NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Massively Parallel Aerodynamic Shape Optimization

C.E. Orozco, O.N. Ghattas

EDRC 12-53-92

Massively Parallel Aerodynamic Shape Optimization

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

July 22, 1992

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

Contents

Abstract

We consider the problem of finding the shape of an airfoil which produces a pressure distribution closest to a desired one. The flow is modeled by the nonlinear potential equations of compressible flow. The problem is formulated as an optimization problem constrained by a discrete approximation to a nonlinear boundary value problem. We present a new parallel infeasible path method for this class of optimization problem. The method is based on a null space representation tailored to the structure of the constraint Jacobian matrix. The resulting null space projections formally involve inverses of the stiffness matrix. The algorithm requires only two stiffness matrix solves per optimization iteration, in contrast to a conventional path-following method, which resolves the full physics at each iteration. The algorithm has been implemented on a CM-2, and requires no new data structures or communication patterns beyond those needed for numerical solution of the boundary value problem. We discuss numerical evidence for the superiority of the new method relative to a conventional path-following approach.

Chapter 1

Introduction

In recent years there has been growing interest in developing the capability of performing optimal design of aerospace vehicles, e.g. [11]. Computational aerodynamics has matured to such a point that inviscid transonic flows about geometrically complex three-dimensional bodies can be accurately resolved in a few hours on modern supercomputers, e.g. [19, 29]. Given this capability for *analysis* of certain classes of flow, it is natural to seek methods for automating the process of *design*.

In its most general form, aerodynamic design can be posed as a nonlinear optimization problem consisting of an objective function reflecting design goals, and constraints which include both the partial differential equations governing behavior and additional design constraints. Variables are of 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, velocities). Solving the (discretized form of the) behavioral equations at each optimization iteration simplifies the optimization problem by both reducing the number of constraints and eliminating the state variables from the set of optimization variables. The optimization approach to aerodynamic design has rarely been taken, essentially due to the existence of more efficient techniques for linear flows about simple geometries, and the difficulty of the analysis problem for more complex flows (for example, simulation of high Reynolds number viscous flows is still an open problem). In contrast, a large body of theory, methodology, and application has been developed over the past 30 years for the sensitivity analysis and optimal design of solids and structures (see [16] or [14]). Calculations of this type are routinely performed for finding optimal shapes of structures ranging from dams [30] to aluminum cans [3]. There is increasing recognition that such an approach will be necessary for general aerodynamic design problems [11, 20].

The optimal design problem associated with a complete aerospace vehicle possesses several complicating features, which make much of the methodology of structural optimization unsuitable:

- Aerodynamic design is typically governed by *nonlinear partial differential equations,* which is the exception, not the norm, for structural design problems. The nonlinearities can be quite severe, as in problems with shocks, and may require many iterations

and special techniques (such as continuation and artificial dissipation) to resolve [29]. Intractability can ensue from having to perform analysis iterations within design iterations.

- Aerodynamic design can be *very large scale:* state variables number in the millions for a complete vehicle (without even attempting to resolve viscous effects); design variables can number in the tens of thousands (for example those needed to describe the shape of an entire aircraft).

A successful strategy for aerospace design optimization must address these two factors. It is desirable that such a strategy avoid a full Newton solution at each optimization iteration of the nonlinear algebraic equations that arise upon discretization of the governing partial differential equations. It is desirable that the optimization strategy exploit the structure and characteristics of the problem, especially those of the governing equations. Furthermore, it is desirable that the strategy be compatible with existing analysis software. Advanced architecture computers will be needed to solve these problems due to the considerable expense associated with analysis, and due to the large problem size (both in state and design variables). It is clear that a successful strategy cannot be devised independently of the computers that will deliver the throughputs necessary to solve such problems.

The goal of this report is to present an optimization strategy that:

- is tailored to engineering systems governed by nonlinear partial differential equations,
- avoids full solution of the flow equations at each optimization iteration,
- exploits existing solver technology for large-scale sparse algebraic systems arising from PDE's, and
- maps well onto scalable, massively parallel private memory systems.

We demonstrate the strategy through an application to shape optimization of subsonic airfoils. The physical model we employ is the full-potential approximation to the Navier-Stokes equations. For steady subsonic flows, these equations are elliptic and nonlinear. Although moderately-sized, the problem is a useful vehicle for examining the behavior of our analysis and optimization algorithms. We have implemented our shape optimization strategy on a CM-2, and we report performance on problems employing unstructured finite element meshes.

The rest of the report is organized as follows. In chapter 2 we discuss the general problem of airfoil design and formulate one instance of it as a shape optimization problem. In chapter 3 we present an optimization algorithm that contains the four desirable features itemized above. In chapter 4 we discuss relevant features of the CM-2 implementation, and in chapter 5 we compare our approach with a conventional one. Conclusions are drawn in chapter 6.

Chapter 2

Aerodynamic Design of Airfoils

The designer of modern airfoils is faced with conflicting requirements and constraints dictated by different disciplines such as aerodynamics, solid mechanics, and control theory. With respect to aerodynamics, performance characteristics like low drag and high lift influence the design. From the structural point of view, the thickness of the airfoil and/or its enclosed area are critical, and for control, stall severity and pitching moment govern [10].

