Infeasible Path Methods for an Aerodynamic Shape Optimization Problem

Omar Ghattas, Carlos Orozco

EDRC 12-55-92

Infeasible Path Methods for an Aerodynamic Shape Optimization Problem

Omar Ghattas[1] Carlos Orozco[2]
Carnegie Mellon University, Pittsburgh, PA 15213

[1] Assistant Professor, Department of Civil Engineering and Engineering Design Research Center
[2] Research Assistant, Department of Civil Engineering and Engineering Design Research Center

Abstract

We consider optimal design problems of systems governed by suitable discretizations of nonlinear partial differential equations. We present and examine a coordinate basis infeasible path method tailored to such design problems. We employ a particular null space representation which exploits the structure of the constraint Jacobian. The resulting method avoids resolution of the nonlinear behavior for each design iterate. Three variants of the method are developed which require the solution of either two or three linear systems involving the stiffness matrix of the discrete boundary value problem. The method is used to solve an aerodynamic design problem governed by nonlinear potential flow. Numerical results demonstrate a substantial performance improvement.

Chapter 1

Introduction

In recent years there has been growing interest in developing the capability for computing optimal designs of engineering systems governed by nonlinear partial differential equations. For example, aircraft shape design can involve coupling compressible flow about a complex body with large elastic deformation of the body. Another example is viscous drag reduction by either shape or velocity control, which requires the solution of the Navier-Stokes equations as a subproblem within optimization.

The optimization problems we have in mind consist of an objective function reflecting design goals, and constraints which include both a nonlinear boundary value problem governing system behavior, and additional design constraints. We classify variables according to two types: *design* variables which describe geometry (e.g. shape, thickness), and *state* variables which describe system behavior for a fixed geometry (e.g. pressure, velocity, stress).

A standard approach to such problems is to view the state variables as implicitly depending on the design variables (see, e.g. [5]). Solution of the governing equations at each optimization iteration reduces both the number of constraints and variables, as the state variables are no longer considered optimization variables. We call this approach *path-following,* since the iterates follow a trajectory characterized by solution of (a discretized form of) the governing equations at each design iteration. In addition to this reduction in problem size, another advantage of path following methods is that they generate dense constraint Jacobian and Lagrangian Hessian matrices, so that standard projected Lagrangian methods can be used without consideration of sparsity. The disadvantage of such methods, however, is that they require full solution of the nonlinear equations governing system behavior at each design iteration. This can result in intractability for systems displaying complex or coupled behavior.

In contradistinction to these methods, *infeasible path* methods regard the discrete form of the governing PDE's as equality constraints in the optimization problem, and include the state variables as optimization variables. Projected Lagrangian optimization techniques (such as Sequential Quadratic Programming) then require zero and first order information about the governing equations at only a single point to define the optimization subproblem.

Full solution of the PDE's is thereby avoided. The difficulty with such an approach is that the constraint Jacobian and Lagrangian Hessian matrices are larger and sparse [8]. Use of a general-purpose sparse optimizer, such as MINOS [6], is problematic: the favorable structure of the constraint Jacobian with respect to the state variables (i.e. the "tangent stiffness matrix") cannot be exploited. In this paper we give a new SQP method based on infeasible path ideas for a class of optimal design problems constrained by a discrete nonlinear boundary value problem (BVP). The method is based on a particular null space representation which exploits sparsity of the constraint Jacobian. Several variants of our method result. The first requires solution of two linear systems involving the same coefficient matrix, the tangent stiffness matrix. The second requires the solution of two systems as well, but with stiffness matrices evaluated at different points. The third requires solution of three linear systems, two of which have the same stiffness matrix. In any case, resolution of nonlinear behavior is avoided at each iteration: the method simultaneously finds an optimum design while converging the nonlinear behavior. Furthermore, we update and store only the projected Hessian matrix, resulting in storage requirements equal to those of a path-following method. We refer to our method as the Coordinate Basis Infeasible Path method (CBIP). Its three variants are described in chapter 2.

To demonstrate our proposed method and examine its performance, we apply it to the solution of a shape optimization problem in aerodynamic design, which is defined in chapter 3. The performance of the three variants are compared to a standard path-following approach for a series of design problems in chapter 4, and conclusions are drawn in chapter 5.

Chapter 2

The Coordinate Basis Infeasible Path method

In this chapter we present an optimization strategy tailored to physical systems governed by nonlinear boundary value problems. We assume that numerical approximation of the governing equations gives rise to large, sparse coefficient matrices, such as those produced by finite difference, finite element, or finite volume methods. We limit our discussion to problems that are constrained only by these equations. Extensions to problems that include additional constraints on the parameters can be generalized. The optimal design problem is to find values of parameters of a system that minimize a desired objective.

While in some cases quadraticaHy convergent Gauss Newton methods can be devised for nonlinear least squares problems, we target both large residual problems and more general objective functions. Given the success of Sequential Quadratic Programming (SQP) methods for general problems (e.g. [9]), it is natural to seek an SQP strategy tailored to design problems with discrete BVP constraints. Typically the number of design variables is much smaller than the number of state variables, and the full Hessian is sparse, indefinite, and of order of the total number of variables. It is therefore advantageous to seek a strategy which updates the projected Hessian matrix, which is positive definite at the optimum, and of the order of the design variables. Because of the special structure of the constraint Jacobian, we seek a corresponding basis for its null space that exploits this structure. We begin with a canonical form of the optimization problem:

$$\begin{aligned} &\text{minimize } /(x) \\ &\text{subject to } h(x) = 0 \\ &/ : 3\,T \to \gg, \quad h : f\,t^{n} \to \gg^{m}, \quad x \in \gg^{n} \end{aligned} \tag{2.1}$$

Here, / represents the design objective, and h is a system of nonlinear algebraic equations arising from discretization of the governing BVP. The *n* variables x consist of the m state variables u and *n* — *m* design variables b. Typically *m*^> *n* — m, since a large number of state variables arise from discretization of the domain, but the design is described by a small

number of parameters (especially in shape optimization). The path-following approach to 2.1 eliminates the constraints, thereby resulting in an unconstrained optimization problem. The penalty, of course, is the need to solve the nonlinear system h(x) = 0 at each design iteration.

The infeasible path approach regards as 2.1 as a nonlinearly-constrained optimization problem. SQP can be thought of as a Newton solution of the first order optimality conditions. A Newton step defines the following quadratic program (QP):

$$\text{minimize } p\pounds g_{fc} + \frac{1}{2} plG_L p_k \tag{2.2}$$

$$\text{subject to } AkPk = -hfc \tag{2.3}$$

where g is the gradient of the objective function, GL is the Hessian matrix of the Lagrangian function, A_k is the Jacobian of the discrete PDE's, p*, is the search direction, and the subscript *k* indicates evaluation at x^.

For clarity we drop the subscript fc; it is understood that all quantities depending on x are evaluted at x^. Let us decompose p_k into two components:

$$p = Zp^* + Yp_y \tag{2.4}$$

in which $Z \pounds .sft^{nX(n\sim m)}$ is a matrix whose columns form a basis for the null space of A, and $Y\ 6\ 3ft^{nXm}$ is chosen so that the partitioned matrix Q = [Z Y] is nonsingular. Often, Y is chosen so that its columns span the range space of A^T. The y-space step is completely determined by substituting (2.4) into (2.3), resulting in the m x *m* system:

$$AYp_y = -h \tag{2.5}$$

The null space move is found by substituting (2.4) into (2.2) and minimizing with respect top^:

$$Z^T G_L Z_{Pz} = -Z^T(g + G_L Y_{Py}) \tag{2.6}$$

The (n - *m)* x (n — m) projected Hessian matrix Z^TGx,Z is dense but of the order of the design variables, and is naturally approximated by a Quasi-Newton update. A feasible-path method would require this storage as well, since the optimization variables in that case are just the design variables. On the other hand, the "long and thin" matrix $Z^T G L Y$ would increase storage requirments considerably over a path-following approach. Our approach is to ignore this term; Nocedal and Overton have shown that the resulting algorithm exhibits two-step Q-superlinear convergence (when orthonormal bases are used for Y and Z) [7], in the sense that

$$\frac{\text{''^''f'Uoas^oc}}{\|x^* - x^*\|} \tag{2.7}$$

The critical step is the definition of appropriate *y*- and null space bases. Since A is large and sparse, a standard QR factorization is unacceptable. To examine the structure of the constraint Jacobian, let us consider a partitioning of state and design variables:

$$x^r = [u^T, b^T] \tag{2.8}$$

in which u 6 3ft^m and b £ &$^{n-m}$. The partitioned constraint Jacobian becomes

$$A = [K, \frac{\partial f}{\partial b}] \qquad (»)$$

in which we have identified the Jacobian of the discrete PDE's with respect to the state variables as the tangent stiffness matrix K (in deference to finite element terminology). It is desirable to exploit the inverse of K, since the solution of linear systems with K as a coefficient matrix is a well-studied problem and there exist many direct and iterative methods that exploit its structure. We can define a matrix Z whose columns are orthogonal to the rows of A as:

$$Z = \begin{bmatrix} -K^{-1} \frac{\partial f}{\partial b} \\ I \end{bmatrix} \qquad (2.10)$$

Here, we write K^{-x} formally; we shall however see that its inverse is not required, and the computations can be arranged in such a way that solution of only two or three linear systems involving K is required, using any direct or iterative technique. The y-space basis is defined simply as

$$v-[;] \qquad (2.H)$$

We refer to this choice as a *coordinate* basis. Clearly, the matrix

$$Q = [\, Z \;\; Y \,] \qquad (2.12)$$

is nonsingular provided K is nonsingular, and hence Z and Y form a basis for 9ft^n. The invertibility of K is established by the well-posedness of the boundary value problem. The resulting t/-space step for p_y from (2.5) becomes:

$$Kp_y = -h \qquad (2.13)$$

We observe that this is simply a Newton step for the nonlinear system h = 0. Using the null space definition (2.10), the null space move p_z can be found from

$$B^*p^* = -g, \qquad (2.14)$$

where

$$g_z \equiv Z^T g = -|£^T K-^T g_u + g_6 \qquad (2.15)$$

Here B_2 represents a Quasi-Newton approximation to the projected Hessian, and g_u and g& represent objective gradients with respect to the state and design variables, respectively. Note the close connection between the expression for the projected gradient (2.15), and the gradient of the objective using a path-following method (in conjunction with implicit sensitivity expressions; see e.g. [5]). The difference, of course, is that in (2.15) K need not be evaluated at the u for which h = 0, in contrast to path following methods, which

converge the behavioral equations at each optimization iteration. Using (2.4), the moves in the state and design variables take the form

$$p \wedge - K T^{1} \frac{\partial \mathbf{n}}{\partial \mathbf{b}} p_M + p_y \tag{2.16}$$

$$P6 = P^* \tag{2.17}$$

The recipe for the update of the state variables (2.16) can be interpreted as being comprised of two components. The first term gives a first-order approximation of the change in state variables due to a change in system parameters p&; the second term is the change that would occur if the design were held constant, and the state variables were updated according to a Newton step.

