Optimization Techniques Exploiting Problem Structure: Applications to Aerodynamic Design

Ajit R. Shenoy

Doctoral Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Eugene M. Cliff, Chair
Bernard Grossman
Frederick Lutze
Rakesh K. Kapania
Terry L. Herdman

April, 1997
Blacksburg, Virginia

Keywords: Sequential Quadratic Programming, Reduced Hessian, Trust Region, Sparse Optimization, Airfoil Design

Optimization Techniques Exploiting Problem Structure: Applications to Aerodynamic Design

Ajit R. Shenoy

(ABSTRACT)

The research presented in this dissertation investigates the use of all-at-once methods applied to aerodynamic design. All-at-once schemes are usually based on the assumption of sufficient continuity in the constraints and objectives, and this assumption can be troublesome in the presence of shock discontinuities. Special treatment has to be considered for such problems and we study several approaches.

Our all-at-once methods are based on the Sequential Quadratic Programming method, and are designed to exploit the structure inherent in a given problem. The first method is a Reduced Hessian formulation which projects the optimization problem to a lower dimension design space. The second method exploits the sparse structure in a given problem which can yield significant savings in terms of computational effort as well as storage requirements. An underlying theme in all our applications is that careful analysis of the given problem can often lead to an efficient implementation of these all-at-once methods.

Chapter 2 describes a nozzle design problem involving one-dimensional transonic flow. An initial formulation as an optimal control problem allows us to solve the problem as as two-point boundary problem which provides useful insight into the nature of the problem. Using the Reduced Hessian formulation for this problem, we find that a conventional CFD method based on shock capturing produces poor performance. The numerical difficulties caused by the presence of the shock can be alleviated by reformulating the constraints so that the shock can be treated explicitly. This amount to using a shock fitting technique. In Chapter 3, we study variants of a simplified temperature control problem. The control problem is solved using a sparse SQP scheme. We show that for problems where the underlying infinite-dimensional problem is well-posed, the optimizer performs well, whereas it fails to produce good results for problems where the underlying infinite-dimensional problem is ill-posed. A transonic airfoil design problem is studied in Chapter 4, using the Reduced SQP formulation. We propose a scheme for performing the optimization subtasks that is based on an Euler Implicit time integration scheme. The motivation is to preserve the solution-finding structure used in the analysis algorithm. Preliminary results obtained using this method are promising. Numerical results have been presented for all the problems described.

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Eugene M. Cliff for his guidance. It has been my good fortune to be associated with him, both as a person and as a mentor. His patience and encouragement are really appreciated and I am a better person for the experience of having worked with him. Dr. Matthias Heinkenschloss has been my co-advisor in spirit, even though his name does not appear offically on my committee, and I would like to place on record my appreciation for his guidance. He has played a major role in the present research effort. I am grateful to both of them for sharing their vast knowledge with me. This research has been supported by the Air Force Office of Scientific Research under Grants F49620-93-1-0280 and F49620-96-1-0329.

I would like to express my gratitude to Dr. Grossman, Dr. Lutze, Dr. Kapania and Dr. Herdman for serving on my committee. I am especially grateful to Dr. Grossman for sharing his thoughts and intuitions on the aerodynamic design problems we have attempted, and to Dr. Mason for his suggestions on the airfoil design problem. I would also like to thank Dr. John Betts for providing us with the opportunity to use the Sparse Nonlinear Programming algorithm, and for his enlightening discussions on the subject.

Melissa Chase has been a large part of the success of ICAM. Her radiant efficiency has made life remarkably worry-free for all of us at ICAM, and I would like her to know her efforts are not unappreciated. It has been a pleasure to be greeted with her smile to start every day. I would also like to thank the secretaries in the Aerospace Engineering department, especially Betty Williams, for all their efforts in making graduate life such a smooth ride.

I would like to thank my colleagues, Jeff Borggard, Ravi Krishnamurthy and John Burkardt for their brainstorming and for helping me master the intricacies of LaTeX, and Phillipe Tetrault for being kind enough to spare his time in teaching me how to use GRIDGEN.

I would also like to thank all my family and friends for their support, especially my Mother and my sister, Kavita. It has been a long journey culminating in the completion of this dissertation, and I have been fortunate to have had my wife, Uma, be a part of it. Her love and support have been an integral part of my life, and this work would not have been possible without her patience and encouragement.

Ajit R. Shenoy

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The introduction of the digital computer has revolutionalized the world of science and technology. The benefits of computers have been particularly felt in the case of the aircraft industry.

In the early days of aeronautical research, though there existed reasonable mathematical understanding of the field of aerodynamics, the lack of analytical solutions meant that research had to be based on a *cut and try* approach, which essentially involved physically testing various configurations on a smaller scale in wind tunnels. The design process was thus a tedious and time-consuming process, wherein the designer had to test a configuration and then based on the results and his experience and insight into the problem, he would make changes to the design. This process meant not only a fair amount of effort, but also required a high level of expertise on the part of the designer.

The advent of the computer gave rise to the possibility of approximating the mathematical models developed for the aerodynamic processes, in finite dimensions, and obtaining numerical simulations of the flow. This capability, of simulating flow phenomena, gave the aerodynamicist an additional tool for analysis of configurations. Over the past couple of decades, rapid advances have been made, both in the technology of the computer as well as the efficiency of the numerical algorithms for simulation. Progress in this field has been summarized by Jameson [87, 88]. Several good reference text exist, *e.g.*, Fletcher [50, 51], Hirsch [72, 73] for the interested reader. We now possess the ability to generate flow solutions for complex configurations which may not be reproducible, for practical reasons, in experimental model tests. We can obtain detailed information about the flow in an increasingly cost-effective manner. As solutions to fluid dynamics problems have become less computationally restrictive, Computational Fluid Dynamics (CFD) has greatly aug-

mented the role of wind-tunnel testing in preliminary design. In the present day design process, actual testing of model configurations is often done only in the final stages, when the required design has been more or less achieved.

Design problems can be categorized as *inverse* and *direct* design. Initial forays in the field of numerical design were directed at field-based inverse design problems. The flow field design problem is formulated by specifying some feature throughout the flow, *e.g.*, a shock-free flow. The best known such method is the hodograph method [21, 12, 124] which has been used with striking success in the development of airfoils displaying shock-free transonic flow. This method transforms the potential equations to the hodograph plane, where by superposing simple solutions, complex shock-free flows are constructed. Another such method is the fictitious gas method formulated by Sobieczky *et al.* [135]. An example of recent research done in this field is the work by De Ponte *et al.* [39].

A more common approach to inverse design is surface flow design. Here a desired flow property is specified along some boundary, *e.g.*, the pressure distribution along the surface of an airfoil. This method was pioneered by Lighthill [98]. He showed that solutions for the inverse problem to design an airfoil existed only if a certain constraint relating the surface velocity to the freestream velocity was satisfied. In addition, if the trailing edge of the airfoil was required to be closed, two more constraints appear. The problem is well-posed provided the target distribution is formulated in terms of parameters which guarantee the constraints are satisfied. For incompressible flow, such a parameterization can be explicitly developed, *e.g.*, the work by Strand [139]. For compressible flow, these explicit relationships have been found only for Karman-Tsein type gas [156]. There have been attempts to solve this problem, but only two of the constraints were satisfied [146, 31, 126]. A formulation of the inverse problem for airfoil design in transonic flow was finally developed by Volpe and Melnik [153]. Some of the recent material published in this field are the works by Bocci [19], Drela [41], van Egmond [148], Sobieczky [136], Volpe [152], Bartelheimer [11] and Brock and Ng [25]. Progress in this field has been summarized by Dulikravich [42].

One of the disadvantages of inverse design is that the designer is required to specify a desired flow profile. Selecting a target profile calls for intervention on the part of the designer as he has to be able to quantify his design needs in terms of the flow profile. This calls for sufficient experience on his part. Another disadvantage is that inverse design problems might not have a solution, as mentioned earlier, in which case we would try to achieve a flow profile that is "close" to the target. Whether this "close" profile will satisfy the conceived design goals is not clear, and the designer might need to tweak his target profile in order to achieve a sufficiently "good" solution.

In contrast, for *direct* design problems some objective is quantified as a function of the design variables, *e.g.*, to maximize the lift of an airfoil. These methods have been recently

gaining in popularity. One key advantage to solving the direct problem is that the designer does not have to rely on experience in order to formulate a desired flow profile.

In the direct design process, we couple an analysis (simulation) code along with an optimizer. The process generally involves using the analysis code to obtain a flow solution for a given configuration and then trying to improve the configuration via the optimizer, using the information available. Optimization methods may be classified as gradient-based or non-gradient based. Non-gradient based methods such as Simulated Annealing [128, 131], Genetic Algorithms [103, 159, 29] and Response Surface Methods [106, 107, 61, 147] are gaining in popularity because they alleviate the problems associated with noisy simulation data, which often cause convergence difficulties for gradient-based optimization. Response surface methods can be considered as an intermediate step in (possibly) gradient-based optimization. The method involves fitting a surface to approximate flow data evaluated for several configurations. Optimization is the performed on the response surface model. However, it should be noted that these methods require evaluation of a large number of flow solutions, and in general may not be practically feasible except in parallel computing environments. Gradient-based methods are much more popular, though the requirement for computing gradient information can often result in prohibitive costs. Initially, optimization was applied using a black-box approach, *e.g.*, by employing finite differences to obtain gradient information. This meant that obtaining gradients involved evaluating the flow solutions for several perturbations of the design configuration. The cost of obtaining these solutions was a major factor in determining the viability of these methods. Also, if the simulation produced noisy data, we end up with inaccuracies in the gradient, which results in poor convergence properties. Some of the issues involved in gradient-based optimization are addressed by Shubin [129], Shubin and Frank [130], Hallman [64], Young *et al.* [160], and Jou *et al.* [91]. Methods such as ADIFOR [18, 30, 74, 111], sensitivity-based schemes [143, 22, 56, 144, 23] and adjoint-based schemes [86, 97, 79] are now available which can be used to evaluate gradients in an effective manner. There have also been attempts to use hybrid techniques which try to use gradient and non-gradient based methods in conjunction with each other to improve efficiency [54, 24].

Whereas initial attempts to use CFD in the design process were confined to potential flow methods [132, 99, 154, 142], present day technology has enabled the use of Euler equations [102], and even the Navier-Stokes equations [71, 76, 101, 47, 66]. The use of parallel processing has also been incorporated to speed up the process of aerodynamic simulation [33, 65, 94, 121] and there have been attempts to incorporate parallel computing in the design process [83]. However, it should be noted that most endeavors have been limited to two-dimensional flows or simple three-dimensional flows, though there have been attempts with positive results for complex configurations [108, 115]. Several authors, *e.g.*, [20, 57, 93, 69, 28], have reported their experiences with using different optimization

techniques on a variety of applications. A good overview of the issues involved is given by Ta'asan [141].

Among the notable works in this field are the efforts of Jameson and Reuther [84, 114, 86, 115, 116], who pioneered the use of optimal control theory to yield an adjoint formulation which can be used to obtain gradient information in an efficient manner. Similar research has been conducted by Huan and Modi [75] and by Pironneau [112]. An interesting method, proposed by Giles *et al.* [58], uses a grid discretization based on the streamlines of the flow to obtain solutions efficiently using a Newton method. The method has been successfully used by Giles and Drela [59] for a variety of aerodynamic design applications. Another potentially useful technique is the use of Krylov subspace based methods [108, 113].

Even with the most efficient analysis codes it can become prohibitively expensive if the analysis code and optimization algorithm are linked directly together in a blackbox approach. An attractive scheme which has been gaining in popularity is the *all-at-once* method, wherein we attempt to approach feasibility and optimality simultaneously. At each iterate, we do not need to compute a feasible solution but only need a solution that moves towards feasibility. Using such a scheme can greatly reduce the costs. An intriguing technique is the one-shot method proposed by Kuruvila *et al.* [97], which is based on an adjoint formulation. The technique uses a pseudo-time concept which enables the use of the flow solver to drive the residual of the adjoint equations towards feasibility along with the governing flow equations. The method has been used to solve a variety of problems [140, 80, 81]. Another scheme has been proposed by Rizk [117].