Although our ultimate objective is to solve interdisciplinary optimal design problems, we focus here on a simplified airfoil design problem obtained by neglecting the coupling among the different behavioral equations. The corresponding airfoil design problem is often reduced to that of finding the shape that will exhibit a prescribed velocity or pressure distribution. This distribution is generally known to favor a desirable performance feature, like low drag or favorable boundary layer behavior. It is important that the flow remains attached to the surface of the airfoil or that it separates very close to the trailing edge. A positive pressure gradient on the upper surface of the airfoil is known to produce an increment in the boundary layer thickness or even complete separation. Target pressure distributions are then devised to avoid separation of the flow, for some values of the design parameters (cruising speed, angle of attack, etc.). Most airfoil design techniques assume that such a target pressure or velocity distribution is available.

Numerous techniques exist for establishing the shape of an airfoil once a pressure or velocity distribution has been identified. The simplest approach is a trial and error procedure: the shape of the airfoil is manually modified with the aid of an analysis code until the target distribution is obtained. A more sophisticated idea is to take an "inverse problem" approach and attempt to manipulate coefficients of the governing equations to produce the desired pressure profile. This was first suggested by Lighthill [21], who conformally mapped the profile to the unit circle and made use of the known solution for incompressible flow to find the mapping function. Generalizations of these techniques to compressible flow become iterative in nature, e.g. [28, 27], and it becomes useful to examine them within the context of optimization problems. Optimization approaches, e.g. [17], have the advantage of generality in choice of objective and constraints functions, and applicability to more complex geometry and flows. They historically have been used in conjunction with *path-following*

methods, i.e. resolving the flow fully at each design iteration, and have been considered inefficient relative to other techniques [20]. Motivated by a desire to improve the efficiency of optimization-based methods, we present in the next chapter an efficient *infeasible path* method, which simultaneously converges the flow equations while improving the design. In the remainder of this chapter, we present a mathematical formulation of the design problem as a nonlinearly-constrained optimization problem.

2.1 Shape Optimization approach

We first define our flow approximation, employing a Galerkin finite element method, although this is not critical for our optimization method. Consider the problem shown in Figure 2.1, where ft represents 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, UQQ the freestream velocity, and *a* the angle of attack of the airfoil. The problem is to find the pressure distribu-

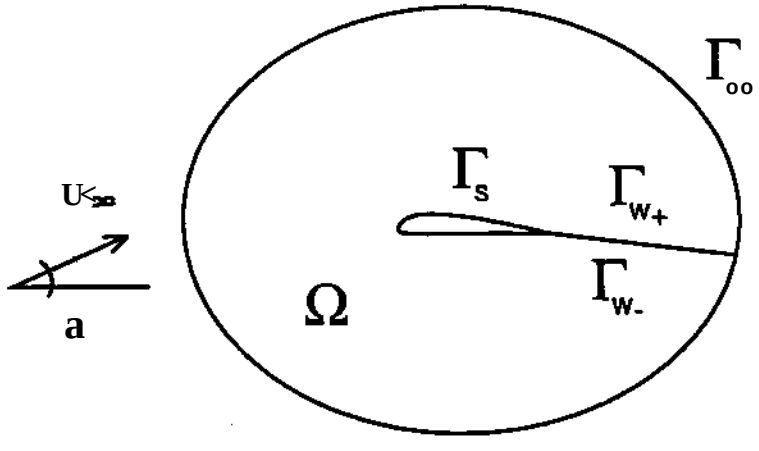

Figure 2.1: Two dimensional domain for airfoil analysis problem.

tion over T_s for subsonic flows with Mach numbers above 0.4, so that compressibility effects cannot be neglected. The equation of continuity V • (pu) = 0 and boundary conditions take the form:

$$\begin{aligned} \nabla\cdot\{[1-\wedge=\tfrac{1}{2}(\nabla<;b)^2]^{\frac{1}{7=i}}\nabla(f>\} &= 0 \quad \text{in ft} \\ \nabla\phi\cdot n &= 0 \quad \text{on } T_s \\ \nabla\phi\cdot n &= \mathbf{u}_\infty\cdot n \quad \text{on } \Gamma_\infty \end{aligned} \tag{2.1}$$

in which *(/>* is the velocity potential, 7 is a constant and poo is the freestream density. A numerical approximation can be found by solving the Galerkin problem, which is to find *<t>h€Uh* such that:

$$/\ [1-\wedge\wedge(\nabla\wedge)^2]\wedge\nabla\wedge\cdot\nabla v_i^* <*ft= /\ w_h\ (/wioo)\cdot n\ dT\ ,\quad Vw_h£Uk \tag{2.2}$$

Introducing the finite element interpolation $(j>h = N^T\$$, where N represents a vector of global basis functions and # nodal potentials, and taking $Q \equiv VN\,(VN)^T$, we obtain a set of nonlinear algebraic equations of the form h($) = f:

$$/\ [1 - \wedge (\wedge^{T} Q *)] \wedge Q *\ dil\ =\ [\quad N\ (pooUoo).\ n\ dT \tag{2.3}$$

The Jacobian of this nonlinear system of equations represents the "stiffness" matrix of the system and is obtained from (2.3) by differentiating the left hand side with respect to <&

$$\mathbf{K(\#) = /_{./n_h}\ [1 - I\frac{-}{2}\wedge(\$^T Q\ \#)]\wedge r\ Q - [1 - \wedge\frac{-}{2}\wedge(*^T Q\ \#)]£?\ Q * *^T Q\ dfl} \tag{2.4}$$

This is an *n* x n symmetric matrix that can be constructed using standard finite element ideas. It is positive definite for subsonic flow [6].