Chapter 3

Algorithms

Depending on the specific way in which the updates of the state and design variables are carried out, it is possible to identify several variants of the methods described in the previous chapter. What follows is a definition of algorithms corresponding to each of these variants. Symmetry of K is assumed in all cases.

PFF : This is the path-following approach. A full Newton solution of the nonlinear BVP is performed at each optimization iteration. The initial guess for the state variables at the beginning of each analysis is taken as zero.

PFP : This is a variant of the path-following approach. The Newton solve of the BVP is initiated with the solution of the previous design.

CBIPS : This is the infeasible path method that results from the direct realization of the SQP method of the previous chapter in conjunction with a coordinate basis for the y—space move. It requires one stiffness matrix factorization and two sets of triangular solves per optimization iteration.

CBIPM : This is a variant of algorithm CBIPS in which information about the design variables is used to evaluate the stiffness matrix which in turn is used to update the state variables. It requires two stiffness matrix factorizations and two sets of triangular solves per optimization iteration.

CBIPC : This algorithm results from applying the Coleman-Conn method [3] (closely related to SQP) to the CBIP method. It requires one factorization and three triangular solves per optimization iteration.

3.1 Algorithm PFF

- Set * = 0, H_z° = I, b = b°,
- Solve h(u, b°) = 0 for u°, using u = 0 as the initial guess.

- Solve K°A° = $g°_u$
- **Find $g°_z$ = -(f6^T)°A° + g?**
- Set κ = 0.0 and a^0 = 1.0.
- While ||g£|| > € and k < maxiter do:
  1. Find p^k_z = -1$\mathbf{H}^k_z g^k_z$
  2. Set a^k = $\frac{\mu^k}{1+\kappa*\|g^k_z\|}$
  3. Update design variables: b^{*+1} = b^k + $a^k p^k_z$
  4. Solve h(u,b$^{A;+1}$) = 0 for u^{fc+1}, using **u** = 0 as the initial guess.
  5. Check strong Wolfe conditions: if conditions are satisfied goto 6 else increment K and go back to step 2.
  6. Solve K*+^{1}A*$^{+1}$=gJ+^{1}
  7. Find g^^{1} = -(f^T)/:+iA*+i $_{+g}$*$_b$+i
  8. Find y^k_2 = g$^{*+x}_z$ - g^k_z
  9. Update H£ using y^k_z and p^k_z
  10. Set q^k_z = g^^{1}
  11. Set fc = fc + l
- Endwhile

3.2 Algorithm PFP

- Set k = 0, H$^°_z$ = I, **b** = b°,
- Solve **h(u,b°)** = **0** for u°, using **u** = 0 as the initial guess.
- Solve K°A° = $\underline{si}$
- **Find $\mathbf{g^0_z = -(\frac{\partial h}{\partial b}^T)^0 \lambda^0 + g^0_b}$**
- Set κ = 0.0 and a^0 = 1.0.
- While \\g^k_z\\ > e and k < maxiter do:
  1. Find p^k = -$H^k_z g^k_z$
  2. **Set α^k = $\frac{\alpha^k}{1+\kappa*\|g^k_z\|}$**
  3. Update design variables: b^{fc+1} = h^k + $a^k p^k_z$
  4. Solve h(u,b^{fc+1}) = 0 for u^{fc+1}, using **u** = u^{fc} as the initial guess.

5. **Check strong Wolfe conditions: if conditions are satisfied goto 6 else increment *K* and go back to step 2.**
6. **Solve** $K^{*+1}A^{*+1} = g_u^{*+1}$
7. **Find** $g \wedge i \wedge -(\frac{\partial h}{\partial b}^T)^{k+1}\lambda^{k+1} + g_b^{k+1}$
8. **Find** $y_z^* = g_z^{*+1} - g_z^*$
9. **Update** H_z^* **using y£ and** p_z^*
10. **Set** $g_z^* = g_z^{*+x}$
11. **Set** k **= fc + 1**

- **Endwhile**

Notice that algorithm PFP differs from Algorithm PFF only in Step 4, in which the initial guess for the nonlinear system solve is taken as u* rather than 0.

3.3 Algorithm CBIPS

- **Set** k **= 0,** H_z° **= I, u = u°, b = b°,**
- **Solve K°A° = g2**
- **Find** $g_z^O = -(|\frac{h}{b}^T\}o_A o + g_b^0$
- **Set** κ **= 0.0 and** a^0 **= 1.0.**
- **While** $||g_z^*||$ **> e and** k **< maxiter do:**

1. **Find** $p_z^* = -H_z^* g_z^*$
2. **Set** $\alpha^k = \frac{a^k}{i + \kappa * ||g_z^k||}$
3. Find $d^h = \S\pounds V\, p_z^* + h^{fc}$
4. **Solve** $K^* p\pounds = -d^k$
5. **Update variables:** $u^{fe+1} = u^* + p\pounds$, $b^{*+1} = b^* + a^k p^k{}_z$
6. **Check strong Wolfe conditions: if conditions are satisfied goto 7 else increment *K* and go back to step 2.**
7. **Solve** $K^{k+l}X^{k+1} = g\pounds^{+1}$
8. **Find** $g_z^{**1} = -(ffcV^1 A^{**1} + g_6^{fc+1}$
9. **Find** $y\pounds = g_z^{*+*} - g_z^k$
10. **Update** H_z^* **using y^ and p^**
11. **Set** $g_z^* = g \wedge^1$
12. **Set** $k = k + 1$

- **Endwhile**

3.4 Algorithm CBIPM

- Set $k = 0$, $H_z^\circ = I$, u = u°, b = b°,
- Solve K°A° = g_u°
- **Find $g_z^\circ = -(t^T)^\circ A^\circ + g_b^\circ$**
- Set κ = 0.0 and a* = 1.0.
- While ||g£|| > € and k < maxiter do:
  1. Find p£ = -H*$_z$g£
  2. Set $a^k = \frac{\alpha^k}{1+\kappa*\|g_z^k\|}$
  3. Update the design variables: b^{*+1} = b^ + $\&^k P\mathfrak{L}$
  4. Write d* = h(b^{*+1},u*)
  5. Solve K(b^{fc+1},u*)p_u^* = -d*
  6. Update state variables: $u^{fc+1} = u^k$ + p^
  7. Check strong Wolfe conditions: if conditions are satisfied goto 8 else increment K and go back to step 2.
  8. Solve $K^{fc+1}\lambda^{k+1} = g_u^{k+1}$
  9. Find $g_z^{*+1} = -(\frac{\partial h}{\partial b}^T)^{k+1}\lambda^{k+1} + g_b^{k+1}$
  10. Find y_z^* = g£$^{+1}$ - g^
  11. Update H£ using $y^k{}_2$ and p^
  12. Set g_z^* = g^$^{+1}$
  13. Set $k = k + 1$
- Endwhile

The only difference between CBIPS and CBIPM is in Step 3, in which the latest information about the design variables is used to evaluate the stiffness matrix. The development of this method is motivated by the desire to make CBIPM and PF* equivalent for *linear* BVP's. The numerical results will demonstrate that this reduces the number of optimization iterations at the expense of one additional stiffness matrix factorization per optimization iteration.

3.5 Algorithm CBIPC

- Set $k = 0$, $H_{\mathfrak{L}} = I$, u = u°, b = b°,
- Solve K°A° = *zl*

- Find $g_z^0 = -(\frac{\partial h}{\partial b}^T)^0 \lambda^0 + g_b^0$
- Set AC = 0.0 and a^0 = 1.0.
- While $||g_z^*|| > \epsilon$ and $k <$ maxiter do:

  1. Find $p_z^* = -H_z^* g_z^k$
  2. Set $\alpha^k = \frac{\alpha^k}{1+\kappa*||g_z^k||}$
  3. Update design variables: b^{k+1} = b* + a^p$_z^*$
  4. Find d_2^{fc} = ftVp$_z^*$
  5. Solve $K^{fc}p\wedge_2 = -d_z^*$
  6. Solve $K^{fc}p\wedge = -h(u^* + p\wedge.b\wedge^1)$
  7. Update states variables: $u^{k+1} = u^k + (p_{u2}^* + p_{uy}^*)$
  8. Check strong Wolfe conditions: if conditions are satisfied goto 9 else increment *K* and go back to step 2.
  9. Solve $K^{*+1}A^{fc+1} = g£^{+1}$
  10. Find $g\wedge^1 = -(\S b^T)^{*+1}A^{fc+1} + g£^{+1}$
  11. Find $y_z^* = g\wedge^1 - g_z^*$
  12. Update H_2^k using y_z^* and p_z^*
  13. Set g£ = $g^{k}{}_{2}{}^{+1}$
  14. Set *k* = fc + 1

- Endwhile

The Coleman-Conn method (CBIPC) differs from the other two infeasible path algorithms in its requirement of an extra constraint evaluation (Step 4). This results in an extra triangular solve, because it does not permit combination of righthand sides.