The motivation behind the present research comes from attempts to address the issue of the numerical difficulties that are experienced in optimization procedures when handling flows that have shock discontinuities. The use of numerical optimization for transonic aerodynamic design was pioneered by Hicks and Vanderplaats [70]. There have been a number of advances in this field. However, most of the researchers use some sort of artificial dissipation in the flow solver to smear the shock, *e.g.*, the works by Jameson and Reuther [84, 114] and Feng and Pulliam [48, 49]. Nixon [109] proposed a co-ordinate straining scheme to address the issue of perturbations in transonic flows, and the method has been applied by Stahara [137] to aerodynamic design problems. A similar shock-fitting methodology has been used by Iollo *et al.* [80] for one dimensional flow, and by Wu *et al.* [158] for two-dimensional flow.

The focus of the present research is to explore the effects of the shock discontinuity in transonic flows upon an *all-at-once* method which exploits the structure of the problem. The method is described in the next section. We also show how using such methods can greatly enhance the efficiency of the optimization process.

1.1 Sequential Quadratic Programming

In an SQP framework [60], we formulate the following problem,

$$\min_z J(z)$$

subject to the (nonlinear) constraints:

$$C(z) = 0$$

Here, $z \in \mathbb{R}^N$, and $C : \mathbb{R}^N \to \mathbb{R}^M$ are a given set of constraints.

The objective function, J, and the constraints, C, are assumed to be sufficiently smooth. We define the Lagrangian function to be

$$L(z, \lambda) = J(z) + \lambda^T(z)C(z)$$

with (first order) necessary optimality conditions given by

$$\begin{array}{rcl} \nabla_z L & = & 0 \\ \nabla_\lambda L & = & 0 \end{array} \tag{1.1}$$

In the *full* SQP procedure [60], we solve a series of successive quadratic approximations of the problem. At the kth step, we have

$$\begin{array}{c} \min_s g_k^T s + \frac{1}{2} s^T H_k s \\ \text{subject to the linear constraints:} \\ A_k s = -C_k \end{array} \tag{1.2}$$

Here $g_k = \nabla_z L_k$ is the gradient, and $H_k = \nabla_{zz} L_k$ is the Hessian of the Lagrangian function at the point, z_k. In most cases, the Hessian is very expensive to compute, and we use an approximation B_k, computed by a method such as the BFGS method [60]. The constraints are linearized, with $A_k = C_z$ corresponding to the Jacobian of the constraints, C, at the point, z_k. If s_k denotes the solution to the above subproblem, we have,

$$z_{k+1} = z_k + \alpha_k s_k,$$

where α_k is a positive scalar indicating the step length, chosen so as to achieve sufficient reduction in an appropriately chosen merit function. The procedure is continued till the optimality conditions are achieved. This is an *all-at-once* method since optimality and feasibility are approached simultaneously. Note that, for $\alpha_k = 1$, we are applying Newton's method to the optimality system (1.1). For further details, see reference [60].

1.1.1 The Reduced Hessian Approach

For a class of problems, such as optimal control problems, the variables, z, can be classified into state variables, q, and control variables, w, depending on the problem structure. In these cases, we can solve for q locally, given some w, and obtain a unique solution for the given constraints, *i.e.*, C_q is invertible. Usually, we might think of the state variables as those that are affected by some physical phenomena modeled by the constraints, while the control variables may be thought of as parameters that affect the constraint physics. In terms of Aerodynamics, the state variables would correspond to the variables that describe the properties of the flow, such as velocity, pressure, etc. The control variables may be parameters that determine the aerodynamic configuration, *e.g.*, the shape of an airfoil, or conditions such as freestream conditions or boundary conditions that affect the nature of the flow. The above problem can be rewritten

$$\min_z J(q, w)$$

subject to the (nonlinear) constraints:

$$\begin{aligned} C(q, w) &= 0 \\ q &\in \mathbb{R}^n \\ w &\in \mathcal{W} \subset \mathbb{R}^m \end{aligned}$$

Here, q is the state variable, and $C : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are the state equations corresponding to q, typically the equations governing the flow, and w is the control (design) variable, with $\mathcal{W}$ denoting some admissible set for the control. Note, $z = [q \quad w]$.

In the *reduced* SQP procedure [67], we make use of the fact that the state and control variables are interdependent, to formulate the problem on a reduced scale. Separating the state and control variables, we have, the optimality conditions (1.1) can be rewritten as,

$$\begin{aligned} \nabla_q J(z) + C_q(z)^T \lambda &= 0 \\ \nabla_w J(z) + C_w(z)^T \lambda &= 0 \\ C(z) &= 0 \end{aligned} \tag{1.3}$$

If Newton's method is applied to the optimality system (1.3), we solve the following system of equations:

$$\begin{pmatrix} \nabla_{zz} L(z, \lambda) & C_z(z)^T \\ C_z(z) & 0 \end{pmatrix} \begin{pmatrix} s_z \\ s_\lambda \end{pmatrix} = - \begin{pmatrix} \nabla_z J(z) + C_z(z)^T \lambda \\ C(z) \end{pmatrix} \tag{1.4}$$

The linearized state equation from the above system can be written as

$$C_q(z) s_q + C_w(z) s_w = -C(z)$$

from which we can solve for s_q in terms of s_w to obtain,

$$s_q = -C_q(z)^{-1}C_w(z)s_w - C_q(z)^{-1}C(z) \tag{1.5}$$

Thus, we can represent the set of solutions for the linearized state equation by,

$$s_z = r(z) + N(z)s_w \tag{1.6}$$

where

$$r(z) = \begin{pmatrix} -C_q(z)^{-1}C(z) \\ 0 \end{pmatrix}, \quad N(z) = \begin{pmatrix} -C_q(z)^{-1}C_w(z) \\ I \end{pmatrix} \tag{1.7}$$

The $(n+m) \times n$ matrix $N(z)$ characterizes the null space of $C_z(z)$. Using this fact, and equations (1.6) and (1.7), we can rewrite (1.4) in the reduced form,

$$\begin{pmatrix} \nabla_{zz}L(z,\lambda)N(z) & C_z(z)^T \end{pmatrix} \begin{pmatrix} s_w \\ s_\lambda \end{pmatrix} = -\left(\nabla_z J(z) + C_z(z)^T\lambda + \nabla_{zz}L(z,\lambda)r(z)\right) \tag{1.8}$$

We can solve the above system for s_w to obtain,

$$s_w = -H(z,\lambda)^{-1}N(z)^T(\nabla_z J(z) + \nabla_{zz}L(z,\lambda)r(z)) \tag{1.9}$$

where, $H(z,\lambda)$ is the reduced Hessian, assumed to be nonsingular, given by,

$$H(z,\lambda) = N(z)^T\nabla_{zz}L(z,\lambda)N(z)$$

Note that the Hessian is projected into the null space of the linearization of the constraints. With the underlying hypothesis that C_q is invertible, the adjoint state can be solved using the first equation in (1.3). The reduced gradient is given by

$$N(z)^T\nabla_z J = \nabla_w F(z) + C_w(z)^T\lambda(z)$$

which can be compared to the left hand side of the second equation in (1.3). If second order information is too expensive to compute, the reduced Hessian is approximated using a quasi-Newton method. The term $N(z)^T\nabla_{zz}L(z,\lambda)r(z)$ is also approximated, using lower order information. The algorithm in Table 1.1 elucidates the basic solution process using a BFGS method [67], with $N(z)^T\nabla_{zz}L(z,\lambda)r(z) = 0$. The algorithm, Trust Region Interior-point algorithms for optimal Control and Engineering (TRICE) has been developed by Heinkenschloss, Vicente and Dennis [40, 68, 67, 151]. The method is globalized using a Trust Region Interior Point algorithm. The augmented Lagrangian is chosen as the merit function,

$$\Phi(z,\lambda;\varrho) = L(z,\lambda) + \varrho\|C(z)\|^2 \tag{1.10}$$

Table 1.1: Reduced SQP Algorithm

(with BFGS approximation of reduced Hessian)

Initialization:
Given $z_0 \in \mathbb{R}^{m+n}$ and $H_0 \in \mathbb{R}^{m\times m}$, H_0 positive definite
Set $k = 0$.

Step k:

Compute the reduced gradient:
(a) Solve $C_q(z_k)^T \lambda = -\nabla_q J(z_k)$.
(b) Compute $N(z_k)^T \nabla_w J(z_k) = C_w(z_k)^T \lambda + \nabla_w J(z_k)$.

(c) Solve $H_k s_w = -N(z_k)^T \nabla_w J(z_k)$.

(d) Solve $C_q(z_k) s_q = -C(z_k) - C_w(z_k) s_w$.

Set $y_{k+1} = y_k + s_y$, $q_{k+1} = q_k + s_q$.

Update H_{k+1}:
Compute $v = N(z_k + N(z_k)s_w)^T \nabla J(z_k + N(z_k)s_w) - T(z_k)^T \nabla J(z_k)$.

If $v^* s_w > 0$, set

$$H_{k+1} = H_k + \frac{vv^T}{v^T s_w} - \frac{(H_k s_w)(H_k s_w)^T}{s_w^T H_k s_w}.$$

Else set $H_{k+1} = H_k$.

where ϱ is the penalty parameter, which can be used to place emphasis on the feasibility of the solution. The penalty parameter is updated using El-Alem's scheme [45]. Details may be found in publications by the authors [40, 68, 67, 151]. Similar work has been done by Kupfer and Sachs [96, 95], by Biegler *et al.* [17, 16], by Byrd *et al.* [27] and by Murray and Prieto [104].

An important observation about the above algorithm is that it can be combined in a natural way with the tools usually available for the solution of optimal control problems. To apply the above algorithm the user would need to provide the solution to the adjoint equation (a), the solution to the linearized state equation (d), and compute the reduced gradient (b). Refer Table 1.1. Since many existing solution procedures would have such information available, either explicitly, or in some implicit form, it is easy to integrate the optimization procedure within existing techniques. Note that we do not compute the null space representation $N(z)$ explicitly, but rather compute the effect of $N(z)$ on the gradient term, $\nabla_w J(z_k)$ in step (b). Progress is simultaneously made towards optimality, with steps (a)–(c), and towards feasibility, with step (d), within a single iteration. The problem is feasible only in the limit as the iteration is converged.

1.1.2 Sparse Sequential Quadratic Programming

For a lot of problems, the quadratic subproblem to be solved at each iterate in the SQP method involves Jacobian and Hessian matrices that are sparse, *i.e.*, they have a lot of zero elements. A lot of savings can be obtained in terms of storage if the zero elements are not stored. A number of techniques have been developed for storing such matrices and for solving the resulting linear systems in an efficient manner[62, 10]. Betts [14, 15, 77, 13] has developed a Sparse Nonlinear Programming (SNLP) algorithm based on the SQP method which can be used to efficiently solve large scale optimization problems having a sparse structure. The algorithm requires the user to provide routines to compute the objective function to be minimized, to evaluate the (nonlinear) constraints, and to provide gradient and Hessian data in sparse form.

1.2 Applications to Aerodynamic Design

As an example of how we can use the *all-at-once* method described in the previous section to improve the efficiency of the design process, in Chapter 2 we apply the algorithm described to obtain solutions to an inverse design problem to obtain the shape of a nozzle housing one-dimensional transonic flow. The flow is governed by the Euler equations.

Initially the problem is cast as an optimal control problem, which enables us to 'solve' it as a two-point boundary value problem. This provides good insight into the nature of the solutions. Using a shock-capturing discretization with the reduced SQP algorithm yields poor performance. We show that the poor convergence behavior exhibited is due to the lack of continuity in the infinite-dimensional problem. The difficulty is caused by the presence of the shock. We show how this problem can be alleviated by using a shock-fitting formulation.

