange-Newton method.

The key issues in developing an efficient gradient-based optimizer are using an efficient flow solver for each design iteration and reducing the cost of computing the gradient. Originally the gradient was found using a finite-difference approach; however the associated computational cost was high. Since then, other approaches have been developed. Currently, the adjoint-based gradient computation approach is popular. This approach can be implemented in either a continuous [5] or discrete form [6].

The optimizer presented in this paper solves the same set of governing equations defined by the discrete adjoint method in a completely different manner. It couples all of the governing equations associated with the shape design optimization problem, including the flow and adjoint equations as well as the optimality conditions, into a single system. This coupled system is then solved for the optimum shape and its corresponding flowfield solution using an iterative method which combines an inexact Newton method with a linesearch. The system to be solved is a little over twice the size of the discrete flow system. The advantage is that this approach does not require that the analysis problem be repeatedly solved to convergence. Instead, all of the components in the coupled system converge together. The final converged solution is identical to that obtained using the discrete adjoint method. Similar to the gradient-based discrete adjoint method, this algorithm also finds local optima. Preliminary results in 1D have been presented [7].

1.2 Project Definition and Statement of Objectives

The primary objective of this thesis is to implement the fully-coupled algorithm for quasi-1D nozzle design problems and to extend the algorithm to 2D, inviscid, airfoil shape design problems. Inverse design problems are only considered for this project; however, the algorithm is applicable to more general problems.

Secondary to these is objectives is the issue of making the process more efficient. By making components of the process more efficient, the development stage of the optimizer is shortened, since diagnostic runs require less time. Many improvements have been made and are presented, although future modifications are suggested in the concluding remarks.

Chapter 2

Governing Equations

In this chapter, the governing equations for the inverse design optimization problem are presented. In industry, inverse design optimization is one approach for designing airfoil geometries. In this method, a target pressure distribution that is needed to satisfy a given set of performance criteria is given to the optimizer in the form of an objective function. The optimizer then returns an airfoil geometry that best optimizes this objective function.

In this thesis, only inverse design optimization is considered, but more general objective functions can be minimized. We use an inverse problem with a known solution for testing and development purposes. It is constructed as follows:

1. Solve the analysis problem for a given shape.
2. Store the pressure distribution from this flow solve.
3. Pick a new shape as an initial guess.
4. Pick an initial flow distribution:
   - it does not have to match the initial guess shape
   - this thesis uses the converged flow distribution for this new shape for simplicity
5. Define an objective function based on the pressure distribution.
6. Run the optimizer to minimize the objective function.

The first step in the approach used for this thesis does not exist in industry. It is final result. However, by by prescribing the target pressure distribution based on a pre-defined shape, the optimization should yield a shape that is identical, within numerical accuracy, to the original shape, thus avoiding any issues related to many local optima.

The first section describes the analysis problem for both the 1D nozzle and the 2D airfoil. In the second section, the fully-coupled formulation is introduced. Finally the chapter concludes with the definition of various objective functions that are used.

2.1 Analysis Problem

For the converging-diverging nozzle optimization, the analysis problem consists of the quasi-1D Euler equations as described in [8]. For the airfoil design optimization, the analysis problem consists of the 2D Euler equations as described in [9].

2.1.1 Quasi-1D Euler Equations

The steady, quasi-1D Euler Equations are given by

$$-\frac{\partial \hat{f}(\mathcal{Q},\mathcal{S})}{\partial x} + \hat{g}(\mathcal{Q},\mathcal{S}) = 0. \tag{2.1}$$

where the ($\hat{}$) symbol signifies a continuous function or variable. The continuous flow variables, $\mathcal{Q}$, which have a dimension of 3×1, are given by

$$\mathcal{Q} = \begin{pmatrix} \rho \\ \rho u \\ e \end{pmatrix}, \tag{2.2}$$

where ρ, u, and e denote density, velocity and total energy respectively. We use the following continuous nozzle area function, $\mathcal{S}$:

$$\mathcal{S}(x) = \begin{cases} 1 + 1.5(1 - \frac{x}{5})^2 & \text{for } 0 \leq x \leq 5 \\ 1 + 0.5(1 - \frac{x}{5})^2 & \text{for } 5 < x \leq 10. \end{cases} \tag{2.3}$$

The flux term, $\hat{f}$ is given by

$$\hat{f} = \mathcal{S} \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (e+p)u \end{pmatrix}, \tag{2.4}$$

and the source term, $\hat{g}$ is given by

$$\hat{g} = \begin{pmatrix} 0 \\ p\frac{dS}{dx} \\ 0 \end{pmatrix}, \tag{2.5}$$

where p denotes pressure and is given by the equation of state for a perfect gas

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho u^2\right). \tag{2.6}$$

The equations are non-dimensionalized by inlet quantities.

The equations are then discretized on an equally-spaced grid and artificial dissipation, D_i, is added giving

$$-(\delta_x f)_i + g_i + D_i = 0, \tag{2.7}$$

where $i \in [1, n]$. Dirichlet boundary conditions are used for simplicity. A central differencing scheme is used to calculate $(\delta_x f)_i$ as follows

$$(\delta_x f)_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x}, \tag{2.8}$$

where Δx is the distance between any two adjacent nodes. The artificial dissipation stencil used for D_i is given in [10].

Finally, the discretized flow equations are written as

$$R_{\text{flow1D}}(Q, S) = 0, \tag{2.9}$$

where R_{flow1D}, Q and S are all vectors of length $3 \times n$.

2.1.2 2D Euler Equations

The 2D Euler equations written in generalized curvilinear coordinates are

$$\frac{\partial \hat{E}}{\partial \xi} + \frac{\partial \hat{F}}{\partial \eta} = 0, \tag{2.10}$$

where the continuous 4×1 flow variables vector Q is given by

$$Q = J^{-1}\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix}. \tag{2.11}$$

The convective flux vectors are

$$\hat{E} = J^{-1} \begin{pmatrix} \rho U \\ \rho U u + \xi_x p \\ \rho U v + \xi_y p \\ (e+p)U \end{pmatrix} \tag{2.12}$$

$$\hat{F} = J^{-1} \begin{pmatrix} \rho V \\ \rho V u + \eta_x p \\ \rho V v + \eta_y p \\ (e+p)V \end{pmatrix} \tag{2.13}$$

with

$$U = \xi_x u + \xi_y v, \quad V = \eta_x u + \eta_y v \tag{2.14}$$

the contravariant velocities. Pressure is related to the conservative flow variables, Q, by the equation of state for a perfect gas, as follows

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right). \tag{2.15}$$

The variable J represents the metric Jacobian of the transformation and is given by

$$J^{-1} = (x_\xi y_\eta - x_\eta y_\xi). \tag{2.16}$$

The equations are non-dimensionalized by freestream quantities.

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

$$(\delta_\xi E)_i + (\delta_\eta F)_i + D_i = 0 \tag{2.17}$$

where the (δ_ξ) and (δ_η) operators indicate second-order central difference approximations to the derivatives in the ξ and η directions, respectively. The boundary conditions used follow the method described by Pulliam [9]. A nonlinear scalar numerical dissipation model is used following the method described in [10].

The discretized flow equations are written as

$$R_{\text{flow2D}}(Q) = 0 \tag{2.18}$$

where R_{flow2D} and Q are both of length $4 \times n$, where n indicates the number of nodes in the computational grid.

2.2 Fully-Coupled Design Formulation

To reduce the size of the overall system being formed, cubic spline points are used to describe the shape of a given nozzle and B-spline control points are used to describe a given airfoil geometry. These points are the design variables, X, for the optimization problem. Hence, the flow equations, which act as a constraint on the optimization problem, and were previously written as (2.9) and (2.18), are rewritten as

$$R(Q, X) = 0. \tag{2.19}$$

The objective function for the optimization problem is denoted as $J(Q, X)$. However, for inverse design optimization problems this function generally does not depend on the design variables. Hence, the dependency of J on X is eliminated. (Note that the symbol J from now only will indicate the objective function, rather than the metric Jacobian).

A new objective functional is defined as

$$L(Q, \Phi, X) \equiv R(Q, X)^T \Phi - J(Q), \tag{2.20}$$

where the variables Φ represent the adjoint variables and are equivalent to Lagrange multipliers. The stationary values of (2.20) with respect to Q, Φ, and X yield the governing equations for the optimization problem. They include the original flow equations (2.19), the adjoint equations

$$(\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial J}{\partial Q})^T = 0 \tag{2.21}$$

and the optimality conditions

$$(\frac{\partial R}{\partial X})^T \Phi = 0. \tag{2.22}$$

The methodology for deriving the governing equations may be found in [11]. By combining these three equations, a fully-coupled system of nonlinear equations is created

$$\Gamma(K) = \begin{pmatrix} R \\ (\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial J}{\partial Q})^T \\ (\frac{\partial R}{\partial X})^T \Phi \end{pmatrix} = 0 \tag{2.23}$$

where the state variables, K, are given by

$$K = \begin{pmatrix} Q \\ \Phi \\ X \end{pmatrix}. \tag{2.24}$$

2.3 Objective Functions

For the nozzle inverse design problems, a desired velocity distribution is given by

$$u_i^* \ \forall \ i \in [1, n] \tag{2.25}$$

where n denotes the number of nodes on the grid. The objective function to be minimized is then defined as

$$J(Q) \equiv \frac{1}{2} \sum_{i=1}^{n-1} (u_i - u_i^*)^2. \tag{2.26}$$

For the airfoil inverse design problems, a desired pressure distribution is given by $C_{p_i}^*$ for all i on the airfoil surface. The objective function is then defined as

$$J(Q) \equiv \frac{1}{2} \sum_i ((C_p)_i - (C_p{}^*)_i)^2. \tag{2.27}$$

Chapter 3

Solution Procedure

In the previous chapter, the governing equations for the optimization problem were presented. They were summarized by (2.23). In this chapter, the fully-coupled algorithm solution procedure is described. The basic iterative method consists of an inexact Newton iteration, followed by a linesearch on the update. Details of the formation of the Jacobian are then given, followed by a description of the block inversion algorithm that is used to compute the update. The chapter concludes with a presentation of the linesearch methods that are used.

3.1 Iterative Method

A Newton update for K_n is first determined by solving

$$\Theta(K_n)\Delta K_n = -\Gamma(K_n) \tag{3.1}$$

for ΔK_n, where $\Theta(K_n)$ is the Jacobian of the system. Next, a linesearch is used to determine the parameter $\alpha_n \in (0, 1]$ which restricts the update, ΔK_n, as follows

$$K_{n+1} = K_n + \alpha_n \Delta K_n. \tag{3.2}$$

3.2 System Components and Decomposition

The Jacobian in the Newton update equation (3.1) is given by

$$\Theta(K) = \frac{\partial \Gamma(K)}{\partial K} \tag{3.3}$$

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

$$\Theta(K) = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}, \tag{3.4}$$

where

$$\begin{aligned} \theta_{11} &= \frac{\partial R}{\partial Q} \\ \theta_{12} &= \mathbf{0} \\ \theta_{13} &= \frac{\partial R}{\partial X} \\ \theta_{21} &= \frac{\partial}{\partial Q}[(\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial J}{\partial Q})^T] \\ \theta_{22} &= (\frac{\partial R}{\partial Q})^T \\ \theta_{23} &= \frac{\partial}{\partial X}[(\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial J}{\partial Q})^T] \\ \theta_{31} &= \frac{\partial}{\partial Q}[(\frac{\partial R}{\partial X})^T \Phi] \\ \theta_{32} &= (\frac{\partial R}{\partial X})^T \\ \theta_{33} &= \frac{\partial}{\partial X}[(\frac{\partial R}{\partial X})^T \Phi]. \end{aligned} \tag{3.5}$$

The dimension n is the length of the flow variable vector, Q, and the dimension m is the length of design variable vector, X.

Some of the elements are extremely difficult to compute analytically. For instance, θ_{21}, θ_{23}, θ_{31}, and θ_{33} contain second-order derivatives with respect to the flow variables and/or design variables. For example, the formation of just the first derivative of the residual vector with respect to the flow variables (i.e. the Jacobian) requires the hand-calculation of of many elements which are situated at various regions of the grid. Imagine trying to then take derivatives of the flow Jacobian with respect to the flow variables again or even the design variables. Although the majority of the cross-derivative terms would be zero, a great deal of care and time would be involved in completing that task.

Thus, a first-order, forward finite-difference formulation of the Jacobian is used. In this formulation, the i^{th} column of the Jacobian at iteration n, $\Theta_i(K_n)$, is formed as follows

$$\Theta_i(K_n) = \frac{\partial \Gamma(K_n)}{\partial k_i} = \frac{\Gamma(K_n + \epsilon k_i V_i) - \Gamma(K_n)}{\epsilon k_i}, \tag{3.6}$$

where k_i is the i^{th} element of K_n and ϵ is a small number given by

$$\epsilon \parallel V_i \parallel \equiv \sqrt{\epsilon_{mz}}, \tag{3.7}$$

and ϵ_{mz} is machine zero. V_i is defined as

$$V_i \equiv \{v_1 \ldots v_{n-1}\}^T, \quad \text{where} \quad v_j = \delta_{ij} \tag{3.8}$$

and δ_{ij} is the Kronecker delta function. Consequently,

$$\parallel V_i \parallel = 1 \quad \forall \quad i \in [1, n]. \tag{3.9}$$

Preliminary tests show that this process is very expensive and inefficient. In fact there are some similarities and simplifications that can be found in the Jacobian that should be taken advantage of. For instance, the flow Jacobian, θ_{11}, occurs twice, as does the block term given by θ_{13}. The flow Jacobian, rather than be approximated by finite differences can be more cheaply, but less accurately, approximated by a linearization used by Pueyo [12]. Furthermore, the component θ_{12}, which accounts for roughly one-quarter of the entire fully-coupled system, is zero for all optimization problems.

The many similarities are exploited in the block inversion algorithm which is described in Appendix A.1. The algorithm consists of solving the 3x3 block system in an analogous manner to a 3x3 scalar system. Furthermore, the approximate formation of some elements is implemented to make the method more efficient. This will be discussed in more detail in Chapter 5 where airfoil design is considered and the discretized system becomes much larger.

3.3 Linesearch Methods

The linesearch parameter, α_n, found in (3.2) must satisfy two conditions: First,

$$\parallel \Gamma(K_n + \alpha_n \Delta K_n) \parallel_2 \; < \; \beta_n \parallel \Gamma(K_n) \parallel_2 . \tag{3.10}$$

where $\beta_n > 1$ is a parameter to allow for some increase in the residual. Its effect is seen mainly in startup iterations for transonic cases.

Second, α_n cannot make the update, K_{n+1}, violate any geometric constraints that are implicitly understood. For example, in 1D, the nozzle cannot have a downstream area

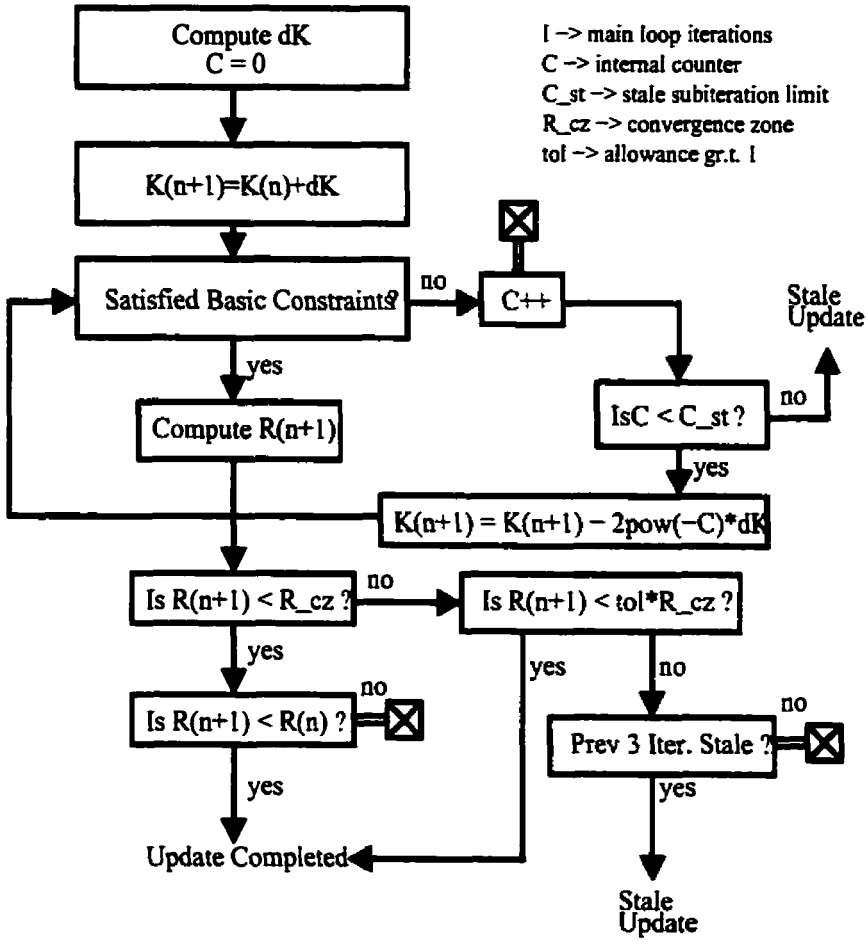

Figure 3.1: Detailed Halving Linesearch Algorithm.

which exceeds the inlet area. In other words, the nozzle cannot turn into a diffuser. Similarly, in 2D, the airfoil geometry can not cross over itself and must exceed some predefined minimum thickness. If either of these conditions are violated, then α_n is reduced by half and so on. If α_n becomes smaller than a predefined lower bound (e.g. 0.01) then α_n is no longer reduced. This method will be referred to as the *halving linesearch method*. A more detailed description of the algorithm is given in Figures 3.1 and 3.2.

To demonstrate that the linesearch method plays an important role in the efficiency of the optimizer, a more advanced *backtracking linesearch method* is developed. If a Newton update is not possible, the method defines a scalar function to represent the vector function being minimized. A quadratic polynomial in α_n is first used to approximate this scalar

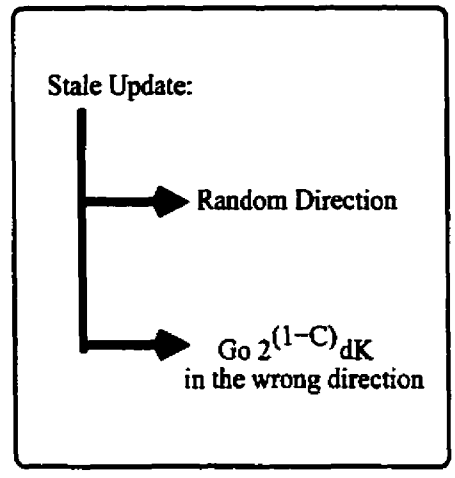

Figure 3.2: Stale Update Algorithm.

function. If the quadratic polynomial has no minimum value for $\alpha_n \in (0, 1]$ or even a value that is less than its value for $\alpha_n = 0$, then the method approximates successive cubics of decreasing domain size in α_n until a value of α_n is found where all of the conditions for the linesearch are satisfied. Similar to the backtracking linesearch method, if α_n becomes smaller than a predefined lower bound (e.g. 0.01) then α_n is no longer reduced. The same constraint is used on the design variable update also; however, the linesearch internally replaces condition (3.10).

Chapter 4

Quasi-1D Nozzle Design Results

We solve the inverse design problems for the converging-diverging nozzle listed in Table 4.1. The inlet and outlet boundary conditions for these cases are given in Table 4.2. A grid with 65 internal nodes is used. The inlet and outlet areas are fixed to $2.5\,\mathrm{m}^2$ and $1.5\,\mathrm{m}^2$ respectively. The initial shape guess for both the 2 and 10 design variable cases is a converging duct which matches the area boundary constraints, as shown in Figures 4.1 and 4.2. The initial state guess is the corresponding flow distribution for the initial shape guess. The target velocity distributions for all four cases are given in Figures 4.3 - 4.6.

The convergence criterion for all of the nozzle optimization problems considered is

$$\| \Gamma(K_n) \|_2 \; < \; 1.0 \times 10^{-10} \tag{4.1}$$

where $\Gamma(K_n)$ is the fully-coupled right hand side at iteration n.

For the halving linesearch method, the parameter β_n in equation 3.10 is set to 2.0 for the first iteration, and to 1.2 for all subsequent iterations to allow for some increase in the residual. This is necessary for the transonic cases, when transient effects are experienced as the flow solution is developing (e.g. shock formation).

The backtracking linesearch method is not used to solve the transonic cases. The algorithm requires some modifications to deal with the formation of the shock, as did the halving linesearch method. Since the fully-coupled algorithm is working, satisfying the objective of this thesis, the further development of the backtracking linesearch method is left as a future area of research in order to move on to work in airoil shape design optimization.

Case	Flow	Design Variables	Linsearch Method(s)
1	Subsonic	2	Halving & Backtracking
2	Subsonic	10	Halving & Backtracking
3	Transonic	2	Halving
4	Transonic	10	Halving

Table 4.1: 1D Nozzle Design Test Cases.

Quantity	Units	Subsonic	Transonic
ρ_{in}	kg/m^3	1.141	1.128
u_{in}	m/s	6.546×10^{1}	8.291×10^{1}
e_{in}	$kg/m/s^2$	2.463×10^{6}	2.440×10^{6}
ρ_{out}	kg/m^3	1.101	1.026
u_{out}	m/s	1.131×10^{2}	1.517×10^{2}
e_{out}	$kg/m/s^2$	2.390×10^{6}	2.243×10^{6}

Table 4.2: Boundary Conditions.

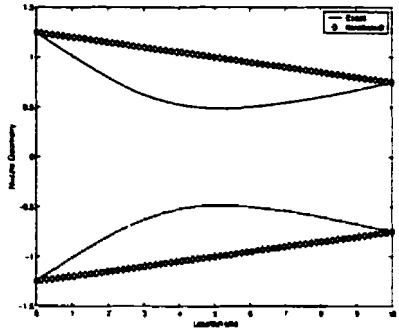

Figure 4.1: Initial shape guess for 2 design variable cases.

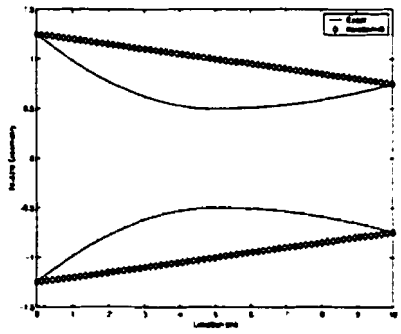

Figure 4.2: Initial shape guess for 10 design variable cases.

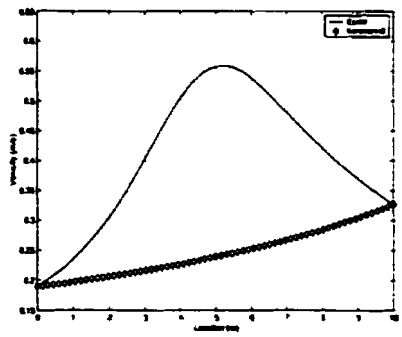

Figure 4.3: Target velocity and initial guess for subsonic, 2 design variable case.

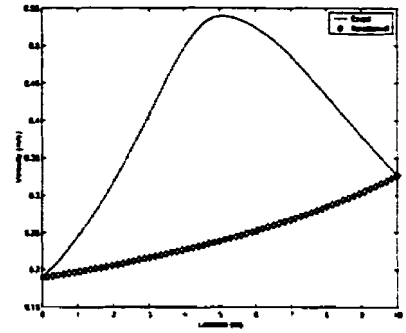

Figure 4.4: Target velocity and initial guess for subsonic, 10 design variable case.

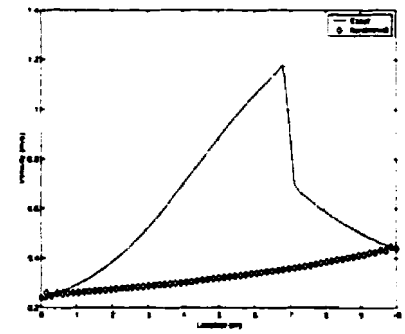

Figure 4.5: Target velocity and initial guess for transonic, 2 design variable case.

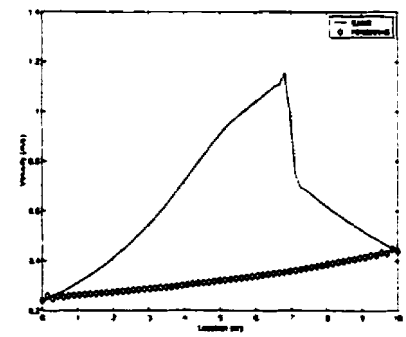

Figure 4.6: Target velocity and initial guess for transonic, 10 design variable case.

4.1 Case 1: Subsonic flow, 2 design variables

The subsonic 2 design variable case is first solved using the halving linesearch algorithm to compute the update parameter α_n in equation 3.2. The acronym *HLA* in the captions refers to the *halving linesearch algorithm*. Plots from various iterations of the optimization are included in Figures 4.7 - 4.12. In Figure 4.10 it is evident that the flow distribution does not approach the optimum solution as efficiently as possible, since it passes the optimum solution. The convergence history for the halving linesearch approach is in included in Figure 4.19.

The same case is then solved using the backtracking linesearch method. The acronym *BLA* in the captions refers to the *backtracking linesearch algorithm*. Plots from various iterations of the optimization are included in Figures 4.13 - 4.18. In contrast to Figure 4.10, Figure 4.16 shows that the flow distribution does not overshoot the optimum solution target. The convergence history for the backtracking linesearch algorithm is included in Figure 4.19. While both approaches took the same number of iterations to converge the fully-coupled governing equations, the backtracking linesearch method performed better than the halving linesearch method.

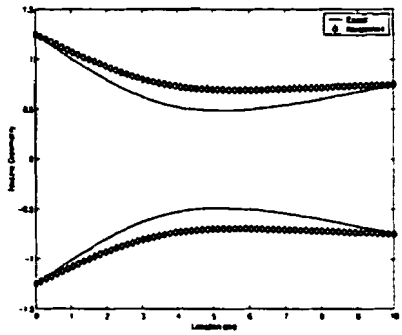

Figure 4.7: Nozzle shape, 1 iteration, subsonic case, 2 design variables, HLA.

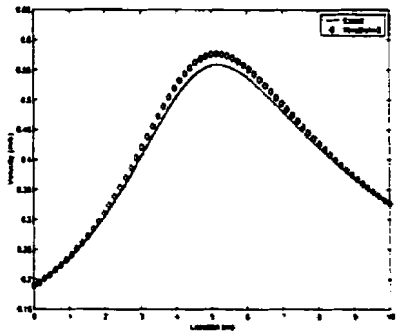

Figure 4.10: Velocity profile, 3 iterations, subsonic case, 2 design variables, HLA.

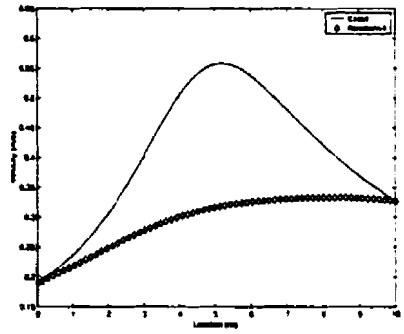

Figure 4.8: Velocity profile, 1 iteration, subsonic case, 2 design variables, HLA.

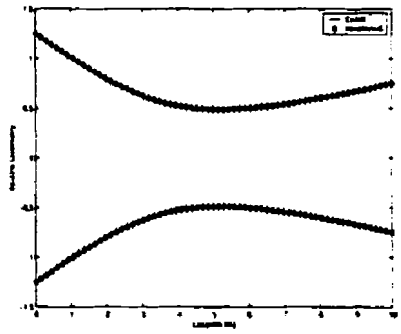

Figure 4.11: Final nozzle shape, subsonic case, 2 design variables, HLA.