Chapter 4

An aerodynamic design problem

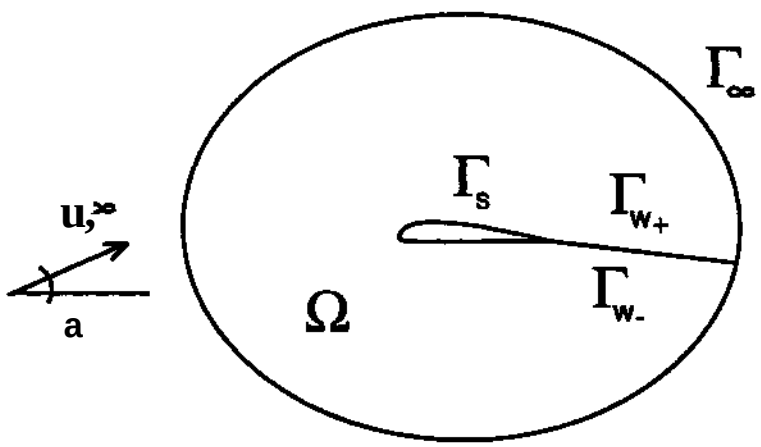

Figure 4.1: Two dimensional domain for airfoil analysis problem

We define in this chapter a design problem of a system governed by a nonlinear partial differential equation. The problem we consider is to find the shape of the airfoil that exhibits a prescribed pressure distribution. Such problems are of importance to the aerospace industry, and there is growing interest in developing an optimal design capability for a complete aircraft [4], The flow around the airfoil is modeled by the nonlinear full-potential equations. Though parameterized by a small number of variables and only two-dimensional, the problem captures the salient features of more general problems, which increase in difficulty as dimensionality, number of system parameters, nonlinearity, and coupling among the physical processes increase.

Let ft represent the domain of definition, T_s the surface of the airfoil, T_w a split boundary intended to model the wake, Too the farfield boundary, Uoo the freestream velocity, and *a* the angle of attack of the airfoil.

The *analysis* problem is to find the pressure distribution over $T_\$$ for subsonic flows with Mach numbers above 0.4, so that compressibility effects cannot be neglected. The equation

of continuity V • (/>u) = 0 and boundary conditions take the form:

$$V \cdot \{[1 - lZ_{\frac{1}{2}}^{\cdot 1}(v^{\wedge})^2]^{\cdot} F^{\frac{1}{1}} T\ V\langle\pounds\} = 0 \text{ in } \quad \text{fi} \tag{4.1}$$

$$V\langle f\rangle \cdot n = 0 \text{ on } T_s \tag{4.2}$$

$$V\langle f\rangle \cdot n = UQO \cdot n \text{ on } Too \tag{4.3}$$

in which $\langle f\rangle$ is the velocity potential, 7 is a constant and $p^{\wedge}$ is the freestream density.

The shape optimization problem of finding parameters describing a shape that induces a desired pressure distribution can be stated as:
minimize:

$$f(\Gamma_s, \Phi) = \int_{\Gamma_s} [p(\Phi) - p^*]^2 \ d\Gamma \tag{4.4}$$

subject to:

$$\int_{JQI_h} [1 - \frac{\gamma - 1}{2}(\Phi^T\ Q\ \Phi)]^{\frac{1}{\gamma - 1}}\ Q\ \Phi\ d\Omega$$

$$-L_{\cdot\infty}\ N^{\wedge}ooUoo)\text{-}!!\ dT \tag{4.5}$$

where $p(\&)$ is the predicted pressure on the airfoil and p* is the prescribed pressure. The constraint (4) reflects the Galerkin form of the BVP, in which the finite element interpolation $(f)_h = N^r 4\rangle$ has been introduced. Here, N represents a vector of global basis functions and $4\rangle$ are nodal potentials, and $Q = VN\ (VN)^T$. The boundary conditions and additional constraints on the state variables are implied in this nonlinear algebraic system. This problem is of the form (2.1).

The Jacobian (with respect to state variables) of the conservation equations (4 is the stiffness matrix, and is obtained by differentiating the left hand side with respect to $4\rangle$:

$$K(\$) = \int_{Jn_h} [1 - {}^{\wedge}\text{-}{}^{\wedge}\frac{1}{2}(\$^T Q\ \$)]T^{\frac{1}{1}} T\ Q$$

$$-[1 - \frac{\gamma - 1}{y}(\Phi^T Q\ \Phi)]^{\frac{2}{\gamma}} * Q\ \$\ *^T Q\ dtt \tag{4.6}$$

This is an *n* x *n* symmetric matrix that can be constructed using standard finite element ideas. It is positive definite for subsonic flow, and indefinite for transonic flow [2].

Chapter 5

Numerical results

In this chapter we compare the performance of the five algorithms for the aerodynamic design problem defined in the previous chapter. We do this first for NACA airfoils. The design variables are then the basic parameters of the NACA family of airfoils, namely, the maximum thickness r, the position of the maximum camber p, and the maximum camber *e* [1]. These parameters are illustrated in Figure 5.2. To allow for more than three design variables a second parameterization based on cubic splines is used to describe the shape of the airfoil in the second set of examples. This parameterization is shown in Figure 5.5. The leading edge of the airfoil is modeled with a circular arc of radius r_0. This arc extends 011 and *821* above and below a line that is inclined *OQ* with respect to the horizontal axis. All these quantities can have any positive value and are therfore taken as design variables. For the purpose of the FE discretization the arc portions corresponding to *9* and #21 are divided into *n* and T121 subintervals. The upper portion of the airfoil is divided into *rt\2* intervals. For purposes of the FE discretization each of these intervals is divided into no subintervals. 7112 — *ly* coordinates are then used to fit a cubic spline between the end of the arc and the trailing edge. These *y* coordinates are also taken as design variables. The lower portion of the airfoil is modeled in an analogous way. Angles *O12* and *622* at the trailing edge complete the list of design variables. The total number of design variables with this modeled is thus 7112 + *TI22* + 4.

The algorithms were implemented on an IBM RS6000-320H workstation. The linear systems involving K were solved using MA-28, the general direct sparse solver available from the Harwell library.

Table 5.1: Summary of Numerical Results - NACA family

PFP							
Moo	0.40	0.45	0.50	0.55	0.60	0.65	0.68
f	0.5440D-12	0.1302D-11	0.9653D-14	0.2739D-12	0.3576D-13	0.2133D-12	0.9593D-14
‖g z‖	0.5043D-05	0.8289D-05	0.1065D-05	0.5042D-05	0.2227D-05	0.6399D-05	0.8375D-06
CPU time	964	975	1051	1258	1252	1385	2865
Iter.	10	10	11	11	11	12	13
time/iter.	96.4	97.5	95.5	114.4	113.8	115.4	220.4

PFP							
Moo	0.40	0.45	0.50	0.55	0.60	0.65	0.68
f	0.4767D-09	0.1771D-09	0.3997D-08	0.3091D-09	0.2129D-09	0.3074-07	0.8996D-11
‖g z‖	0.8482D-04	0.4235D-04	0.4678D-04	0.9004D-04	0.5074D-04	0.1064D-04	0.1483D-04
CPU time	724	790	775	801	822	961	1742
Iter.	9	10	10	10	10	11	11
time/iter.	80.4	79.0	77.5	80.1	82.2	87.4	158.4

CBIPS							
Moo	0.40	0.45	0.50	0.55	0.60	0.65	0.68
f	0.6672D-11	0.1642D-09	0.4018D-08	0.4230D-12	0.1234D-10	0.3074D-07	0.1778D-09
‖g z‖	0.9446D-05	0.1216D-05	0.1170D-05	0.3890D-05	0.3781D-05	0.7041D-06	.6362D-04
CPU time	644	685	691	667	668	735	1224
Iter.	15	16	16	15	15	17	24
time/iter.	42.9	42.8	43.2	44.5	44.5	43.2	51.0

CBIPM							
Moo	0.40	0.45	0.50	0.55	0.60	0.65	0.68
f	0.5063D-11	0.1641D-09	0.4018D-08	0.2323D-11	0.1223D-10	0.3074D-07	0.8677D-10
‖g z‖	0.1016D-05	0.2225D-06	0.1343D-05	0.3731D-05	0.1106D-05	0.9497D-05	0.5191D-04
CPU time	637	639	639	699	703	997	1153
Iter.	10	10	10	11	11	16	18
time/iter.	63.7	63.9	63.9	63.5	63.9	62.3	64.1

CBIPC							
Moo	0.40	0.45	0.50	0.55	0.60	0.65	0.68
f	0.5067D-11	0.1642D-09	0.4019D-08	0.4291D-12	0.1331D-10	0.3073D-07	0.4326D-10
‖g z‖	0.1292D-05	0.1708D-05	0.4766D-05	0.5013D-05	0.8745D-05	0.2116D-05	0.2515D-04
CPU time	527	526	528	538	543	571	907
Iter.	10	10	11	11	11	12	15
time/iter.	47.9	47.8	48.0	48.9	49.4	47.6	60.5

5.1 NACA family

In these examples, the target pressure distribution was taken as that of a NACA 2412 airfoil at an angle of attack of one degree and at the corresponding Mach number. The parameters of the NACA 2412 are r = 0.12, p = 0.40, e = 0.02. In Figure 5.3 we show the Mach number distribution for the target airfoil at *Moo* = 0.68 and angle of attack = 1.0 degrees. The corresponding pressure distribution is shown in Figure 5.4 (actually, the figure shows the pressure coefficient defined as $C_v = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty u_\infty}$).