Other aspects of this design problem and other methods for its solution have been studied by Iollo, Salas and Ta'asan [82], by Borggaard [22], by Narducci, Grossman, and Haftka [105], and by Feng and Pulliam [48, 49]. In [82] a target pressure distribution is used along with the full 1-D Euler equations. The unsteady Euler equations are used in conjunction with a pseudo-transient adjoint equation formulation to drive the system to optimality. Both shock-free and transonic solutions are investigated. In [22] the flow variables are viewed as functions of the designs and a sensitivity equation approach is used to compute the gradient. In [105] sensitivities for various (semi-) discretizations are studied. Although the shock location is treated implicitly, it enters the discretization scheme for the objective function. A coordinate straining method is used to smoothen the objective function. The design problem is solved numerically using the black-box-approach with a steepest descent method. Feng and Pulliam [48] have used an all-at-once reduced Hessian SQP scheme, which does not require the solution of the adjoint equations. They use nonlinear artificial dissipation with the full 1-D Euler equations. In [49] they use a solution refining scheme to improve the efficiency and robustness of the process.

In Chapter 3 we analyze a temperature control problem. The temperature along the side of a duct is to be regulated by controlling the temperature profile of the incoming flow to the duct. The flow is governed by the two-dimensional Navier-Stokes equations. The problem has been analyzed by Gunzburger and Lee [63]. As a simplification, the flow is assumed to be uncoupled from the energy equation. We show that the optimization problem is an exact QP problem. The structure of the problem results in a sparsity pattern which can be efficiently exploited using the SNLP algorithm described in Section 1.1.2.

A two-dimensional conceptualization of the shock-fitting method described in Chapter 2 would correspond to the co-ordinate straining method proposed by Nixon [109]. However, the method would not be practical for higher dimensions. In Chapter 4, we develop a technique based on an Euler Implicit regularization which can be extended to three dimensions. In Chapter 4 we explore solutions for an airfoil design problem using the Reduced Hessian formulation. The objective is to design the shape of an airfoil in order to maximize the lift. The shape of the airfoil is parameterized in terms of four existing NACA airfoils. The problem has been formulated by Vanderplaats [149] and solved by Joh *et al.* [90] and Narducci [107]. Vanderplaats [149] and Joh *et al.* [90] used sequential approx-

imation optimization to obtain solutions while Narducci *et al.* [106, 107] used a response surface method. An alternate formulation has been proposed by Verhoff *et al.* [150], using Chebychev polynomials, and by Sadrehaghighi *et al.* [120], where the airfoil shape has been parameterized using Bezier curves.

The dissertation ends with a few concluding remarks about the present research and opportunities for further efforts in this direction.

Chapter 2

One-Dimensional Nozzle Design

2.1 Introduction

The one-dimensional compressible flow of an inviscid fluid in a duct with spatially variable cross-sectional area $A(x)$ is considered here. The steady flow is governed by the Euler equations[55]

$$\mathcal{F}_x + \mathcal{G} = 0, \quad 0 \leq x \leq 1, \tag{2.1}$$

where

$$\mathcal{F} = \begin{pmatrix} (\rho u)A \\ (\rho u^2 + p)A \\ (\rho E + p)uA \end{pmatrix}, \quad \mathcal{G} = \begin{pmatrix} 0 \\ -pA_x \\ 0 \end{pmatrix}.$$

Standard notation has been used, with ρ being the fluid density, u the velocity, $E \equiv e + u^2/2$, where e is specific internal energy, and p is the fluid static pressure, computed from the equation of state for a perfect gas, $p = (\gamma - 1)\rho e$, where $\gamma > 1$ is a gas constant (for air, $\gamma = 1.4$). The subscript x denotes differentiation with respect to the position along the streamwise direction x. Details on the derivation of the Euler equations may be found in any book on compressible fluid dynamics, such as Anderson [8]. A good reference text on the subject is provided by Eberle *et al.* [43]. It is assumed that the cross-sectional area $A(x)$ of the duct along the streamwise direction x is at least piecewise smooth and monotonically increasing:

$$A(x) > 0, \quad A_x(x) > 0, \quad x \in [0, 1]. \tag{2.2}$$

Frank and Shubin [55] note that solutions of this simple model exhibit phenomena quite similar to those in two dimensional inviscid flow over an airfoil. The one-dimensional

transonic duct flow problem, hence, serves a useful purpose in that it provides valuable insight into the nature of the relatively complex problem of airfoil design, addressed in Chapter 4. The existence of a well-known analytic solution for the flow, the form of which is relatively simple to interpret, makes it a good test-case for trying out solution procedures used in aerodynamic design. As a result, it has been extensively used as a *benchmark* for testing various numerical methods for the solution of flows with shocks.

In the analysis problem, given a particular area distribution $A(x)$ which is monotonically non-decreasing in nature (*i.e.*, $A_x \geq 0$), we determine the flow solution. We are interested in an (inverse) design problem governed by the equations (2.1). More precisely, given a desired velocity $\hat{u}$ we want to find an area profile $A(x)$ obeying (2.2) and generating a flow that best fits the desired flow $\hat{u}$ in the least squares sense. In a first formulation the design problem can be stated as follows:

$$\text{Minimize} \int_0^1 (u(x) - \hat{u}(x))^2 \, dx \tag{2.3}$$

over all u and A satisfying (2.1), (2.2), and certain boundary conditions to be specified. A detailed formulation of the state equation will be given in Section 2.2.

As an initial study, we formulate the problem as a problem of optimal control [127]. In Section 2.3 Pontryagin's Minimum Principle is used to characterize candidate minimizers which are shown to contain singular arcs. Numerical studies show that generic solutions include interior singular arcs, and that arcs with bounded control occur at the inlet, the outlet, and adjacent to the shock.

The results obtained in Section 2.3 indicate that we need to impose the stricter condition of absolute continuity on the area distribution in order to obtain more realistic results. Here, we parameterize the area by a few variables, while the governing equations are approximated by a finite difference scheme. The finite difference approximation can result in a large scale optimization problem, depending on the level of discretization. In Section 2.4 we show the results obtained for a case where we include the linearized finite-dimensional constraints within the optimization algorithm, which effectively reduces the dimensionality [118].

This design problem is difficult to solve numerically, because the duct flow has a shock. Many good numerical shock-capturing schemes, such as the Godunov scheme, used in Section 2.4, have low continuity properties, see *e.g.*, [72, 73]. On the other hand, efficient numerical optimization schemes require sufficiently smooth cost and constraint functions. A straightforward combination of off-the-shelf discretization schemes for the flow equations and of off-the-shelf optimization methods often leads to very unsatisfactory results. Frank and Shubin [55] have considered this problem. For the numerical solution of the design problem they used standard schemes for the discretization of the flow equation and then

applied two optimization approaches to solve the discretized problem. The first optimization approach considered in [55] is the black-box method in which the flow variables are considered to be functions of the design variables A, implicitly defined by the flow equation. The resulting optimization problem is then solved using a Newton-like method. The other optimization approach considered was the all-at-once method in which the flow and the design are treated as independent variables. The flow equations are viewed as constraints of the optimization problem. A sequential quadratic programming (SQP) method is applied to the solution of this optimization problem. One of the findings reported in [55] is that their implementations of the black-box approach were more robust than their implementation of the all-at-once method. Even though there may be a lack of smoothness in the functions, the black-box-approach using *e.g.*, finite difference derivative approximations seems to tolerate this. However, if the all-at-once method converged, then, as expected, it was much faster than the black-box approach. These findings are collaborated by our study in Section 2.4.

The failure of the SQP method is related to the presence of shocks in the flow. In particular, if shock-capturing schemes with low continuity properties, like the Godunov scheme, are used, then the SQP method, which is designed for smooth problems, behaves poorly. This seeming incompatibility of good shock-capturing schemes for the discretization of the flow equations and efficient SQP methods for the solution of the optimization problem motivated the use of a shock-fitting formulation [36]. Our approach to this problem, in Section 2.5 is an extended and refined version of the all-at-once approach in [55]. The important extension is that we include the shock location as a state variable. This guarantees differentiability of the appropriate mappings. Moreover, we use a discretization of the flow equations that is very similar to the Godunov scheme. The formulation gives a sharp shock and, since the shock location is an explicit variable, the map from the design parameters to the flow is differentiable. In this section, we give a rigorous analysis of the infinite dimensional design problem including existence of optimal designs, existence of Fréchet derivatives, and existence of Lagrange multipliers. In particular we will show that the co-state is discontinuous at the shock location, unless the target velocity can be matched perfectly. The second part of the section is devoted to the numerical solution of the problem. We discuss the discretized design problem and investigate the relation between the finite dimensional problem and the discretized one. The careful study of this relation gives valuable insight and reveals information that is shown to be important for the performance of the optimization algorithm.

2.2 The One-Dimensional Nozzle Flow

It has been shown by Frank and Shubin [55] that equation (2.1) can be reduced to a single ordinary differential equation in u. In fact, (2.1) is equivalent to the ODE

$$(f(u))_x + g(u, A) = 0, \tag{2.4}$$

where

$$f(u) \equiv u + \bar{H}/u, \quad g(u, A) \equiv \frac{A_x}{A}(\bar{\gamma}u - \bar{H}/u), \tag{2.5}$$

and where

$$\bar{\gamma} = (\gamma - 1)/(\gamma + 1), \quad \bar{H} = 2H\bar{\gamma}$$

are given constants. The constant $\gamma > 1$ is the gas constant (for air, $\gamma = 1.4$), and the constant H is the total enthalpy. The flow is supersonic for $u > u_* \equiv \sqrt{\bar{H}}$ and subsonic for $u < u_*$.

In addition, we impose the following boundary conditions

$$u(0) = u_{\text{in}}, \quad u(1) = u_{\text{out}}. \tag{2.6}$$

We choose boundary data $u_{\text{in}} > u_* > u_{\text{out}}$ so that a solution u of (2.4), (2.6) has a jump from supersonic to subsonic at some point x_s. At the shock location x_s the flow is required to satisfy the Rankine-Hugoniot relation

$$f(u(x_s^-)) = f(u(x_s^+)) \tag{2.7}$$

or, equivalently,

$$u(x_s^-) \cdot u(x_s^+) = \bar{H}. \tag{2.8}$$

As usual, $u(x_s^-) = \lim_{h \to 0^+} u(x_s - h)$ and $u(x_s^+) = \lim_{h \to 0^+} u(x_s + h)$. Equation (2.4) along with the above conditions (2.6) and (2.7) defines the flow profile.

For sake of completeness, we review the results presented by Frank and Shubin [55] concerning the existence of solutions to (2.4), (2.6), and (2.7). This existence result will not be needed in this form, but the arguments applied for its proof give some important insight into the structure of the problem.