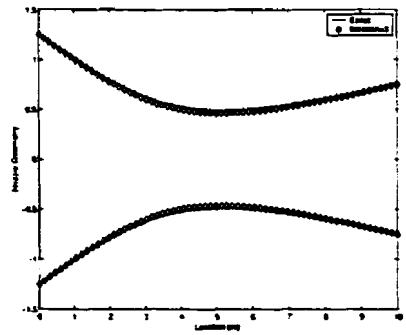

Figure 4.9: Nozzle shape, 3 iterations, subsonic case, 2 design variables, HLA.

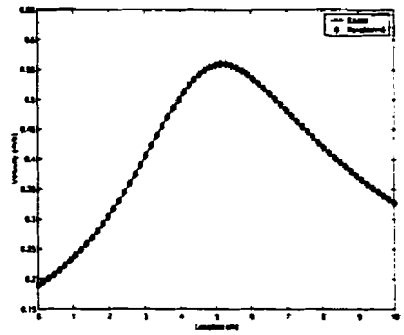

Figure 4.12: Final velocity profile, subsonic case, 2 design variables, HLA.

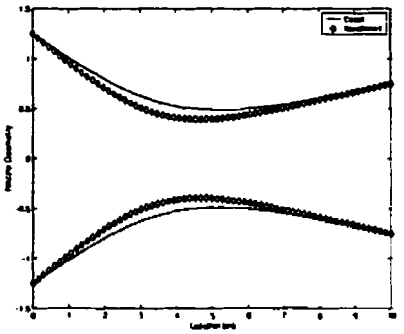

Figure 4.13: Nozzle shape, 1 iteration, subsonic case, 2 design variables, BLA.

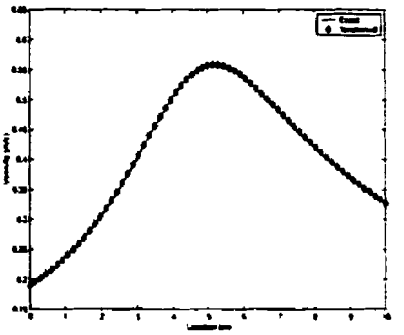

Figure 4.16: Velocity profile, 3 iterations, subsonic case, 2 design variables, BLA.

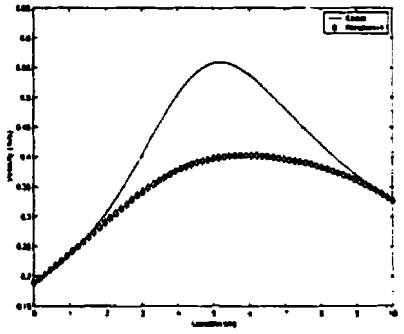

Figure 4.14: Velocity profile, 1 iteration, subsonic case, 2 design variables, BLA.

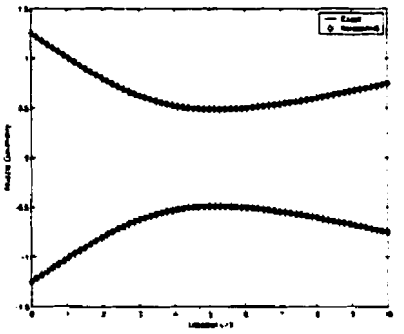

Figure 4.17: Final nozzle shape, subsonic case, 2 design variables, BLA.

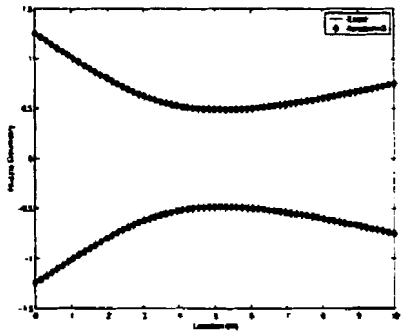

Figure 4.15: Nozzle shape, 3 iterations, subsonic case, 2 design variables, BLA.

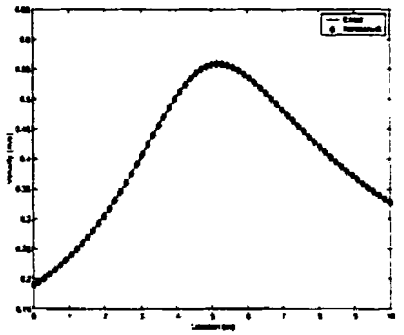

Figure 4.18: Final velocity profile, subsonic case, 2 design variables, BLA.

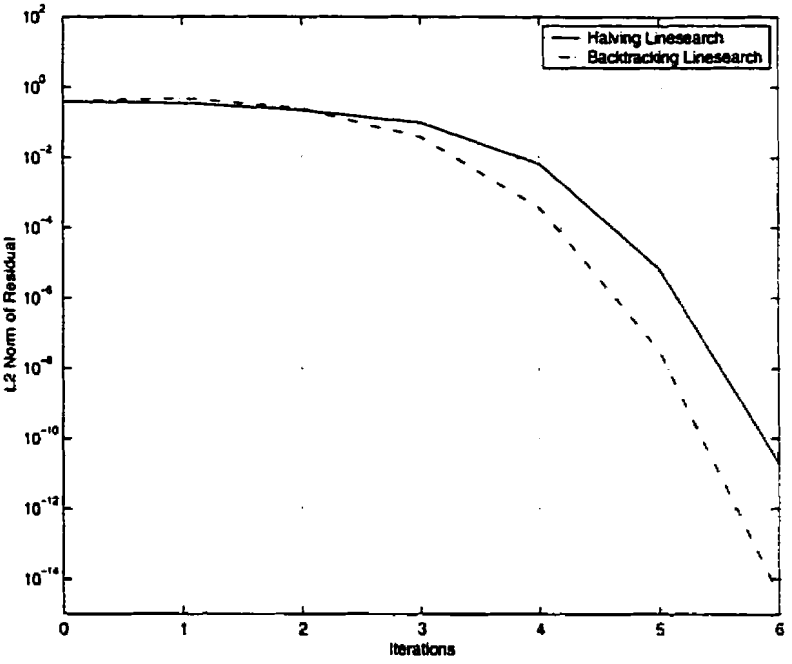

Figure 4.19: Convergence history, subsonic case, 2 design variables, HLA and BLA.

4.2 Case 2: Subsonic flow, 10 design variables

The subsonic 10 design variable case is also first solved using the halving linesearch algorithm. Plots from various iterations of the optimization are included in Figures 4.20 - 4.25. The convergence history for the halving linesearch applied to this case included in Figure 4.32.

The case is also solved using the backtracking linesearch algorithm. Plots from the same intermediate iterations of the optimization as well as the final iteration are included in Figures 4.26 - 4.31. In contrast to Figure 4.22, Figure 4.28 shows that the backtracking linesearch method approaches the optimum solution much faster than the halving linesearch method. The convergence history for this case, included in Figure 4.32, verifies this fact. The backtracking linesearch requires only 6 iterations, as opposed to 10 for the halving linesearch, to converge the fully-coupled system's equations.

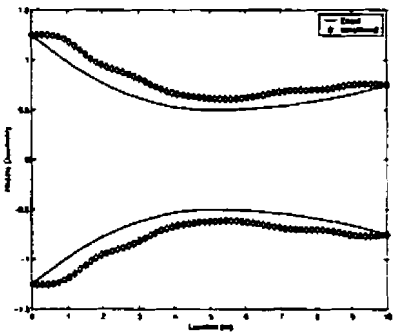

Figure 4.20: Nozzle shape, 2 iterations, subsonic case, 10 design variables, HLA.

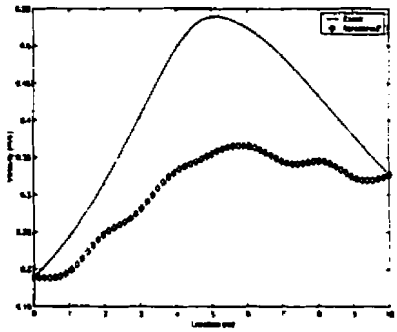

Figure 4.21: Velocity profile, 2 iterations, subsonic case, 10 design variables, HLA.

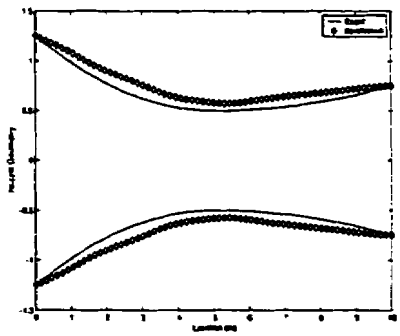

Figure 4.22: Nozzle shape, 4 iterations, subsonic case, 10 design variables, HLA.

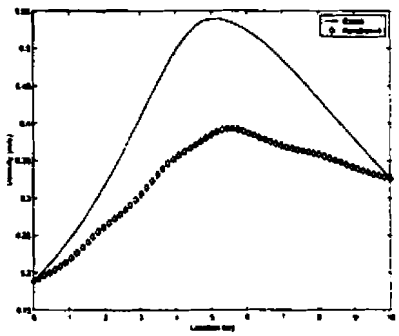

Figure 4.23: Velocity profile, 4 iterations, subsonic case, 10 design variables, HLA.

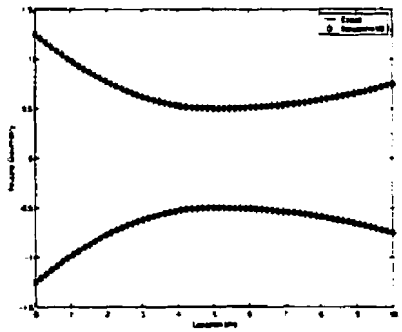

Figure 4.24: Final nozzle shape, subsonic case, 10 design variables, HLA.

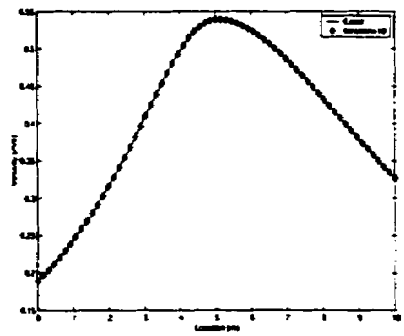

Figure 4.25: Final velocity profile, subsonic case, 10 design variables, HLA.

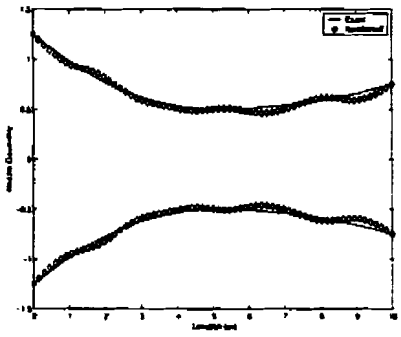

Figure 4.26: Nozzle shape, 2 iterations, subsonic case, 10 design variables, BLA.

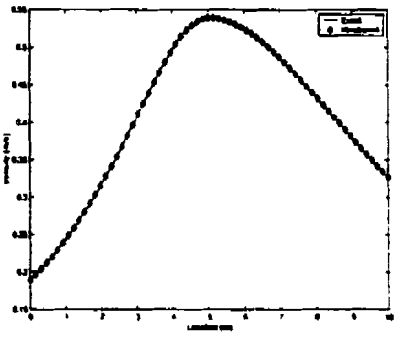

Figure 4.29: Velocity profile, 4 iterations, subsonic case, 10 design variables, BLA.

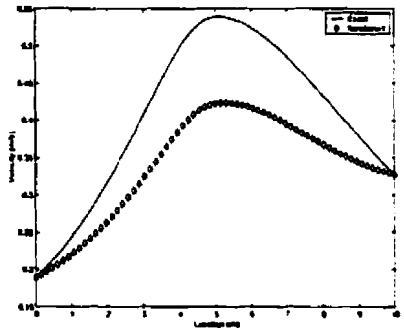

Figure 4.27: Velocity profile, 2 iterations, subsonic case, 10 design variables, BLA.

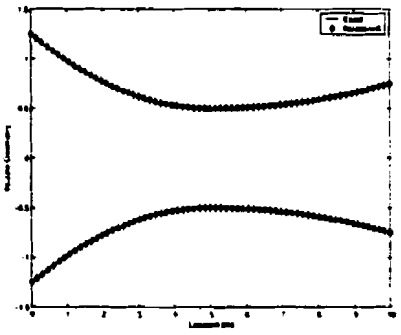

Figure 4.30: Final nozzle shape, subsonic case, 10 design variables, BLA.

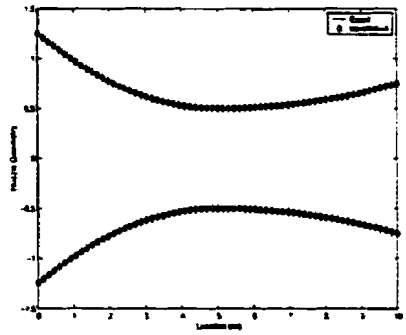

Figure 4.28: Nozzle shape, 4 iterations, subsonic case, 10 design variables, BLA.

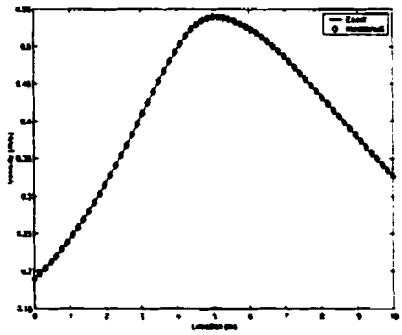

Figure 4.31: Final velocity profile, subsonic case, 10 design variables, BLA.

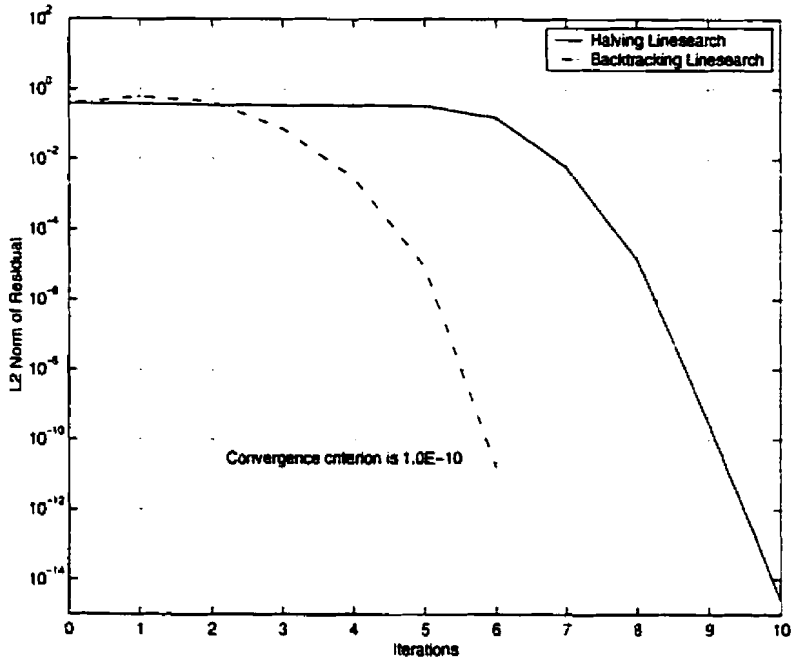

Figure 4.32: Convergence history, subsonic case, 10 design variables, HLA and BLA.

4.3 Case 3: Transonic flow, 2 design variables

The transonic 2 design variable case is solved using only the halving linesearch method. Plots from various iterations are included in Figures 4.33 - 4.44. Figures 4.34, 4.36, and 4.38 show how the shock is formed in the early stages of the optimization process. Ensuring the shock is formed and located properly is critical to the efficiency of the optimization process. Since the flow solver is embedded in the nonlinear system of equations, the optimization requires additional iterations to form the shock, just as a conventional flow solver would. By the 14th iteration, the flow distribution appears to have reached the target as indicated in Figure 4.42. The convergence history for this case is included in Figure 4.45. The case required 18 iterations as opposed to 6 for the 2 design variable case with the halving linesearch algorithm.

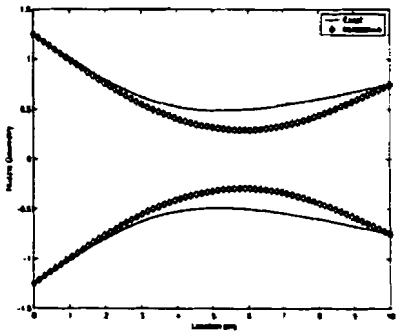

Figure 4.33: Nozzle shape, 4 iterations, transonic case, 2 design variables, HLA.

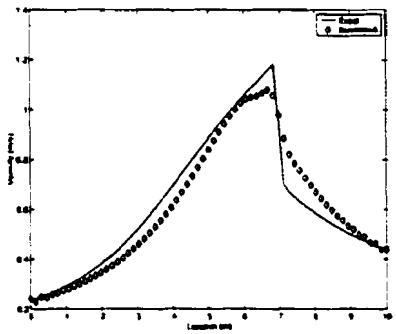

Figure 4.36: Velocity profile, 5 iterations, transonic case, 2 design variables, HLA.

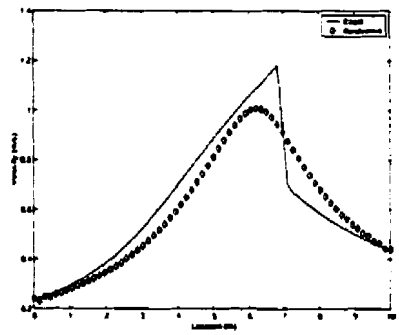

Figure 4.34: Velocity profile, 4 iterations, transonic case, 2 design variables, HLA.

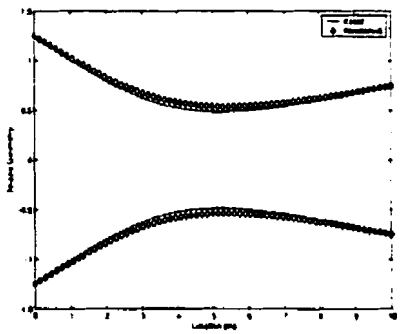

Figure 4.37: Nozzle shape, 6 iterations, transonic case, 2 design variables, HLA.

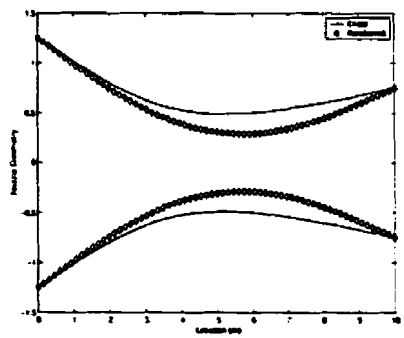

Figure 4.35: Nozzle shape, 5 iterations, transonic case, 2 design variables, HLA.

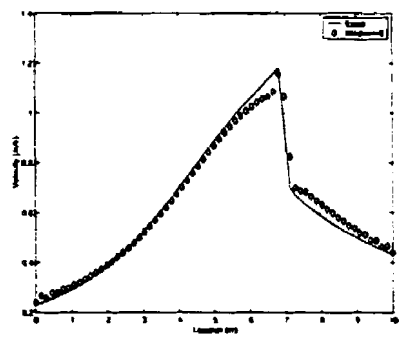

Figure 4.38: Velocity profile, 6 iterations, transonic case, 2 design variables, HLA.

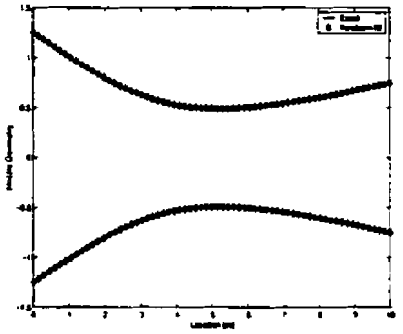

Figure 4.39: Nozzle shape, 10 iterations, transonic case, 2 design variables, HLA.

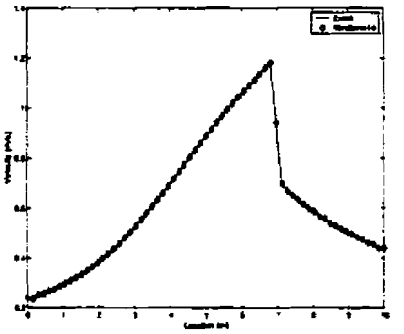

Figure 4.42: Velocity profile, 14 iterations, transonic case, 2 design variables, HLA.

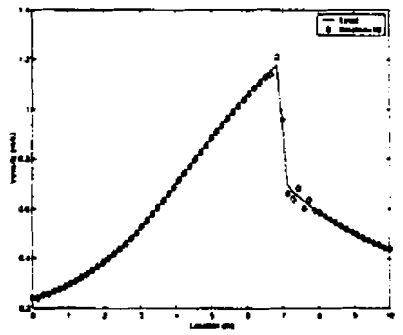

Figure 4.40: Velocity profile, 10 iterations, transonic case, 2 design variables, HLA.

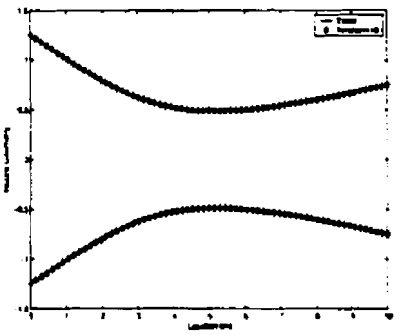

Figure 4.43: Final nozzle shape, transonic case, 2 design variables, HLA.

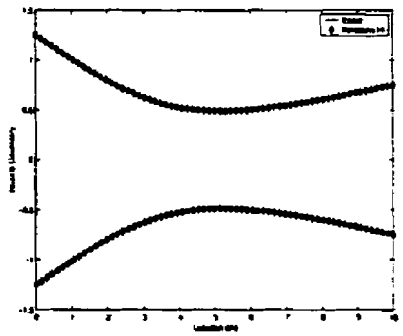

Figure 4.41: Nozzle shape, 14 iterations, transonic case, 2 design variables, HLA.

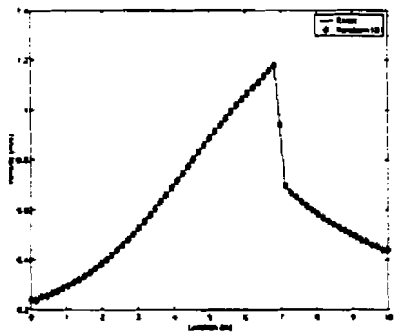

Figure 4.44: Final velocity profile, transonic case, 2 design variables, HLA.

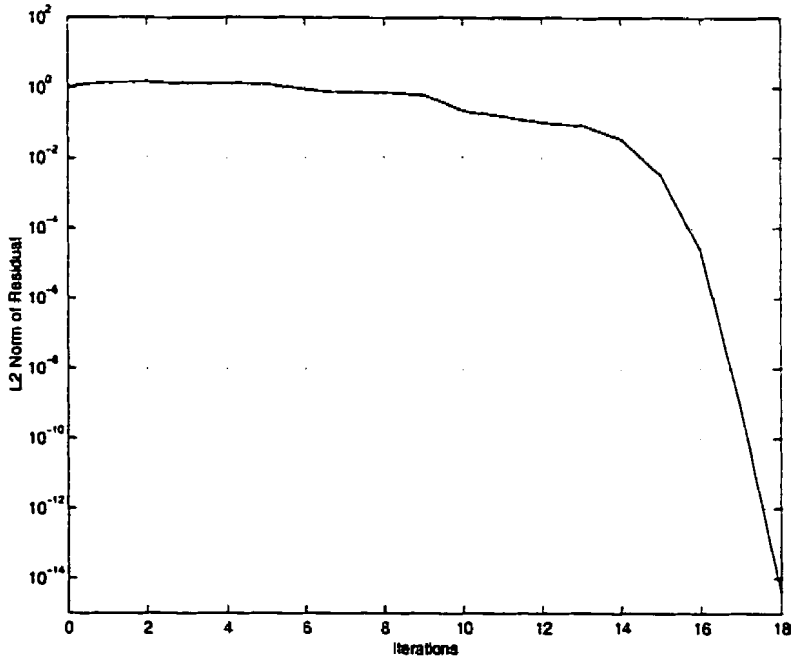

Figure 4.45: Convergence history, transonic case, 2 design variables. HLA.

4.4 Case 4: Transonic flow, 10 design variables

The transonic 10 design variable case is solved using only the halving linesearch method. Plots from various iterations are included in Figures 4.46 - 4.57. Similar to the 2 design variable case, additional iterations are required to form the shock, as indicated in Figures 4.47 and 4.49. By the 14th iteration, to the reader's visual perception, the solution has roughly the optimum. The case converged in 17 iterations as indicated by the convergence plot in Figure 4.58. The case took roughly the same amount of iterations as the 2 design variable transonic case. Clearly the bottleneck in this fully-coupled algorithm, is converging the flow solve efficiently and accurately. This fact is validated because, when compared to the subsonic case with the halving linesearch, the transonic case required 7 additional iterations.

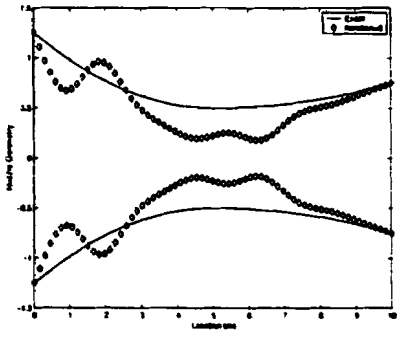

Figure 4.46: Nozzle shape, 8 iterations, transonic case, 10 design variables, HLA.

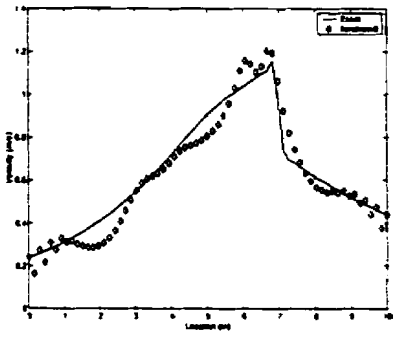

Figure 4.49: Velocity profile, 9 iterations, transonic case, 10 design variables, HLA.

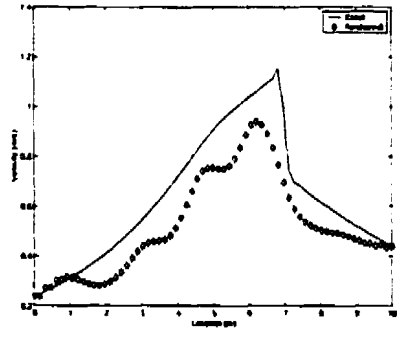

Figure 4.47: Velocity profile, 8 iterations, transonic case, 10 design variables, HLA.

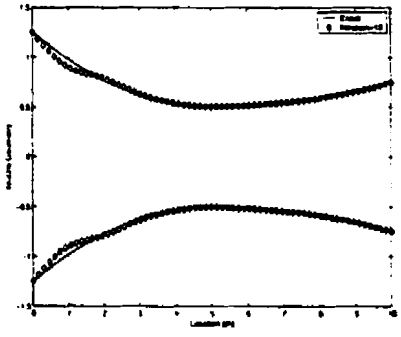

Figure 4.50: Nozzle shape, 12 iterations, transonic case, 10 design variables, HLA.