Figure 5.1 shows the mesh topology used in our numerical simulations.

The fixed data used for these examples are as follows:

- Target pressure distribution: NACA 2412 at a = 1.0 (Variable Mach number).
- Initial values of parameters: r = 0.07, p = 0.30, e = 0.00
- Number of finite elements: 3,280
- Number of nodes: 1,743

Numerical results are summarized in Table 5.1. CPU times for the different algorithms are plotted against Mach numbers in Figure 5.7. It is clearly seen from this graph that

Table 5.2: Summary of Numerical Results - Splines Model

PPP							
Opt. Vars.	10	12	16	22	28	40	90
/	0.2549D-02	0.1319D-02	0.3859D-03	0.1660D-03	0.1052-03	0.5141D-04	0.2377D-07
‖gz‖	0.1179D-03	0.3655D-03	0.4002D-03	0.9426D-03	0.4934D-03	0.6625D-03	0.8192D-03
CPU time	5317	5734	6927	7979	10328	19482	124,889
Iter.	33	37	39	33	42	63	99
time/iter.	161.1	155.0	177.6	241.78	245.9	309.2	1261.5
CBIPM							
Opt. Vars.	10	12	16	22	28	40	90
/	0.2549D-02	0.1363D-02	0.3866D-03	0.1567D-03	0.1052D-03	0.7100D-04	0.9260D-08
‖gz‖	0.2449D-03	0.9867D-03	0.7529D-03	0.5502D-03	0.3153D-03	0.5408D-03	0.4695D-03
CPU time	2791	1903	3281	4411	5020	8242	45422
Iter.	38	24	38	42	42	49	103
time/iter.	73.4	79.3	86.3	105.02	119.5	168.2	440.99

path-following methods are the most sensitive to the increase in nonlinearity induced by high Mach numbers. Cost per iteration, in CPU seconds, is shown in Figure 5.8. Again, as expected, the path-following methods are the most sensitive to the increase in the freestream Mach number. The cost per iteration for the infeasible path algorithms remains essentially constant with increasing nonlinearity, up to Mach number of about 0.65. For larger Mach numbers the physical solution to the flow problem becomes more and more difficult to get when a Newton solver is used. This means that the values of the potential $<f>$ is not very accurate, specially for very distorted shapes that arise in intermediate optimization steps. This results in poor values of the gradients and the search directions, thereby producing an increase in the number of iterations.

From the results presented in Table 5.1 and displayed in Figures 5.7 and 5.8 the following additional remarks can be made:

- With the exception of the CBIPS algorithm, all algorithms take essentially the same number of optimization iterations.
- The accuracy of the objective function value at optimality is best for the PFF. All other algorithms with the exception of the PFP exhibit essentially the same accuracy. This can clearly be observed in Figure 5.9.
- The worst accuracy is that of the PFP algorithm. In fact, if the tolerance in the norm of the projected gradient is lowered from $10^{\sim 4}$ in Table 5.1 to $10^{\sim 5}$, the algorithm oscillates about the solution, reaching it only by chance after a large number of iterations.
- The most efficient algorithm in terms of CPU time is the CBIPC algorithm. This algorithm also exhibits the least sensitivity to increases in the freestream Mach number (See Figure 5.7).
- The lowest cost per iteration is exhibited by algorithm CBIPS, as can be seen in Figure 5.8. However, this algorithm takes the largest number of iterations, and therefore its efficiency is diminished.

It is important to note that no formal line search was implemented for the cases presented here. From a limited number of tests made it was found that a line search was expensive and did not improve the convergence characteristics of the algorithms. However, it was sometimes necessary to limit the step length of the algorithms due to the fact that some iterates generated correspond to undefined values of K and h (specially for high Mach numbers). In addition, for the case with $M^{\wedge} = 0.68$, it was necessary to limit the step length to ensure a decrease in the objective function. In general, the step lengths were equal to one, except for the first few iterations.

The progress in the objective function when $M^{\wedge} = 0.65$ is presented in figure 5.9. The corresponding sequence of norms of the projected gradients is shown in Figure 5.10. The sequence of airfoil shapes generated by algorithm CBIPS for $M^{\wedge} = 0.65$ is displayed in Figure 5.11. The final shape is indistinguishable from the target.

5.2 Parameterization using the spline model

Figure 5.13 illustrates the initial conditions for a case using the splines model and $n\backslash 2 = 7122 = 9$ (22 design variables). The objective is again the NACA 2412 at $M^{\wedge} = 0.6$ and $a = 1.0$. The target pressure distribution and airfoil together with the optima found are shown in Figure 5.14. It can be seen that the two shapes are virtually indistinguishible from each other. The target conditions and optima for a case with $n\backslash 2 = 822 = 3$ (10 design variables) is shown in Figure 5.12. Not surprisingly, the agreement of the pressure distributions for the 22-variable case is much better than for the 10-variable case.

The number of optimization iterations against the number of design variables is depicted in Figure 5.15. The number of iterations increases with the number of design variables as can be expected. Again, it can be observed that the number of iterations is basically the same for the CBIPM and PFF algorithms. The cost per iteration, however, is much larger for the *path-following* method as can be seen in Figure 5.16.

The results of Figures 5.15 and 5.16 are summarized in Table 5.2.

The norm of the residual of the behavioral equations against the number of iterations has been plotted in Figure 5.17 for all algorithms. The convergence tolerance for the Newton solver was set to $10''''^{7}$. Although not necessary for the infeasible path algorithms, the starting point used here corresponds to a fully converged solution of the flow equations. This can be observed in the Figure by noting that the residual for all algorithms at iteration zero is $10''^{7}$. After that initial point, all infeasible path methods (that is, the CBIPS, CBIPM and CBIPC) depart considerably from strict feasibility, with a fully converged solution for the flow equations ocurring only at the last iteration. It can also be observed from Figure 5.17 that the path following algorithms (PFF and PFP) produce a residual that is lower than the tolerance for some intermediate iterations. This is due to the fact that even though the $10''^{7}$ value of the tolerance is acceptable for engineering purposes, the machine can provide a much higher accuracy for this particular problem. A problem with 28 design variables was used to produce the results shown in Figure 5.17.

All results presented in this section, with the exception of the 90-variable case correspond to an airfoil problem with $M^{\wedge} = 0.6$, $a = 1.0$, and the following discretization characteristics:

- Number of finite elements: 3,952
- Number of nodes : 2079
- Target pressure distribution: NACA 2412 at 1.0 degree angle of attack. $M^{\wedge} = 0.6$.

The results for the 90-variable case do not correspond to an airfoil problem but to a unit cilynder under the following conditions:

- Moo = 0.35.
- Target Pressure Distribution: Unit Cilynder (r = 0.50).
- Design variables: Radial distances to the surface of the Cilynder.
- Initial values of all design variables: 0.456
- Number of finite elements: 4,416
- Number of nodes: 2,325

In lieu of a line search, the step length was limited to avoid undefined values of K and h and to ensure a decrease in the objective function, in all cases presented here.