The split boundary in Figure 2.1 is provided to allow for lifting flows. The potential <*f*> is discontinuous across this boundary, i.e.

$$P = 0r+ - <t>r. \tag{2.5}$$

In this way the jump in <*j)* is equal to the circulation around the airfoil. The introduction of this line of discontinuity or split boundary corresponds mathematically to transforming the originally multiply connected domain into a simply connected domain for which the value of the potential is unique. The value of the jump in <*f*> across the wake is unknown. This value is determined by enforcing the Kutta condition which establishes that the circulation *(3* is such that the flow leaves the trailing edge smoothly and the velocity is continuous [1]. This condition implies that the velocities of the flow immediately above and below the line of discontinuity, at the trailing edge, are equal and that their direction is parallel to the bisector of the angle of the airfoil at that point. It also means that the pressure has a unique value at the trailing edge.

The airfoil design problem of finding a shape that induces a desired pressure distribution can now be stated as:

$$\text{minimize } /((!,\#) = /\ \backslash p(\&)-p^*]^2\ dT \tag{2.6}$$

$$\text{subject to } /_{J_{toh}}\ [1 - \wedge -\dot{\wedge}(*^T Q\ \#)]\wedge" Q *\ dil\ =\ \int_{Whoa} N\ (pooUoo) \bullet n\ dT \tag{2.7}$$

$$p(\Omega, \Phi)_{TE+} - p(\Omega, *)\,T\,S - = 0 \tag{2.8}$$

where *p(&)* is the predicted pressure on the airfoil, *p** is the prescribed pressure, and the last equation represents the Kutta condition written in terms of pressures at the trailing edge.

Chapter 3

An Infeasible Path Method for Optimization of Systems Governed by Nonlinear Partial Differential Equations

In this chapter we present an optimization strategy tailored to systems governed by nonlinear boundary value problems. We limit our discussion to problems that are constrained only by (a suitable discretization of) the governing partial differential equations. Extensions to problems that include additional constraints on design variables will be discussed in a future article.

Our approach is to avoid solution of the governing equations at each optimization iteration by including the governing PDE's as constraints in conjunction with projected Lagrangian optimization methods, which do not in general satisfy nonlinear constraints until the final iteration. Such strategies are termed *infeasible path.* Though well-established in chemical engineering optimization problems ([8, 4, 9]), their use in structural optimization [7, 18, 13, 15, 24] has been the exception rather than the rule. There are two principal difficulties with such an approach: first, inclusion of the governing equations forces the *analysis* software to be embedded within the optimizer; and second, the PDE constraints induce sparse constraint Jacobian and Lagrangian Hessian matrices [24], thus necessitating sparse optimization strategies for large problems. Addressing the second problem with a general-purpose sparse optimizer, such as MINOS [22], is problematic: the favorable structure of the constraint Jacobian with respect to state variables (i.e. the tangent stiffness matrix, which in the subsonic flow case is symmetric positive definite with nonzeroes corresponding to edges of a planar graph) cannot be exploited. An alternative approach to projected Lagrangian methods which does not require sparsity considerations is to employ the infeasible path idea within a quadratic penalty method [13] and minimize by a matrix-free method such as nonlinear conjugate gradients; of course, such an approach implies inherent

ill-conditioning [12].

Although special nonlinear least squares techniques (e.g. employing Gauss-Newton ideas) can be developed for (2.6)-(2.8), we have here in mind both large residual problems and problems with more general objective functions. Given the success of Sequential Quadratic Programming (SQP) methods for general problems (e.g. [26]), it is natural to seek an SQP strategy tailored to optimal design problems with discrete PDE 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.

3.1 An SQP Method

We begin with a generalization of the optimization problem (2.6)-(2.8):

$$\text{minimize } /(x) \tag{3.1}$$

$$\text{subject to } h(x) = 0 \tag{3.2}$$

$$/ : » ^{n} \wedge K, \quad h : » ^{n} \wedge » ^{m}, \quad x \; G \; f \tag{3.3}$$

Here, / represents the design objective, and h is a system of nonlinear algebraic equations arising from discretization of a boundary value problem, for example by a Galerkin finite element method. The *n* variables x consist of the *m* state variables u and n — *m* design variables b. Typically *m* >• *n* — ra, especially in shape optimization in which a shape is parameterized by a small number of variables, and a large number of state variables arise from discretization of the domain.