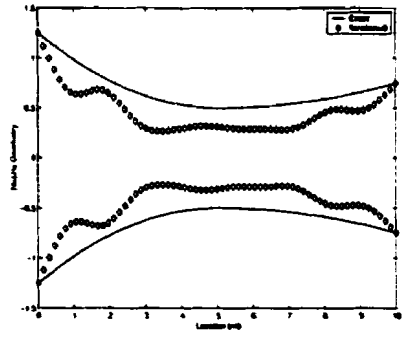

Figure 4.48: Nozzle shape, 9 iterations, transonic case, 10 design variables, HLA.

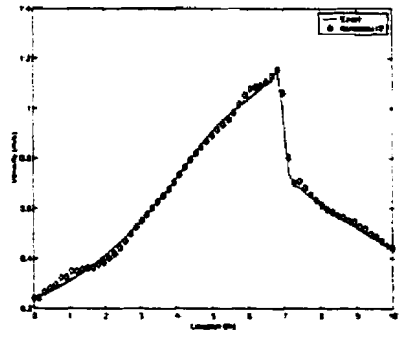

Figure 4.51: Velocity profile, 12 iterations, transonic case, 10 design variables, HLA.

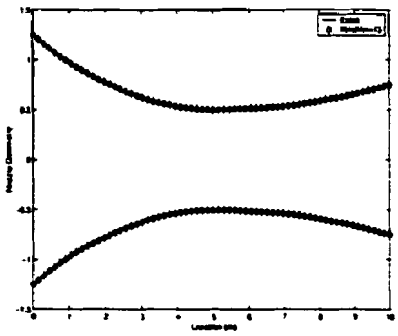

Figure 4.52: Nozzle shape, 13 iterations, transonic case, 10 design variables, HLA.

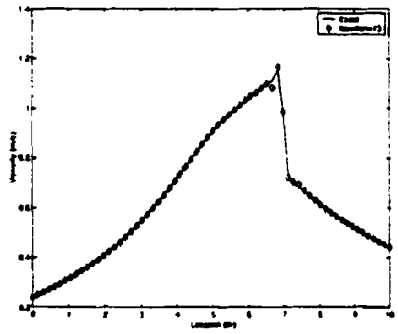

Figure 4.53: Velocity profile, 13 iterations, transonic case, 10 design variables, HLA.

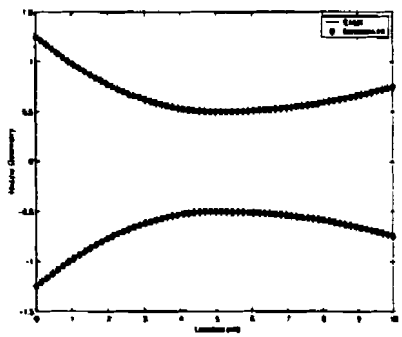

Figure 4.54: Nozzle shape, 14 iterations, transonic case, 10 design variables, HLA.

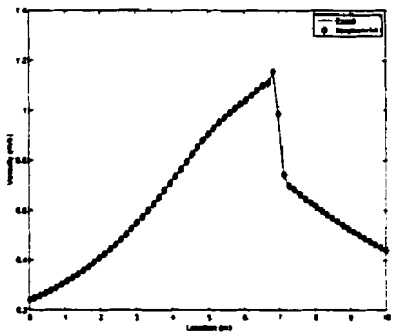

Figure 4.55: Velocity profile, 14 iterations, transonic case, 10 design variables, HLA.

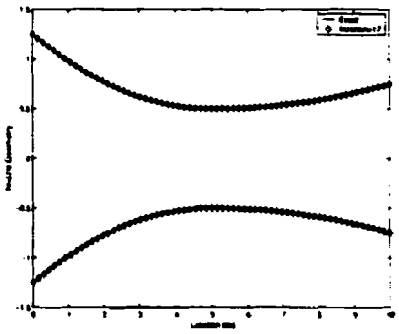

Figure 4.56: Final nozzle shape, transonic case, 10 design variables, HLA.

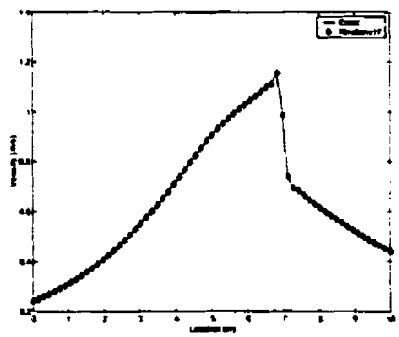

Figure 4.57: Final velocity profile, transonic case, 10 design variables, HLA.

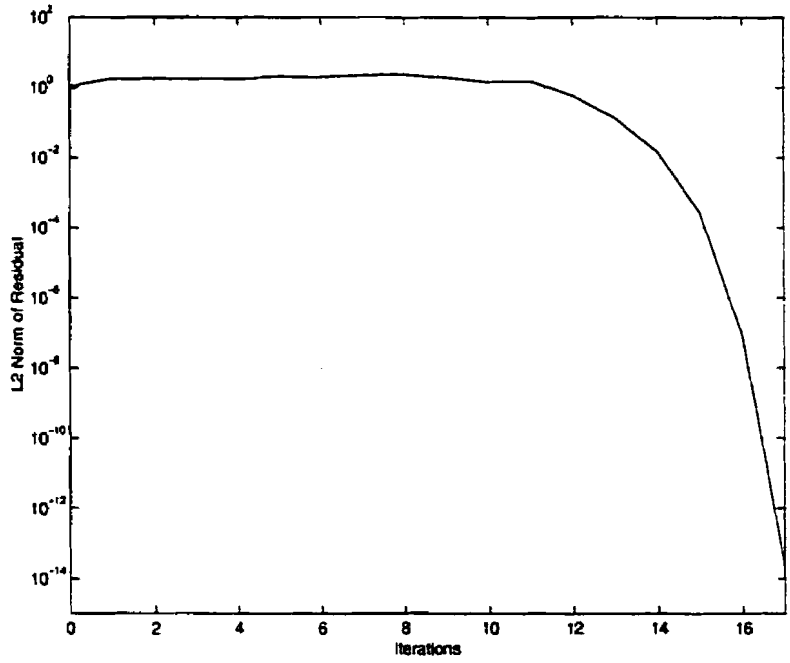

Figure 4.58: Convergence history, transonic case, 10 design variables, HLA.

Chapter 5

Inviscid 2D Airfoil Design Results

We solve the inverse design problems for inviscid airfoil optimization listed in Table 5.1. Cases 1 and 2 are designed to test whether the optimizer is working properly. The target shape design is the NACA0012 airfoil. In the first case, 2 of the 15 B-spline control points representing the NACA0012 airfoil are perturbed and are defined as design variables in the optimization. The optimizer then returns the shape to the original NACA0012 airfoil, by changing the design variables back to their original values. The second case involves 10 design variables instead of 2, but a similar process is followed. In cases 3 and 4, the target airfoil geometry is the RAE2822 airfoil, and the initial airfoil geometry is a modified NACA0012 airfoil. The modifications made on the airfoil are at the leading and trailing edges, due to considerations involving grid generation. This is discussed in detail in Section 5.3.

Case	Initial Design	Target Design	Flow	M_∞	α	Design Variables
1	Perturbed NACA 0012	NACA 0012	Subsonic	0.50	2°	2
2	Perturbed NACA 0012	NACA 0012	Subsonic	0.50	2°	10
3	Modified NACA 0012	RAE 2822	Subsonic	0.50	2°	8
4	Modified NACA 0012	RAE 2822	Transonic	0.74	2°	8

Table 5.1: 2D Airfoil Design Test Cases.

All cases are run on a high-performance engineering workstation. The halving linesearch algorithm is used for the first three subsonic cases, since they are run on very coarse grids and a comparison of the two linesearch methods is not very dramatic in terms

of CPU time for the optimization. However, for the fourth case, which is more difficult since it is transonic and is run on a finer grid, both the halving and backtracking linesearch algorithms are compared. Every initial flow distribution is the converged flow solution for the initial shape guess that is used for any given case. Thus the convergence plots for all of the flow equations presented begin with a residual of machine zero.

For the shape design optimization of airfoils, the norms of the flow equations, adjoint equations and optimality conditions are calculated separately, which differs from the way they were calculated for the whole system in the nozzle shape design optimization. This is done in order to get a better sense of how each of the equations is behaving while the fully-coupled system is converging as a whole. Therefore, three convergence criteria are used for the 2D version of the code. The L2-norm of the flow equations must be below 1.0×10^{-12} and the L2-norms of the adjoint equations and optimality conditions must be below 1.0×10^{-10}. It is found that the optimality and adjoint equations can converge below this tolerance. All convergence plots show the separate convergence of each of the governing equations which are part of the fully-coupled system. For case 4 the optimizer was forced to go many more iterations past the convergence criterion, and the optimum solution did not diverge.

To construct the grids, the *hygrid* grid generator is used. This grid generator forms hyperbolic C-meshes about airfoils and is adequate for the optimization cases that are considered. Details about the grids used are given in Table 5.2. Grid 3 is first generated and then restricted to coarser grids 2 and 1 respectively. The parameters for the generation of grid 3 are given in Table 5.3. Grids 3 for the NACA0012 and RAE2822 airfoils are given in Figures 5.1 and 5.2 respectively.

Grid	JMAX	KMAX	JTAIL1	JTAIL2
1	22	8	4	19
2	43	15	7	37
3	85	29	13	73

Table 5.2: Grid Specifications.

In Chapter 3, the flow Jacobian and its transpose are approximated by replacing the finite-difference Jacobian with a linearized equivalent. Furthermore, one of the other finite-difference formed blocks of the Jacobian, θ_{21} in equation (3.5), is replaced by only one

Characteristic	Value(s)
Dimensions:	
JMAX	85
KMAX	29
Cluster Points:	
Upper LE	0.01
Upper TE	0.001
Lower TE	0.01
Number of Grid Lines:	
Upper	30
Lower	30
Wake Cut	13
Chords to Farfield	8
Offwall Spacing	2.0×10^{-3}

Table 5.3: Grid Generation Parameters for *hygrid*.

of its two components, as shown in Appendix C. The surviving term is the Hessian of the objective function and its structure is shown in Figure 5.3 for Grid 1. The remaining terms in the Jacobian that are formed using finite differences are currently one of the two remaining bottlenecks in making the fully-coupled algorithm more efficient. The other difficulty lies with solving the various smaller systems as described in Appendix A. Currently an LU decomposition is used and the resulting lower and upper triangular matrices are used to perform a number of backsolves equal to twice the number of design variables plus two. Future improvements to the optimizer are discussed in the final chapter.

No cases are presented here for the fully-coupled Jacobian with the finite-difference formulation, since the block inversion algorithm and the previously-mentioned approximations provide much faster and equally accurate results. However, plots of the Jacobian, formed by finite differences, for the first and second iteration for case 3 on grid 1 are shown in Figures 5.4 and 5.5. Initially, the θ_{21} block term of the Jacobian is exactly the same as the Hessian approximation, since the adjoint variables are set to zero. However, after one iteration, the adjoint variables are not zero and extra elements can be seen in the block term. These additional terms are generally small when compared to the Hessian elements, making the matrix dominant along the block diagonal and justifying the Hessian approximation.

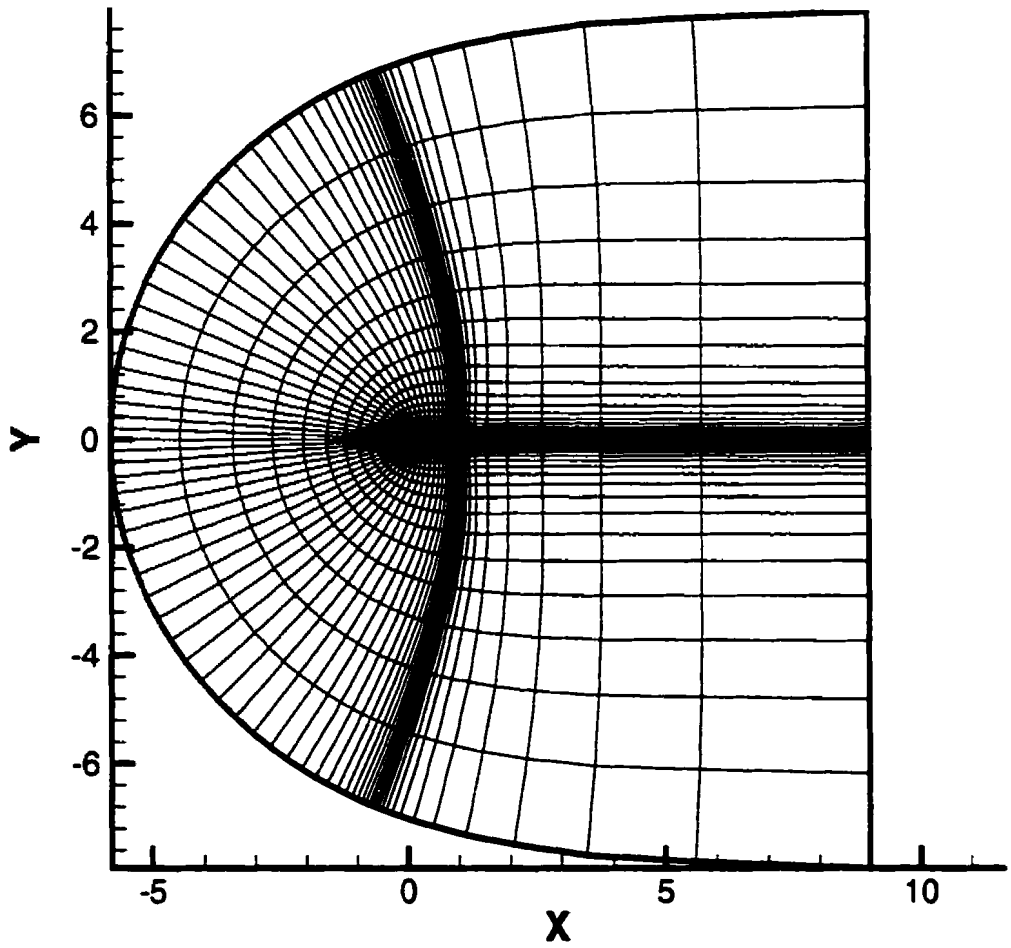

Figure 5.1: Grid 3 for the NACA0012 airfoil.

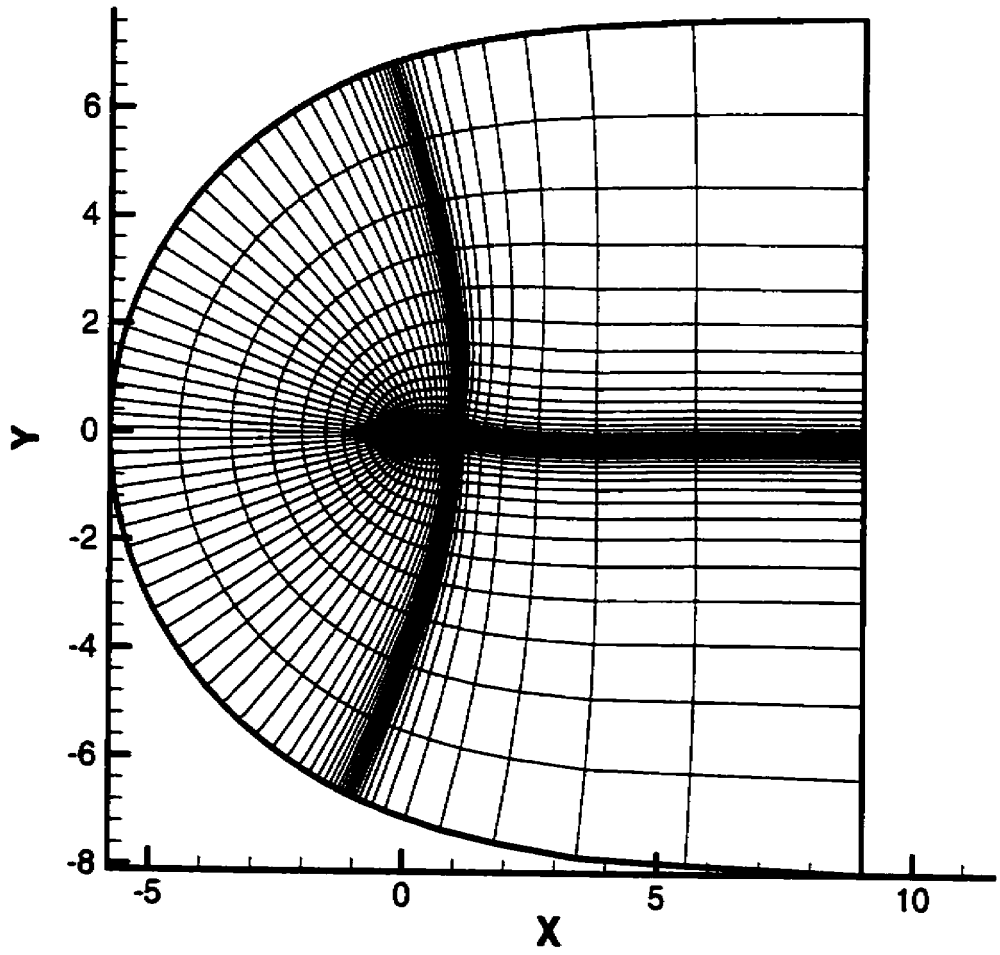

Figure 5.2: Grid 3 for the RAE2822 airfoil.

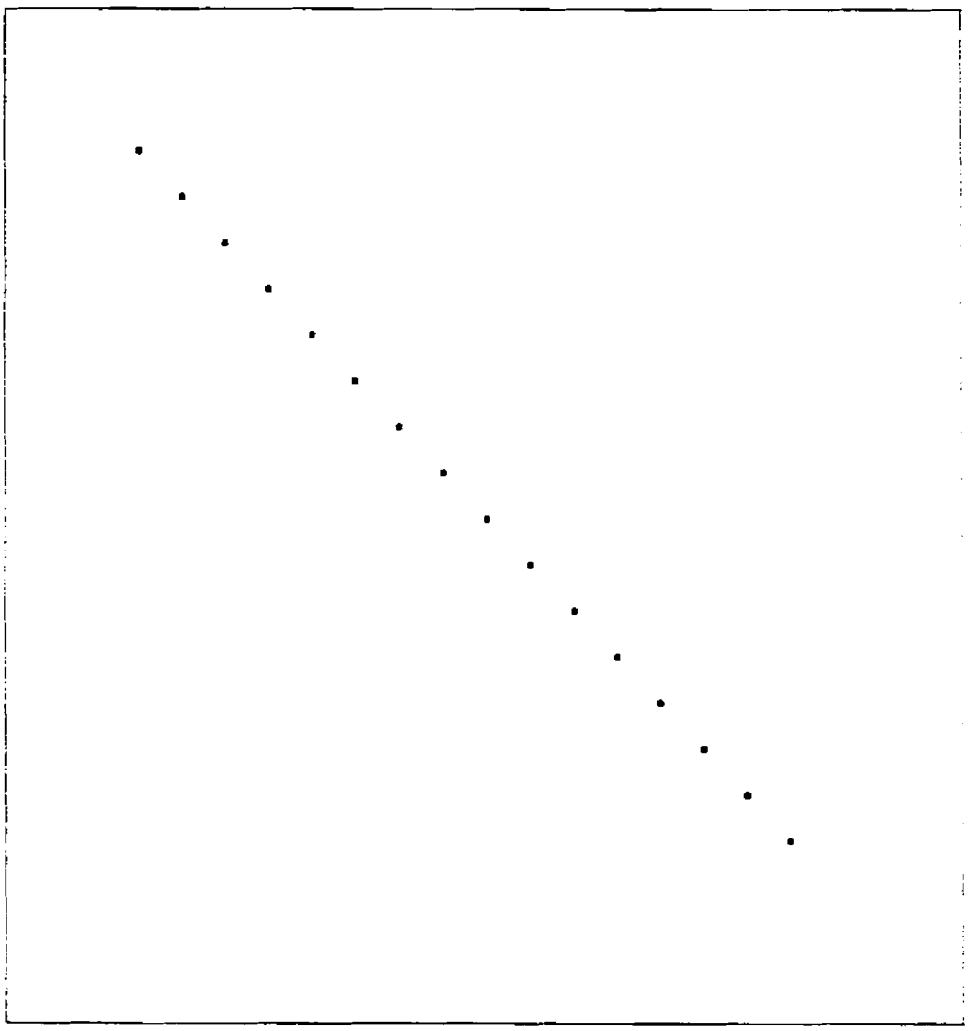

Figure 5.3: Structure of the Hessian of the objective function.

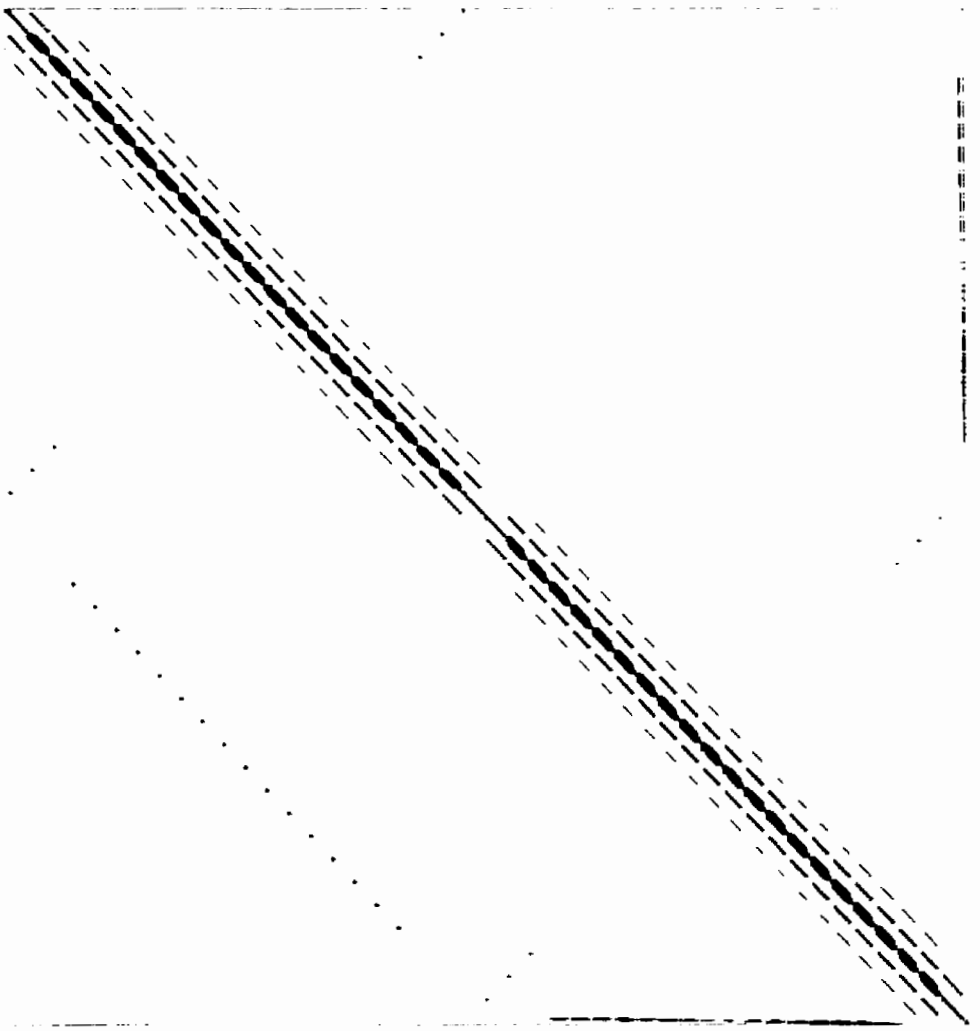

Figure 5.4: Structure of the fully coupled Jacobian on first iteration.

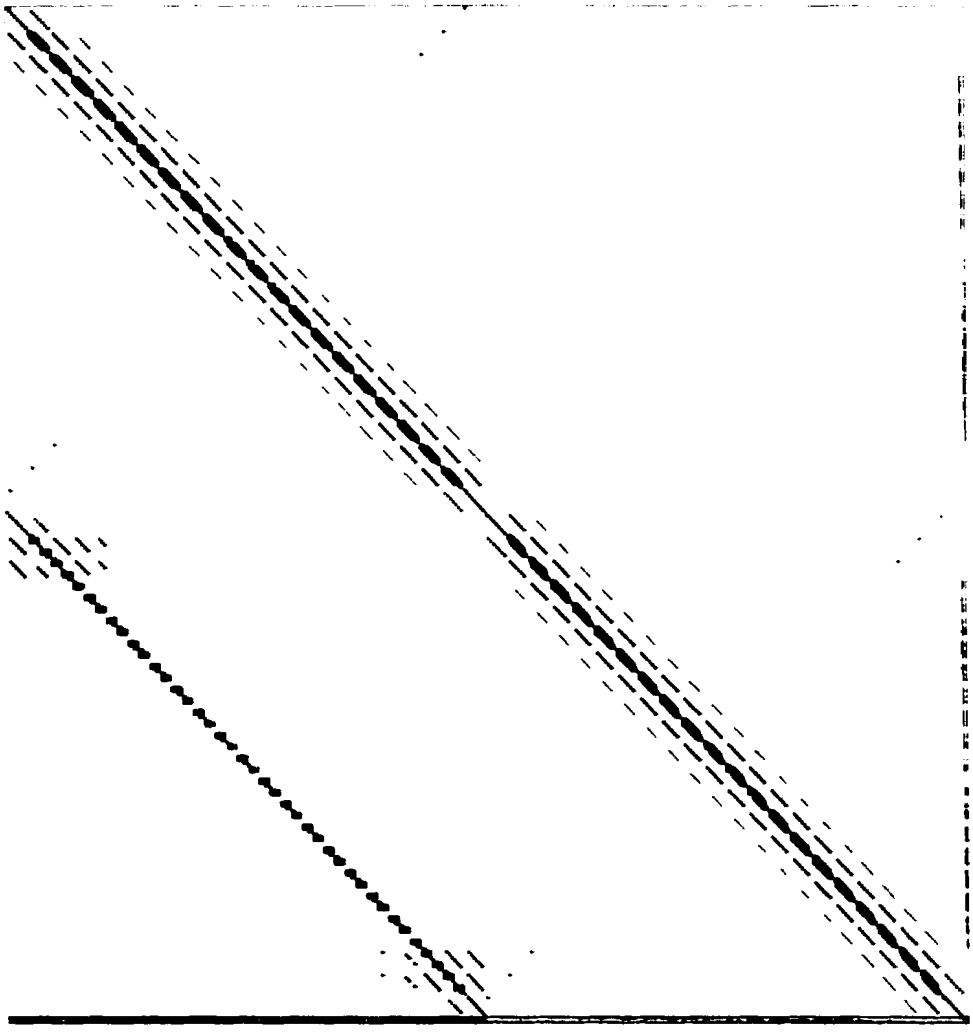

Figure 5.5: Structure of the fully coupled Jacobian on second iteration.

5.1 Case 1: Subsonic flow, 2 design variables

Case 1 is subsonic and has 2 design variables. The freestream Mach number is 0.5 and the angle of attack of the airfoil is fixed at 2°. The target shape is the NACA0012 airfoil, and the initial geometry is the NACA0012 airfoil, with 2 of its 15 control points perturbed and defined to be the design variables. The perturbations are shown in Table 5.4.

Design Variable	Control Point	Initial Value	Target Value	% Change
1	10	0.125	0.061626222	102.8%
2	11	0.105	0.060966288	72.2%

Table 5.4: Case 1 Initial Design Variables.

Two runs are made for this case. The first run is performed using a 22 × 8 grid and the second run is performed using a 43 × 15 grid. Details about the runs in terms of CPU time are shown in Table 5.5.

Run	Grid	Linesearch	Optimization Time	Iterations	Time / Iteration
1	22 × 8	HLA	13.2 s	18	0.7 s
2	43 × 15	HLA	8 m 55.4 s	57	9.4 s

Table 5.5: Case 1 Runs.