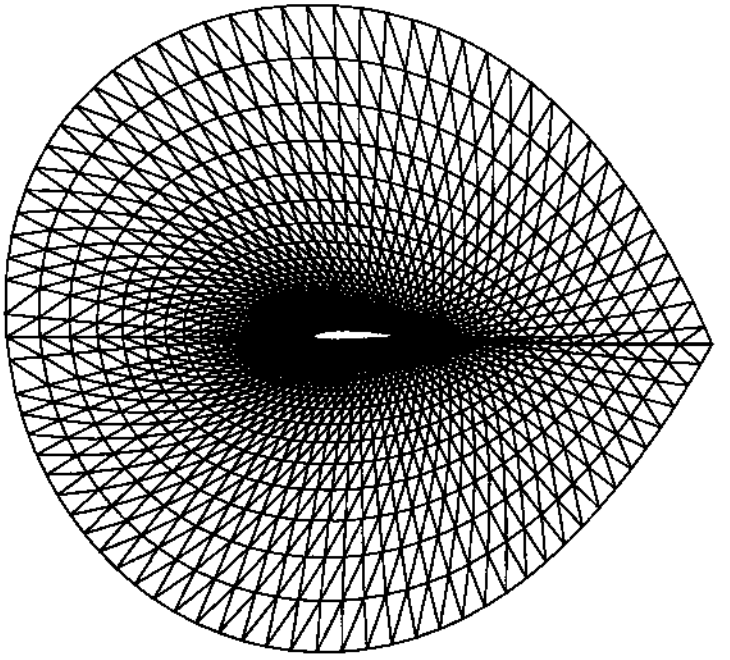

Figure 5.1: Discretized two dimensional domain with airfoil

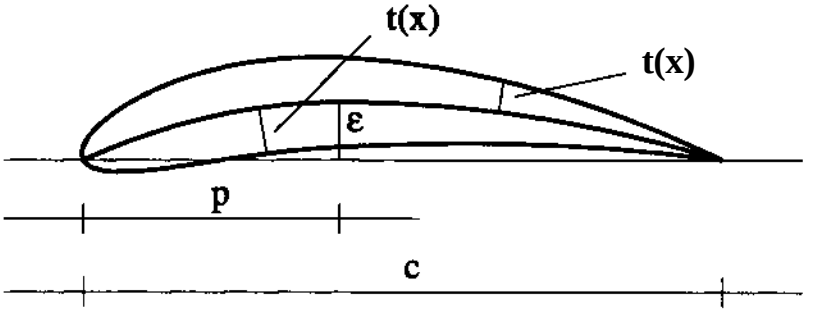

Figure 5.2: Four-digit NACA airfoil

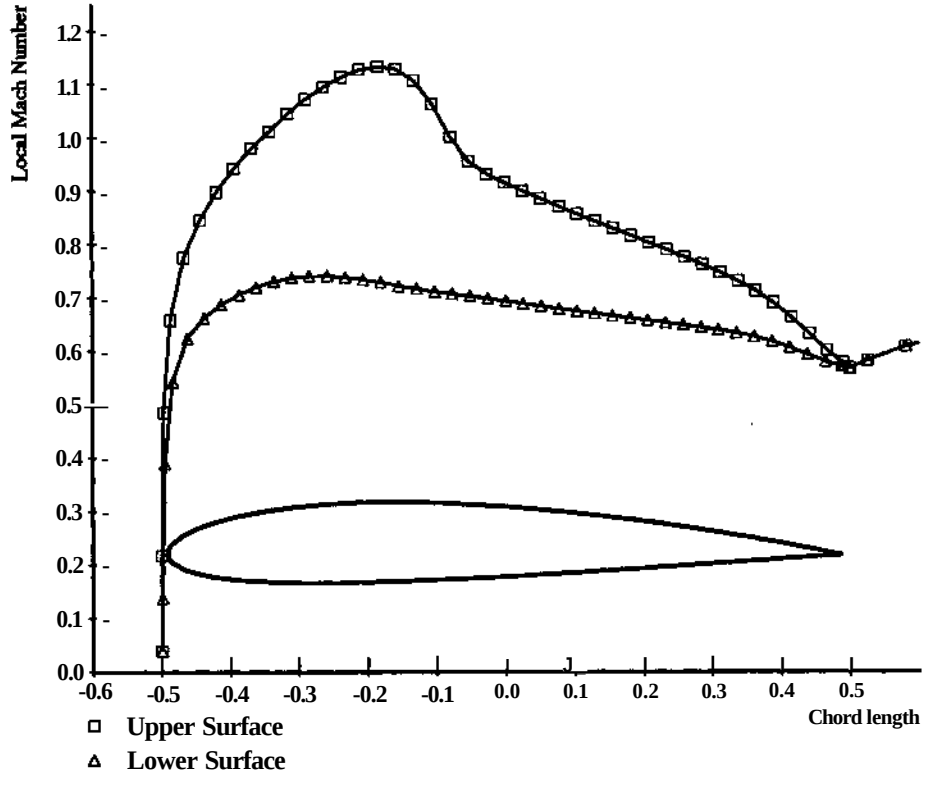

Figure 5.3: Mach Number Distribution : NACA 2412 at *a* = 1.0 and M^ = 0.68

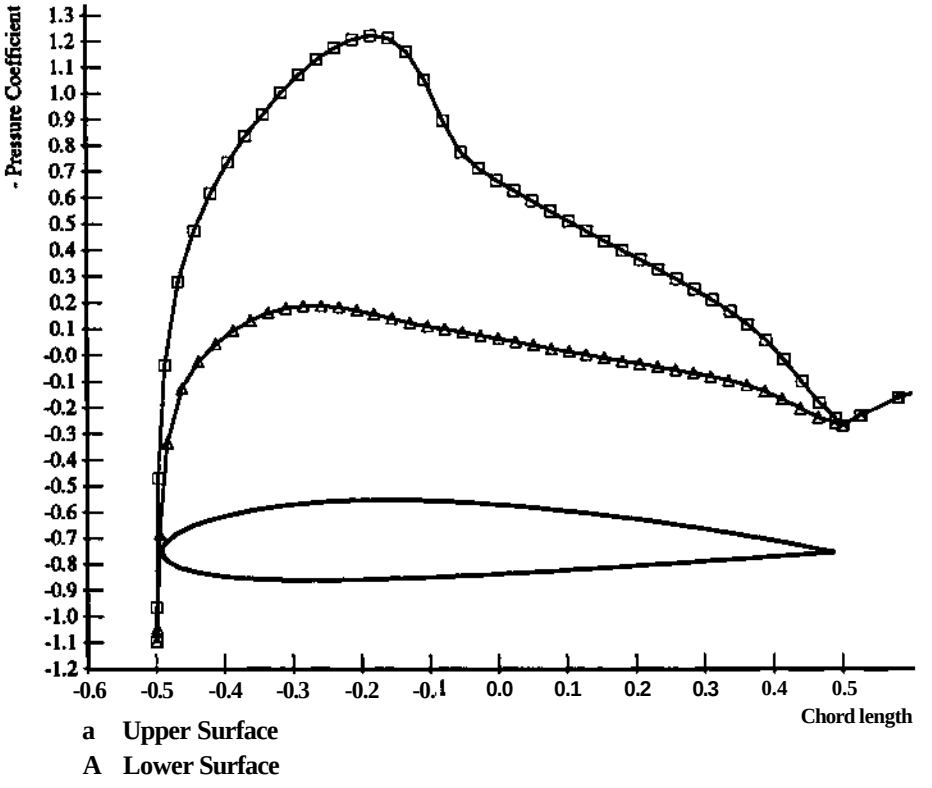

Figure 5.4: Pressure coefficient: NACA 2412 at a = 1.0 and $M^{\wedge}$ = 0.68

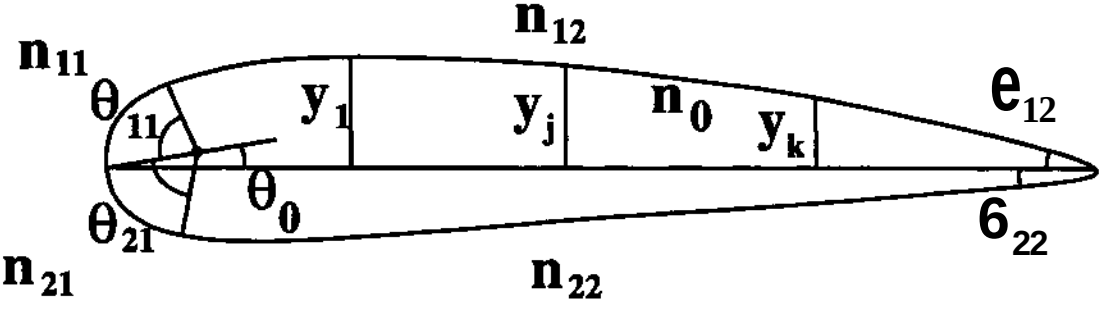

Figure 5.5: Parameterization of airfoil using splines.

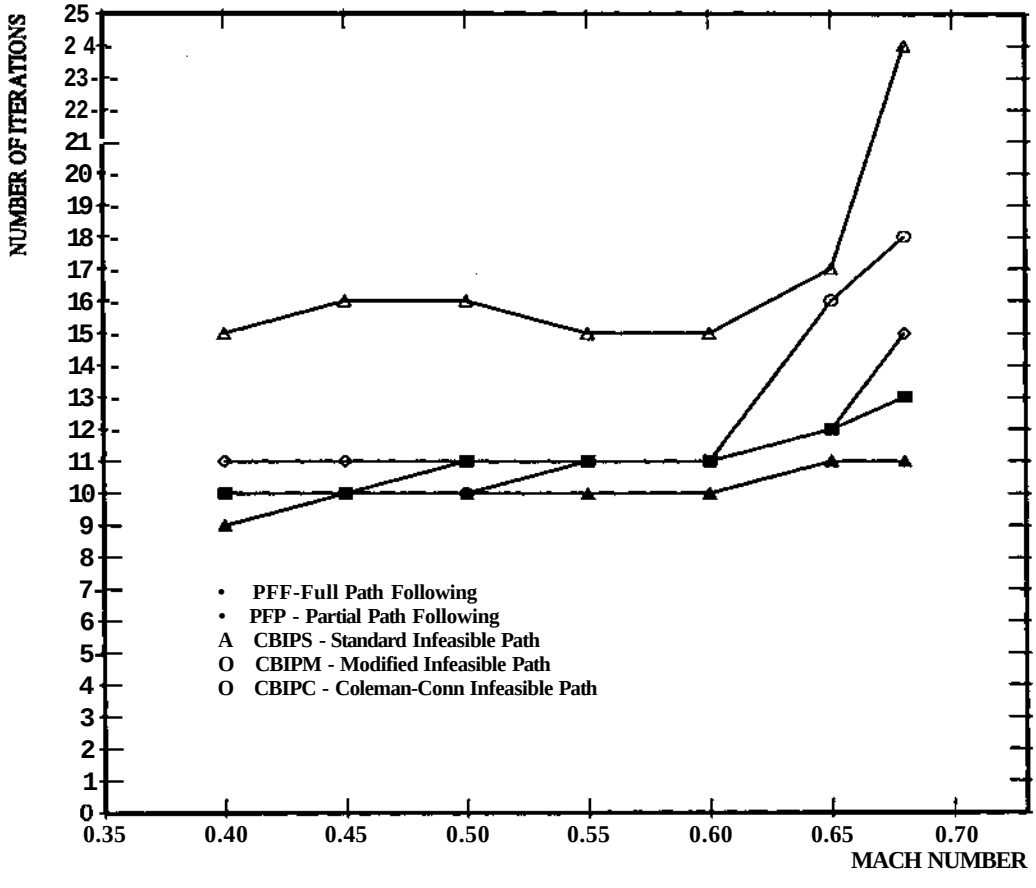

Figure 5.6: Number of Iterations vs. Mach number.