SQP can be considered as a Newton solution of the first order optimality conditions. A Newton step defines the following quadratic program (QP):

$$\text{minimize } pjfg^{*} + \frac{1}{2}p£G_{L}pfc \tag{3.4}$$

$$\text{subject to } AATPA; = -h^{*} \tag{3.5}$$

where g is the gradient of the objective function, G^ is the Hessian matrix of the Lagrangian function, *Ak* 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 *k;* it is understood that all quantities depending on x are evaluated at x&. Let us write p^ as the sum of range and null space components:

$$p = Zp^{*} + Yp_{y} \tag{3.6}$$

in which Z G $K^{nx\wedge n-m}$) is a matrix whose columns form a basis for the null space of A, and Y £ $\$ft^{nXm}$ is a matrix whose columns span the range space of A^T. The range space step is completely determined by substituting (3.6) into (3.5), resulting in the m x *in* system:

$$\mathbf{AY}\mathbf{p}_y = -h \tag{3.7}$$

The null space move is found by substituting (3.6) into (3.4) and minimizing with respect to $\mathbf{p}_{\omega}$:

$$\mathbf{Z}^T\mathbf{G}_L\mathbf{Z}\mathbf{p}, = -\mathbf{Z}^T(\mathbf{g} + \mathbf{G}_L\mathbf{Y}_{Py}) \tag{3.8}$$

The (n — *m)* x (n — m) projected Hessian matrix Z^TGiZ 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 $\mathbf{Z}^T\mathbf{G}\wedge\mathbf{Y}$ would increase storage requirements 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 [23], in the sense that

$$\frac{\| \mathbf{x}_{k+2} - X^* \|}{\| \mathbf{x}_k - \mathbf{x}^* \|} \rightarrow^{Q} \mathbf{as}^{k} \rightarrow \infty \tag{39}$$

3.2 Decomposition Strategy

The critical step is the definition of appropriate range 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:

$$\mathbf{x}^T = [\mathbf{u}^T, \mathbf{b}^T] \tag{3.10}$$

in which $\mathbf{u} \in U^m$ and $\mathbf{b}$ 6 $\&^{n\sim m}$. The partitioned constraint Jacobian becomes

$$\mathbf{A} - [\mathbf{K}, \frac{\partial \mathbf{R}}{\partial \mathbf{b}}] \tag{MI}$$

in which we have identified the Jacobian of the discrete PDE's with respect to the state variables as the "tangent" stiffness matrix K. It is desirable to exploit the inverse of K, since this is the central computation of a Newton step of the analysis problem. We can define a matrix Z whose columns are orthogonal to the rows of A as:

$$\mathbf{Z} = \begin{bmatrix} K_x & SE \end{bmatrix} \tag{3.12}$$

Here, we write K^{-1} 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 linear systems involving K is required, using any direct or iterative technique.

The range space basis is defined simply as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} \tag{3.13}$$

Clearly, the matrix

$$Q = [\, Z \; Y \,] \tag{3.14}$$

is nonsingular provided K is nonsingular, and hence Z and Y form a basis for $3?^n$. The invertibility of K is established by the well-posedness of the boundary value problem. The resulting range space step for p_y from (3.7) becomes:

$$Kp_y = -h \tag{3.15}$$

We observe that this is simply a Newton step for the nonlinear system h = 0.

Using the null space definition (3.12), the null space move p_2 can be found from

$$B^* p_2 = -g_2 \tag{3.16}$$

where

$$g_2 = Z^T g = -\left[K^{-T} g_u + g_t \right] \tag{3.17}$$

Here B_2 represents a Quasi-Newton approximation to the projected Hessian, and g_u and gt represent objective gradients with respect to the state and design variables, respectively. Note the close connection between the expression for the projected gradient (3.17), and the gradient of the objective using a path-following method (in conjunction with implicit gradients, e.g. [16]). The difference, of course, is that in (3.17) K need not be evaluated at the u for which h = 0, in contrast to path following methods, which converge the governing equations at each design iteration.

Using (3.6), the moves in the state and design variables take the form

$$p_u = -K^{-1} \frac{\partial h}{\partial b} p_z + p_y \tag{3.18}$$

$$p_b = p_z \tag{3.19}$$

The recipe for the update of the state variables (3.18) 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 design variables p_z; 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.

3.3 Algorithm

The steps of the method can be arranged into the following algorithm, in which we have made use of the symmetry of K:

- Set $k = 0$, $H_z^\circ = I$, $u = u_0$, $b = b_0$,
- Solve $K^\circ A^\circ = g£$
- Find $g_z^\circ = -(\frac{\partial h}{\partial b}^T)^0 \lambda^0 + g_b^0$
- While Hg^ll > e and $k <$ maxiter do:
  - Find $p_z^k = -H_z^* g$,
  - Find $d^k = §|_y^{**} p_z^* + h^k$
  - Solve $K^{fc} p£ = -d^\wedge$
  - Find a^k from a line search
  - Update variables: $u^{fc+1} = u^k + a^k v_u^k$, $b^{\wedge 1} = b^* + a^k p_z^k$
  - Solve $K^{*+1} A^{*+1} = g_u^{*+1}$
  - Find $g_z^{*+1} = -(\frac{\partial h}{\partial b}^T)^{k+1} \lambda^{k+1} + g_b^{k+1}$
  - Find $y_z^k = g_z^{k+1} - g_z^k$
  - Update $H£^{+1}$ using y^ and p_z^k
  - Set $g_z^k = g_z^{k+1}$
  - Set fc = k + 1
- Endwhile

We refer to the system $K^{fc} p£ = -d^{fc}$ as the state variable update, and $K^{fc+1} A^{A;+1} = g£^{+1}$ as the adjoint system, and we note that the main work associated with this algorithm is the solution of these two linear systems.