For all plots, the circles indicate the actual airfoil co-ordinates at a given iteration. This may be confusing on coarser grids, since such a small number of nodes is used. Furthermore, in 1D the target geometry is shown, whereas in this chapter the initial geometry is shown. This is because the 2D version is also being developed for general shape design optimization problems, such as drag minimization, where the target shape is not known, as opposed to the diagnostic inverse shape design optimization problems used in this thesis, where the target shape is known.

For the first run, the initial airfoil geometry and pressure distribution are given in Figures 5.6 and 5.7. Plots after various iterations in the optimization are given in Figures 5.8 - 5.15. By only the second iteration, the pressure distribution has reached the target pressure distribution. The flow distribution does not correspond to the airfoil geometry on any given iteration, except the final converged solution. The final airfoil geometry and

pressure distribution are given in Figures 5.16 and 5.17. The convergence plot for this run is given in Figure 5.18. The convergence of the flow equations, the adjoint equations, and the optimality conditions are separately shown on all convergence plots. The flow solve occurs simultaneously with the optimization. The objective function history is shown in Figure 5.19.

For the second run, the initial airfoil geometry and pressure distribution are given in Figures 5.20 and 5.21. Plots after various iterations in the optimization are shown in Figures 5.22 - 5.29. The final airfoil geometry and pressure distribution are shown in Figures 5.30 and 5.31. The convergence plot and objective function history for this run are given in Figures 5.32 and 5.33.

In the fully-coupled algorithm, the goal is to find values for flow, adjoint, and design variables that drive the discretized equations defining the fully-coupled system to zero. Furthermore, the design variables should match the target control points. Table 5.6 shows the error between the optimum design variables and the target control points. The error is defined as

$$\text{error} \equiv \frac{\text{simulation} - \text{target}}{\text{target}}. \tag{5.1}$$

Although the error decreases as the grid becomes finer, it is not zero. Furthermore, for inverse design optimization problems, the adjoint variables' values should be zero for the optimum solution. This is easily seen in the adjoint equations. When they are rearranged to

$$(\frac{\partial R}{\partial Q})^T \Phi = (\frac{\partial J}{\partial Q})^T, \tag{5.2}$$

the right hand side is zero for the optimum solution. Since $(\frac{\partial R}{\partial Q})^T$ is nonzero, Φ must be zero. However, when the L2-norm is taken of the adjoint variables solution vector, the value is of the order of the largest error in the design variables. One possible explanation for these errors could be because the fully-coupled equations do not converge to machine zero.

Design Variable	Target Value	Run 1	Run 1 Error	Run 2	Run 2 Error
1	0.061626222	0.061622697	-5.719968E-5	0.061625807	-6.734146E-6
2	0.060966288	0.060976853	1.732924E-4	0.060965575	-1.169499E-5

Table 5.6: Case 1 Final Design Variables.

Two possible reasons for why the fully-coupled system does not converge to ma-

chine zero are because the linearization of the residual used to form the flow Jacobian in the adjoint equations is not complete and the optimality conditions are formed using finite differences. The only equations that are able to converge close to machine zero are the flow equations, since they are taken from the CYCLONE flow solver, which itself converges to machine zero.

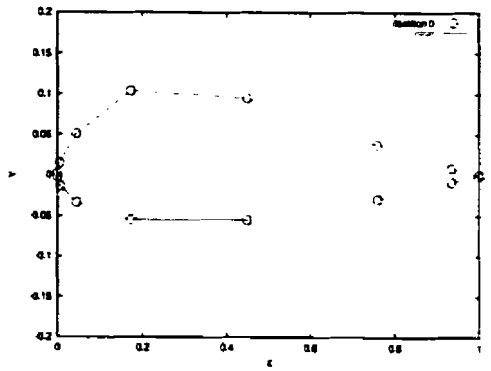

Figure 5.6: Initial airfoil geometry, case 1, run 1, HLA.

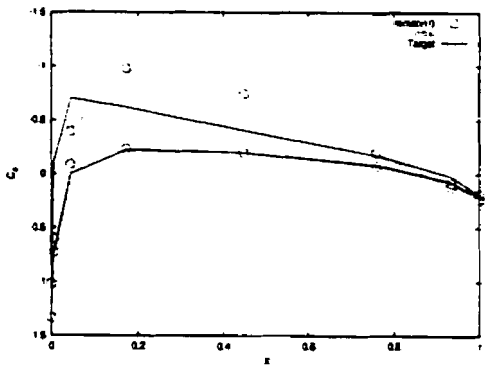

Figure 5.7: Initial pressure distribution, case 1, run 1, HLA.

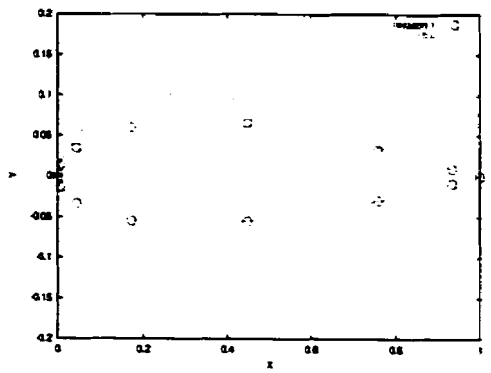

Figure 5.8: Airfoil geometry, 1 iteration, case 1, run 1, HLA.

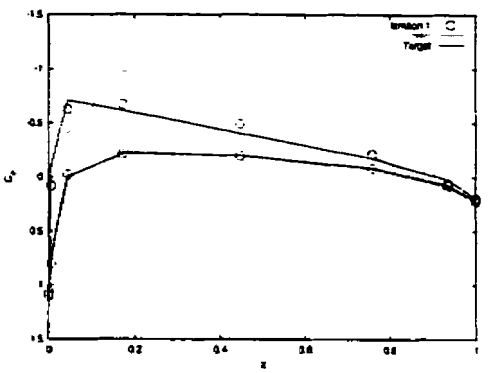

Figure 5.9: Pressure distribution, 1 iteration, case 1, run 1, HLA.

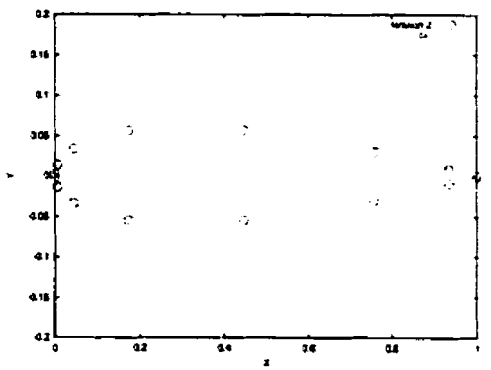

Figure 5.10: Airfoil geometry, 2 iterations, case 1, run 1, HLA.

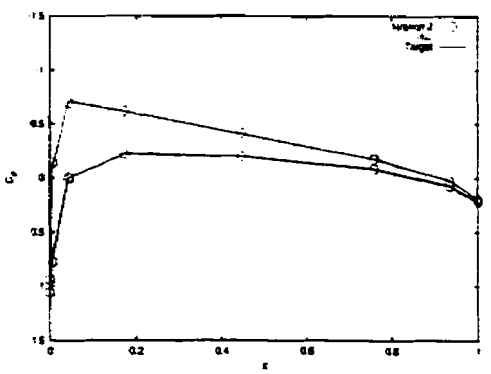

Figure 5.11: Pressure distribution, 2 iterations, case 1, run 1, HLA.

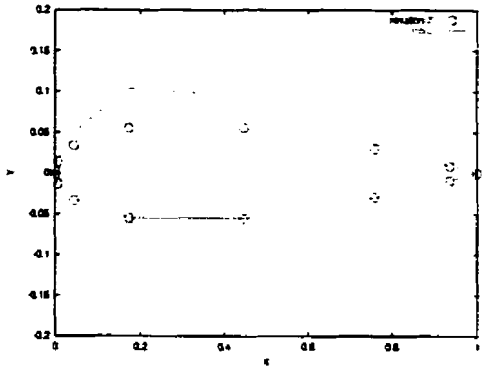

Figure 5.12: Airfoil geometry, 7 iterations, case 1, run 1, HLA.

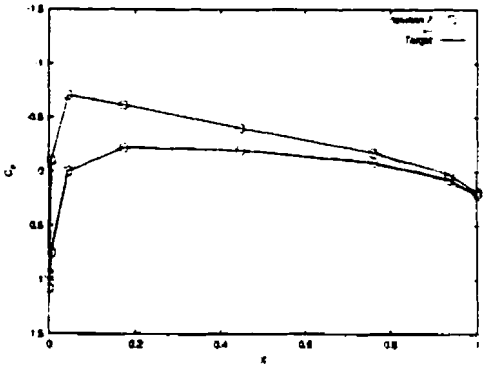

Figure 5.13: Pressure distribution, 7 iterations, case 1, run 1, HLA.

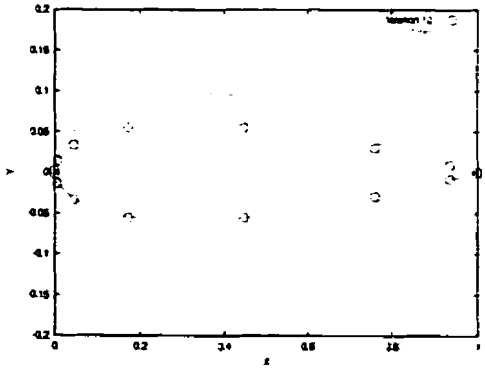

Figure 5.14: Airfoil geometry, 12 iterations, case 1, run 1, HLA.

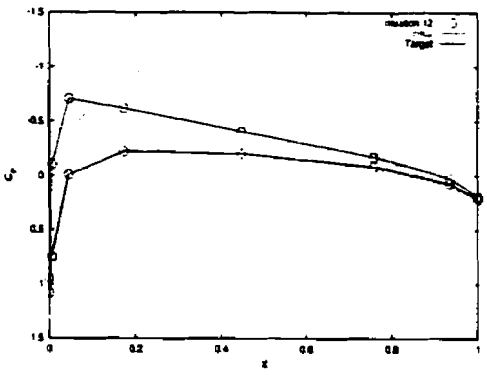

Figure 5.15: Pressure distribution, 12 iterations, case 1, run 1, HLA.

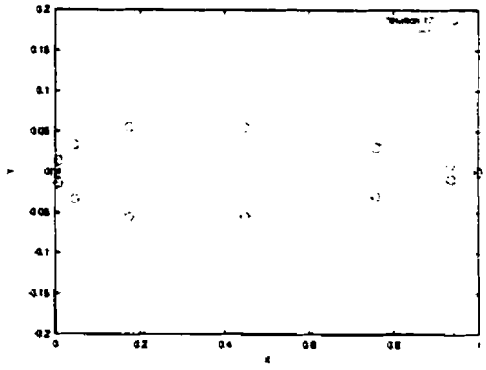

Figure 5.16: Final airfoil geometry, case 1, run 1, HLA.

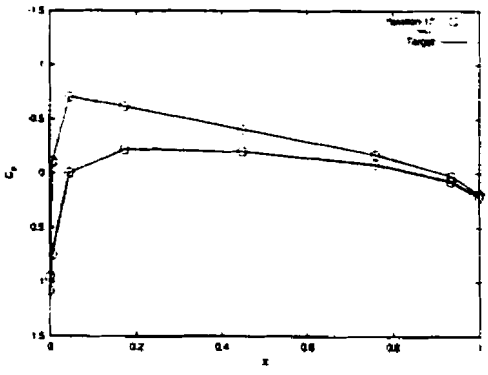

Figure 5.17: Final pressure distribution, case 1, run 1, HLA.

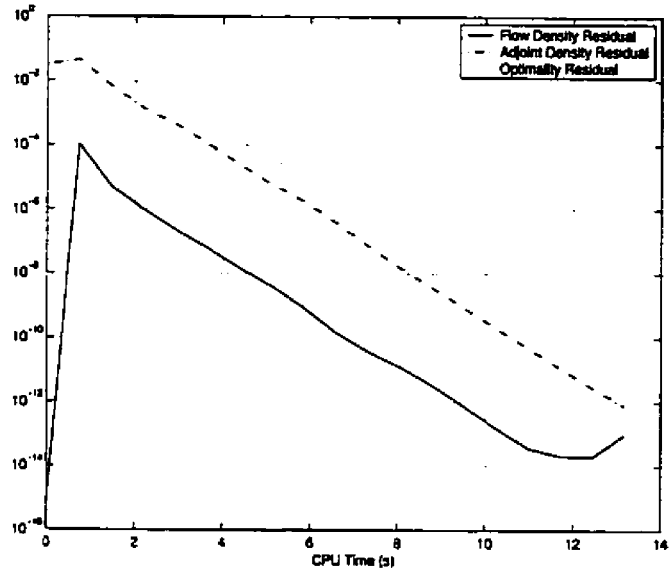

Figure 5.18: Convergence history, case 1, run 1, HLA.

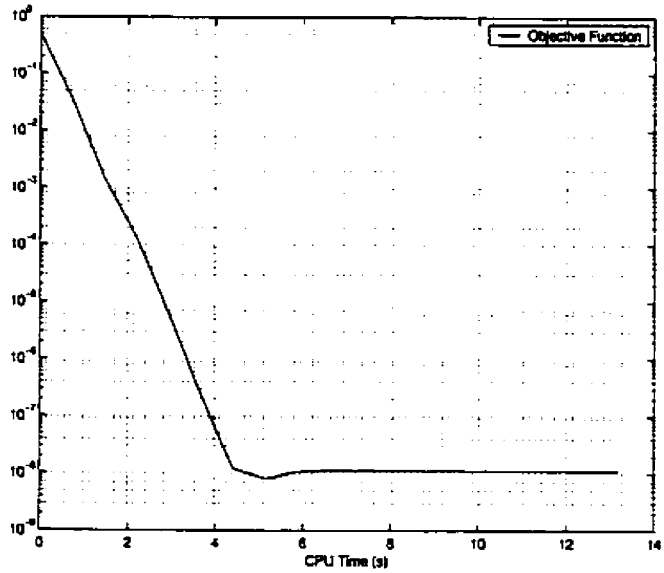

Figure 5.19: Objective function history, case 1, run 1, HLA.

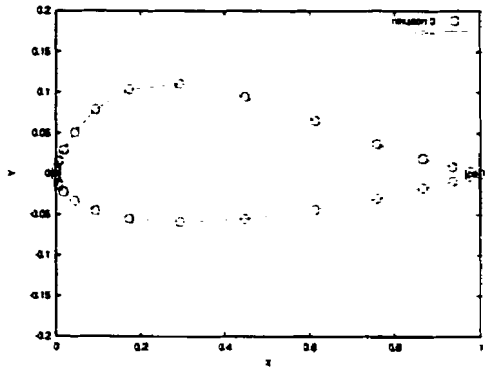

Figure 5.20: Initial airfoil geometry, case 1, run 2, HLA.

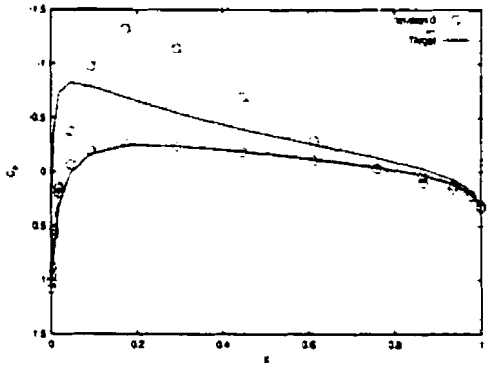

Figure 5.21: Initial pressure distribution, case 1, run 2, HLA.

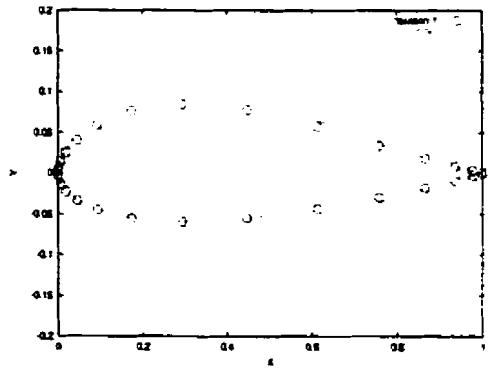

Figure 5.22: Airfoil geometry, 1 iteration, case 1, run 2, HLA.

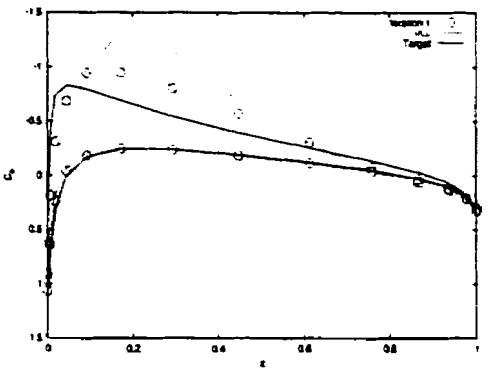

Figure 5.23: Pressure distribution, 1 iteration, case 1, run 2, HLA.

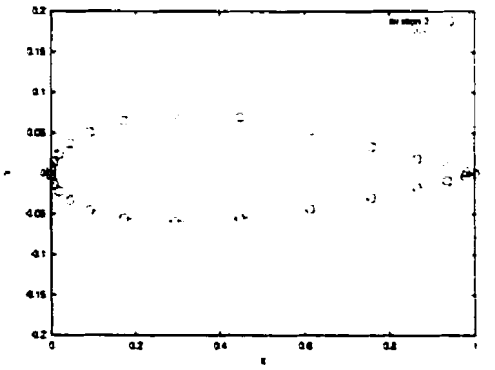

Figure 5.24: Airfoil geometry, 2 iterations, case 1, run 2, HLA.

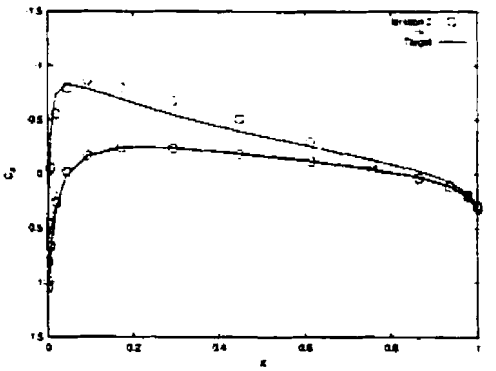

Figure 5.25: Pressure distribution, 2 iterations, case 1, run 2, HLA.

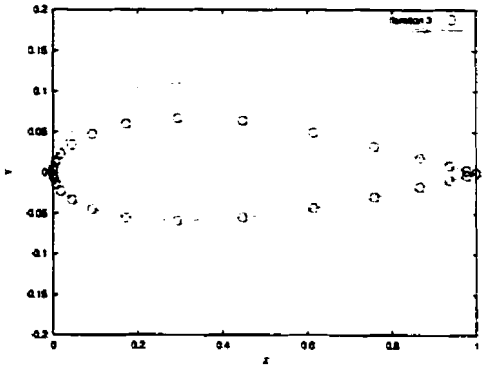

Figure 5.26: Airfoil geometry, 3 iterations, case 1, run 2, HLA.

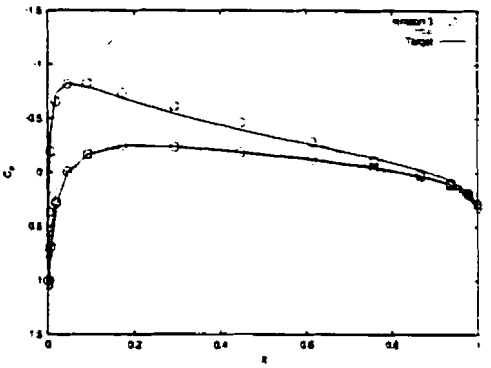

Figure 5.27: Pressure distribution, 3 iterations, case 1, run 2, HLA.

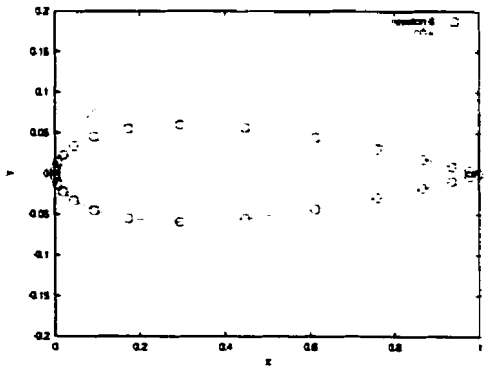

Figure 5.28: Airfoil geometry, 6 iterations, case 1, run 2, HLA.

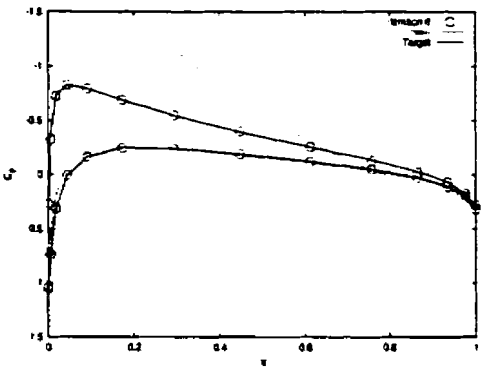

Figure 5.29: Pressure distribution, 6 iterations, case 1, run 2, HLA.

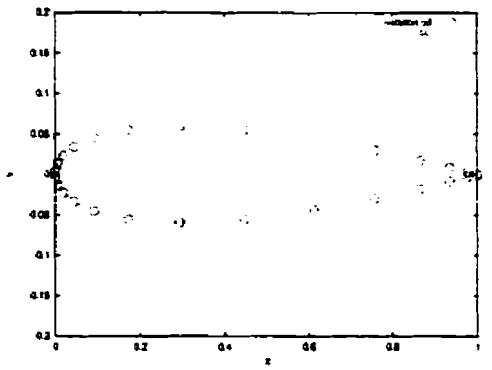

Figure 5.30: Final airfoil geometry, case 1, run 2, HLA.

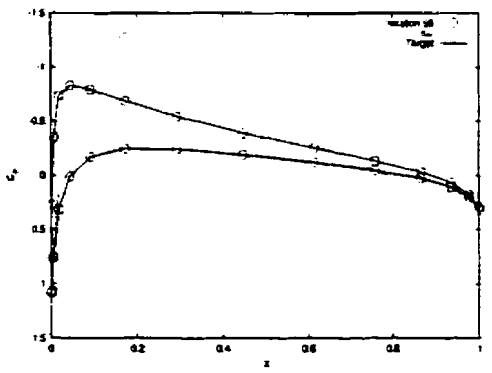

Figure 5.31: Final pressure distribution, case 1, run 2, HLA.

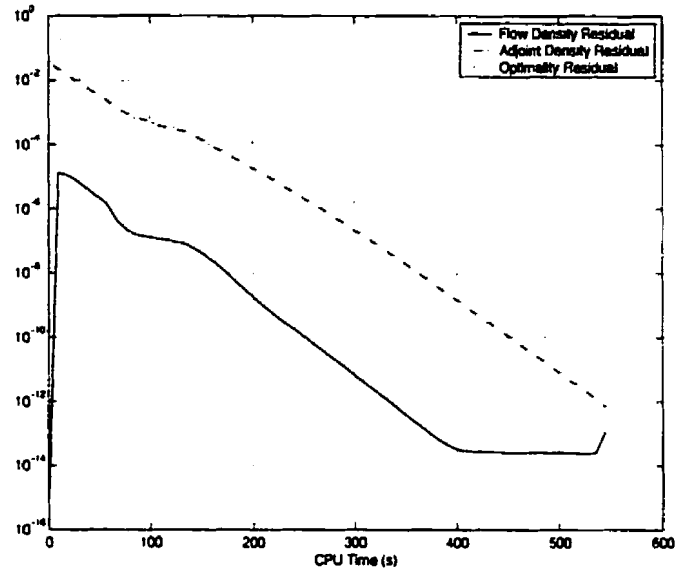

Figure 5.32: Convergence history, case 1, run 2, HLA.

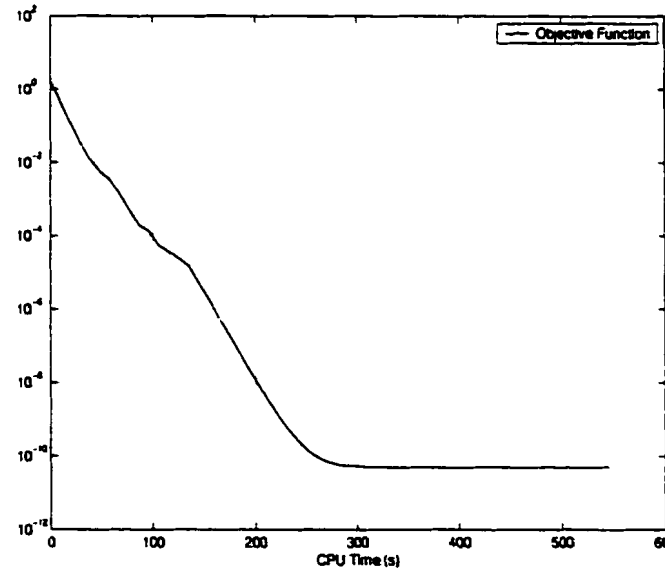

Figure 5.33: Objective function history, case 1, run 2, HLA.

5.2 Case 2: Subsonic flow, 10 design variables

Case 2 is subsonic and has 10 design variables. The Mach number and angle of attack are 0.5 and 2° respectively. This case is very similar to case 1, however, more design variables are used. A pressure distribution is found using the NACA0012 airfoil. The airfoil is then perturbed at 10 of its 15 control points. These control points become the design variables for the inverse design optimization problem. The perturbations made in the NACA0012 airfoil are given in Table 5.7.

Design Variable	Control Point	Initial Value	Target Value	% Change
1	3	-0.014	-0.008237090	-70.0%
2	4	-0.060	-0.029184925	-105.6%
3	5	-0.150	-0.060966288	-146.0%
4	6	-0.150	-0.061626222	-143.4%
5	7	-0.050	-0.026014304	-92.2%
6	9	0.050	0.026014304	92.2%
7	10	0.150	0.061626222	143.4%
8	11	0.150	0.060966288	146.0%
9	12	0.060	0.029184925	105.6%
10	13	0.014	0.008237090	70.0%

Table 5.7: Case 2 Initial Design Variables.

Two runs are made for this case. The first run is performed using a 22 × 8 grid and the second run is performed using a 43 × 15 grid. Details about the runs in terms of CPU time are shown in Table 5.8.

Runs	Grid	Linesearch	Optimization Time	Iterations	Time / Iteration
1	22 × 8	HLA	1 m 16.8 s	23	3.3 s
2	43 × 15	HLA	1 h 43 m 21.6 s	141	44.0 s

Table 5.8: Case 2 Runs.

The intial airfoil geometry and pressure distribution for the first run are given in Figures 5.34 and 5.35. Plots during after various iterations of the optimization are shown in Figures 5.36 - 5.43. The final airfoil geometry and pressure distribution are given in Figures 5.44 and 5.45. The convergence plot and the objective function history are given in Figures 5.46 and 5.47 respectively.

For the second run, the initial airfoil geometry and pressure distribution are given in Figures 5.48 and 5.49. Plots during various iterations of the airfoil geometry and the pressure distribution are shown in Figures 5.50 - 5.57. The final airfoil geometry and pressure distribution are shown in Figures 5.58 and 5.59 respectively. The convergence plot and the objective function history are shown in Figures 5.60 and 5.61.

Table 5.9 shows the optimum design variables in comparison to the control variables that were used to generate the target pressure distribution. The second run, being done on a finer grid, provides a more accurate result. The error is not zero for reasons explained earlier in case 1.