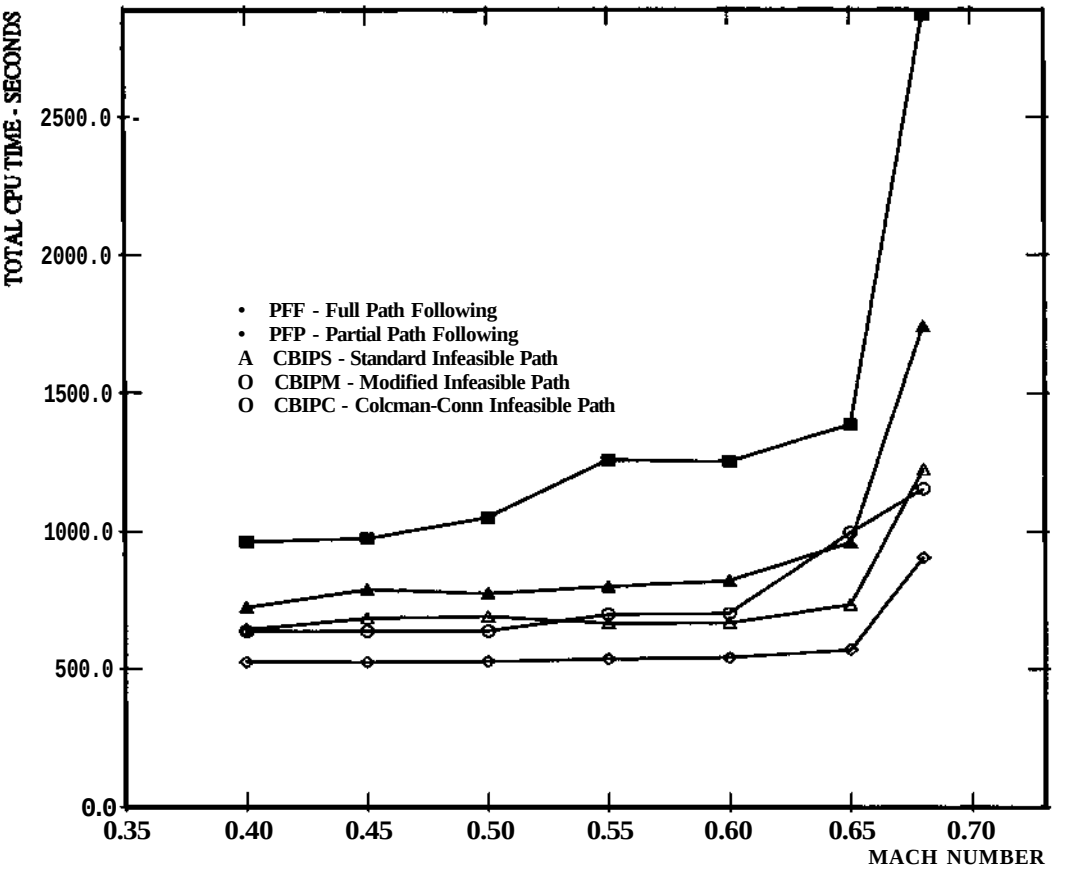

Figure 5.7: CPU time vs. Mach number.

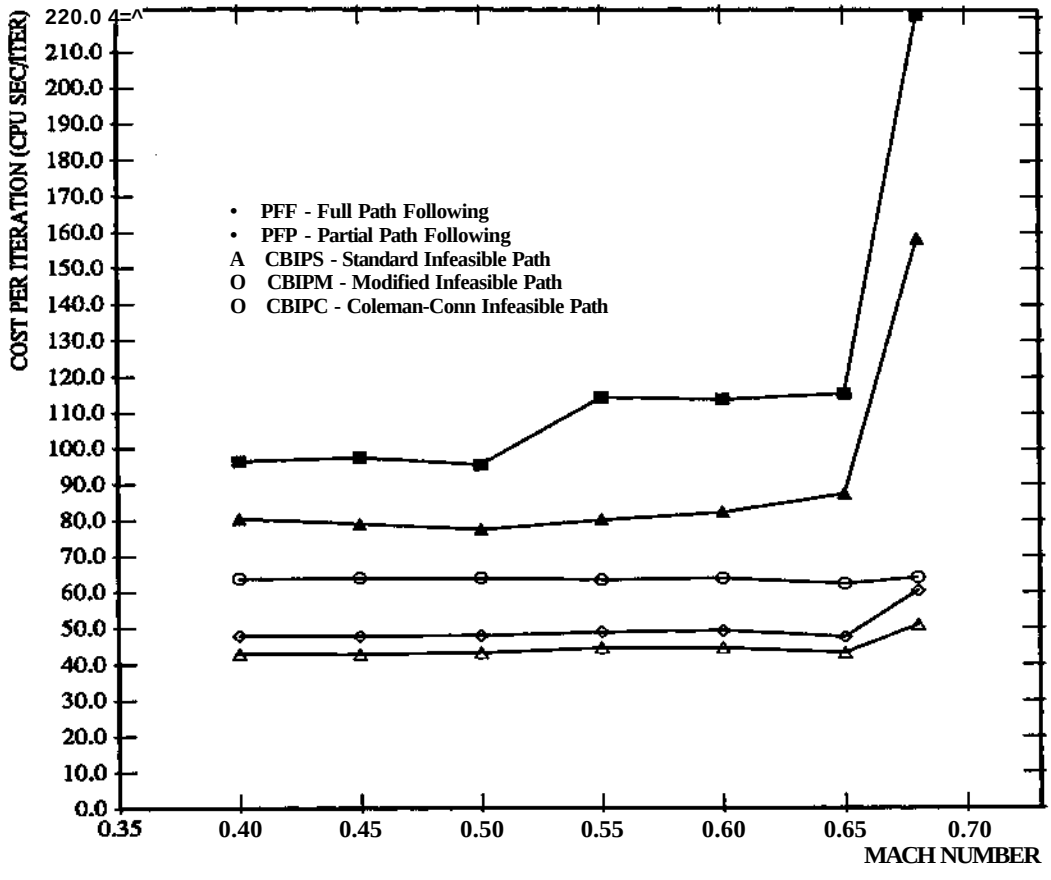

Figure 5.8: Cost per Iteration vs. Mach number.

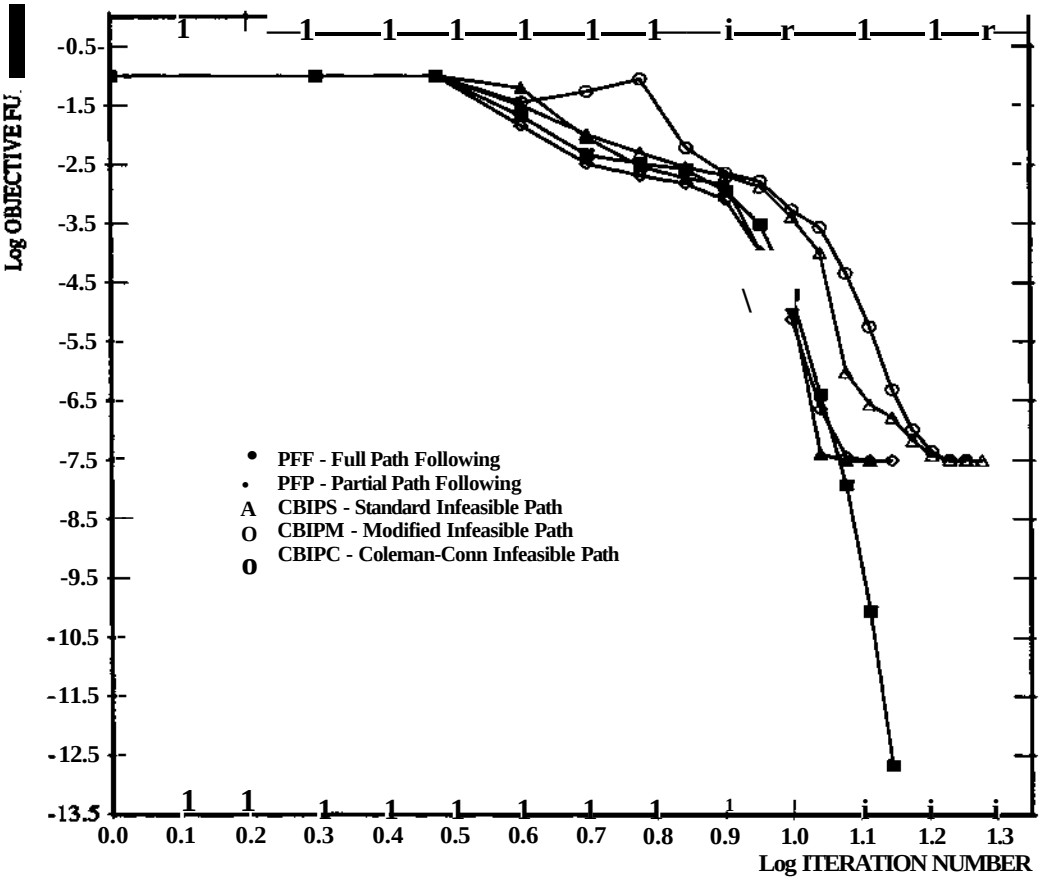

Figure 5.9: Progress of Objective Function. $M^{\wedge} = 0.65$.