Chapter 4

Implementation on the Connection Machine CM-2

The infeasible path optimization method presented in the previous chapter simultaneously drives the design towards an optimum one while converging the state equations. It derives its advantage over the path-following method by relaxing the need to fully resolve the state equations at each optimization iteration—an advantage which grows with increasing nonlinearity of the state equations. As a consequence of the particular range and null space bases employed, the primary effort per iteration associated with this algorithm is the solution of two linear systems involving K, the state variable update system and the adjoint system. In fact, these two systems are precisely "analysis"-like problems, or more properly single steps of an analysis problem. Thus, for these two subproblems, we can exploit massively parallel algorithms for the numerical solution of partial differential equations. Typically, K is large and sparse, and often (as in subsonic flow) symmetric positive definite. Its sparsity structure is typical of weighted-residual schemes which employ low-order, compactly-supported basis functions. Efficient iterative solvers have been developed for such problems.

In our current CM-2 implementation, we assume that the number of design variables is "small", and therefore the null space move (3.16) is computed on the front end workstation, rather than distributing it to the processors. Accordingly, the null space vectors p^*_z, g£, b*, g£, and y^k_z are stored on the front end. Under these conditions, the algorithm defined in the previous chapter induces *no new communication patterns nor parallel data structures* beyond those required for the analysis problem h = 0. In particular, the range space vectors d*, J^ ^{k}, p£, A^{fc}, and g£ are of the same dimension and share the same relationship to the underlying mesh topology as the state equations h* and the state variables u*. Therefore, they are partitioned and mapped to processors and manipulated by the data-parallel routines in the same way as are the state variables. In our CM-2 implementation, we employ a simple Cuthill-McKee ordering and unstructured element-based partitioning; however, more sophisticated graph partitioning and mapping methods (e.g. [5]) can be invoked for the range space vectors.

The adjoint linear system is solved in identical fashion to the state variable update. In

our current CM-2 implementation, we have used an unpreconditioned unstructured-mesh conjugate gradient solver for the two linear systems involving K; again, more sophisticated solver-preconditioner pairs may be used, e.g. [2]. Since the linear systems involve K, the same issues that arise in massively parallel solution of PDE's arise here as well. In particular, routing contention associated with unstructured meshes can degrade performance [25]. We employ the FASTGRAPH communication compiler to speed up global communications induced by unstructured meshes. We repeat for emphasis that the *optimization problem can be decomposed to a form that, with respect to parallelism, requires no new tools beyond those developed for the analysis problem.*

Chapter 5

Numerical results

In this chapter we compare, using two examples, a CM-2 implementation of our infeasible path method with sequential versions of the both the infeasible path method and a feasible path method. The design variables are chosen as 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*. These parameters are illustrated in Figure 5.1. The Kutta

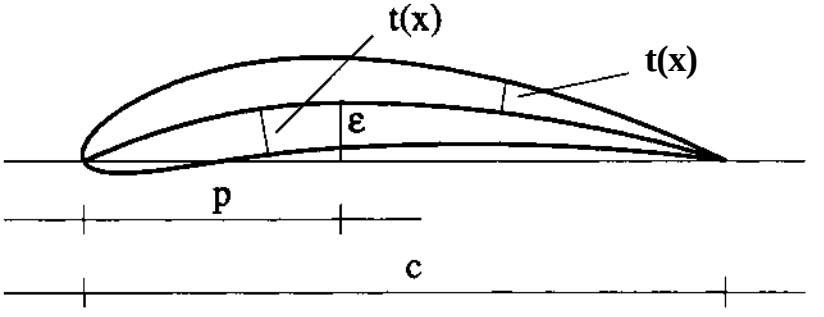

Figure 5.1: Four-digit NACA airfoil

condition is satisfied by treating the circulation as a variable, and including the square of (2.8) in the objective function.

When dealing with pressure distributions over bodies, it is convenient to refer to the pressure coefficient, instead of the absolute pressure. The pressure coefficient is defined as C_p = [illegible]. In what follows, we refer to the distribution of the pressure coefficient over the airfoil simply as the pressure distribution. In both examples, the target pressure distribution corresponds to a NACA 2412 airfoil, for which r = 0.12, p = 0.40, e = 0.02.

Figure 5.2 shows the mesh topology used in our numerical simulation. This mesh is shown for illustration purposes only; it was not used to obtain the results reported here. To obtain accurate results for pressure distributions, the exterior boundary must be placed much farther away from the airfoil than shown in Figure 5.2. Although the mesh is structured, we do not treat it as such, in anticipation of more general problems.

Example 1

The data for the first example are:

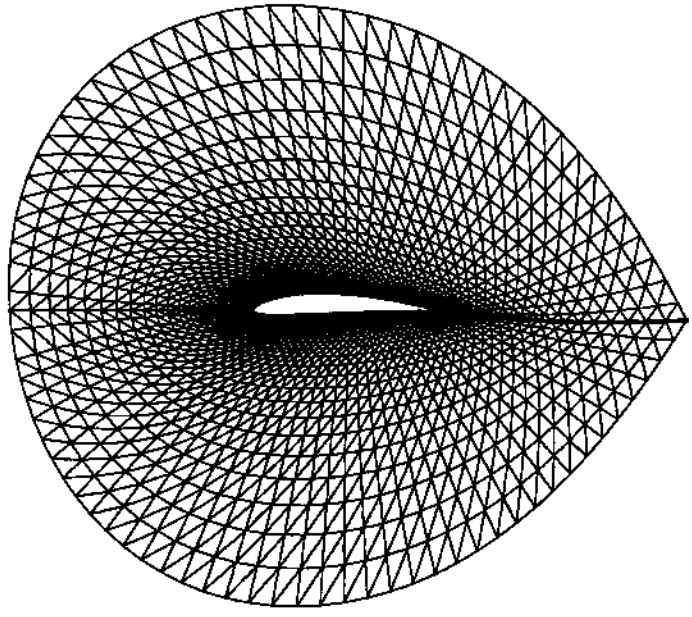

Figure 5.2: Discretized two dimensional domain with airfoil

- Target pressure distribution: NACA 2412 at zero angle of attack, $M^{\wedge} = 0.57$
- Initial values of parameters: r = 0.08, p = 0.35, e = 0.01
- Number of finite elements: 16,200
- Number of nodes: 8,296

A sequence of airfoil shapes is displayed in Figure 5.3. The final shape is indistinguishable from the target. Numerical results are tabulated in Table 5.1. The first two columns present

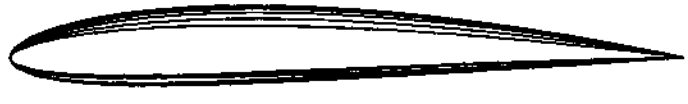

Figure 5.3: Sequence of airfoil shapes

results from a sequential implementation of the path following (PF) method (using SQP)

Table 5.1: Algorithm performance

PATH FOLLOWING, IBM	**INFEASIBLE PATH, IBM**	INFEASIBLE PATH, CM-2
Optimal values of parameters		
0.12000308 0.39982421 0.02000018	0.11990874 0.39977783 0.01998744	0.11987302 0.39913902 0.01998177
Optimal objective		
0.2309865D-06	0.6310934D-06	0.445447D-05
Norm of projected gradient		
0.1697D-03	0.3834D-03	0.7606D-03
Number of optimization iterations		
20	18	19
Average inumber of conjugate gradient iterations per solve		
—	—	1,102
Total CPU time (seconds)		
12,641	5,932	1,511

and the proposed infeasible path (IP) algorithm on an IBM RS6000-320H workstation. The linear systems involving K are solved with a direct band solver. The number of iterations is about the same for the two methods—which is not surprising since the dimension of the null space is identical in both. The IP method derives its advantage by avoiding the need to resolve the flow at each optimization iteration. This problem is not particularly nonlinear—convergence of the flow equations is usually obtained in 6 to 7 Newton iterations for freestream Mach numbers of the order of 0.6. Yet, the CPU times reflect a 53% reduction in effort. With greater nonlinearity imposed by higher Mach numbers, the improvement will become more dramatic.

The third column of Table 5.1 lists results obtained on a single 8K sequencer of the CM-2. The algorithm differs from that in the second column: a conjugate gradient method, rather than a direct band solver, is used to solve the linear systems at each iteration. The CM-2 provides only a factor of four improvement, for several reasons: (1) the expressions $\frac{\partial b}{\partial x_i}$ are computed by finite differences; since the controlling null space vectors reside on the front end, this induces frequent front end-to-processor communication (which we have not optimized); (2) the conjugate gradient solver is inefficient relative to a direct method for a problem of this small size (8296 unknowns); (3) the conjugate gradient solver is not preconditioned; and (4) the unstructured mesh induces global communication. The first three problems can be easily addressed when we solve larger problems by introducing a preconditioner and analytic derivatives. The last is inherent to the CM-2, and can be

alleviated by moving to a coarser grained machine, and using an algorithm that has a gvreater computation-to-communication ratio. The algorithm of the previous chapter maps well to such an architecture, again requiring only an appropriate solver for the "analysis" problems.

Example 2

To examine the effect of an incorrect physical model on the optimal design, we solve the airfoil design problem assuming linear (incompressible) flow but with a target distribution based on nonlinear (compressible) flow. The target pressure distribution corresponds to a NACA 2412 airfoil at zero angle of attack and 0.6015 freestream Mach number. The pressure distribution is shown in Figure 5.4.

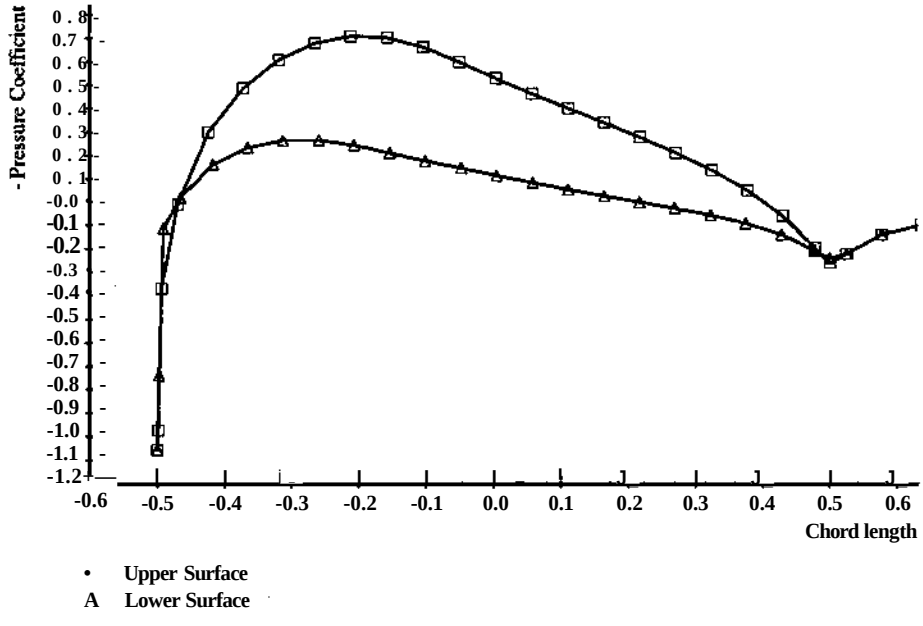

Figure 5.4: Pressure coefficient: NACA 2412 at zero angle of attack, Moo = 0.6015