Design Variable	Target Value	Run 1	Run 1 Error	Run 2	Run 2 Error
1	-0.008237090	-0.008227309	-1.187433E-3	-0.008236950	-1.699629E-5
2	-0.029184925	-0.029198074	4.505408E-4	-0.029184967	1.439099E-6
3	-0.060966288	-0.060939812	-4.342728E-4	-0.060966026	-4.297457E-6
4	-0.061626222	-0.061638613	2.010670E-4	-0.061626725	8.162110E-6
5	-0.026014304	-0.026007330	-2.680832E-4	-0.026014228	-2.921470E-6
6	0.026014304	0.026020953	2.555901E-4	0.026014432	4.920370E-6
7	0.061626222	0.061622201	-6.524819E-5	0.061625775	-7.253406E-6
8	0.060966288	0.060953033	-2.174152E-4	0.060964895	-2.284869E-5
9	0.029184925	0.029187136	7.575829E-5	0.029185414	1.675523E-5
10	0.008237090	0.008234085	-3.648133E-4	0.008237173	1.007637E-5

Table 5.9: Case 2 Final Design Variables.

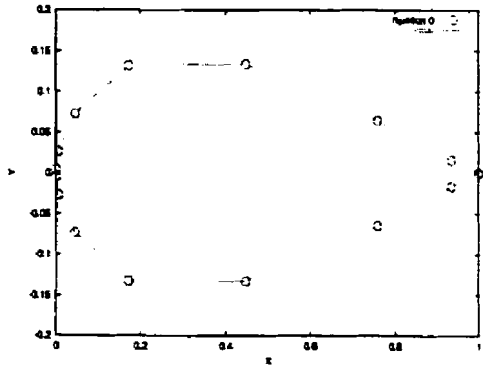

Figure 5.34: Initial airfoil geometry, case 2, run 1, HLA.

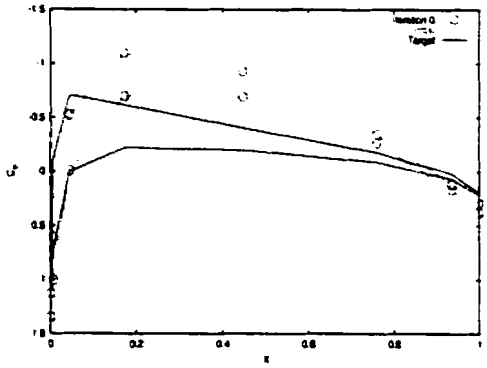

Figure 5.35: Initial pressure distribution, case 2, run 1, HLA.

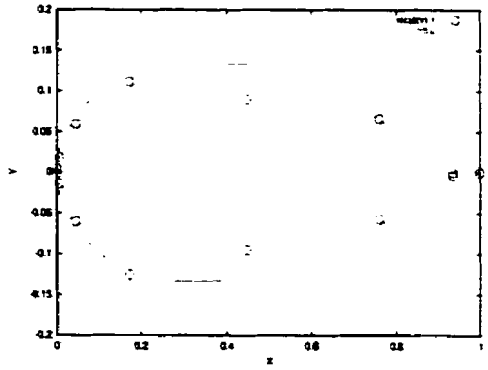

Figure 5.36: Airfoil geometry, 1 iteration, case 2, run 1, HLA.

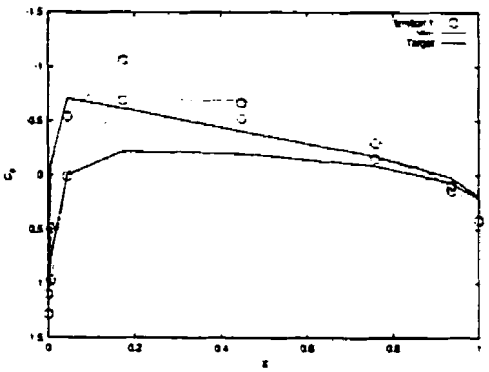

Figure 5.37: Pressure distribution, 1 iteration, case 2, run 1, HLA.

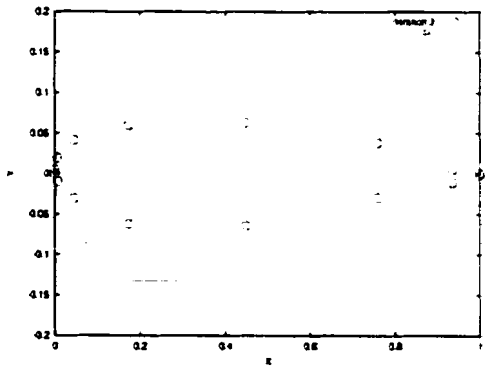

Figure 5.38: Airfoil geometry, 3 iterations, case 2, run 1, HLA.

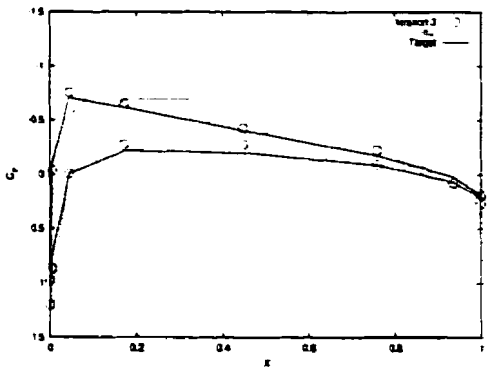

Figure 5.39: Pressure distribution, 3 iterations, case 2, run 1, HLA.

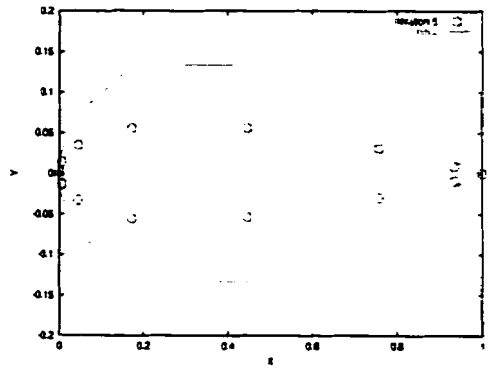

Figure 5.40: Airfoil geometry, 5 iterations, case 2, run 1, HLA.

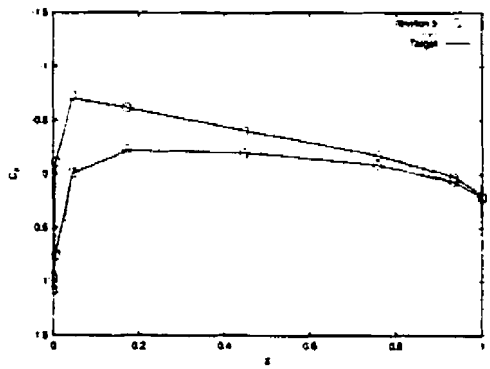

Figure 5.41: Pressure distribution, 5 iterations, case 2, run 1, HLA.

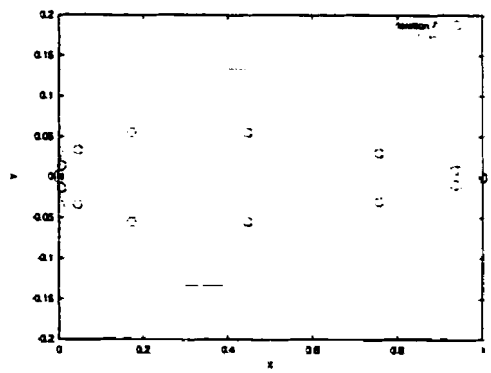

Figure 5.42: Airfoil geometry, 7 iterations, case 2, run 1, HLA.

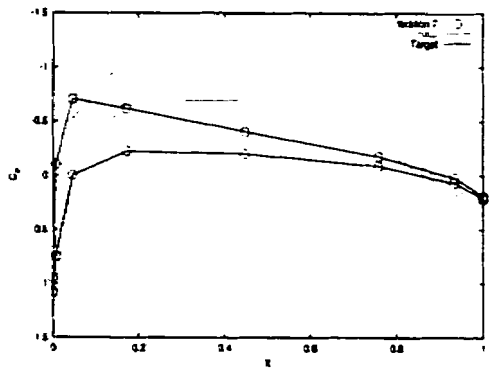

Figure 5.43: Pressure distribution, 7 iterations, case 2, run 1, HLA.

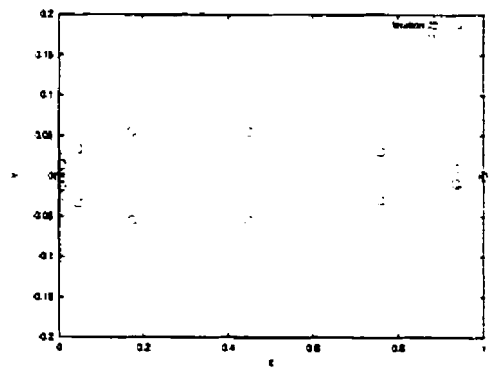

Figure 5.44: Final airfoil geometry, case 2, run 1, HLA.

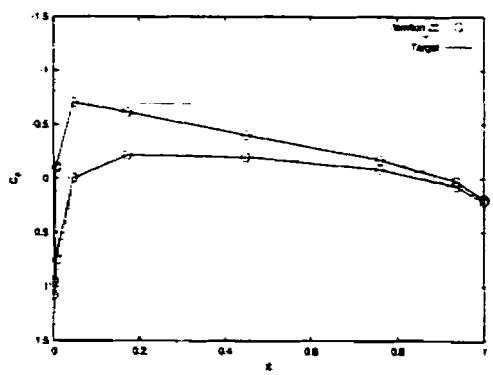

Figure 5.45: Final pressure distribution, case 2, run 1, HLA.

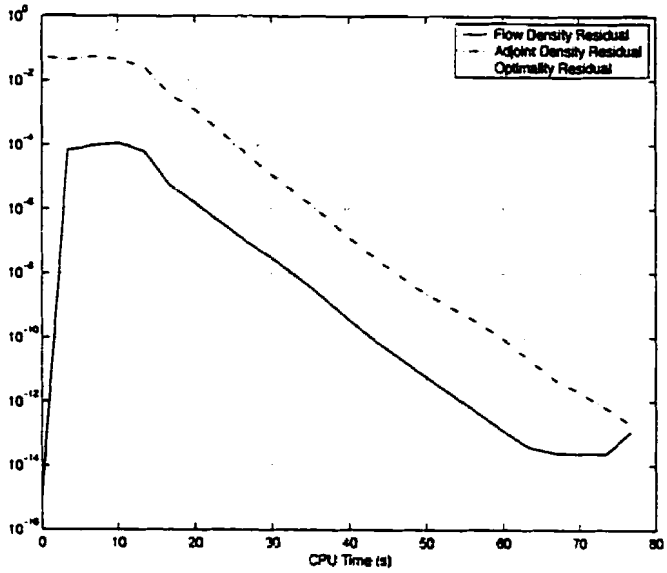

Figure 5.46: Convergence history, case 2, run 1, HLA.

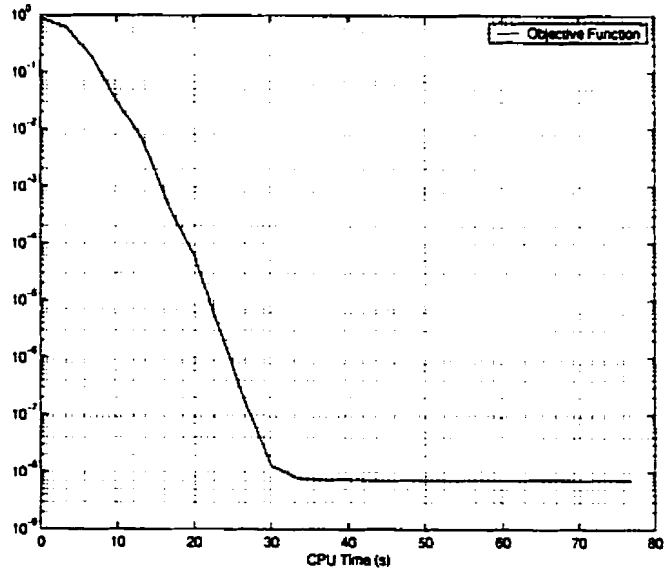

Figure 5.47: Objective function history, case 2, run 1, HLA.

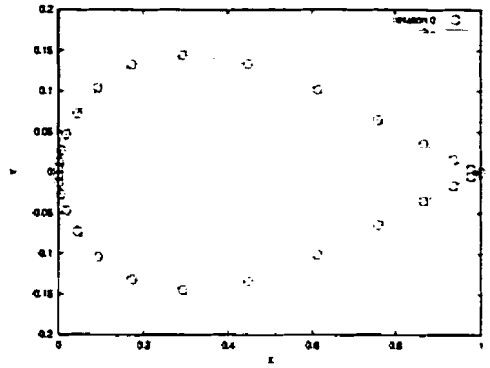

Figure 5.48: Initial airfoil geometry, case 2, run 2, HLA.

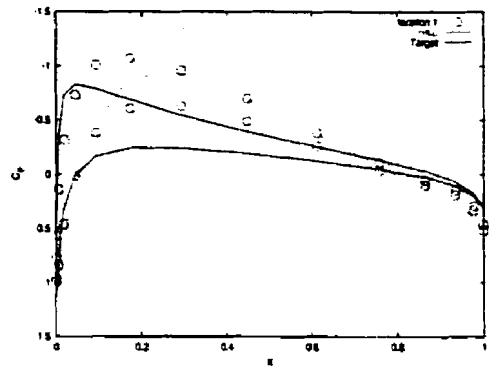

Figure 5.51: Pressure distribution, 1 iteration, case 2, run 2, HLA.

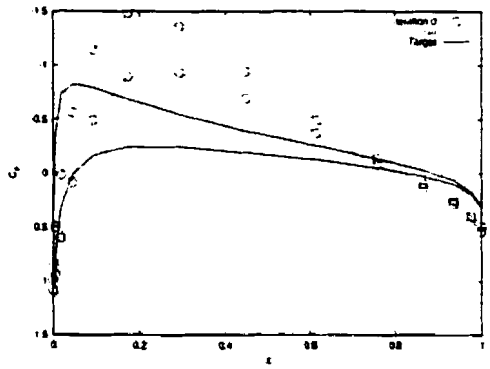

Figure 5.49: Initial pressure distribution, case 2, run 2, HLA.

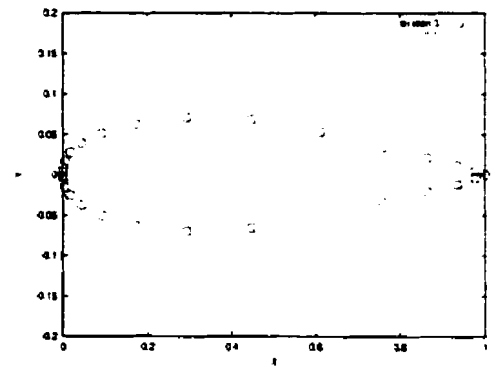

Figure 5.52: Airfoil geometry, 3 iterations, case 2, run 2, HLA.

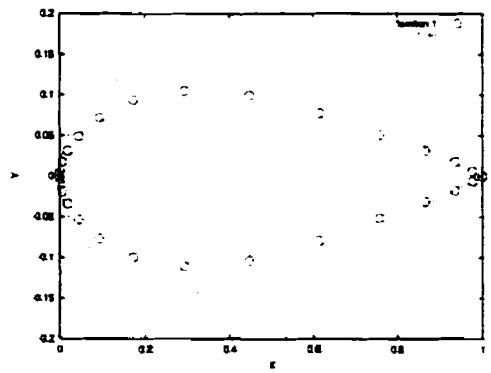

Figure 5.50: Airfoil geometry, 1 iteration, case 2, run 2, HLA.

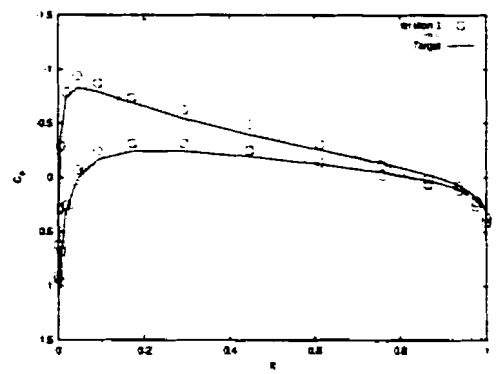

Figure 5.53: Pressure distribution, 3 iterations, case 2, run 2, HLA.

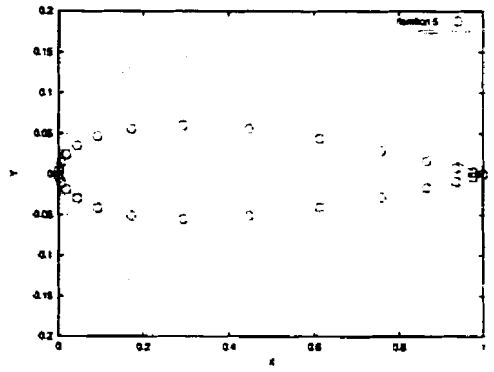

Figure 5.54: Airfoil geometry, 5 iterations, case 2, run 2, HLA.

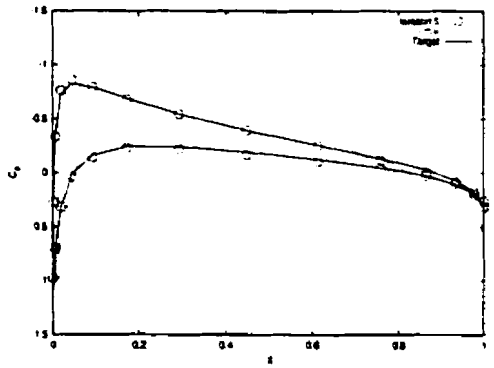

Figure 5.55: Pressure distribution, 5 iterations, case 2, run 2, HLA.

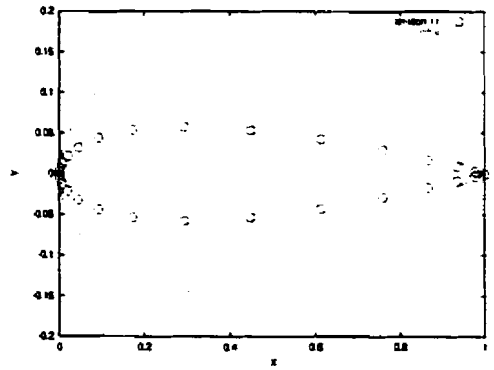

Figure 5.56: Airfoil geometry, 11 iterations, case 2, run 2, HLA.

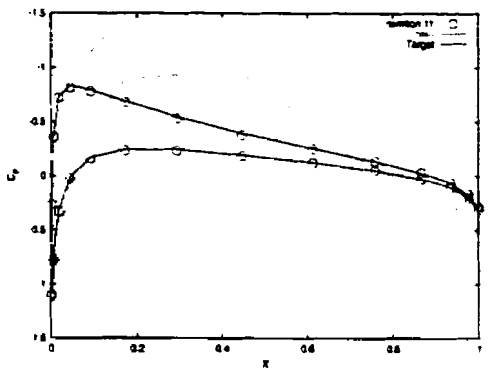

Figure 5.57: Pressure distribution, 11 iterations, case 2, run 2, HLA.

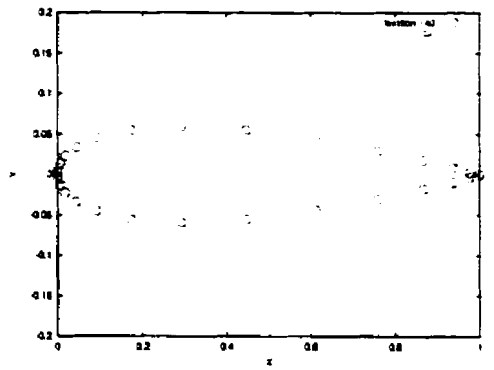

Figure 5.58: Final airfoil geometry, case 2, run 2, HLA.

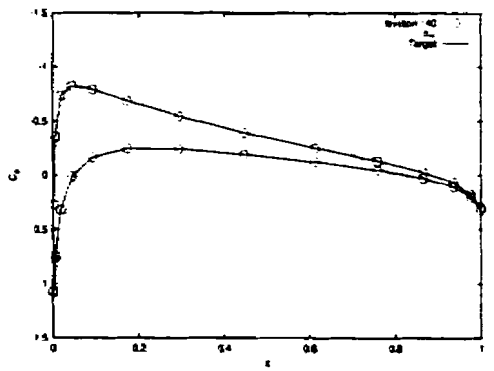

Figure 5.59: Final pressure distribution, case 2, run 2, HLA.

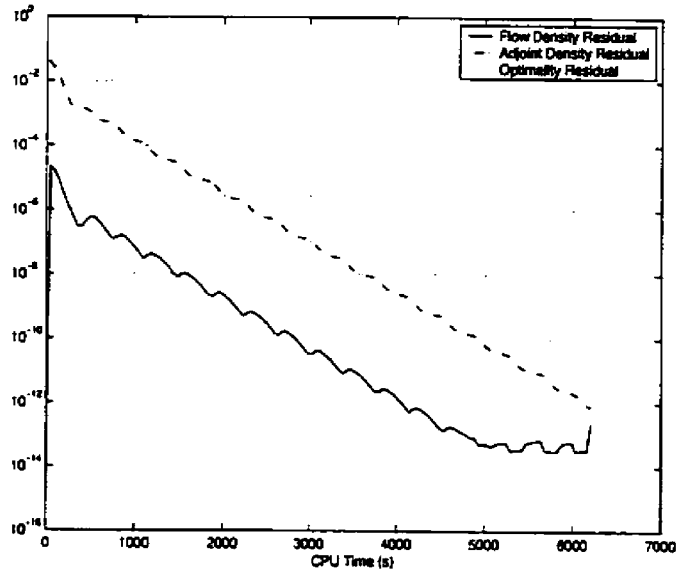

Figure 5.60: Convergence history, case 2, run 2, HLA.

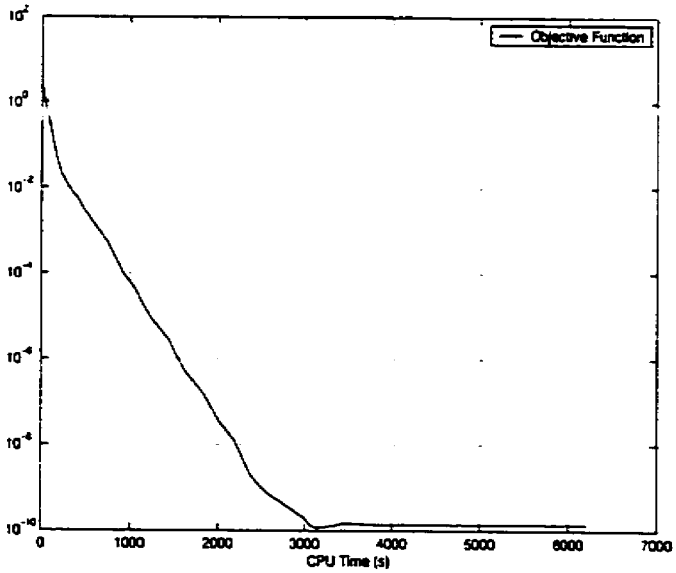

Figure 5.61: Objective function history, case 2, run 2, HLA.

5.3 Case 3: Subsonic flow, 8 design variables

Case 3 is a classic inverse design optimization problem. The objective is to start with one airfoil and to reach another airfoil that is used to generate the target objective pressure distribution. The conditions are the same as those for cases 1 and 2: A freestream Mach number of 0.5 and an angle of attack of 2°. The target geometry is the RAE2822 airfoil and the initial guess is the NACA0012 airfoil. Only 8 design variables are used, since the remaining 5 affect the leading and trailing edge. Since the grid at the wake of the RAE2822 airfoil differs from the grid for the NACA0012 airfoil, a modified NACA0012 is actually used. Basically, the airfoil consists of a NACA0012 airfoil with an RAE2822 leading and trailing edge. The design variables are listed in Table 5.10.

Design Variable	Control Point	Initial Value	Target Value	% Change
1	3	-0.008237091	0.002083546	-495.3%
2	4	-0.029184926	-0.006431745	353.8%
3	5	-0.060966288	-0.070838106	-13.9%
4	6	-0.061626221	-0.050526554	22.0%
5	10	0.061626222	0.049307073	25.0%
6	11	0.060966288	0.072702675	-16.1%
7	12	0.029184924	0.042692709	-31.6%
8	13	0.008237094	0.010584477	-22.2%

Table 5.10: Case 3 Initial Design Variables.

Two runs are made for this case. The first run is performed using a 22 × 8 grid and the second run is performed using a 43 × 15 grid. Details about the runs in terms of CPU time are shown in Table 5.11. When comparing Tables 5.5, 5.8, and 5.11, one can see a linear relationship between the number design variables and the amount of CPU time to run the optimization. This can be attributed to the two bottlenecks that remain in the efficiency of the optimization. Both the block inversion algorithm and the formation of the remaining elements that are formed by finite differences are a linear function of the number of design variables.

The initial airfoil geometry and pressure distribution for the first run are given in Figures 5.62 and 5.63. Various plots at subsequent iterations are given in Figures 5.64 - 5.71. The final airfoil shape, which is the RAE2822 airfoil and the corresponding pressure distribution are given in Figures 5.72 and 5.73. The convergence plot and the objective

Run	Grid	Linesearch	Optimization Time	Iterations	Time / Iteration
1	22 x 8	HLA	48.8 s	18	2.7 s
2	43 x 15	HLA	23 m 14.5 s	41	34.0 s

Table 5.11: Case 3 Runs.

function history are given in Figures 5.74 and 5.75.

For the second run, the initial airfoil geometry and pressure distribution are given in figures 5.76 and 5.77. For various iterations, plots are shown in Figures 5.78 - 5.85. The final airfoil shape and the corresponding pressure distribution are shown in Figures 5.86 and 5.87. The convergence plot and objective funtion history are given in Figures 5.88 and 5.89 respectively.

Table 5.12 shows the optimum design variables in comparison to the control variables that were used to generate the target pressure distribution. The second run, being done on a finer grid, provides a more accurate result.

Design Variable	Target Value	Run 1	Run 1 Error	Run 2	Run 2 Error
1	0.002083546	0.002083652	5.068283E-5	0.002083553	3.455647E-6
2	-0.006431745	-0.006431828	1.284255E-5	-0.006431762	2.720879E-6
3	-0.070838106	-0.070838068	-5.364344E-7	-0.070838108	2.823339E-8
4	-0.050526554	-0.050526590	7.124966E-7	-0.050526555	1.979157E-8
5	0.049307073	0.049307029	-8.923669E-7	0.049307058	-3.042160E-7
6	0.072702675	0.072702748	1.004090E-6	0.072702705	4.126396E-7
7	0.042692709	0.042692614	-2.225204E-6	0.042692677	-7.495425E-7
8	0.010584477	0.010584480	2.834339E-7	0.010584482	4.723900E-7

Table 5.12: Case 3 Final Design Variables.

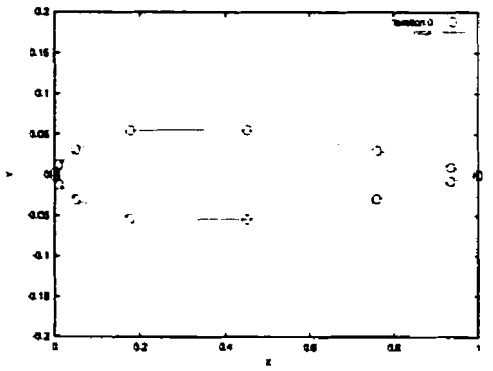

Figure 5.62: Initial airfoil geometry, case 3, run 1, HLA.