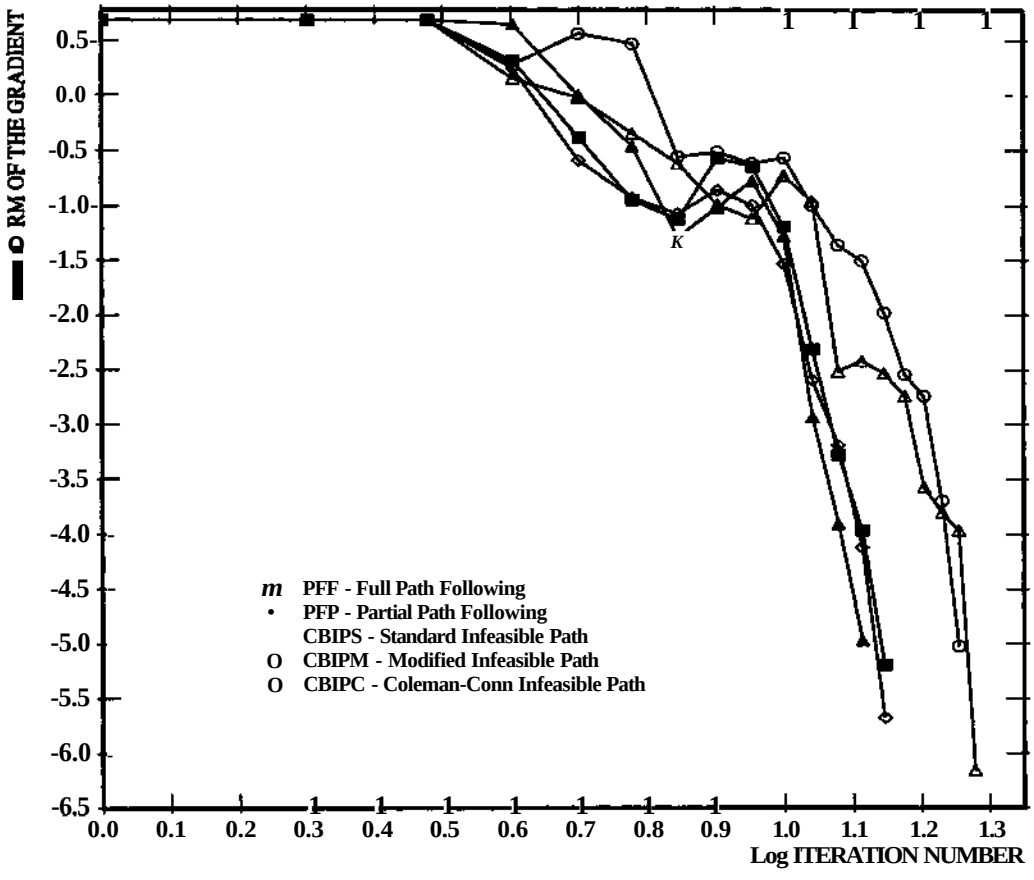

Figure 5.10: Sequence of norms of projected gradient. $M^{\wedge}$ = 0.65.

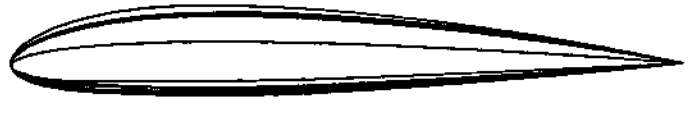

Figure 5.11: Sequence of airfoil shapes. CBIPS algorithm, $M^{\wedge}$ = 0.65.

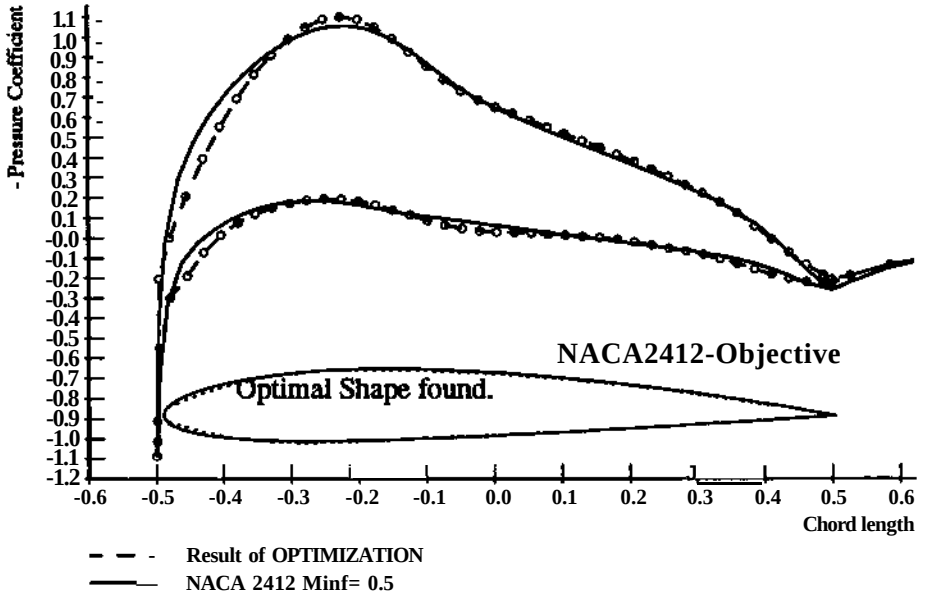

Figure 5.12: Comparison between Optimum and target conditions. NACA 2412 modeled using splines *(n\2* = ^22 = 3). *M^* = 0.5, *a* = 1.0 degrees.

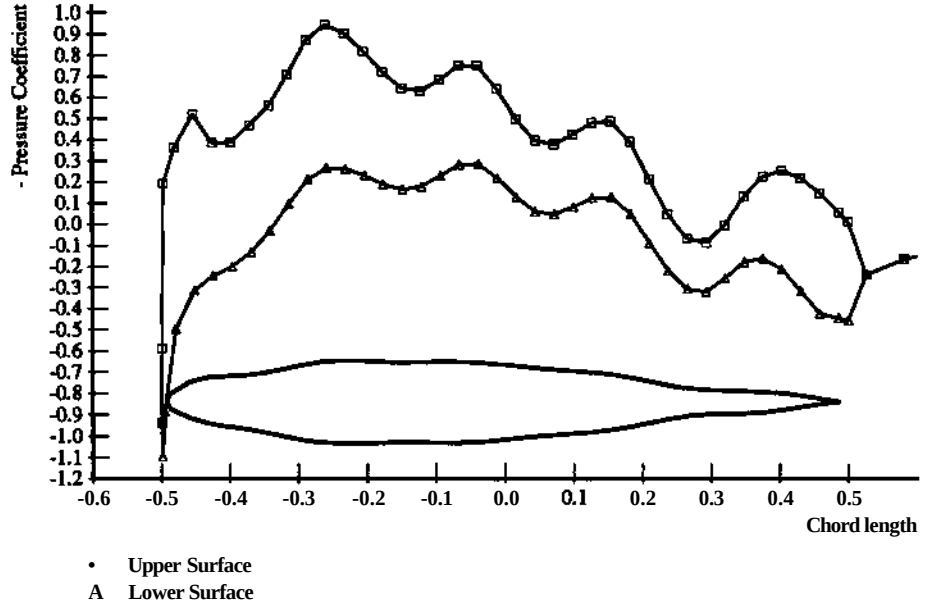

Figure 5.13: Initial conditions. NACA 2412 modeled using splines *(n\2* = ^22 = 9). M_∞ = 0.6, a = 1.0 degrees.

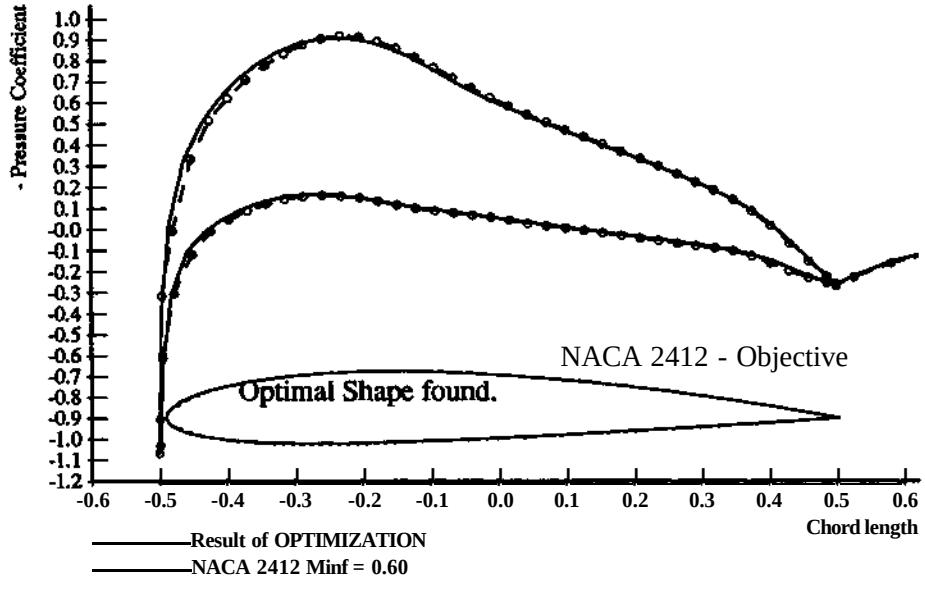

Figure 5.14: Comparison between Optimum and target conditions. NACA 2412 modeled using 9 splines. $M^{\wedge}$ = 0.6, a = 1.0 degrees.

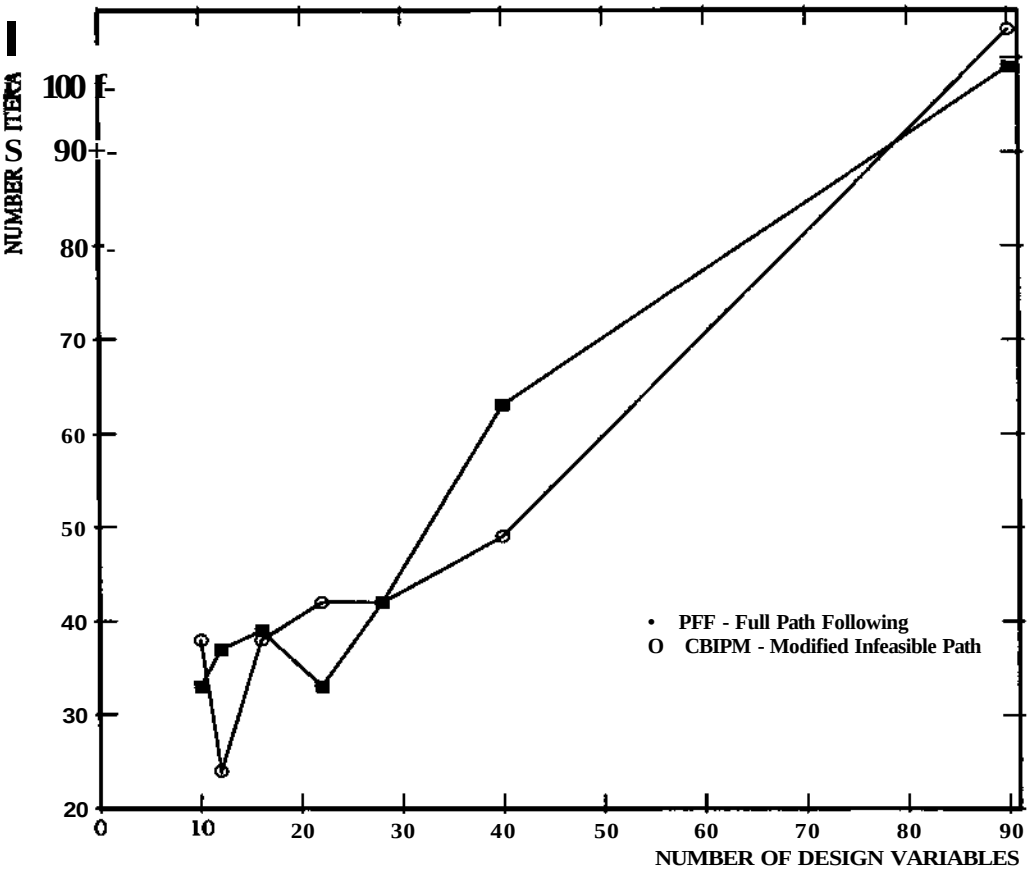

Figure 5.15: Number of Iterations vs. Number of Design Variables.