The initial guesses for the parameters of the airfoil are taken as r = 0.02, p = 0.35, e = 0.0002. The airfoil with the corresponding pressure distribution is shown in Figure 5.5. Since this airfoil is essentially symmetric, it produces no lift force and therefore the circulation is zero. This value was taken as the initial guess for the circulation. Since there is no lift force, the pressures for the upper and lower surfaces practically coincide, and since this airfoil is very thin, the pressure coefficient is very close to zero.

For linear flow, the full potential equations reduce to Laplace's equation. A discretization using 2520 elements and 1333 nodes is used. The algorithm performs 24 optimization iterations that take 286 seconds of CPU time on an IBM RS6000-320H workstation. The algorithm terminates for a value of the projected gradient of 0.9410 X 10^{-7}, corresponding to an objective value of 0.7773114 X 10^{-3}. The optimal values of the design variables

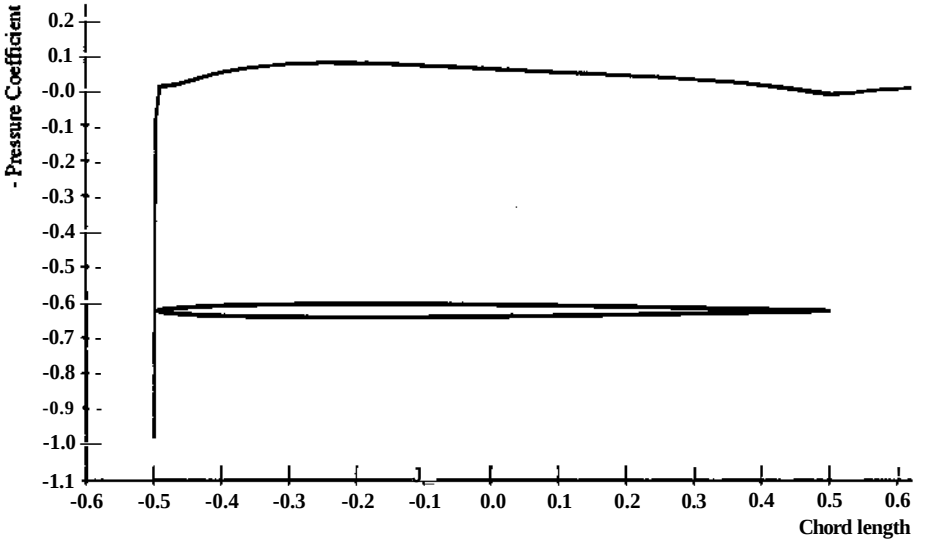

Figure 5.5: Initial airfoil and pressure distribution, Example 1

are r = 0.14983768, p = 0.40093774, e = 0.02608220. The final value of the circulation is $/3$ = 26.83437313. The resulting airfoil shape and pressure distributions are shown in Figure 5.6. As can be observed, although the target and optimal pressure distributions agree well, the optimal and target shapes differ due to the differing models employed. In fact, the optimal shape is 25% thicker than the target one.

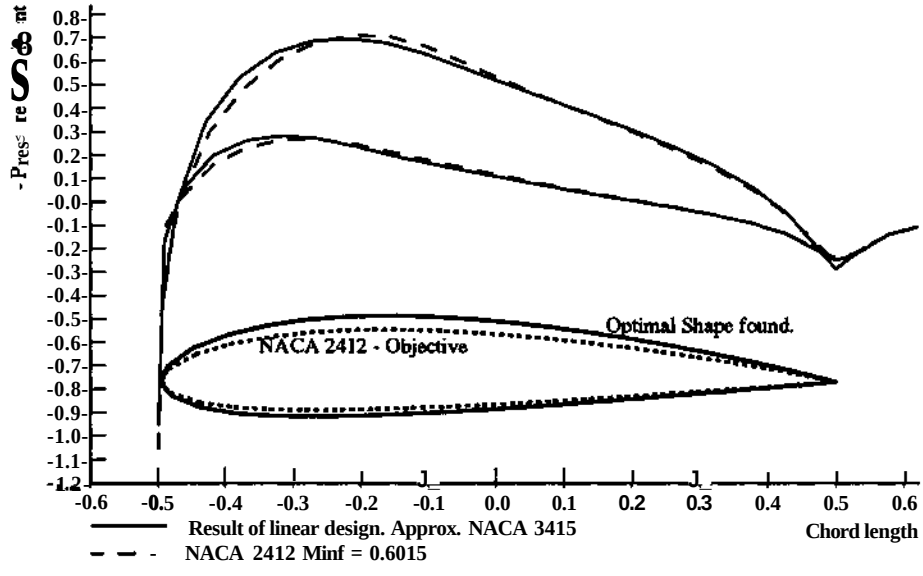

Figure 5.6: Target airfoil shape and optimal (based on linear flow) shape with corresponding pressure distributions, Example 1