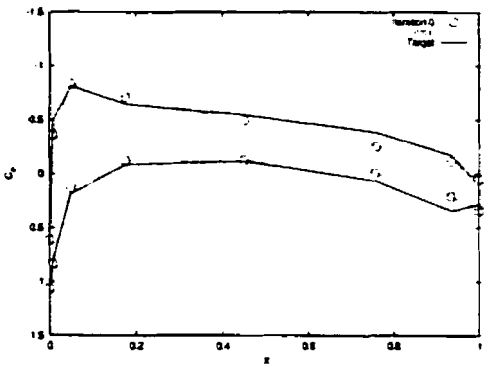

Figure 5.63: Initial pressure distribution, case 3, run 1, HLA.

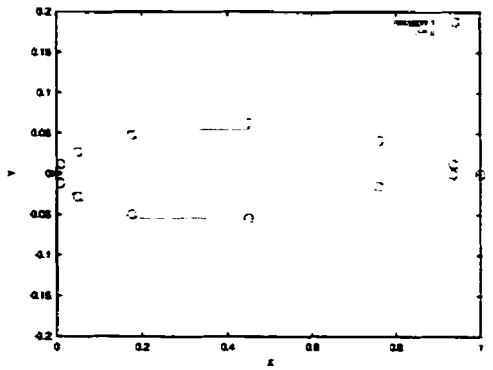

Figure 5.64: Airfoil geometry, 1 iteration, case 3, run 1, HLA.

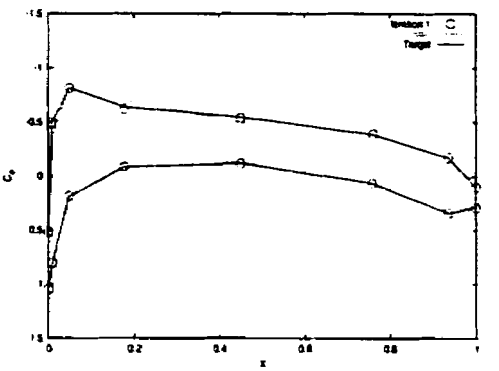

Figure 5.65: Pressure distribution, 1 iteration, case 3, run 1, HLA.

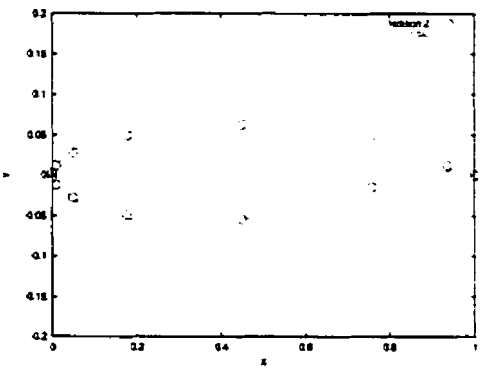

Figure 5.66: Airfoil geometry, 2 iterations, case 3, run 1, HLA.

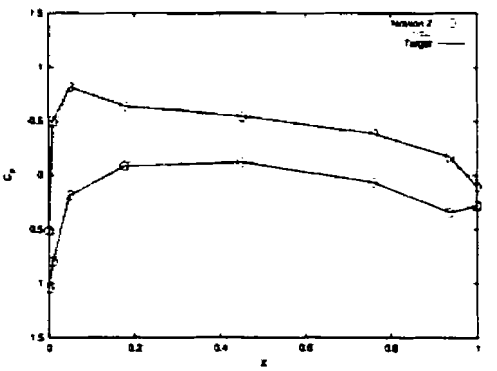

Figure 5.67: Pressure distribution, 2 iterations, case 3, run 1, HLA.

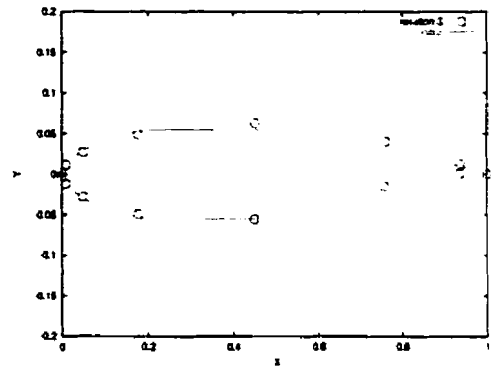

Figure 5.68: Airfoil geometry, 3 iterations, case 3, run 1, HLA.

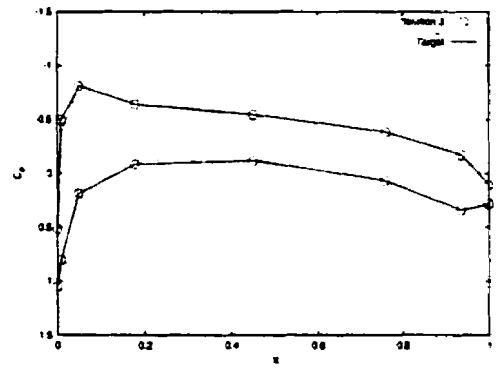

Figure 5.69: Pressure distribution, 3 iterations, case 3, run 1, HLA.

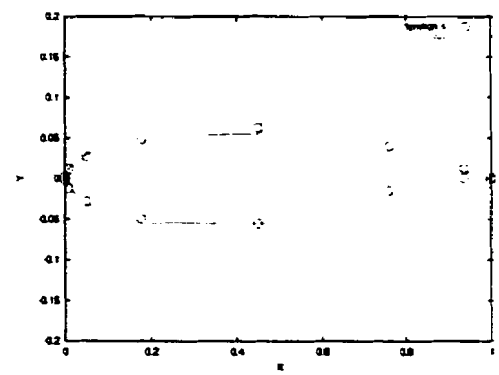

Figure 5.70: Airfoil geometry, 4 iterations, case 3, run 1, HLA.

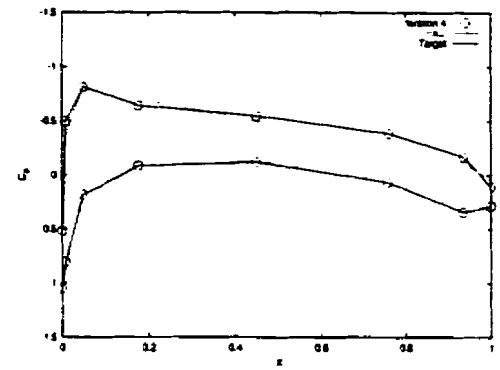

Figure 5.71: Pressure distribution, 4 iterations, case 3, run 1, HLA.

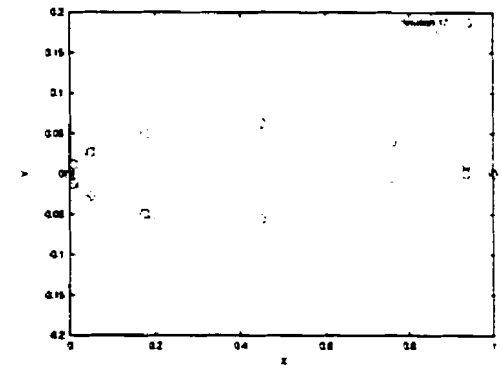

Figure 5.72: Final airfoil geometry, case 3, run 1, HLA.

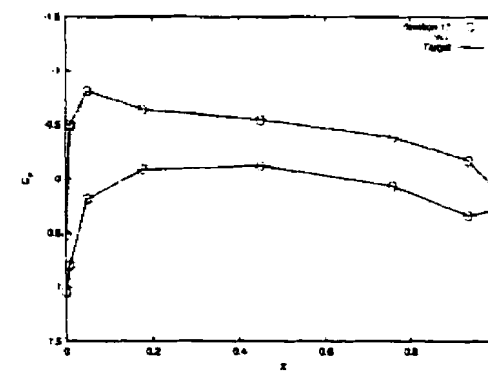

Figure 5.73: Final pressure distribution, case 3, run 1, HLA.

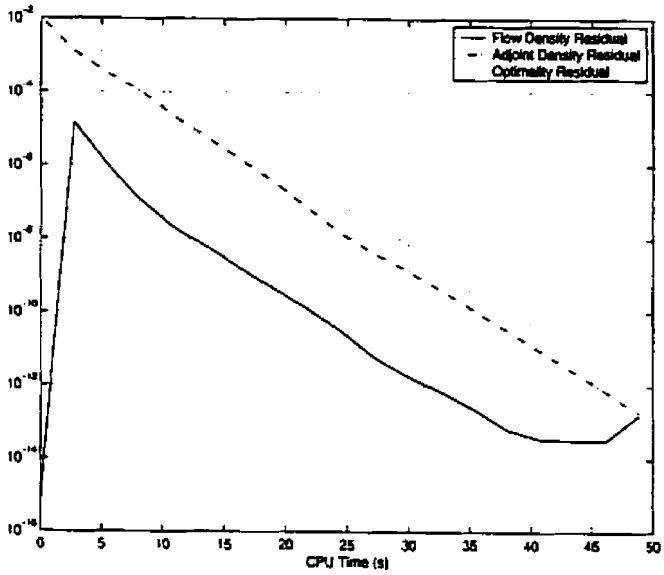

Figure 5.74: Convergence history, case 3, run 1, HLA.

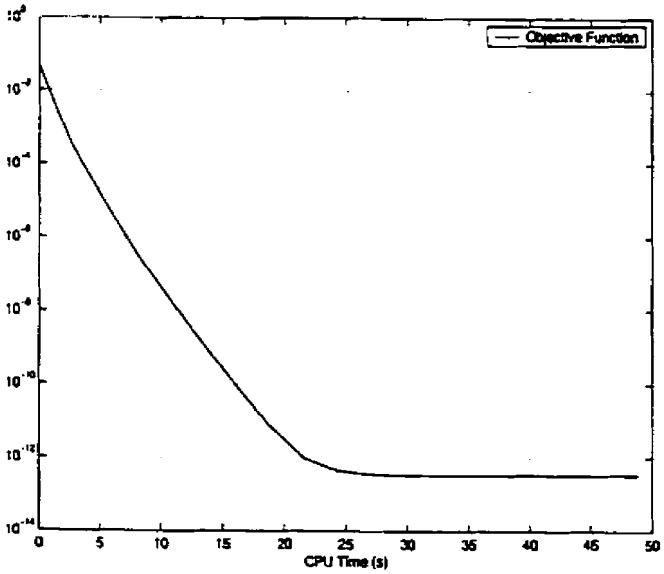

Figure 5.75: Objective function history, case 3, run 1, HLA.

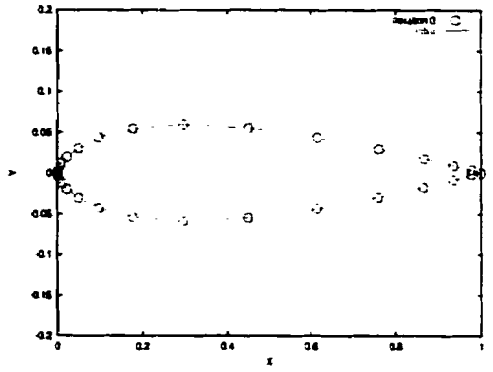

Figure 5.76: Initial airfoil geometry, case 3, run 2, HLA.

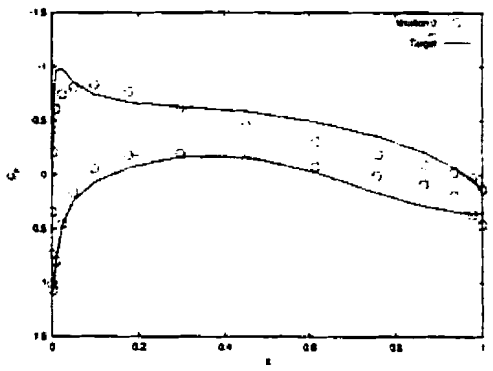

Figure 5.77: Initial pressure distribution, case 3, run 2, HLA.

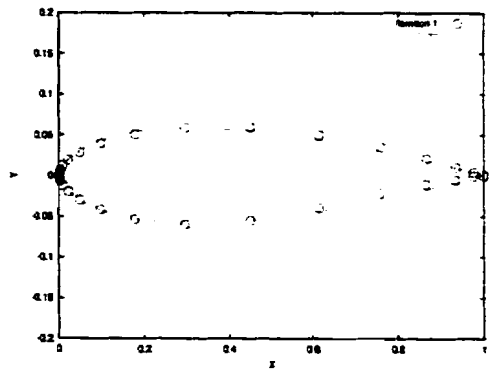

Figure 5.78: Airfoil geometry, 1 iteration, case 3, run 2, HLA.

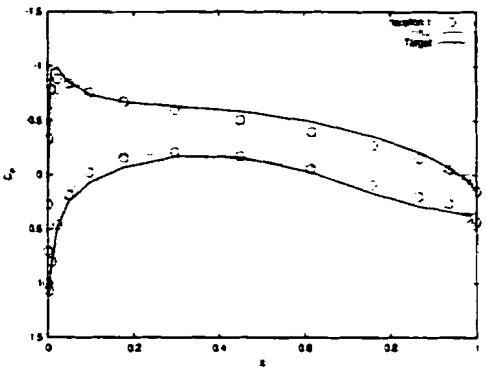

Figure 5.79: Pressure distribution, 1 iteration, case 3, run 2, HLA.

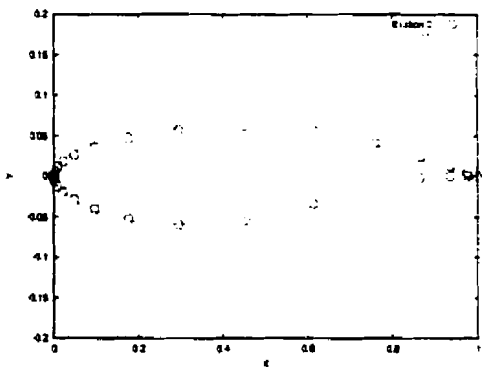

Figure 5.80: Airfoil geometry, 2 iterations, case 3, run 2, HLA.

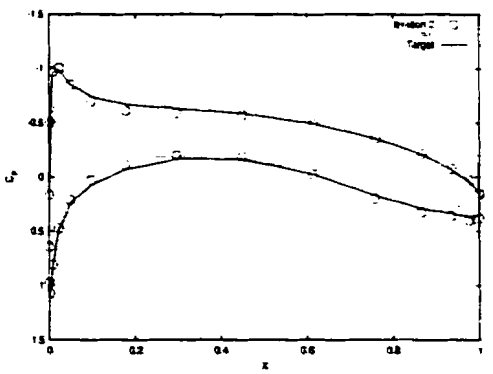

Figure 5.81: Pressure distribution, 2 iterations, case 3, run 2, HLA.

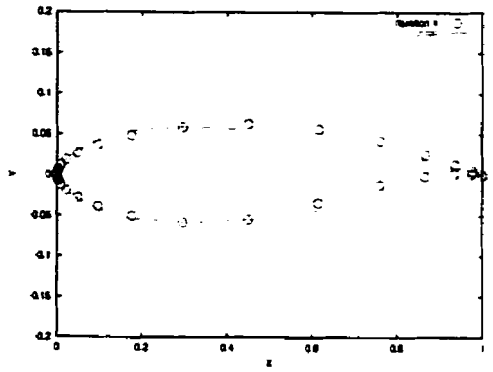

Figure 5.82: Airfoil geometry, 4 iterations, case 3, run 2, HLA.

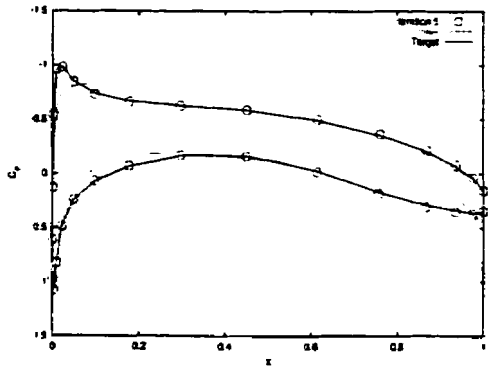

Figure 5.85: Pressure distribution, 5 iterations, case 3, run 2, HLA.

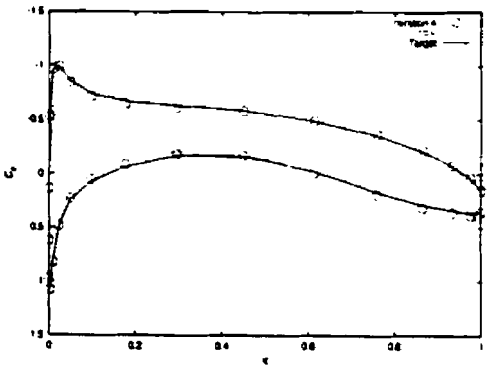

Figure 5.83: Pressure distribution, 4 iterations, case 3, run 2, HLA.

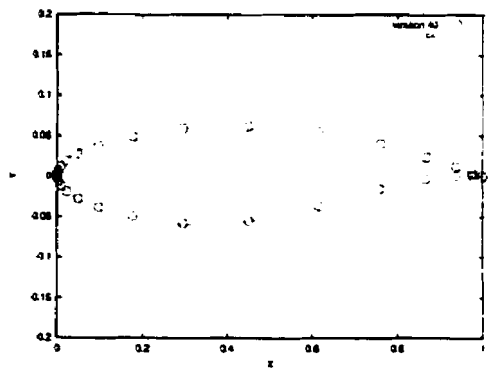

Figure 5.86: Final airfoil geometry, case 3, run 2, HLA.

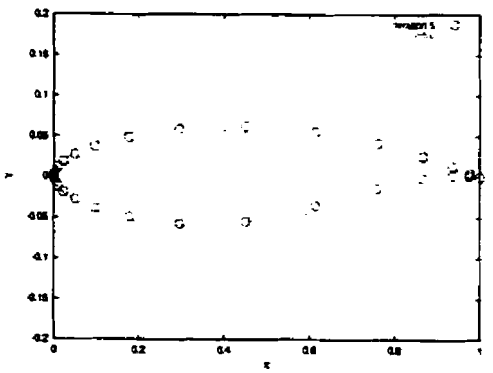

Figure 5.84: Airfoil geometry, 5 iterations, case 3, run 2, HLA.

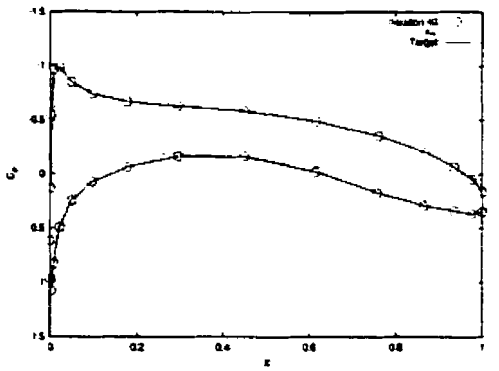

Figure 5.87: Final pressure distribution, case 3, run 2, HLA.

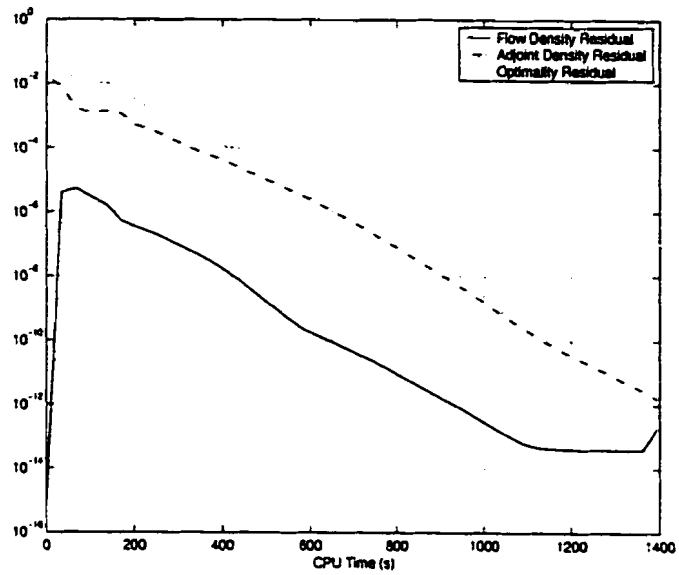

Figure 5.88: Convergence history, case 3, run 2, HLA.

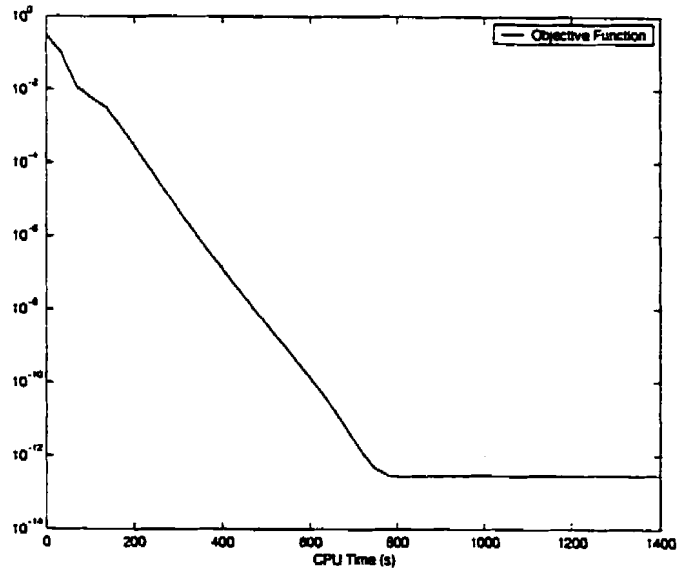

Figure 5.89: Objective function history, case 3, run 2, HLA.

5.4 Case 4: Transonic flow, 8 design variables

Case 4 is an extension of case 3 to the transonic flow regime. Apart from the Mach number being increased to 0.74 to prescribe the transonic flow, the initial design variables and angle of attack of the airfoil are the same as in case 3 as shown in Table 5.13. A finest grid (grid 3) of 85 x 29 nodes is needed to converge the flow solution for the analysis problem and is therefore used in the optimization. No grid study is performed for this case, but rather a comparison of the two linesearch methods is made. In the first run, the halving linesearch algorithm is used, and in the second run, the backtracking linesearch algorithm is used.

Design Variable	Control Point	Initial Value	Target Value	% Change
1	3	-0.008237091	0.002083546	-495.3%
2	4	-0.029184926	-0.006431745	353.8%
3	5	-0.060966288	-0.070838106	-13.9%
4	6	-0.061626221	-0.050526554	22.0%
5	10	0.061626222	0.049307073	25.0%
6	11	0.060966288	0.072702675	-16.1%
7	12	0.029184924	0.042692709	-31.6%
8	13	0.008237094	0.010584477	-22.2%

Table 5.13: Case 4 Initial Design Variables.

Details about the runs in terms of CPU time are given in Table 5.14. The backtracking linesearch algorithm requires fewer iterations than the halving linesearch algorithm. Furthermore, the BLA is slightly faster than the HLA in terms of CPU time per iteration. Each iteration takes much longer when compared to case 3, since the grid that is used is much finer. On the coarse 22 x 8 grid, the iteration time was 2.7 seconds, on the middle 43 x 15 grid, the iteration time was 34.0 seconds, and on the finest 85 x 29 grid, the iteration time was 105955.4 seconds. Clearly the growth is exponential as is expected with the refinement of the grid. The times stated are all for 8 design variable cases, since the iteration time is also a function of the number of design variables.

The initial airfoil geometry and pressure distribution for the first run are given in Figures 5.90 and 5.91. Various plots at subsequent iterations are given in Figures 5.92 - 5.99. In Figures 5.96 and 5.97 the airfoil geometry and the pressure distribution move in the wrong direction near the trailing edge. Clearly the linesearch method affects the

Run	Grid	Linesearch	Optimization Time	Iterations	Time / Iteration
1	85 x 29	HLA	29 h 25 m 55.4 s	192	9 m 11.9 s
2	85 x 29	BLA	27 h 14 m 51.2 s	180	9 m 5.0 s

Table 5.14: Case 4 Runs.

efficiency of the optimizer. The final airfoil shape, which is the RAE2822 airfoil, and the corresponding pressure distribution are given in Figures 5.100 and 5.101. The convergence plot and the objective function history are given in Figures 5.102 and 5.103.

For the second run, the initial airfoil geometry and pressure distribution are given in figures 5.104 and 5.105. For various iterations, plots are shown in Figures 5.106 - 5.113. The movement of the design variables throughout the entire optimization is almost completely monotonic. The final airfoil shape and the corresponding pressure distribution are shown in Figures 5.114 and 5.115. The convergence plot and objective funtion history are given in Figures 5.116 and 5.117 respectively.

Table 5.15 shows the optimum design variables in comparison to the control variables that were used to generate the target pressure distribution. Both the BLA and the HLA provide equally-accurate results in terms of the error in the target design variables. Both linesearch methods were able to handle this weak shock case, thus verifying the fully-coupled algorithm is applicable to more difficult design problems.

Design Variable	Target Value	Run 1	Run 1 Error	Run 2	Run 2 Error
1	0.002083546	0.002083602	2.687726E-4	0.002083642	4.607529E-5
2	-0.006431745	-0.006431738	-1.088352E-6	-0.006431627	-1.834649E-5
3	-0.070838106	-0.070838131	3.529173E-7	-0.070838064	-5.929012E-7
4	-0.050526554	-0.050526546	-1.583326E-7	-0.050526576	4.354146E-7
5	0.049307073	0.049307070	-6.084320E-8	0.049307070	-6.084319E-8
6	0.072702675	0.072702733	7.977698E-7	0.072702854	2.462082E-6
7	0.042692709	0.042692703	-1.405392E-7	0.042692688	-4.918872E-7
8	0.010584477	0.010584488	1.039258E-6	0.010584516	3.684641E-6

Table 5.15: Case 4 Final Design Variables.

Finally, a comparison is made to the discrete adjoint method using the KSOPT [13] optimizer for the same parameters. Details about the discrete adjoint method that is used in the comparison can be found in [14]. The design variables vary slightly since

this discrete adjoint method optimization software contains a modified portion of code that is shared between it and the fully-coupled algorithm. However this inconsistency is not considered in detail since this is a qualitative analysis. Figures 5.118 and 5.119 show the initial and final airfoil geometries. The optimum solution, which is the RAE2822 airfoil, is reached. Figure 5.120 shows the pressure distribution at the end of the optimization. The target pressure distribution is reached as required. A more meaningful comparison can really be made between the two methods when general optimization problems, such as drag minimization, are considered. However, in the scope of this thesis, those problems are not considered. The fully-coupled algorithm reaches the optimum solution as required and is qualitatively verified by the disrete adjoint method optimizer.

In terms of CPU time, the discrete adjoint method took far less time to reach the optimum. The optimization took roughly 10 minutes on the same computer. However, the method requires 46 fully-converged flow evaluations. And in each of the 46 flowfield evaluations, a gradient evaluation is also required. The fully-coupled algorithm converges the flowfield solution only once. With future improvements to the fully-coupled algorithm, a fair comparison can be made between the two approaches in terms of CPU time. The objective function history is shown in Figure 5.121. While the discrete adjoint method converges very fast near the optimum solution, in the inital stages of the design, its progress is much slower. However, the fully-coupled algorithm makes significant progress throughout the design process. In fact the objective function history plot for all cases shown are nearly linear. This is more useful to designers who wish to see meaningful results in the early stages of the design. Although the airfoil configuration and the pressure distribution (i.e. flowfield state variables) do not match at any given iteration in the fully-coupled algorithm, an intermediate airfoil shape could easily be given to a flow solver and its flowfield could be converged properly, using the current state values as a warm-start initial guess.