First we consider the initial value problems

$$(f(u))_x + g(u, A) = 0, \quad u(0) = u_{\text{in}} \tag{2.9}$$

and

$$(f(u))_x + g(u, A) = 0, \quad u(1) = u_{\text{out}}. \tag{2.10}$$

Since $f_u(u) > 0$ for $u > \sqrt{\bar{H}}$, there exists a solution of (2.9) in a neighborhood $[0, x_L)$ of $x = 0$ provided $u_{\text{in}} > \sqrt{\bar{H}}$. If $u_{\text{in}} \in (\sqrt{\bar{H}}, \sqrt{2H})$ and $u(1) = u_{\text{out}} < \sqrt{\bar{H}}$, then (2.2) and the definitions of f, g imply that

$$u_x(x) = -\frac{g(u(x), A(x))}{f_u(u(x))} \quad \begin{cases} > 0 & x = 0, \\ < 0 & x = 1. \end{cases}$$

Using the continuity of the solution and bootstrapping, we can deduce that unique solutions of (2.9) and (2.10) exist on some interval $[0, x_L)$ and $(x_R, 1]$, respectively. Moreover, the solutions are monotonically increasing and decreasing, respectively. It is easy to verify that the solutions are implicitly given by

$$A(x)u(x)\Big(2H - u^2(x)\Big)^r = \begin{cases} K_L, & x \in [0, x_L), \\ K_R, & x \in (x_R, 1], \end{cases} \tag{2.11}$$

where $r = 1/(\gamma - 1)$ and where the constants K_L, K_R are determined from $A(0) = A_{\text{in}}$, $u(0) = u_{\text{in}}$ and $A(1) = A_{\text{out}}$, $u(1) = u_{\text{out}}$. Due to the restrictions on the boundary conditions and due to the fact that $A(x) > 0$, the constants K_L, K_R are positive. Equation (2.11) defines two functions $A_L(u), A_R(u)$. It can be easily verified that these functions have a minimum at $u_* = \sqrt{\bar{H}}$ and are strictly monotone on $(0, \sqrt{\bar{H}}]$ and on $[\sqrt{\bar{H}}, \sqrt{2H})$. Thus, the initial condition $u_{\text{in}} \in (\sqrt{\bar{H}}, \sqrt{2H})$ guarantees that the solution u of (2.9) exists on $[0, x_L) = [0, \infty)$. The point x_R is the uniquely defined point satisfying $A(x_R) = A_R^*$. The situation is sketched in Figure 2.1.

Using (2.8), (2.11), and the continuity of A, the shock position x_s can be characterized by

$$l(u(x_s^-)) = \frac{1}{K_L} u(x_s^-) \left(2H - u^2(x_s^-)\right)^r - \frac{1}{K_R} \frac{\bar{H}}{u(x_s^-)} \left(2H - \left(\frac{\bar{H}}{u(x_s^-)}\right)^2\right)^r = 0. \tag{2.12}$$

It is easy to see that $\lim_{u \to \sqrt{2H}^-} l(u) < 0$. Hence, given A there exists a boundary conditions $u_{\text{in}} \in (\sqrt{\bar{H}}, \sqrt{2H})$, $u_{\text{out}} \in (0, \sqrt{\bar{H}})$, *i.e.*, K_L, K_R, such that $\lim_{u \to u_{\text{in}}^+} l(u) > 0$. In this case there exists $u(x_s^-)$ such that $l(u(x_s^-)) = 0$. Since the area A is monotonically increasing, the shock condition can then be computed from (2.11). Thus, we can conclude the following result:

Theorem 2.2.1 *Suppose the area function satisfies (2.2). Then there exist boundary conditions* $u_{\text{in}} \in (\sqrt{\bar{H}}, \sqrt{2H})$ *and* $u_{\text{out}} \in (0, \sqrt{\bar{H}})$ *such that the equations (2.4), (2.7), and (2.6)*

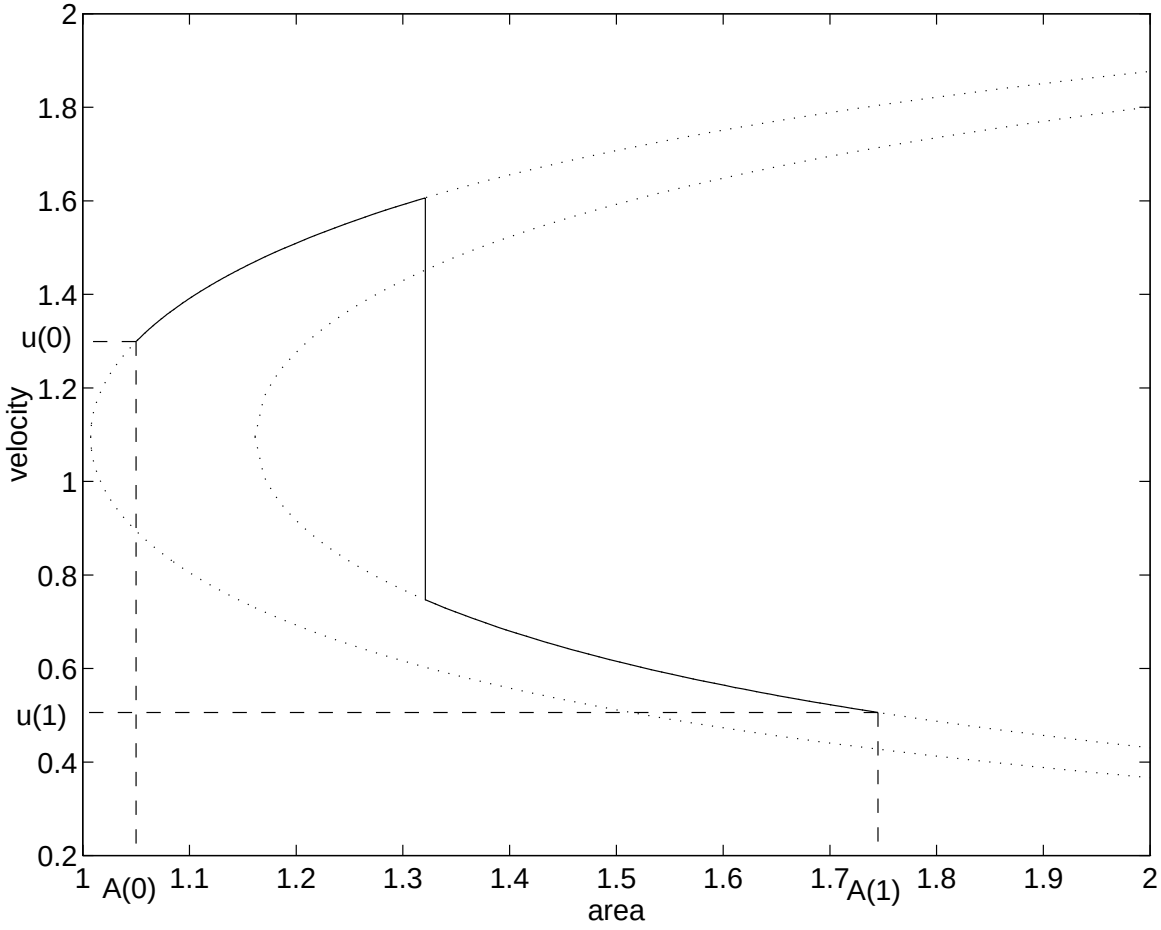


Figure 2.1: Sketch of the velocity as a function of the area.

admit a unique solution u which is supersonic and monotonically increasing on $(0, x_s)$ *and subsonic and monotonically decreasing on* $(x_s, 1)$. *Moreover, the function* $u(\cdot)$ *obeys the inequalities*

$$\begin{array}{rll} \sqrt{\bar{H}} < u_{\text{in}} \leq \; u(x) & < \min\left\{\sqrt{2\bar{H}}, \bar{H}/u_{\text{out}}\right\}, & x \in [0, x_s), \\ u_{\text{out}} \leq \; u(x) & < \bar{H}/u_{\text{in}} < \sqrt{\bar{H}}, & x \in (x_s, 1]. \end{array} \tag{2.13}$$

2.3 An Optimal Control Formulation of the Problem

A particularly convenient aspect of this problem for the "control engineer" is that the problem entails only one independent variable. Thus it lends itself very easily to a standard optimal control formulation. This observation has lead to the motivation for the present study. Simply stated, one can pose the problem thus: the fluid velocity and the area are considered as states, and the variation of the area, *i.e.*, the "slope" of the area distribution curve, is treated as a control. Loosely said, the problem, then, is to determine how to "control" the area distribution in order to minimize the deviation of the obtained velocity from a desired profile. Rewriting equation (2.4) we pose the problem as follows:

Given,

$$\begin{aligned} u'(x) &= -\frac{w(x)}{A(x)}\frac{(\bar{\gamma}u(x) - \bar{H}/u(x))}{(1 - \bar{H}/u(x)^2)}, \quad \text{and} \\ A'(x) &= w(x), \end{aligned}$$

we seek a function $w(\cdot)$ to

$$\min_w J = \frac{1}{2}\int_0^1 (u(x) - \hat{u}(x))^2 dx$$

subject to:

$$\begin{aligned} w_{\min} \leq \; w &\leq w_{\max}, \\ u(0) &= u_{\text{in}}, \\ A(0) &= A_{\text{in}}, \\ u(1) &= u_{\text{out}}, \\ A(1) &= A_{\text{out}}, \quad \text{with} \end{aligned}$$

$$u_L \cdot u_R - \bar{H} = 0 \quad \text{at the shock position, } x = x_s.$$

A potential difficulty with this formulation is that u' is not defined at $x = x_s$. This could lead to numerical difficulties. One way to circumvent this is to divide the domain into the pre-shock and post-shock regions. As a matter of convenience, we also prefer to reformulate the above problem as a Mayer type, *i.e.*, we include $J(= \frac{1}{2}\int(u-\hat{u})^2)$ as an additional state and then minimize J(1).

We rescale the problem so that the scaled length in each region is unity. In the post-shock region we integrate backwards so that the shock appears at the end on the interval. The terminal condition on the states then becomes an initial condition on the new scale, and the Rankine-Hugoniot relation becomes a terminal condition for the problem.

For domain L (supersonic region), using the transformation, $\xi = x/x_s$, we have,

$$\begin{aligned} u'_L(\xi) &= -x_s \frac{w_L(\xi)}{A_L(\xi)} \frac{(\bar{\gamma} u_L(\xi) - \bar{H}/u_L(\xi))}{(1 - \bar{H}/u_L(\xi)^2)} \\ A'_L(\xi) &= x_s w_L(\xi) \\ J'_L(\xi) &= \frac{x_s}{2}(u_L(\xi) - \hat{u_L}(\xi))^2 \end{aligned} \tag{2.14}$$

Here, the subscript L denotes the variables as measured in domain L. Note that now the independent variable is scaled. Similarly, for domain R, we use the transformation $\xi = (1-x)/(1-x_s)$, to obtain

$$\begin{aligned} u'_R(\xi) &= (1 - x_s) \frac{w_R(\xi)}{A_R(\xi)} \frac{(\bar{\gamma} u_R(\xi) - \bar{H}/u_R(\xi))}{(1 - \bar{H}/u_R(\xi)^2)} \\ A'_R(\xi) &= -(1 - x_s) w_R(\xi) \\ J'_R(\xi) &= \frac{(1 - x_s)}{R}(u_R(\xi) - \hat{u_R}(\xi))^2 \end{aligned} \tag{2.15}$$

Note that we have mapped each region to the interval [0,1]. Thus, we can treat the two sets of equations as a single system of differential equations. The problem then becomes,

Given (2.14) and (2.15),

$$\min_{w_L, w_R} J_L(1) + J_R(1) \tag{2.16}$$

subject to:

$$\begin{aligned} w_{\min} &\leq w_L \leq w_{\max}, \\ w_{\min} &\leq w_R \leq w_{\max}, \end{aligned}$$

$$\begin{aligned} u_L(0) &= u_{\text{in}}, \\ A_L(0) &= A_{\text{in}}, \\ J_L(0) &= 0, \\ u_R(0) &= u_{\text{out}}, \\ A_R(0) &= A_{\text{out}}, \\ J_R(0) &= 0, \end{aligned} \tag{2.17}$$

with

$$\begin{aligned} u_L(\xi) \cdot u_R(\xi) - \bar{H} &= 0, \quad \text{and} \\ A_L(\xi) - A_R(\xi) &= 0 \quad \text{at } \xi = 1. \end{aligned}$$

Note that our problem includes specified boundary conditions on the inlet and outlet areas. Specification of either one (say, the inlet area) amounts to normalization. In a real design problem this might be replaced by specification of reservoir conditions and a mass-flow rate. The outlet area could be specified or left free. The former is typical of geometric constraints that arise in nozzle re-design.