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. Nonorthogonal range and null space bases for the constraint Jacobian are defined, resulting in an algorithm which requires solution of just two linear systems at each optimization iteration. The coefficient matrix of these two systems is just the finite element stiffness matrix, thereby enabling the method to leverage efficient finite element solvers. The algorithm has been shown to map well to massively parallel systems, in the sense that it requires no new parallel data structures nor communication patterns beyond those required for the analysis problem. An example demonstrates that the overall number of iterations is about the same as a path-following method, while significantly reducing the work per iteration. Even for a mildly nonlinear problem, the resulting reduction in CPU time is over 50%.

6.1 Acknowledgments

We thank Larry Biegler for many useful discussions. Use of the CM-2 at the Pittsburgh Supercomputing Center is gratefully acknowledged. 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] I.H. Abott and A.E. von Doenhoff. *Theory of Wing Sections.* Dover, 1959.

[2] H. Berryman, J. Salz, W. Gropp, and R. Mirchandaney. Krylov methods preconditioned with incompletely factored matrices on the CM-2. Technical Report 89-54, ICASE, NASA Langley Research Center, Hampton, VA, December 1989.

[3] N.D. Bhakuni, A.B. Trageser, S. Sundaresan, and K. Ishii. Structural optimization methods for aluminum beverage can bottoms. Unpublished report, Alcoa Laboratories, 1992.

[4] L.T. Biegler. On the simultaneous solution and optimization of large scale engineering systems. *Computers and Chemical Engineering,* 12(5):357-369, 1988.

[5] G.E. Blelloch, A. Feldmann, 0. Ghattas, J.R. Gilbert, G.L. Miller, D.R. O'Hallaron, E.J. Schwabe, J.R. Shewchuk, and S.H. Teng. A separator-based framework for automated partitioning and mapping of parallel algorithms for numerical solution of PDEs. In *Issues and Obstacles in the Practical Implementation of Parallel Algorithms and the Use of Parallel Machines,* 1992, to appear.

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

[7] N.H. Chao, S.J. Fenves, and A.W. Westerberg. Application of a reduced quadratic programming technique to optimal structural design. In E. Atrek, R.H. Gallagher, K.M. Ragsdell, and O.C. Zienkiewicz, editors, *New Directions in Optimum Structural Design,* pages 413-427. Wiley, 1984.

[8] J.E. Cuthrell and L.T. Biegler. Simultaneous solution and optimization of process flowsheets with differential equation models. *Chemical Engineering Research and Design,* 64:341-357,1986.

[9] J.E. Cuthrell and L.T. Biegler. Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering,* 13:49-62, 1989.

[10] M. Drela. Elements of airfoil design methodology. In P.A. Henne, editor, *Applied Computational Aerodynamics.* AIAA, 1990.

[11] 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.

[12] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[13] R.T. Haftka. Simultaneous analysis and design. *AIAA Journal,* 23(7):1099-1103,1985.

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

[15] R.T. Haftka and M.P. Kamat. Simultaneous nonlinear structural analysis and design. *Computational Mechanics,* 4(6):409-416, 1989.

[16] E.J. Haug and J.S. Arora. *Applied Optimal Design.* Wiley-Interscience, 1979.

[17] R. Hicks and P.A. Henne. Wing design by numerical optimization. In *Proceedings of the AIAA Aircraft Systems and Technology Meeting,* 1977.

[18] A.N. Hrymak, G.J.McRae, and A.W. Westerberg. Combined analysis and optimization of extended heat transfer surfaces. *Journal of Heat Transfer,* 107:527-532, 1985.

[19] A. Jameson. Successes and challenges in computational aerodynamics. In *8th Computational Fluid Dynamics Conference.* AIAA, 1987.

[20] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing,* 3:233-260, 1988.

[21] M.J. Lighthill. *A new method of two-dimensional aerodynamic design.* Aeronautical Research Council, London, 1945.

[22] 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.

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

[24] C.E. Orozco and O.N. Ghattas. A sparse approach to simultaneous analysis and design of geometrically nonlinear structures. *AIAA Journal,* 1992, to appear.

[25] J. Saltz, S. Petiton, H. Berryman, and A. Rifkin. Performance effects of irregular communication patterns on massively parallel multiprocessors. *Journal of Parallel and Distributed Computing,* 13:202-212, 1991.

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

[27] G. Volpe. Inverse airfoil design: A classical approach updated for transonic applications. In P.A. Henne, editor, *Applied Computational Aerodynamics.* AIAA, 1990.

[28] G. Volpe and R.E. Melnik. The design of transonic aerofoils by a well posed inverse method. *International Journal for Numerical Methods in Engineering,* 22:341-361, 1986.

[29] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, and J.E. Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics,* 92:1-66, 1991.

[30] O.C. Zienkiewicz and J.S. Campbell. Shape optimization and sequential linear programming. In R.H. Gallgher and O.C. Zienkiewicz, editors, *Optimum Structural Design,* pages 109-126. Wiley, 1973.