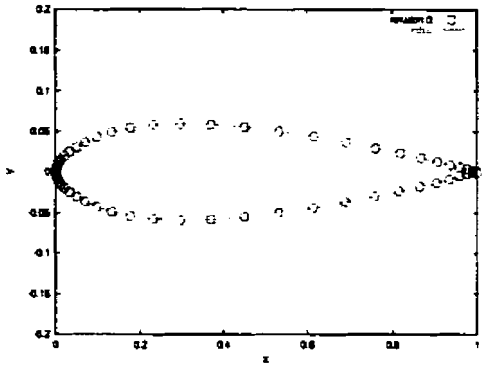

Figure 5.90: Initial airfoil geometry, case 4, run 1, HLA.

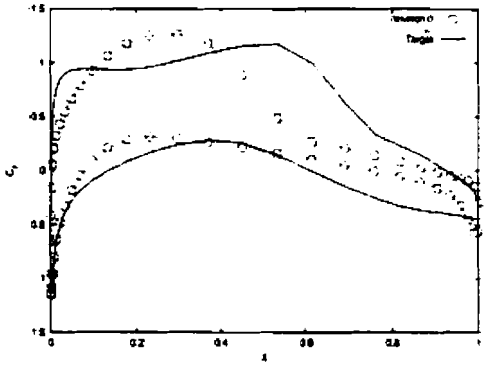

Figure 5.91: Initial pressure distribution, case 4, run 1, HLA.

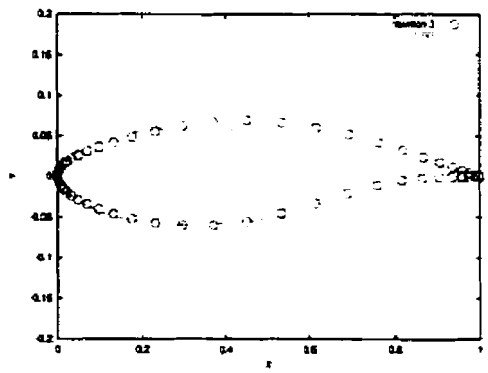

Figure 5.92: Airfoil geometry, 3 iterations, case 4, run 1, HLA.

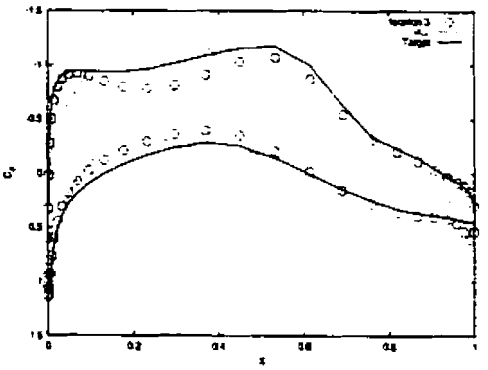

Figure 5.93: Pressure distribution, 3 iteration, case 4, run 1, HLA.

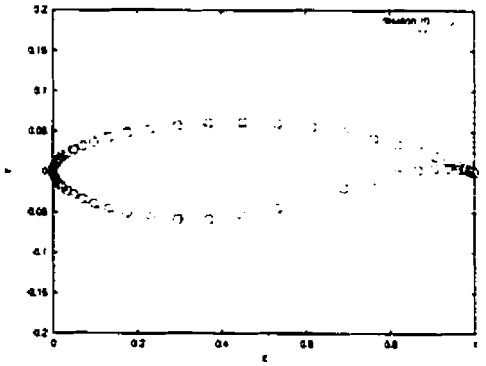

Figure 5.94: Airfoil geometry, 10 iterations. case 4, run 1, HLA.

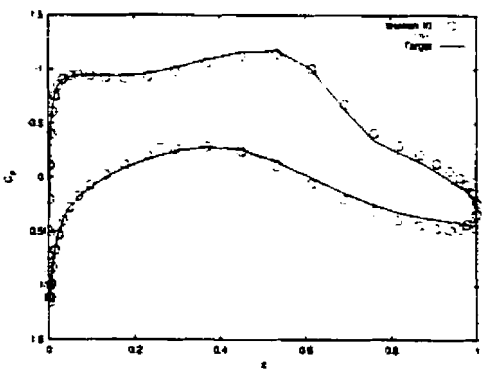

Figure 5.95: Pressure distribution, 10 iterations, case 4, run 1, HLA.

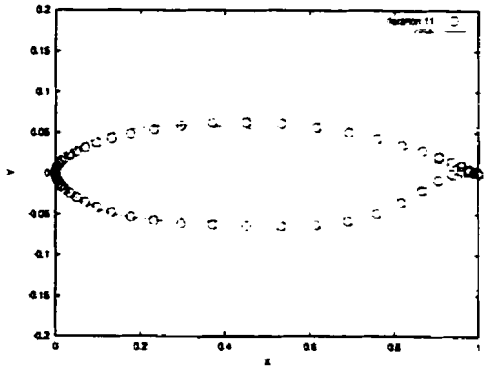

Figure 5.96: Airfoil geometry, 11 iterations, case 4, run 1, HLA.

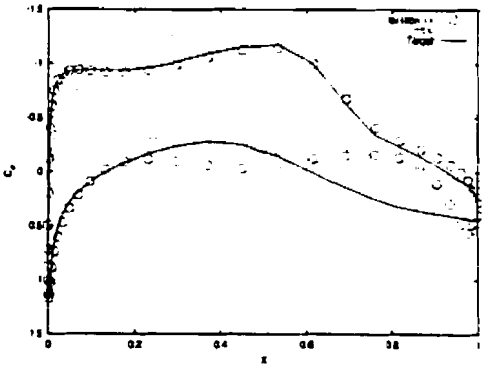

Figure 5.97: Pressure distribution, 11 iterations, case 4, run 1, HLA.

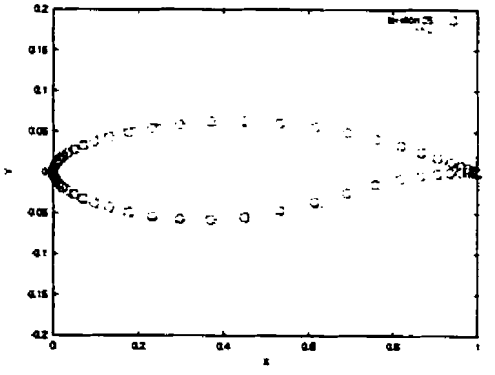

Figure 5.98: Airfoil geometry, 25 iterations, case 4, run 1, HLA.

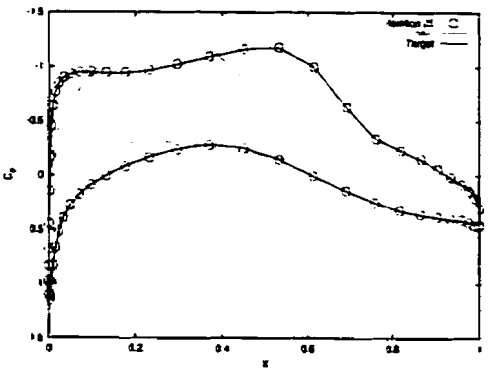

Figure 5.99: Pressure distribution, 25 iterations, case 4, run 1, HLA.

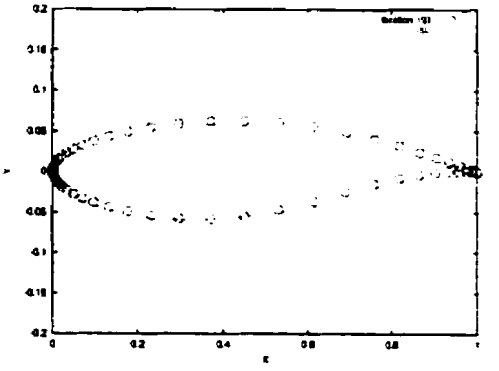

Figure 5.100: Final airfoil geometry, case 4, run 1, HLA.

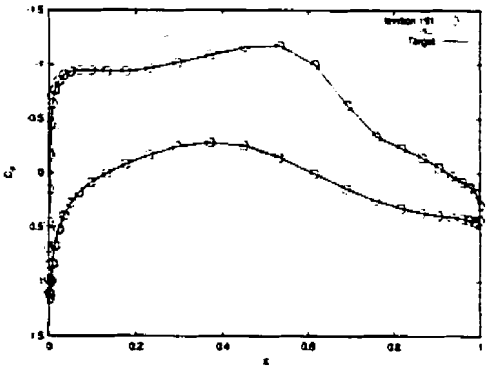

Figure 5.101: Final pressure distribution, case 4, run 1, HLA.

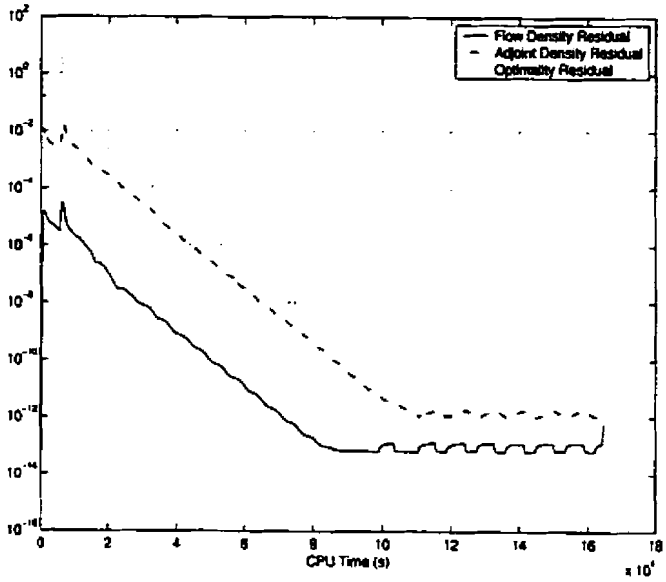

Figure 5.102: Convergence history, case 4, run 1, HLA.

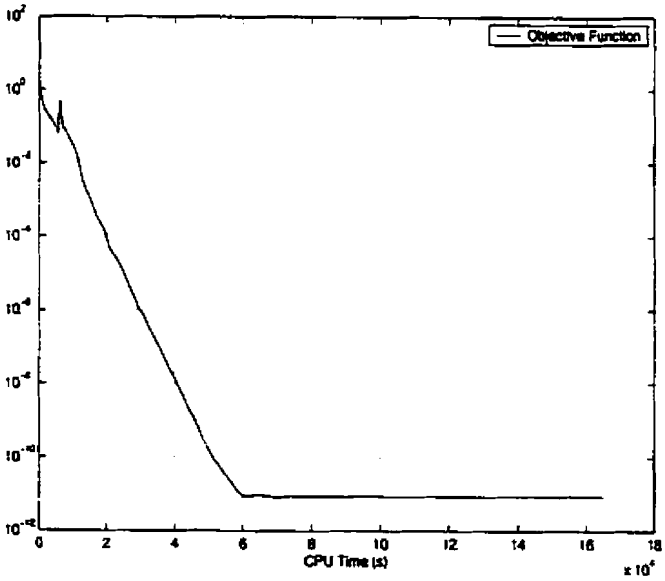

Figure 5.103: Objective function history, case 4, run 1, HLA.

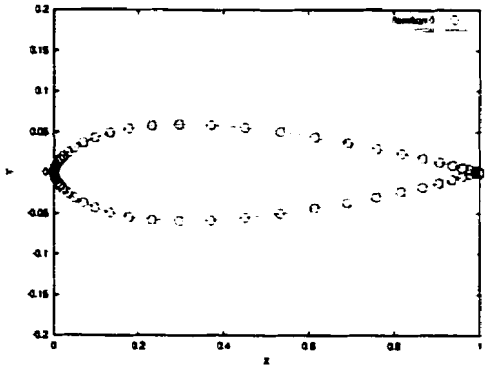

Figure 5.104: Initial airfoil geometry, case 4, run 2, BLA.

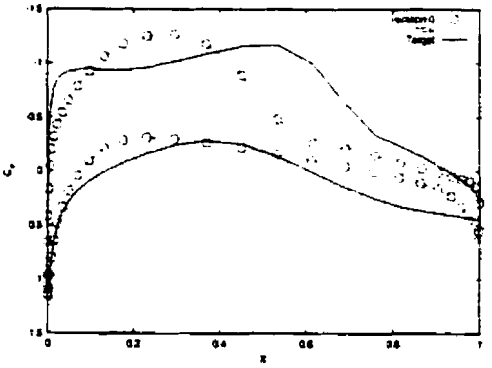

Figure 5.105: Initial pressure distribution, case 4, run 2, BLA.

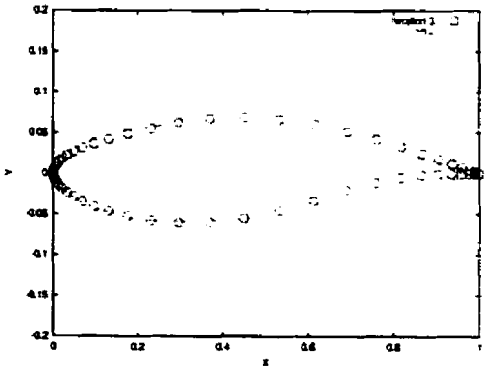

Figure 5.106: Airfoil geometry, 3 iterations, case 4, run 2, BLA.

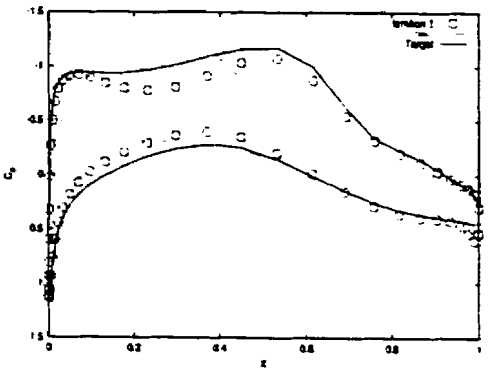

Figure 5.107: Pressure distribution, 3 iteration, case 4, run 2, BLA.

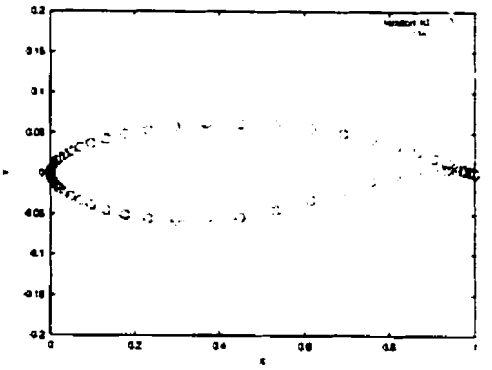

Figure 5.108: Airfoil geometry, 10 iterations, case 4, run 2, BLA.

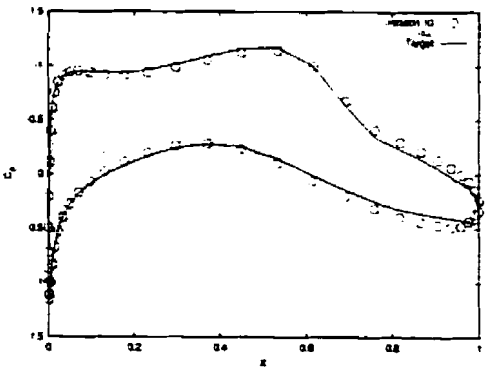

Figure 5.109: Pressure distribution, 10 iterations, case 4, run 2, BLA.

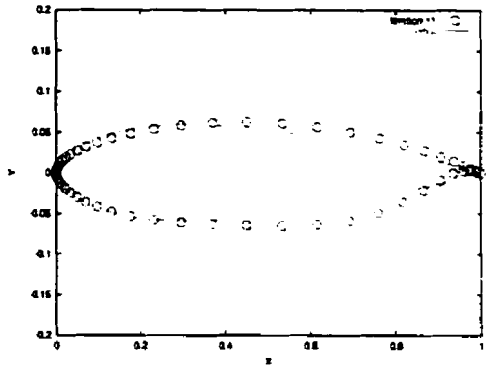

Figure 5.110: Airfoil geometry, 11 iterations, case 4, run 2, BLA.

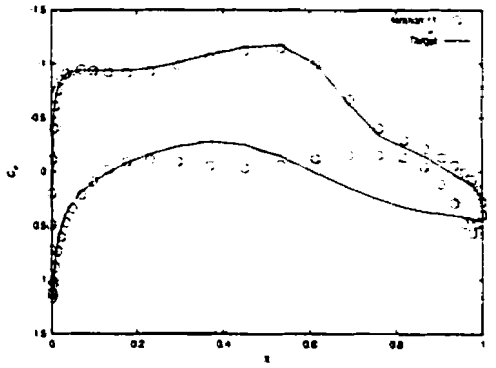

Figure 5.111: Pressure distribution, 11 iterations, case 4, run 2, BLA.

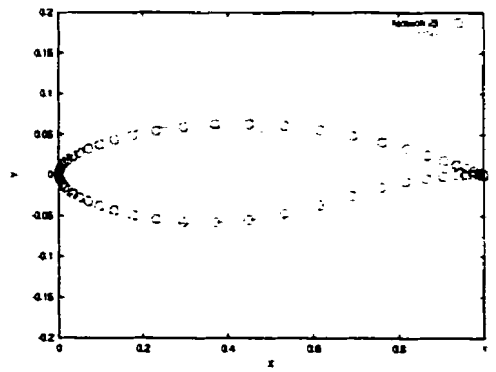

Figure 5.112: Airfoil geometry, 25 iterations, case 4, run 2, BLA.

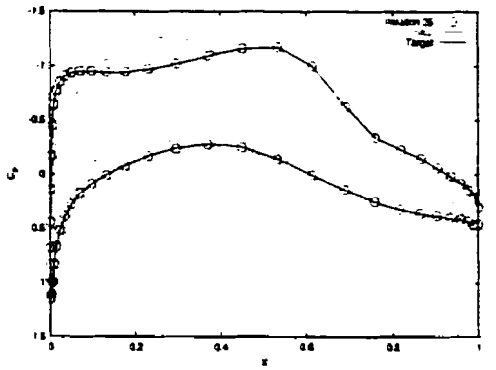

Figure 5.113: Pressure distribution, 25 iterations, case 4, run 2, BLA.

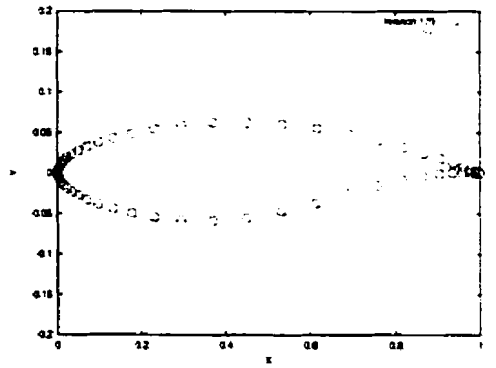

Figure 5.114: Final airfoil geometry, case 4, run 2, BLA.

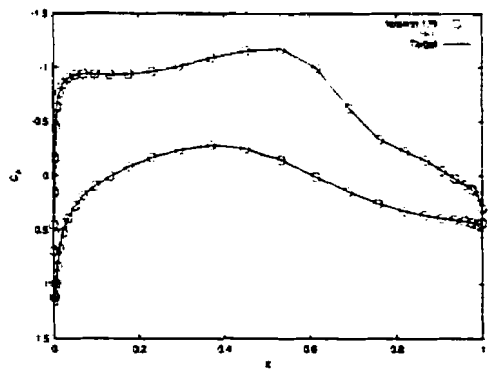

Figure 5.115: Final pressure distribution, case 4, run 2, BLA.

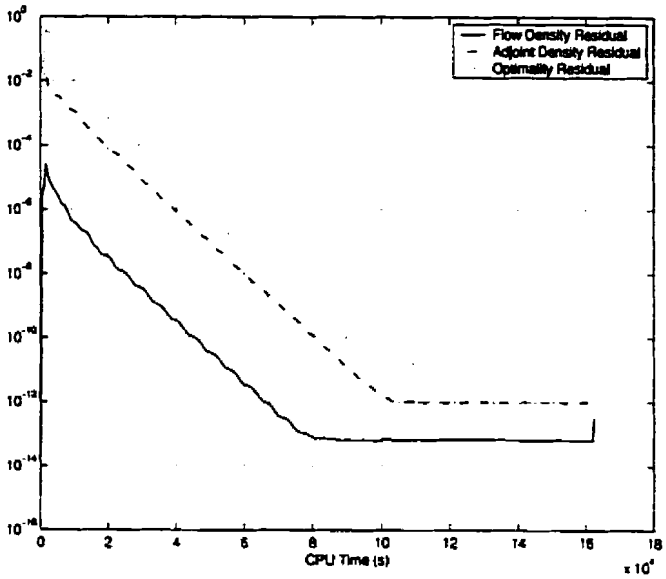

Figure 5.116: Convergence history, case 4, run 2, BLA.

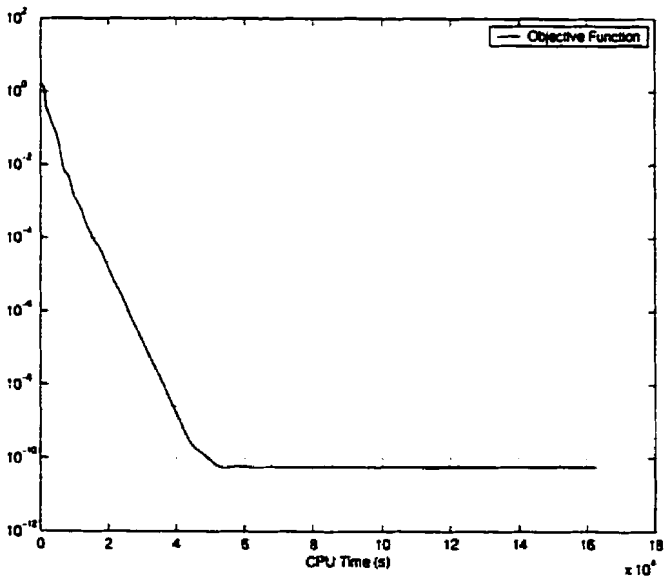

Figure 5.117: Objective function history, case 4, run 2, BLA.

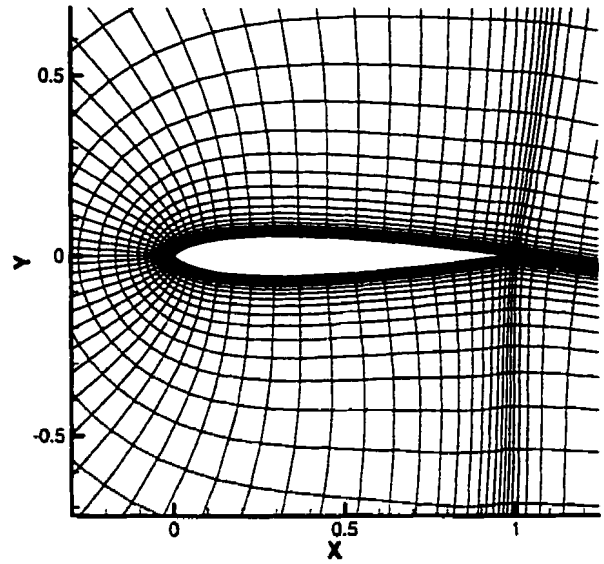

Figure 5.118: Initial NACA0012 airfoil with RAE2822 leading and trailing edges.

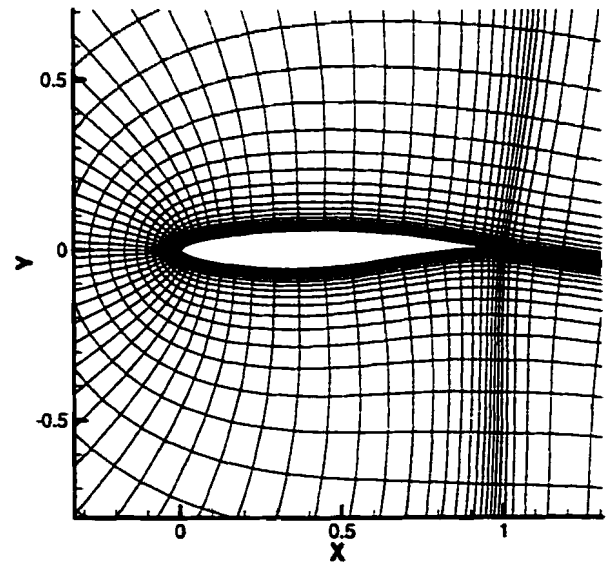

Figure 5.119: Optimum solution (RAE2822 airfoil) using the discrete adjoint method.

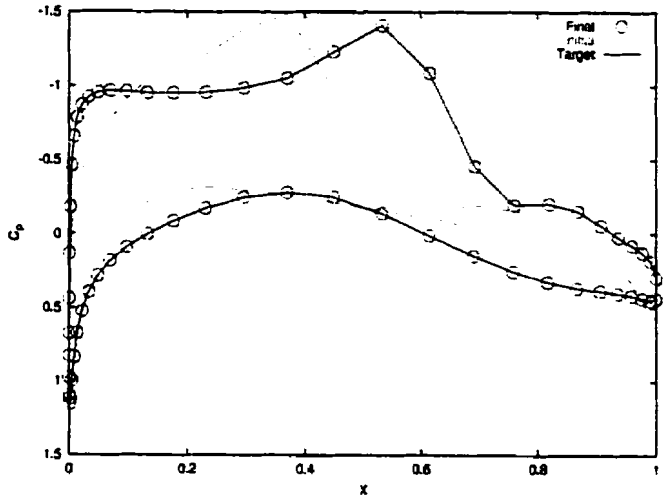

Figure 5.120: Pressure distribution using the discrete adjoint method.

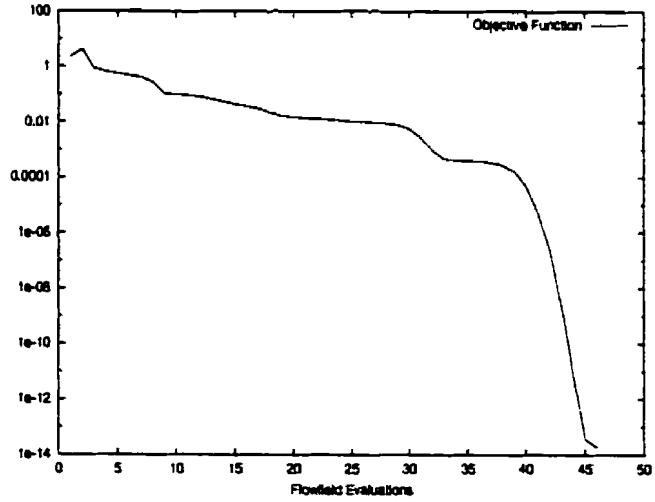

Figure 5.121: Objective function history using the discrete adjoint method.

Chapter 6

Conclusions

The fully-coupled algorithm for aerodynamic design optimization consists of combining all of the governing equations defining the optimization problem into a single system and solving that system only once. The equations defining the optimization problem include the flow equations, the adjoint equations, and the optimality conditions. The method produces the same results as those found using the discrete adjoint method, however the flow equations are only converged once, which is the key motivation for implementing and developing this method.

In this research, the primary objective was to implement the fully-coupled algorithm for quasi-1D nozzle design problems and to extend the algorithm to 2D, inviscid, airfoil shape design problems. This objective has been satisfied, as shown in Chapters 4 and 5. The results in 1D act as a proof that the algorithm works. The results in 2D show how the algorithm is in the initial stages of becoming a practical tool for realistic aerodynamic design optimization. This thesis is largely an initial step in the direction of creating a powerful, efficient, and robust alternative to conventional gradient-based optimization techniques.