Furthermore, it must be noted that the problem description is somewhat formal in that we have not specified the admissible class of "data" functions $\hat{u}(\cdot)$. For subsequent analysis we assume that $\hat{u}$ is smooth on the intervals $(0,\hat{x}_s)$ and $(\hat{x}_s,1)$ and that $\hat{x}_s \in (0,1)$ is given. In addition we seek flow solutions $u(\cdot)$ that have the same smoothness properties. In particular the shock location, x_s for each flow solution is fixed at the location specified by the data. Thus, one might generalize our problem to allow the shock in the flow solution to be "free"; its location to be determined by optimality. For the flow-matching problem, one would not expect the shock in the optimal "matching" flow to be very far from the shock in the specified flow $\hat{u}(\cdot)$. We shall say more about this feature later.

2.3.1 Solution as a Boundary Value Problem

To solve the optimal control problem [26], we form the usual variational Hamiltonian,

$$\mathcal{H} = \lambda^T C, \quad \text{where,}$$

C is the right hand side of the system differential equations ($q' = C(q,w)$), with q denoting the states, and w denoting the controls. The adjoint variables, λ, are then given by the differential equations,

$$\lambda' = -\frac{\partial \mathcal{H}}{\partial q},$$

with terminal conditions given by,

$$\lambda(1) = \frac{\partial \phi}{\partial q} + \nu^T \frac{\partial \psi}{\partial q}, \quad \text{where}$$

ψ denotes the terminal conditions on the states, ϕ is the terminal cost function, and ν is a vector of Lagrange multipliers. The optimality condition ($\min_w \mathcal{H}$) determines the optimal control.

For our problem, we get,

$$\begin{aligned}
\lambda'_{u_L} &= \lambda_{u_L} x_s \frac{w_L}{A_L}\left[\frac{(\bar{\gamma} + \bar{H}/u_L^2)}{(1-\bar{H}/u_L^2)} - \frac{(\bar{\gamma} u_L - \bar{H}/u_L)}{(1-\bar{H}/u_L^2)^2}\frac{2\bar{H}}{u_L^3}\right] - \lambda_{J_L} x_s (u_L - \hat{u}_L) \\
\lambda'_{A_L} &= -\lambda_{u_L} x_s \frac{w_L}{A_L^2}\frac{(\bar{\gamma} u_L - \bar{H}/u_L)}{(1-\bar{H}/u_L^2)} \\
\lambda'_{J_L} &= 0 \\
\lambda'_{u_R} &= \lambda_{u_R} (x_s - 1) \frac{w_R}{A_R}\left[\frac{(\bar{\gamma} + \bar{H}/u_R^2)}{(1-\bar{H}/u_R^2)} - \frac{(\bar{\gamma} u_R - \bar{H}/u_R)}{(1-\bar{H}/u_R^2)^2}\frac{2\bar{H}}{u_R^3}\right] - \lambda_{J_R} (1 - x_s)(u_R - \hat{u}_R) \\
\lambda'_{A_R} &= \lambda_{u_R} (1 - x_s) \frac{w_R}{A_R^2}\frac{(\bar{\gamma} u_R - \bar{H}/u_R)}{(1-\bar{H}/u_R^2)} \\
\lambda'_{J_R} &= 0
\end{aligned} \tag{2.18}$$

The subscripts for the adjoint variables denote association with the particular state. The transversality conditions for the adjoint variables give,

$$\begin{aligned}
\lambda_{u_L}(1) u_L(1) &= \lambda_{u_R}(1) u_R(1), \\
\lambda_{A_L}(1) + \lambda_{A_R}(1) &= 0, \\
\lambda_{J_L}(1) &= 1, \quad \text{and} \\
\lambda_{J_R}(1) &= 1
\end{aligned} \tag{2.19}$$

Leaving the velocities free at the two ends gives us the conditions,

$$\begin{aligned}
\lambda_{u_L}(0) &= 0, \quad \text{and} \\
\lambda_{u_R}(0) &= 0
\end{aligned} \tag{2.20}$$

The optimality condition ($\min_{w_L} \mathcal{H}$) gives,

$$w_L = \begin{cases} w_{\min} & \text{when } S_L > 0, \\ w_{\max} & \text{when } S_L < 0, \\ \text{singular} & \text{when } S_L = 0, \end{cases} \tag{2.21}$$

where the switching function, S_L, is defined as

$$S_L = -\lambda_{u_L}\frac{(\bar{\gamma}u_L - \bar{H}/u_L)}{A_L(1-\bar{H}/u_L^2)} + \lambda_{A_L}$$

When $S_L = 0$, we have a case of singular control. To find a control $w_L(\cdot)$ that will keep us on the singular arc, we require,

$$S_L' = 0$$
$$\Rightarrow \lambda_{J_L}\frac{(\bar{\gamma}u_L - \bar{H}/u_L)}{A_L(1-\bar{H}/u_L^2)}(u_L - \hat{u}_L) = 0$$

Now, from the transversality conditions (2.19), $\lambda_{J_L} = 1(\neq 0)$. Also, from the physics of the problem it can be seen that $(\bar{\gamma}u_L - \bar{H}/u_L) \neq 0$. This implies that

$$u_L - \hat{u}_L = 0 \tag{2.22}$$

Thus, along a singular arc, the flow solution matches the design profile exactly. From $S_L'' = 0$, we get the singular control,

$$w_L(\xi) = -\frac{A_L\hat{u}_L'(1-\bar{H}/\hat{u}_L^2)}{x_s(\bar{\gamma}\hat{u}_L - \bar{H}/\hat{u}_L)} \tag{2.23}$$

Similarly, the optimality condition ($\min_{w_R}\mathcal{H}$) gives,

$$w_R = \begin{cases} w_{\min} & \text{when } S_R > 0, \\ w_{\max} & \text{when } S_R < 0, \\ \text{singular} & \text{when } S_R = 0, \end{cases} \tag{2.24}$$

where,

$$S_R = \lambda_{u_R}\frac{(\bar{\gamma}u_R - \bar{H}/u_R)}{A_R(1-\bar{H}/u_R^2)} - \lambda_{A_R}$$

Examining the singular case as before, gives

$$S_R' = 0$$
$$\Rightarrow \lambda_{J_R}\frac{(\bar{\gamma}u_R - \bar{H}/u_R)}{A_R(1-\bar{H}/u_R^2)}(u_R - \hat{u}_R) = 0$$

which gives,

$$u_R - \hat{u}_R = 0 \tag{2.25}$$

From $S''_R = 0$, we get the singular control,

$$w_R(\xi) = \frac{A_R \hat{u}'_R (1 - \bar{H}/\hat{u}_R^2)}{(1 - x_s)(\bar{\gamma}\hat{u}_R - \bar{H}/\hat{u}_R)} \tag{2.26}$$

From (2.22) and (2.25) it is clear that the singular case corresponds to the situation wherein we use the flow equation (2.4) to "solve" for A_x, given the flow velocity $u(x) = \hat{u}(x)$. As noted in [55], this amounts to a first-order equation for the area, $A(\cdot)$, and except in contrived cases, one does not expect to satisfy the specified boundary conditions on A. Hence, one expects the solution to consist of singular arcs *and* arcs with controls on the boundary. Thus, the slope bounds $w_{\min}$ and $w_{\max}$ are important; without these bounds the flow-matching problem with generic data will not have a solution.

2.3.2 Numerical Results

To gain some preliminary insight into the nature of the problem, we explored "solutions" using the Graphical Environment for Solving Optimization Problems (GESOP) software on a Silicon Graphics IRIX workstation. XGESOP is implemented in an X-Window environment for the UNIX Operating System. It is a commercial version of the Advanced Launcher Trajectory Optimization Software (XALTOS) [37, 89, 123], used by the European Space Agency for its research, and was made available to the Interdisciplinary Center for Applied Mathematics (ICAM), at Virginia Tech, by Dr. K.-H. Well of the University of Stuttgart.

The software provides a graphical interface to facilitate interaction between the user and the optimization software. The optimization is implemented using either PROMIS or TROPIC, which are Nonlinear Programming codes for an optimal control formulation. PROMIS uses multiple shooting, while TROPIC uses a direct collocation method. Details on these methods can be found in reference [155].

The numerical data that was used for the initial experiment are the same as those used by Frank and Shubin [55]. The values of the area at the two ends of the duct are fixed at $A_{\text{in}} = 1.05$ and $A_{\text{out}} = 1.745$. As a preliminary study, we explore solutions with boundary conditions imposed on the fluid velocity, given by $u_{\text{in}} = 1.299$ and $u_{\text{out}} = 0.506$, respectively. We use $\gamma = 1.4$ and $H = 3.6$.

Thus for our problem, we have the following boundary conditions:

$$\begin{aligned} u_L(0) &= 1.299, \\ A_L(0) &= 1.050, \\ u_R(0) &= 0.506, \\ A_R(0) &= 1.745, \end{aligned}$$

with the terminal conditions

$$\begin{aligned} u_L \cdot u_R - 1.2 &= 0 \quad \text{and} \\ A_L - A_R &= 0 \quad \text{at } \xi = 1. \end{aligned}$$

The analytical solution to this problem using equation (2.11) is shown in Figure 2.1. Note that this results in velocity as a function of area.

For ease of implementation we specify a target velocity profile, $\hat{u}(\cdot)$, which is piecewise linear in x. We have specified the shock location at $x_s = 0.75$. The corresponding profiles for the velocity and area are shown in Figure 2.2.

The optimization was implemented using PROMIS [123], which transcribes the optimal control problem to a finite-dimensional Nonlinear Programming problem (NLP). As described in the preceding section, we expect the solutions for the optimal control to consist of singular arcs, where the flow exactly matches the design profile, and of arcs where the control is up against one of its bounds. For the case described above, the solution would be totally singular. In the NLP formulation the controls are approximated by piecewise cubic polynomials, and the states are solved by numerical integration, using a multiple shooting grid, with the states specified at the grid points. For the nominal case, grid points placed at the ends of the interval suffice, with the controls given by piecewise cubic splines in the shooting meshes. The initial guess was arbitrarily made, keeping in mind, however, the expected trends in the states and controls. The results obtained match very well with the analytical results, and we are able to recover the desired flow profile perfectly. The above case being one where the optimal flow profile generated exactly matches the design velocity distribution, it shall henceforth be referred to as the nominal case.