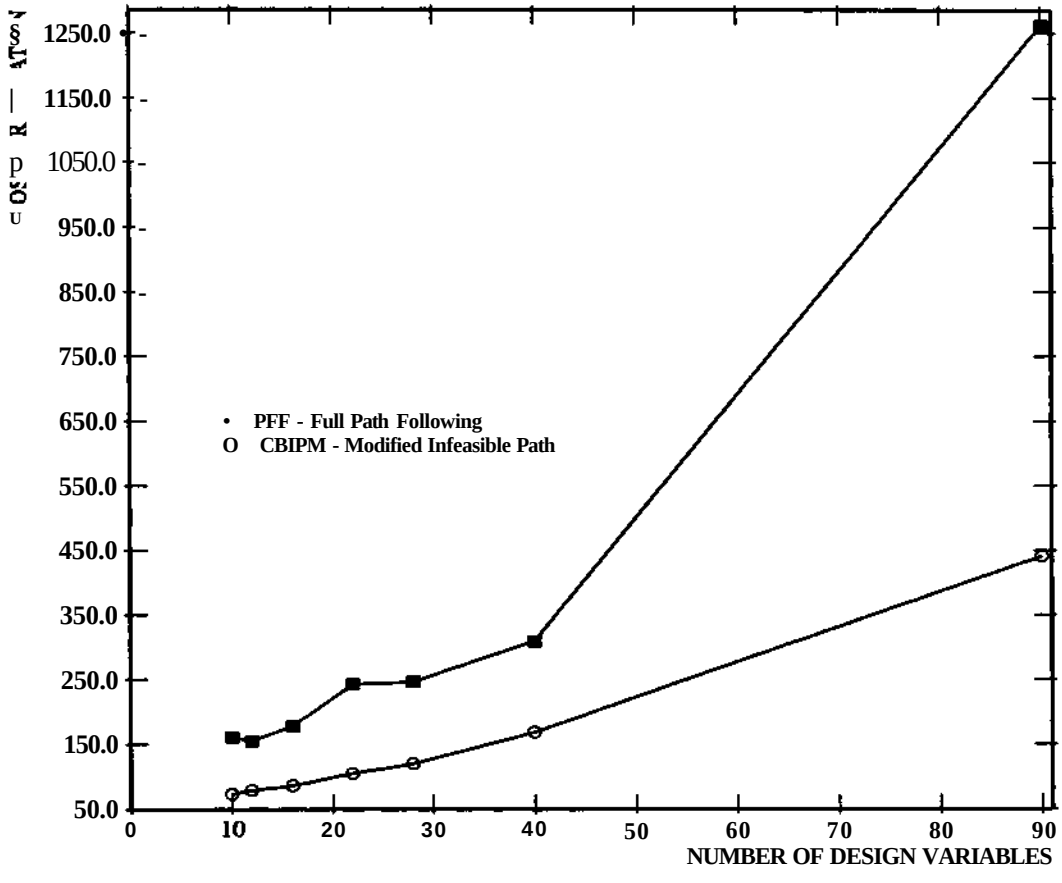

Figure 5.16: Cost per Iteration vs. Number of Design Variables.

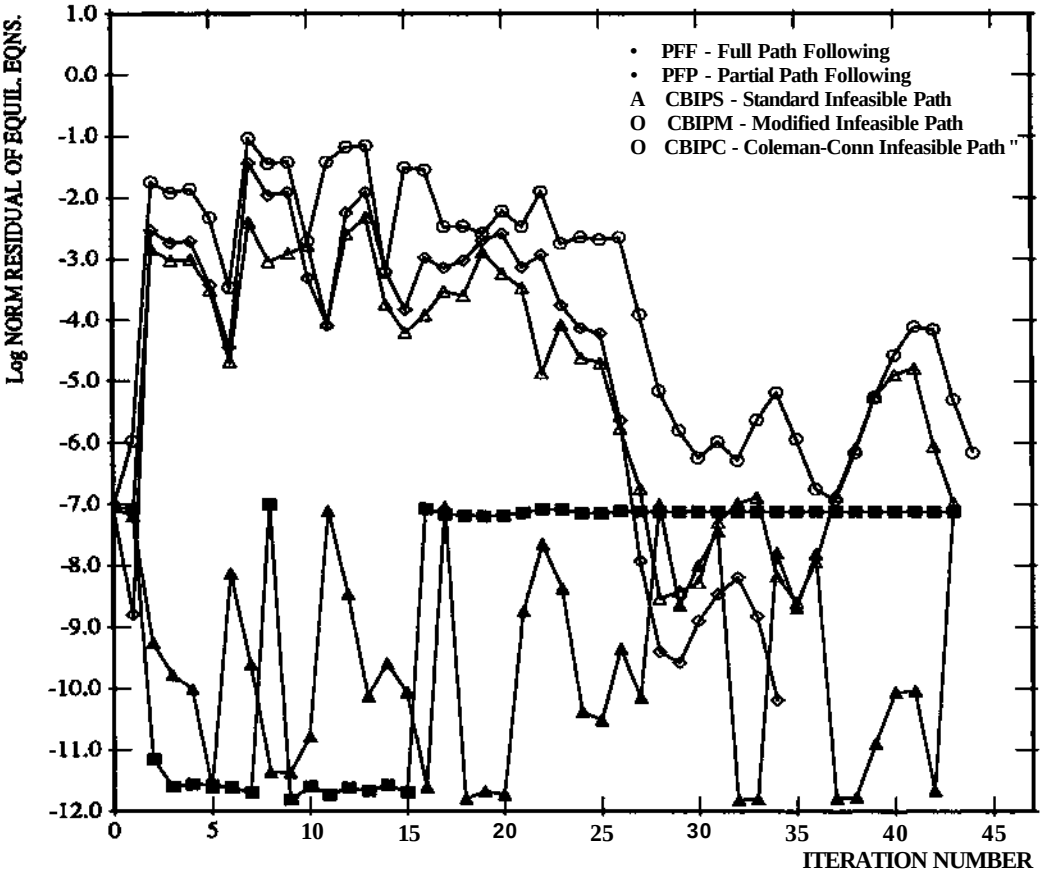

Figure 5.17: Residual of Equilibrium Equations vs. Iteration Number.

Chapter 6

Conclusions

We have presented an infeasible path method for the optimal design of systems governed by nonlinear boundary value problems. In particular, we have considered shape optimization of airfoils in compressible potential flow. The infeasible path method avoids full resolution of the flow at each iteration by including the governing equations as equality constraints. A null space basis for the constraint Jacobian is defined, resulting in a method which requires solution of just two or three linear systems at each optimization iteration. The coefficient matrix of these systems is just the finite element stiffness matrix, thereby enabling the method to leverage efficient finite element solvers. Examples demonstrates that the overall number of iterations is about the same as a path-following method, while significantly reducing the work per iteration.

Acknowledgments

We thank Larry Biegler for many useful discussions. This work was partially supported by the Engineering Design Research Center, an NSF Engineering Research Center at Carnegie Mellon University, and by NSF grants DDM-9009597 and DDM-9114678.

Bibliography

[1] LH. Abott and A.E. von Doenhoff. *Theory of Wing Sections.* Dover, 1959.

[2] G. Carey and J.T. Oden. *Finite Elements: Fluid Mechanics, Vol VI.* Prentice Hall, 1986.

[3] T.F. Coleman and A.R. Conn. Nonlinear programming via an exact penalty function: Asymptotic analysis. *Mathematical Programming,* 24:123-136, 1982.

[4] P.D. Frank and G.R. Shubin. A comparison of optimization-based approaches for a model computational aerodynamics design problem. *Journal of Computational Physics,* 98:74-89, 1992.

[5] R.T. Haftka, Z. Giirdal, and M.P. Kamat. *Elements of Structural Optimization.* Kluwer Academic Publishers, 1990.

[6] B.A. Murtagh and M.A. Saunders. A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. Technical Report SOL 80-1R, Department of Operations Research, Stanford University, 1980.

[7] J. Nocedal and M.L. Overton. Projected Hessian updating algorithms for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis,* 22, 1985.

[8] C.E. Orozco and O.N. Ghattas. Sparse approach to simultaneous analysis and design of geometrically nonlinear structures. *AIAA Journal,* 30(7):1877-1885,1992.

[9] K. Schittkowski. NLPQL: A FORTRAN subroutine for solving constrained nonlinear programming problems. *Annals of Operations Research,* 5:485-500, 1985/6.