6.1 The Fully-Coupled Algorithm and its Development

Inverse shape design optimization problems were considered. For this thesis the inverse design process consists of converging the analysis problem for a given shape and storing the pressure distribution from this flow solution. Next, an initial shape guess is chosen along with converged flow solution for this shape. An objective function is defined based on

the current and target pressure distributions and the optimizer iteratively solves the equations defining the optimization problem until they have converged to below a predefined tolerance. Furthermore, the objective function is also driven to a small value.

To solve the fully-coupled system, Newton's method is used followed by a linesearch on the update. Two linesearch algorithms were tested, including the halving linesearch algorithm and the backtracking linesearch algorithm. For Newton's method the formation of the system Jacobian is required. Initially this was done using finite differences.

In the development of this algorithm, it became obvious after preliminary tests in 1D that the initial finite-difference formulation of the system Jacobian would be a bottleneck in the optimization process. For this reason, a significant amount of time was invested in approximating many terms. For example the flow Jacobian, which occurs twice, is not a complete linearization of the flow residual vector. The other finite-difference block term, which has the same dimensions as the flow Jacobian, was the next major bottleneck in the process of making the algorithm more efficient. This term requires the formation of a flow Jacobian and the evaluation of a matrix-vector product with an adjoint-variable vector for each of the flow variable perturbations in the finite-difference process. This block term was approximated by the Hessian of the objective function, as shown in Appendix C, with no significant penalty to the number of iterations. This approximation provided the most significant improvement in the efficiency of the fully-coupled algorithm, since the major finite-difference term was removed.

A novel block inversion algorithm is used to compute the Newton update. It consists of breaking up the larger system Jacobian into a 3×3 matrix of blocks, where the largest block dimensions are of the same size as the flow Jacobian. It is described in Appendix A.

The linesearch component of the fully-coupled algorithm was explored in some detail. The backtracking linesearch algorithm performed better than the halving linesearch algorithm for the cases where they were compared.

6.2 Future Work

There are still four remaining terms in the system Jacobian that pose a problem to the efficiency of the optimization. The block terms are described in (3.5). Three of those four terms, including θ_{13}, θ_{23}, and θ_{33}, are not significantly expensive to form since they

only require the formation of a number of right hand sides equal to the number of design variables. Appendix D proves that the final finite-difference formed block term, θ_{31}, is equal to θ_{23} meaning this term will not have to be formed at all, since it can be copied from the inexpensively-formed θ_{23} block term.

Currently in the block inversion method, the sub-systems with the flow Jacobian are solved using a single LU decomposition with a number of backsolves equal to the number of design variables plus one. Another future improvement would be to use matrix-free GMRES in the SPARSKIT [15] library to perform the sub-system solves. Furthermore, the sub-systems with the transpose of the flow Jacobian, are also currently back-solved with the same LU decomposition. GMRES can be used to iteratively solve this system, however not in a matrix-free form, since a system with a left hand side matrix equal to the transpose of the flow Jacobian cannot be solved with matrix-free GMRES. Furthermore, a comparison should be made to when the system is solved without the block decomposition. GMRES can also be applied here, however significant research would be needed in the development of a preconditioner. In this full system solution method, when the equations and variables are rearranged (i.e. rearranging the blocks of the Jacobian) the system is put in KKT (Karush-Kuhn-Tucker) form [16], which has some algebraic and computational benefits in terms of being solved efficiently. For example, the KKT system is symmetric. In either case, a reordering strategy such as RCM (Reverse Cuthill-McKee) would be useful in speeding up the solution process.

There is some future research that can be made in area of linesearch techniques. For example, in the initial iterations, the noise that corrupts the backtracking linesearch's cubic and quadratic approximations can be smoothed. However, this avenue of research is not as important at the moment as are the implementation of a better system solution method and a more efficient formulation of the fully-coupled system Jacobian and/or preconditioner.

While the classic inverse design problem was an excellent tool for verifying the usefulness and accuracy of the fully-coupled algorithm, more realistic design problems must be considered. The current algorithm incorporates geometric constraints in the linesearching process as another condition for the linesearch parameter to satisfy. One example of a realistic design problem would be the minimization of drag with specified geometric constraints in a given set of freestream conditions. Another case would be the minimization of the ratio of drag to lift. The results of these cases are essential in making a meaningful compari-

son between this method and the state-of-the-art discrete adjoint method. Furthermore, multi-point design optimization can be investigated.

References

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

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] Hirsch, C., *Numerical Computation of Internal and External Flows*, Vol. 2, John Wiley & Sons, 1994.

[9] 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.

[10] Jameson, A. and Schmidt, W. and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping," AIAA Paper 81-1259, June 1981.

[11] 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.

[12] Pueyo, A., *An Efficient Newton-Krylov Method for the Euler and Navier-Stokes Equations*, Ph.D. thesis, University of Toronto, Dec. 1997.

[13] Wrenn, G. A., "An Indirect Method for Numerical Optimization Using the Kreisselmeier-Steinhauser Function," Tech. rep., NASA CR 4220, March 1989.

[14] Nemec, M. and Zingg, D.W., *Towards Efficient Aerodynamic Shape Optimization Based on the Navier-Stokes Equations*, No. AIAA-2001-2532, Anaheim, CA, June 2001.

[15] Saad, Y., "SPARSKIT: a basic tool kit for sparse matrix computations," Tech. rep., http:// www.cs.umn.edu/ Research/ arpa/ SPARSKIT/ sparskit.html, 1994.

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

Appendix A

Block Inversion Algorithm

We wish to solve the block linear system

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \Delta\vec{Q} \\ \Delta\vec{\Phi} \\ \Delta\vec{X} \end{pmatrix} = \begin{pmatrix} \vec{R} \\ \vec{S} \\ \vec{T} \end{pmatrix} \tag{A.1}$$

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

$$\mathbf{J}\Delta\vec{K} = \vec{\Gamma}. \tag{A.2}$$

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:

$$\mathbf{E} = \mathbf{A}^T \tag{A.3}$$

$$\mathbf{H} = \mathbf{C}^T \tag{A.4}$$

$$\mathbf{B} = \mathbf{0}. \tag{A.5}$$

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

The system is separated into three equations and, at the moment, only simplification (A.5) is applied. The resulting equations are

$$\mathbf{A}\Delta\vec{Q} + \mathbf{C}\Delta\vec{X} = \vec{R} \tag{A.6}$$

$$\mathbf{D}\Delta\vec{Q} + \mathbf{E}\Delta\vec{\Phi} + \mathbf{F}\Delta\vec{X} = \vec{S} \tag{A.7}$$

$$\mathbf{G}\Delta\vec{Q} + \mathbf{H}\Delta\vec{\Phi} + \mathbf{I}\Delta\vec{X} = \vec{T}. \tag{A.8}$$

Equations (A.6) and (A.7) are of equal size and are much larger then (A.8).

Intermediate variables are introduced along the way and are denoted by the (′) operator. Isolating $\Delta\vec{Q}$ in (A.6) we have

$$\Delta\vec{Q} = \vec{R}' - \mathbf{C}'\Delta\vec{X} \tag{A.9}$$

where

$$\vec{R}' = \mathbf{A}^{-1}\vec{R} \tag{A.10}$$

and

$$\mathbf{C}' = \mathbf{A}^{-1}\mathbf{C}. \tag{A.11}$$

Substituting $\Delta\vec{Q}$ into (A.7) gives

$$\mathbf{E}\Delta\vec{\Phi} + \mathbf{F}'\Delta\vec{X} = \vec{S}' \tag{A.12}$$

where

$$\mathbf{F}' = \mathbf{F} - \mathbf{D}\mathbf{C}' \tag{A.13}$$

and

$$\vec{S}' = \vec{S} - \mathbf{D}\vec{R}'. \tag{A.14}$$

Substituting $\Delta\vec{Q}$ into (A.8) gives

$$\mathbf{H}\Delta\vec{\Phi} + \mathbf{I}'\Delta\vec{X} = \vec{T}' \tag{A.15}$$

where

$$\mathbf{I}' = \mathbf{I} - \mathbf{G}\mathbf{C}' \tag{A.16}$$

and

$$\vec{T}' = \vec{T} - \mathbf{G}\vec{R}'. \tag{A.17}$$

Isolating $\Delta\vec{\Phi}$ in (A.12) we have

$$\Delta\vec{\Phi} = \vec{S}'' - \mathbf{F}''\Delta\vec{X} \tag{A.18}$$

where

$$\vec{S}'' = \mathbf{E}^{-1}\vec{S}' \tag{A.19}$$

and

$$\mathbf{F}'' = \mathbf{E}^{-1}\mathbf{F}'. \tag{A.20}$$

Substituting $\Delta\vec{\Phi}$ into (A.15) gives

$$\mathbf{I}''\Delta\vec{X} = \vec{T}'' \tag{A.21}$$

where

$$\mathbf{I}'' = \mathbf{I}' - \mathbf{H}\mathbf{F}'' \tag{A.22}$$

and

$$\vec{T}'' = \vec{T}' - \mathbf{H}\vec{S}''. \tag{A.23}$$

Finally, (A.21) 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 (A.18) and (A.9) to find $\Delta\vec{\Phi}$ and $\Delta\vec{Q}$ respectively.

Although simplification (A.5) has been applied, this inversion process is likely more expensive than (A.2). However, once the simplifications (A.3) and (A.4) are also applied, especially (A.3), 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 (A.3) 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. $\vec{R}' = \mathbf{A}^{-1}\vec{R}$
2. $\mathbf{C}' = \mathbf{A}^{-1}\mathbf{C}$
3. $\mathbf{F}' = \mathbf{F} - \mathbf{D}\mathbf{C}'$
4. $\vec{S}' = \vec{S} - \mathbf{D}\vec{R}'$
5. $\mathbf{I}' = \mathbf{I} - \mathbf{G}\mathbf{C}'$
6. $\vec{T}' = \vec{T} - \mathbf{G}\vec{R}'$
7. $\vec{S}'' = \mathbf{E}^{-1}\vec{S}' = (\mathbf{A}^{-1})^T\vec{S}'$
8. $\mathbf{F}'' = \mathbf{E}^{-1}\mathbf{F}' = (\mathbf{A}^{-1})^T\mathbf{F}'$
9. $\mathbf{I}'' = \mathbf{I}' - \mathbf{H}\mathbf{F}''$
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. $\Delta\vec{Q} = \vec{R}' - \mathbf{C}'\Delta\vec{X}$

Figure A.1: Block Inversion Algorithm

Appendix B

Backtracking Linesearch

We wish to solve the nonlinear system

$$\vec{R}(\vec{x}) = \vec{0}. \tag{B.1}$$

Iterative techniques are commonly used to solve this system if an analytic solution is not known. One approach is to compute a Newton update, $\vec{p}_k$, at iteration k

$$\vec{p}_k = \vec{\Delta x}_k = -J(\vec{x}_k)^{-1}\vec{R}(\vec{x}_k) \tag{B.2}$$

where $J(\vec{x}_k)$ is the Jacobian of the system, and to restrict the update by some scalar, $\lambda_k \in (0, 1]$ as follows:

$$\vec{x}_{k+1} = \vec{x}_k + \lambda_k \vec{p}_k. \tag{B.3}$$

The problem of solving system (B.1) may also be posed as

$$\min_{\vec{x}\in\Re} f(\vec{x}) \tag{B.4}$$

where $f(\vec{x})$ is given by

$$f(\vec{x}) \equiv \frac{1}{2}\vec{R}(\vec{x})^T\vec{R}(\vec{x}). \tag{B.5}$$

The backtracking linesearch then uses this scalar function to compute the best possible λ_k for the update equation (B.3). The gradient of $f(\vec{x})$, $\vec{\nabla} f$, is given by

$$\vec{\nabla} f = \left[\frac{d}{d\vec{x}}\sum_{i=1}^{n}\frac{1}{2}r_i^2\right]^T = \sum_{i=1}^{n}\vec{\nabla} r_i^T r_i = J(\vec{x})^T\vec{R}(\vec{x}). \tag{B.6}$$

However, only the product $\vec{\nabla} f^T \vec{p}$ is ever explicity needed in the algorithm. This is simply given by

$$\vec{\nabla} f^T \vec{p} = \left[\vec{R}(\vec{x})^T J(\vec{x})\right]\left[-J(\vec{x})^{-1}\vec{R}(\vec{x})\right] = -\vec{R}(\vec{x})^T \vec{R}(\vec{x}). \tag{B.7}$$

Notice how (B.5) and (B.7) differ only by a scalar constant.

The backtracking linesearch framework is outlined in Figure B.1.

Set parameter $\alpha = 1.0 \times 10^{-4}$.
Set parameter $\rho_{\min} = 0.1$.
Set parameter $\rho_{\max} = 0.5$.
Set counter $m = 1$.
Set $\lambda_k = 1$ for a Newton iteration.
Compute $f(x_k + \lambda_k p_k)$.
While $f(x_k + \lambda_k p_k) > f(x_k) + \alpha\lambda_k \vec{\nabla} f(x_k)^T p_k$
1. Set $m = m + 1$.
2. If $m = 2$ perform a quadratic backtrack for parameter $\rho \in [\rho_{\min}, \rho_{\max}]$ which reduces λ_k by that ratio.
3. Else perform a cubic backtrack for parameter $\rho \in [\rho_{\min}, \rho_{\max}]$ which reduces λ_k by that ratio.
End of Loop.
λ_k is now ready to be used in the update of state $\vec{x}_k$.

Figure B.1: Backtracking Linesearch Algorithm

The parameter ρ is an internal parameter to the linesearch method which reduces λ_k during each backtracking step. The parameters α, $\rho_{\min}$, and $\rho_{\max}$ are all empirical. For the first backtrack, a quadratic backtrack is made, and for subsequent backtracks, a cubic backtrack is made. A new function is needed to perform these backtracks is defined as follows:

$$\hat{f}(\lambda_k) \equiv f(\vec{x}_k + \lambda_k \vec{p}_k). \tag{B.8}$$

Polynomials are then fitted to this function using known data from the previous iteration for $\vec{x}$, the tested Newton iteration, and previous backtracks.

B.1 Quadratic Backtrack

With the Newton iteration having failed, the following data is available to fit a quadratic polynomial, $\hat{m}_q(\lambda_k)$, to $\hat{f}(\lambda_k)$:

$$\hat{f}(0) = f(\vec{x}_k) \tag{B.9}$$

$$\hat{f}'(0) = \nabla f(\vec{x}_k)^T \vec{p}_k \tag{B.10}$$

$$\hat{f}(1) = f(\vec{x}_k + \vec{p}_k). \tag{B.11}$$

The quadratic polynomial is defined as

$$\hat{m}_q(\lambda_k) = a\lambda_k^2 + b\lambda_k + c \tag{B.12}$$

where

$$a = \hat{f}(1) - \hat{f}(0) - \hat{f}'(0) \tag{B.13}$$

$$b = \hat{f}'(0) \tag{B.14}$$

$$c = \hat{f}(0). \tag{B.15}$$

The extreme value of $\hat{m}_q(\lambda_k)$ is found at

$$\lambda_k^* = \frac{-\hat{f}'(0)}{2\left[\hat{f}(1) - \hat{f}(0) - \hat{f}'(0)\right]}. \tag{B.16}$$

It is easily verifiable that this λ_k^* is a minimum since

$$\hat{m}_q''(\lambda_k) = 2\left[\hat{f}(1) - \hat{f}(0) - \hat{f}'(0)\right] > 0. \tag{B.17}$$

The backtracking parameter, ρ, is defined as

$$\rho \equiv \lambda_k^*. \tag{B.18}$$

The linesearch parameter, λ_k, is updated as follows:

$$\lambda_k = \begin{cases} \rho & \text{if } \rho \in [\rho_{\min}, \rho_{\max}] \\ \rho_{\min} & \text{if } \rho < \rho_{\min} \\ \rho_{\max} & \text{if } \rho > \rho_{\max} \end{cases}. \tag{B.19}$$

B.2 Cubic Backtrack

With the quadratic backtrack having failed, $\hat{f}(0)$, $\hat{f}'(0)$, λ_k^p, $\hat{f}(\lambda_k^p)$, λ_k^{pp}, and $\hat{f}(\lambda_k^{pp})$ are available to fit a cubic polynomial, $\hat{m}_c(\lambda_k)$ to $\hat{f}(\lambda_k)$. λ_k^p is the λ_k from the previous backtrack and λ_k^{pp} is the λ_k from the backtrack before the previous backtrack. For example, or the first cubic backtrack, λ_k^p is the quadratic backtrack λ_k and λ_k^{pp} is the Newton update λ_k or 1.

The cubic polynomial is defined as

$$\hat{m}_c(\lambda_k) = a\lambda_k^3 + b\lambda_k^2 + c\lambda_k + d \tag{B.20}$$

where the coefficients a and b are given by

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_k^p - \lambda_k^{pp}} \begin{bmatrix} \frac{1}{\lambda_k^{p\,2}} & \frac{-1}{\lambda_k^{pp\,2}} \\ \frac{-\lambda_k^{pp}}{\lambda_k^{p\,2}} & \frac{\lambda_k^p}{\lambda_k^{pp\,2}} \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda_k^p) - \hat{f}(0) - \hat{f}'(0)\lambda_k^p \\ \hat{f}(\lambda_k^{pp}) - \hat{f}(0) - \hat{f}'(0)\lambda_k^{pp} \end{bmatrix} \tag{B.21}$$

and

$$c = \hat{f}'(0) \tag{B.22}$$

$$d = \hat{f}(0). \tag{B.23}$$

The extreme value of $\hat{m}_c(\lambda_k)$ that is used for a possible λ_k update is

$$\lambda_k^* = \frac{-b + \sqrt{b^2 - 3ac}}{3a}. \tag{B.24}$$

This value is the acceptable solution of the quadratic extreme value equation, since λ_k must be positive. The numerator is positive for this solution. The backtracking parameter, ρ, is defined as

$$\rho \equiv \lambda_k^*. \tag{B.25}$$

The linesearch parameter, λ_k, is updated as follows:

$$\lambda_k = \begin{cases} \rho\lambda_k & \text{if } \rho \in [\rho_{\min}, \rho_{\max}] \\ \rho_{\min}\lambda_k & \text{if } \rho < \rho_{\min} \\ \rho_{\max}\lambda_k & \text{if } \rho > \rho_{\max} \end{cases}. \tag{B.26}$$

Appendix C

Fully-Coupled Jacobian Block Element Hessian Approximation

The block element, θ_{21}, of the system Jacobian, $\Theta(K)$, defined in equation (3.3) is as follows:

$$\theta_{21} = \frac{\partial}{\partial Q}[(\frac{\partial R}{\partial Q})^T \Phi - (\frac{\partial J}{\partial Q})^T]. \tag{C.1}$$

A finite-difference approach may be used to form this element. However, this is computationally expensive in both time and memory. Approximations to forming this element are being explored, one of which is neglecting the first term, leaving only the Hessian matrix, $H(Q)$, of the objective function

$$\begin{aligned} \theta_{21} &\approx -\frac{\partial}{\partial Q}(\frac{\partial J}{\partial Q})^T \\ \theta_{21} &\approx -H(Q). \end{aligned} \tag{C.2}$$

Recall, that the objective function for the inverse design problems is given by

$$J(Q) \equiv \frac{1}{2}\sum_i ((C_p)_i - (C_p{}^*)_i)^2. \tag{C.3}$$

where $(C_p)_i$ and $(C_p{}^*)_i$ denote the pressure coefficient and target pressure coefficient on the airfoil boundary. The Hessian matrix is a diagonal matrix of 4x4 blocks (in 2 dimensions) where the blocks only exist for the node values on the airfoil surface. This is true because the pressure coefficient does not depend on neighbouring nodes' state values. Hence for the rest of this derivation, the subscript i will be dropped, since the formation of a single block element is only required to describe all the elements of the Hessian.

State Q at each node is given by

$$Q = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ e \end{pmatrix} \tag{C.4}$$

where q_1 - q_4 are the conservative flow variables. The metric Jacobian is not used in Q for the optimization, but was previously defined with it to conform with the notation of the Euler equations' flow solver. The target pressure coefficient is constant, and therefore any derivatives with respect to it are zero. The pressure coefficient on the airfoil is given by

$$C_p = (p\gamma - 1)\frac{2}{\gamma M_\infty^2} \tag{C.5}$$

where pressure is given by

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho(u^2 + v^2)\right) \tag{C.6}$$

and can be expressed in terms of the conservative variables as

$$p = (\gamma - 1)\left(q_4 - \frac{1}{2q_1}\rho(q_2^2 + q_3^2)\right). \tag{C.7}$$

With all of these definitions considered, the chain rule is applied to get the first derivative of the objective function with respect to the state Q:

$$\frac{\partial J}{\partial Q} = (C_p - C_p{}^*)\frac{\partial C_p}{\partial Q}. \tag{C.8}$$

The result $\frac{\partial J}{\partial Q}$ is a 4×1 array for each node for the 2D optimization problem. The Hessian of the objective function is found when the derivative of (C.8) is taken with respect to state Q again. The matrix is symmetric so it is indexed as follows:

$$H(Q) = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{12} & h_{22} & h_{23} & h_{24} \\ h_{13} & h_{23} & h_{33} & h_{34} \\ h_{14} & h_{24} & h_{34} & h_{44} \end{pmatrix}. \tag{C.9}$$

The intermediate steps are left out, as they are cumbersome to show. Finally, the elements

of the Hessian of the objective function look like:

$$
\begin{aligned}
h_{11} &= \left(\tfrac{(\gamma-1)}{M_\infty^2}\tfrac{(q_2^2+q_3^2)}{q_1^2}\right)^2 - 2(C_p - C_p{}^*)\tfrac{(\gamma-1)}{q_1 M_\infty^2}\tfrac{(q_2^2+q_3^2)}{q_1^2} \\
h_{12} &= -2\tfrac{(\gamma-1)^2}{M_\infty^4}\tfrac{q_2}{q_1}\tfrac{(q_2^2+q_3^2)}{q_1^2} + 2(C_p - C_p{}^*)\tfrac{(\gamma-1)}{q_1 M_\infty^2}\tfrac{q_2}{q_1} \\
h_{13} &= -2\tfrac{(\gamma-1)^2}{M_\infty^4}\tfrac{q_3}{q_1}\tfrac{(q_2^2+q_3^2)}{q_1^2} + 2(C_p - C_p{}^*)\tfrac{(\gamma-1)}{q_1 M_\infty^2}\tfrac{q_3}{q_1} \\
h_{14} &= 2\tfrac{(\gamma-1)^2}{M_\infty^4}\tfrac{(q_2^2+q_3^2)}{q_1^2} \\
h_{22} &= \left(2\tfrac{(\gamma-1)}{M_\infty^2}\tfrac{q_2}{q_1}\right)^2 - 2(C_p - C_p{}^*)\tfrac{(\gamma-1)}{q_1 M_\infty^2} \\
h_{23} &= \left(2\tfrac{(\gamma-1)}{M_\infty^2}\right)^2 \tfrac{q_2}{q_1}\tfrac{q_3}{q_1} \\
h_{24} &= -\left(2\tfrac{(\gamma-1)}{M_\infty^2}\right)^2 \tfrac{q_2}{q_1} \\
h_{33} &= \left(2\tfrac{(\gamma-1)}{M_\infty^2}\tfrac{q_3}{q_1}\right)^2 - 2(C_p - C_p{}^*)\tfrac{(\gamma-1)}{q_1 M_\infty^2} \\
h_{34} &= -\left(2\tfrac{(\gamma-1)}{M_\infty^2}\right)^2 \tfrac{q_3}{q_1} \\
h_{44} &= \left(2\tfrac{(\gamma-1)}{M_\infty^2}\right)^2 .
\end{aligned}
\tag{C.10}
$$

Appendix D

Block Jacobian Cross-Derivative Elements' Equivalence

We wish to show that

$$\left[\frac{\partial}{\partial X}\left(\left(\frac{\partial R}{\partial Q}\right)^T \Phi\right)\right] \equiv \left[\frac{\partial}{\partial Q}\left(\left(\frac{\partial R}{\partial X}\right)^T \Phi\right)\right]^T. \tag{D.1}$$

Let the flow variables, Q, the adjoint variables, Φ, and the design variables, X, be respectively defined as

$$Q = \begin{pmatrix} q_1 & q_2 & \cdots & q_{n-1} & q_n \end{pmatrix}^T \tag{D.2}$$

$$\Phi = \begin{pmatrix} \phi_1 & \phi_2 & \cdots & \phi_{n-1} & \phi_n \end{pmatrix}^T \tag{D.3}$$

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_{m-1} & x_m \end{pmatrix}^T \tag{D.4}$$

where n indicates the number of flow variables, m represents the number of design variables, and $n >> m$. The length of the flow variables vector is the same as the length of the adjoint variables vector.

The Jacobian of R with respect to Q is given by

$$\left(\frac{\partial R}{\partial Q}\right)_{ij} = \frac{\partial r_i}{\partial q_j} \tag{D.5}$$

and its transpose is given by

$$\left(\frac{\partial R}{\partial Q}\right)^T_{ij} = \frac{\partial r_j}{\partial q_i}. \tag{D.6}$$

Similarly, the transpose of the Jacobian of R with respect to X is given by

$$\left(\frac{\partial R}{\partial X}\right)^T_{ij} = \frac{\partial r_j}{\partial x_i}. \tag{D.7}$$

Let the first Jacobian with respect to the flow variables will be called the flow Jacobian and the second Jacobian with respect to the design variables be called the design Jacobian. The product of the transpose of the flow Jacobian with the adjoint variables is given by

$$\left(\left(\frac{\partial R}{\partial Q}\right)^T \Phi\right)_i = \frac{\partial r_j}{\partial q_i}\phi_j. \tag{D.8}$$

This quantity is a column vector, thus requiring only the single index, i. The repeated index, j, indicates that there is a summation of all of the terms within the bounds of j. Similarly, the product of the transpose of the design Jacobian with respect to the adjoint variables is given by

$$\left(\left(\frac{\partial R}{\partial X}\right)^T \Phi\right)_i = \frac{\partial r_j}{\partial x_i}\phi_j. \tag{D.9}$$

The Jacobian of (D.8) with respect to X is given by

$$\frac{\partial}{\partial X}\left(\left(\frac{\partial R}{\partial Q}\right)^T \Phi\right)_{ij} = \frac{\partial}{\partial x_j}\left(\frac{\partial r_k}{\partial q_i}\phi_k\right). \tag{D.10}$$

Similarly, the Jacobian of (D.9) with respect to Q is given by

$$\frac{\partial}{\partial Q}\left(\left(\frac{\partial R}{\partial X}\right)^T \Phi\right)_{ij} = \frac{\partial}{\partial q_j}\left(\frac{\partial r_k}{\partial x_i}\phi_k\right). \tag{D.11}$$

The adjoint variables are not a function of the flow variables and the design variables. Hence, the adjoint variables should be removed from the partial derivatives, leaving

$$\frac{\partial}{\partial X}\left(\left(\frac{\partial R}{\partial Q}\right)^T \Phi\right)_{ij} = \frac{\partial^2 r_k}{\partial x_j \partial q_i}\phi_k \tag{D.12}$$

and

$$\frac{\partial}{\partial Q}\left(\left(\frac{\partial R}{\partial X}\right)^T \Phi\right)_{ij} = \frac{\partial^2 r_k}{\partial q_j \partial x_i}\phi_k. \tag{D.13}$$

Although some discontinuities exist within the R vector, it is assumed to be differentiable and continuous. Therefore, using the mixed partials property, the order of differentiation in either equation can be reversed. For example, (D.13) becomes

$$\frac{\partial}{\partial Q}\left(\left(\frac{\partial R}{\partial X}\right)^T \Phi\right)_{ij} = \frac{\partial^2 r_k}{\partial x_i \partial q_j}\phi_k \tag{D.14}$$

which is the transpose of (D.12). Hence (D.1) is proven.