Changing either of the boundary conditions on the area will mean that the flow profile cannot be exactly matched. In that case, one would try to obtain a least-squares fit to the desired profile. Grid points are placed at points where the controls might be expected to switch from singular to non-singular arcs, to enable the control to be discontinuous at these points. Along the singular arcs the controls are approximated by piecewise cubic splines as before, while along the non-singular arcs they are approximated by piecewise constant functions. As an example, the outlet cross-sectional area was changed from the nominal value of 1.745. Bounds are placed on the controls, $0 \leq w_L, w_R \leq 3$. The imposition of

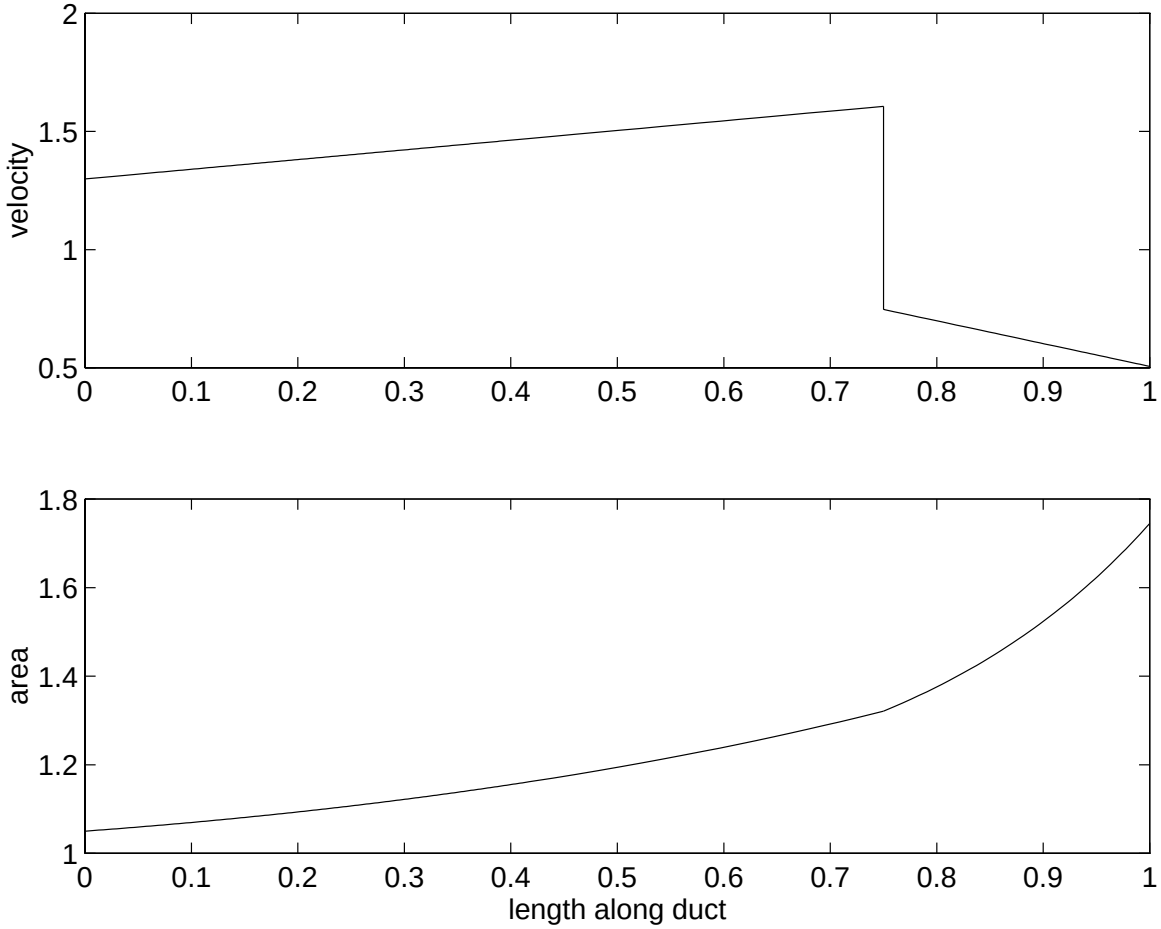


Figure 2.2: Desired linear velocity profile for $x_s = 0.75$, with corresponding Area distribution.

bounds is necessary for the problem to make physical sense, as is discussed previously. The results are shown for the cases $A_{\text{out}} = 1.6$ and $A_{\text{out}} = 2.0$ in Figures 2.3 and 2.4, respectively.

When the outlet area is decreased, the optimal solution indicates that the velocity profile is exactly matched from the inlet to a position near the shock. Approaching the shock, the control, *i.e.*, the slope of the area distribution jumps to the lower bound and following the shock, the control switches to the upper bound. Though it might seem strange at first glance that the slope must hit the upper bound causing an increase in the area, inspite of our requirement that the area be less than the nominal value, this is necessitated by the imposition of the Rankine-Hugoniot condition at the shock location. Towards the outlet, the flow again exactly matches the desired profile. Figure 2.3 shows the obtained velocity profile, the area distribution and the slope of the area along the length of the duct.

Increasing the outlet area causes the control to hit the upper bound approaching the shock position in the supersonic region, following which it jumps to the lower bound in the post-shock subsonic region. The optimal velocity profile and the associated area distribution and slope of the area are shown in Figure 2.4.

In both cases, as the boundary condition on the outlet area is moved further away from the nominal value, the regions where the control is up against the bounds plays an increasingly dominant role. As a matter of practicality, for a flow-matching problem, the ultimate objective would be to obtain the best possible fit to a design flow profile. In general, we would not impose any conditions on the inlet and outlet velocities. Indeed, one would expect to be able to provide a better fit to the desired velocity distribution by leaving the velocity free at the two ends of the duct. Results for the two cases above, with no boundary conditions imposed on the velocity, are shown in Figures 2.5 and 2.6, respectively.

By forcing the control against the bounds for a brief distance at the two ends of the ducts the regions, close to the shock, where the controls were at the bounds earlier, are now reduced considerably. The optimal flow profile, as indicated by the results, now matches the exact profile for much larger regions. The performance index could be seen to be brought down drastically as a result. For the case when the outlet duct area was fixed at 1.6, the performance index decreased from 1.386×10^{-3} to 1.935×10^{-5}. The results are shown in Figure 2.5. When the outlet area was fixed at 2.0, the performance index was reduced from 5.186×10^{-4} to 1.023×10^{-4}. The results for this case are shown in Figure 2.6.

To obtain more precise results, the boundary value problem described in Section 2.3.1, was solved numerically. Note, for this case the end-velocities are free, as in the latter case of our preliminary studies. As before, we confine our studies to a linear velocity profile, for the present. Numerical results were obtained invoking a Gauss-Newton method for solving

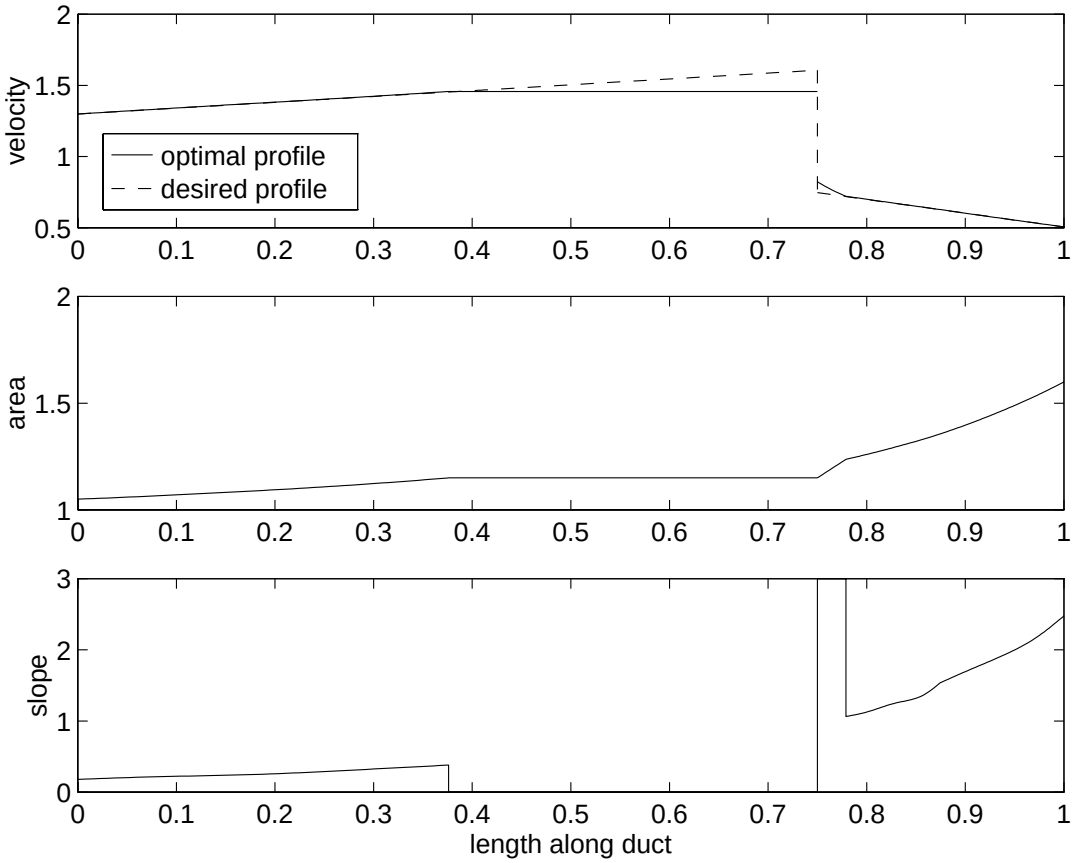


Figure 2.3: Optimal Profile for $A_{\text{out}} = 1.6$, with end-velocities fixed

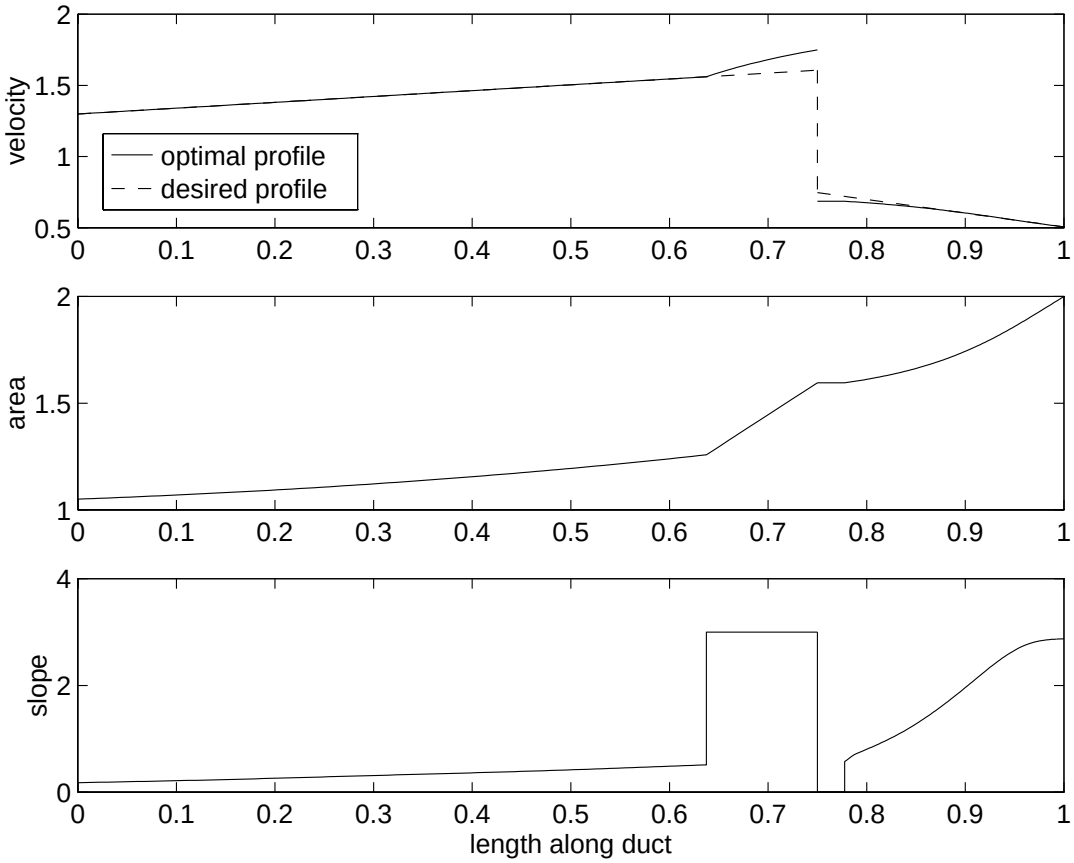


Figure 2.4: Optimal Profile for $A_{\text{out}} = 2.0$, with end-velocities fixed

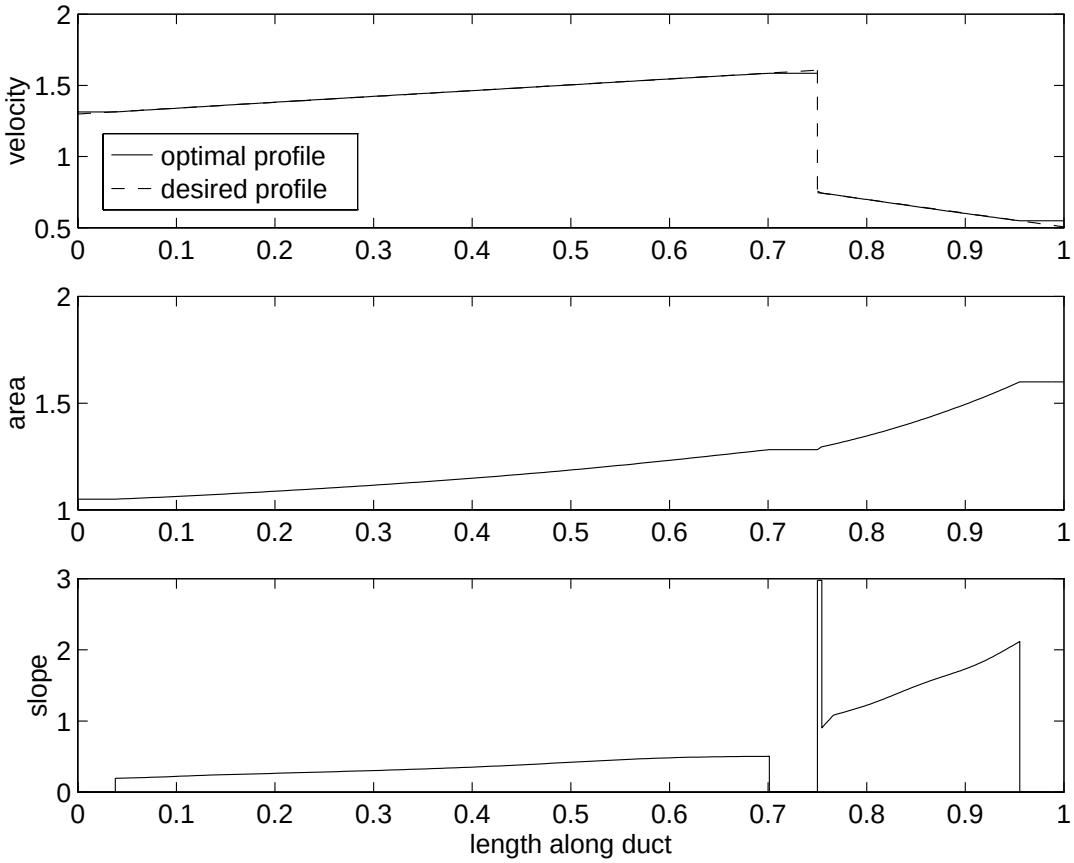


Figure 2.5: Optimal Profile for $A_{\text{out}} = 1.6$, with end-velocities free

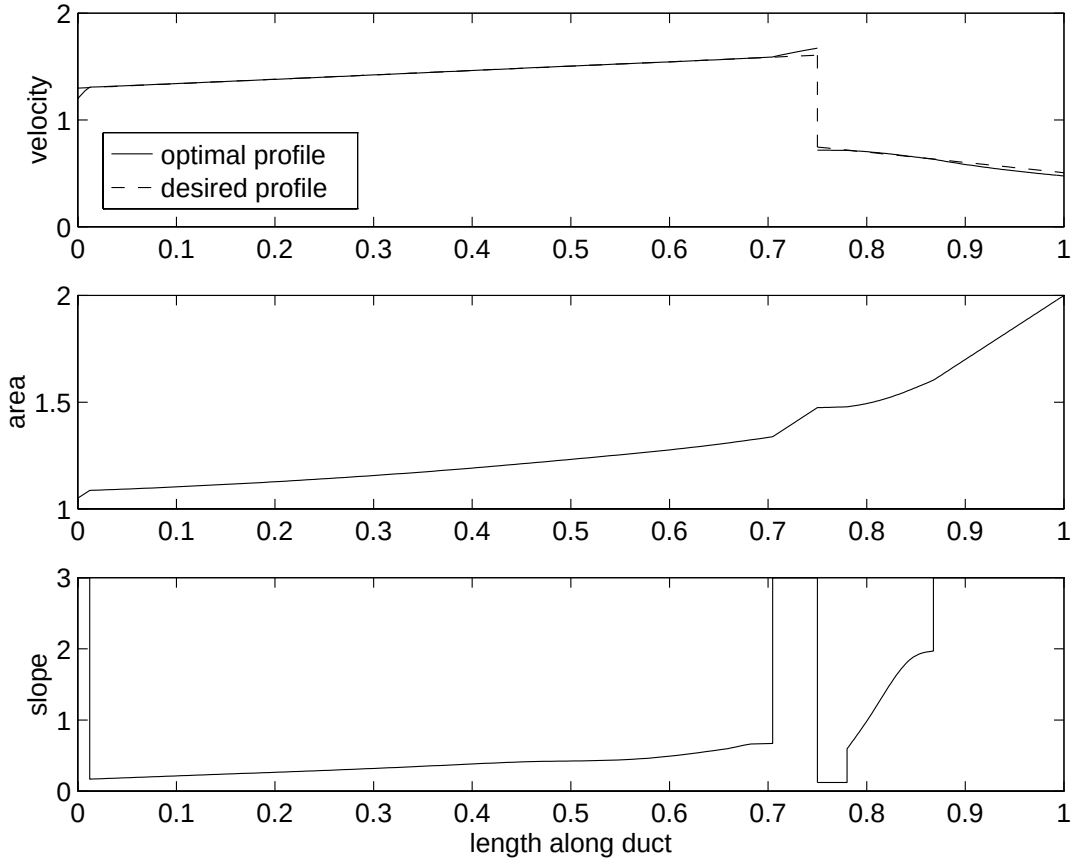


Figure 2.6: Optimal Profile for $A_{\text{out}} = 2.0$, with end-velocities free

the boundary value problem involving the states and the co-states using MATLAB, and also using a FORTRAN-based boundary-value solver, BNDSCO [110]. BNDSCO is an implementation of the multiple shooting algorithm adapted to the solution of the optimal control problem as a multi-point boundary value problem. The MATLAB routine has the advantage of not requiring information about the precise nature of the control, while BNDSCO provides far more accurate results. The initial estimate of the switching structure was made using the MATLAB routine, the results of which were then fed as an initial guess to BNDSCO.

As is expected, for the nominal case, when the cross-sectional areas at the inlet and outlet are fixed at 1.05 and 1.745, the optimal solution is totally singular, and the profile exactly matches the desired flow. When the outlet cross-sectional area is changed from the nominal value of 1.745, we get a "bang-singular-bang" control structure in both the subsonic and supersonic domains. Results were obtained for a range of values of $A_R(0)$, as indicated by Table 2.1.

When the outlet area is *decreased* from the nominal value, the control first hits the lower bound at the inlet, switches to a singular arc, where the flow profile is matched exactly, and then switches to a "bang" arc, up against the lower bound as it approaches the shock location. In the post- shock region, the control is initially up against the upper bound, switches to a singular arc, and then jumps to the lower bound approaching the outlet of the duct. Figures 2.7 and 2.8 show typical results, for the case when the outlet area has been fixed at 1.50. The optimal flow profile, and the corresponding area distribution and control, is shown in Figure 2.7 with the bang arcs shown in closer detail in Figure 2.8.

Once again, it should be noticed that since the specified outlet area is *less* than the nominal value, it is reasonable that arcs occur with $A'(x)$ at its *minimum* value. The region downstream of the shock, with $A'(x)$ at its *maximum* value, seems counterintuitive. However, in view of the flow physics, we see that in the supersonic region upstream of the shock, using $A'(x) = w_{\min}$ will result in a flow velocity that is *lower* than the nominal so that following the shock, from the Rankine-Hugoniot relation, the flow velocity will be *higher* than nominal. To "get back on track", *i.e.*, to produce an exact match to the desired flow profile, the flow must be slowed down more quickly than the nominal flow. In a subsonic flow, this requires *expanding* the area.

Increasing the outlet area from the nominal value yields a similar control structure, except the controls now hit the upper bound where they previously were up against the lower bound, and vice versa. Results for $A_R(0) = 2.0$ are shown in Figures 2.9 and 2.10, respectively. The bang arc at the outlet of the duct plays a dominant role in this case. This is understandable as the nominal control, *i.e.*, the control for the nominal case when $A_{\text{out}} = 1.745$, is a monotonically increasing function and attains its largest magnitude to-

Table 2.1: Switching Configurations for Various Values of the Outlet Duct Area

Outlet Area A_{out}	*Control Structure*		
	interval	*switching function*	$A'(x)$
1.5	$0 \le x \le 0.0384$	$S_L > 0$	$w_{\min}$
	$0.0748 \le x \le 0.662$	$S_L = 0$	singular
	$0.662 \le x \le 0.75$	$S_L > 0$	$w_{\min}$
	$0.75 \le x \le 0.758$	$S_R < 0$	$w_{\max}$
	$0.758 \le x \le 0.919$	$S_R = 0$	singular
	$0.919 \le x \le 1.0$	$S_R > 0$	$w_{\min}$
1.6	$0 \le x \le 0.0384$	$S_L > 0$	$w_{\min}$
	$0.0384 \le x \le 0.702$	$S_L = 0$	singular
	$0.702 \le x \le 0.75$	$S_L > 0$	$w_{\min}$
	$0.75 \le x \le 0.755$	$S_R < 0$	$w_{\max}$
	$0.702 \le x \le 0.956$	$S_R = 0$	singular
	$0.956 \le x \le 1.0$	$S_R > 0$	$w_{\min}$
1.7	$0 \le x \le 0.0105$	$S_L > 0$	$w_{\min}$
	$0.0105 \le x \le 0.736$	$S_L = 0$	singular
	$0.736 \le x \le 0.75$	$S_L > 0$	$w_{\min}$
	$0.75 \le x \le 0.751$	$S_R < 0$	$w_{\max}$
	$0.751 \le x \le 0.987$	$S_R = 0$	singular
	$0.987 \le x \le 1.0$	$S_R > 0$	$w_{\min}$
1.745	$0 \le x \le 1.0$	$S_L, S_R = 0$	singular
1.8	$0 \le x \le 0.0034$	$S_L < 0$	$w_{\max}$
	$0.0034 \le x \le 0.741$	$S_L = 0$	singular
	$0.741 \le x \le 0.75$	$S_L < 0$	$w_{\max}$
	$0.75 \le x \le 0.757$	$S_R > 0$	$w_{\min}$
	$0.757 \le x \le 0.959$	$S_R = 0$	singular
	$0.959 \le x \le 1.0$	$S_R < 0$	$w_{\max}$
1.9	$0 \le x \le 0.0084$	$S_L < 0$	$w_{\max}$
	$0.0084 \le x \le 0.723$	$S_L = 0$	singular
	$0.723 \le x \le 0.75$	$S_L < 0$	$w_{\max}$
	$0.75 \le x \le 0.769$	$S_R > 0$	$w_{\min}$
	$0.769 \le x \le 0.909$	$S_R = 0$	singular
	$0.909 \le x \le 1.0$	$S_R < 0$	$w_{\max}$
2.0	$0 \le x \le 0.0121$	$S_L < 0$	$w_{\max}$
	$0.0121 \le x \le 0.704$	$S_L = 0$	singular
	$0.704 \le x \le 0.75$	$S_L < 0$	$w_{\max}$
	$0.75 \le x \le 0.780$	$S_R > 0$	$w_{\min}$
	$0.780 \le x \le 0.867$	$S_R = 0$	singular
	$0.867 \le x \le 1.0$	$S_R < 0$	$w_{\max}$

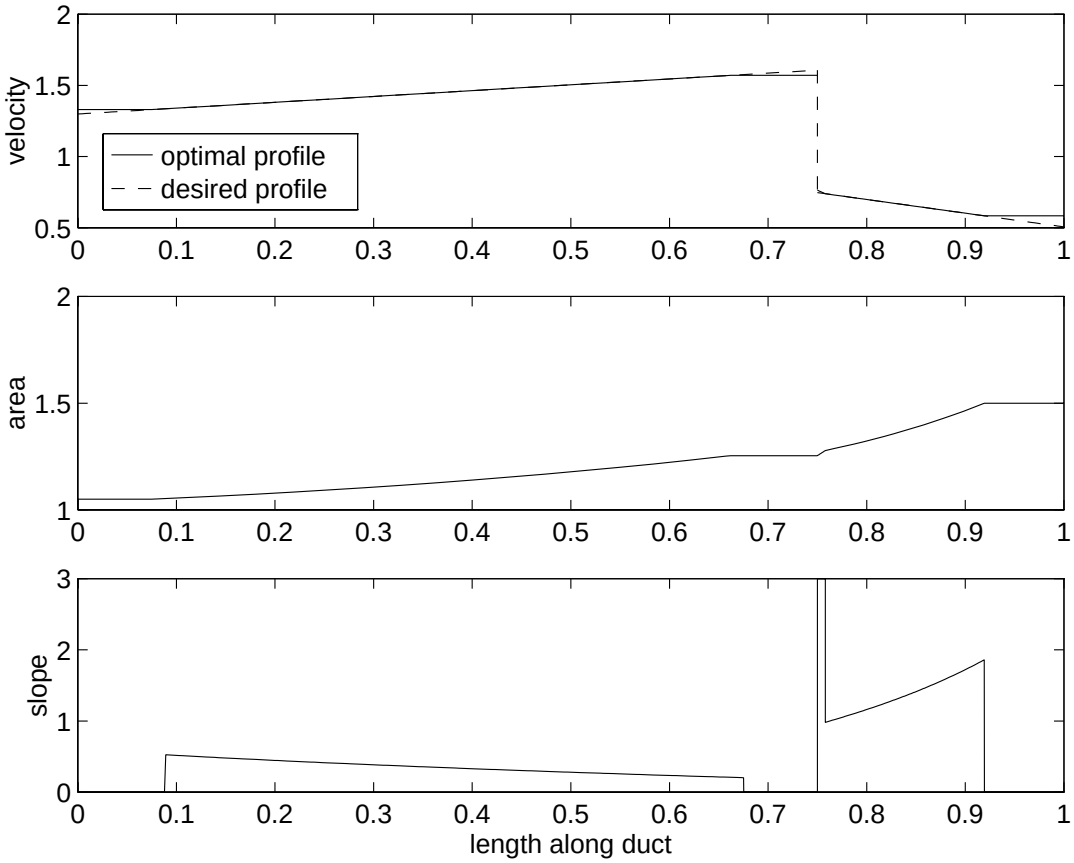


Figure 2.7: Optimal Profile for $A_{\text{out}} = 1.5$, using BNDSCO

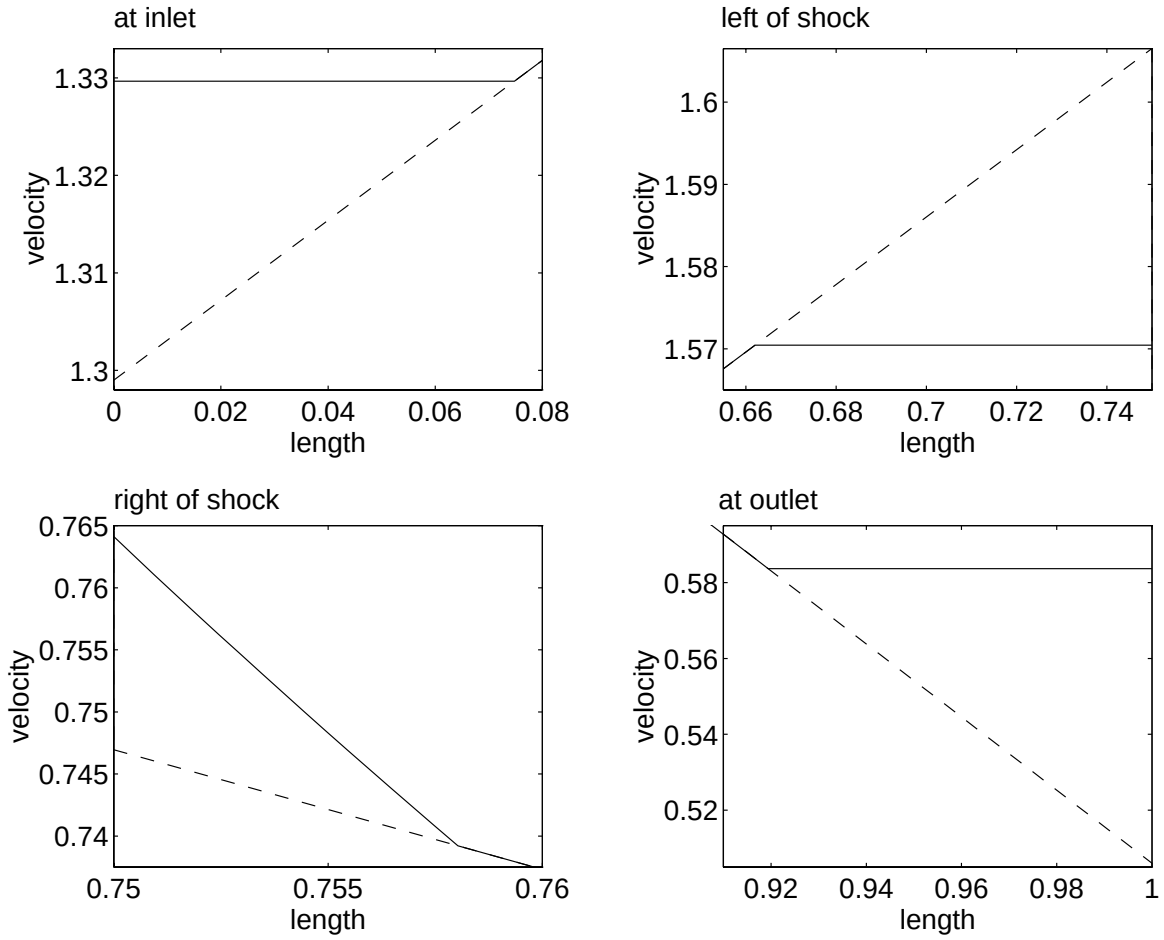


Figure 2.8: Close-up of the Velocity Profile for the Nonsingular Arcs, for $A_{\text{out}} = 1.5$

wards the end of the duct. As a result the deviation from the nominal profile is relatively smaller at this point when the control jumps to its upper bound. Thus, most of required increase in the area from its nominal value can be made up at the end of the duct, without causing too much deviation from the design velocity distribution.

Table 2.1 indicates the precise control switching structure for the cases described above. As would be expected the non-singular arcs play an increasingly prominent role as the area of the duct at the outlet is moved away from the nominal case. The problem seemed to be fairly robust, in that once the switching structure was known, the solution converged fairly quickly using BNDSCO, even when the initial estimate was far from the actual solution.

While the results from GESOP are fairly indicative of the optimal solutions the results are not as accurate as those obtained using BNDSCO. It must be noted that the TROPIC algorithm initially failed to capture the non-singular nature of the control at the ends of the duct, at the outlet when the area is increased, and at the inlet when the area is decreased, as these arcs are of relatively small lengths. It was only after the precise nature of the control was known, through the results of MATLAB and BNDSCO, that we were able to "force" the control to be nonsingular at the said location, by adjusting the initial estimate of the solution accordingly. The results also show that leaving the velocities "free" at the two ends of the duct allows for a far better match of the desired flow profile. When the end-velocities are fixed, we have "singular-bang" controls, which do not do as good a job at flow-matching as in the former case. The difference becomes more marked as the outlet area is forced further away from its nominal value.

Variation of the Performance Index with the Shock Position. Before closing, we revisit the issue of shock-location. Note that there are two (distinct) shocks to consider: the shock in the "data", say $\hat{x}_s$, where $\hat{u}(\cdot)$ is discontinuous; and, the shock in the flow solution, say x_s, where $u(\cdot)$ is discontinuous. The value $\hat{x}_s$ is known because $\hat{u}(\cdot)$ is given. Up to this point we have required that $x_s = \hat{x}_s$. However, given the L_2 performance index, it is reasonable to suppose that if x_s were "free" then optimality would locate it. In general one would expect that $x_s^* \neq \hat{x}_s$, however they would be very close since the integrand in the performance index for the region separating the two shock locations would be "large". Thus, we are motivated to consider

$$J(x_s) \equiv \frac{1}{2}\int_0^1 \left[u(x;x_s) - \hat{u}(x)\right]^2 dx$$

Assuming $x_s > \hat{x}_s$

$$J(x_s) = \frac{1}{2}\int_0^{\hat{x}_s^-} (u-\hat{u})^2 dx + \frac{1}{2}\int_{\hat{x}_s^+}^{x_s^-} (u-\hat{u})^2 dx + \frac{1}{2}\int_{x_s^+}^1 (u-\hat{u})^2 dx$$

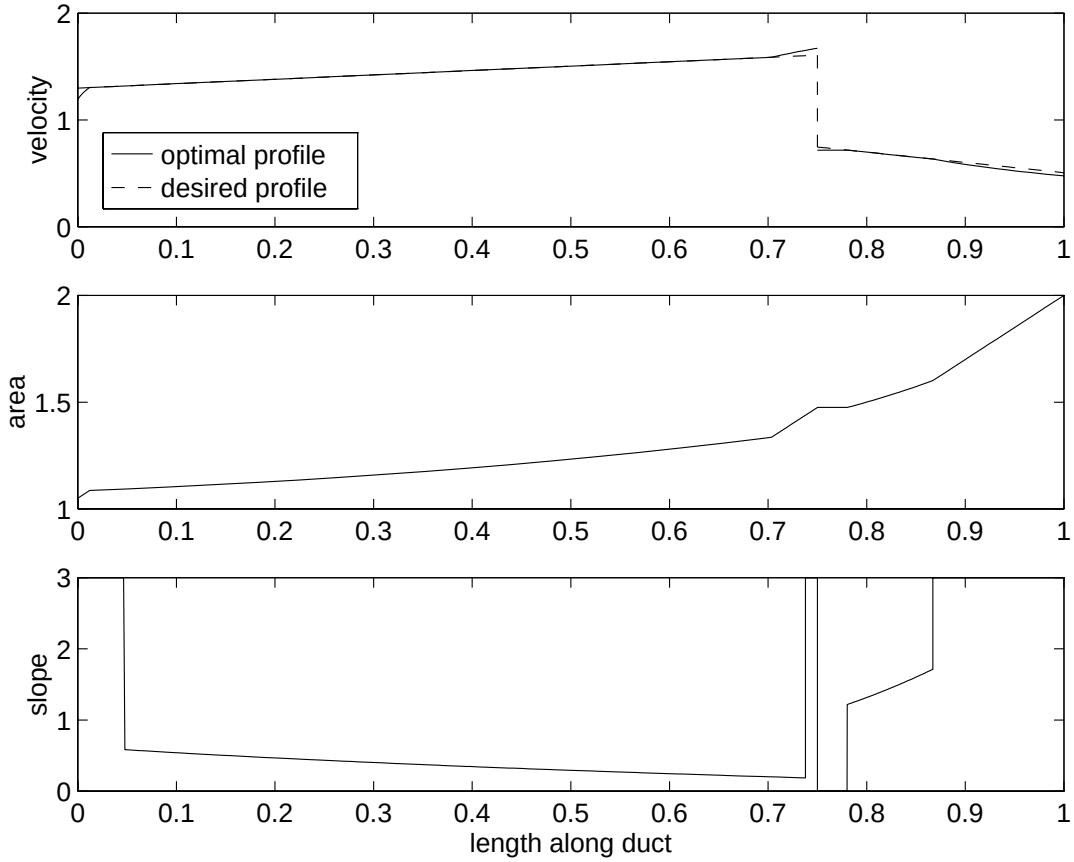


Figure 2.9: Optimal Profile for $A_{\text{out}} = 2.0$, using BNDSCO

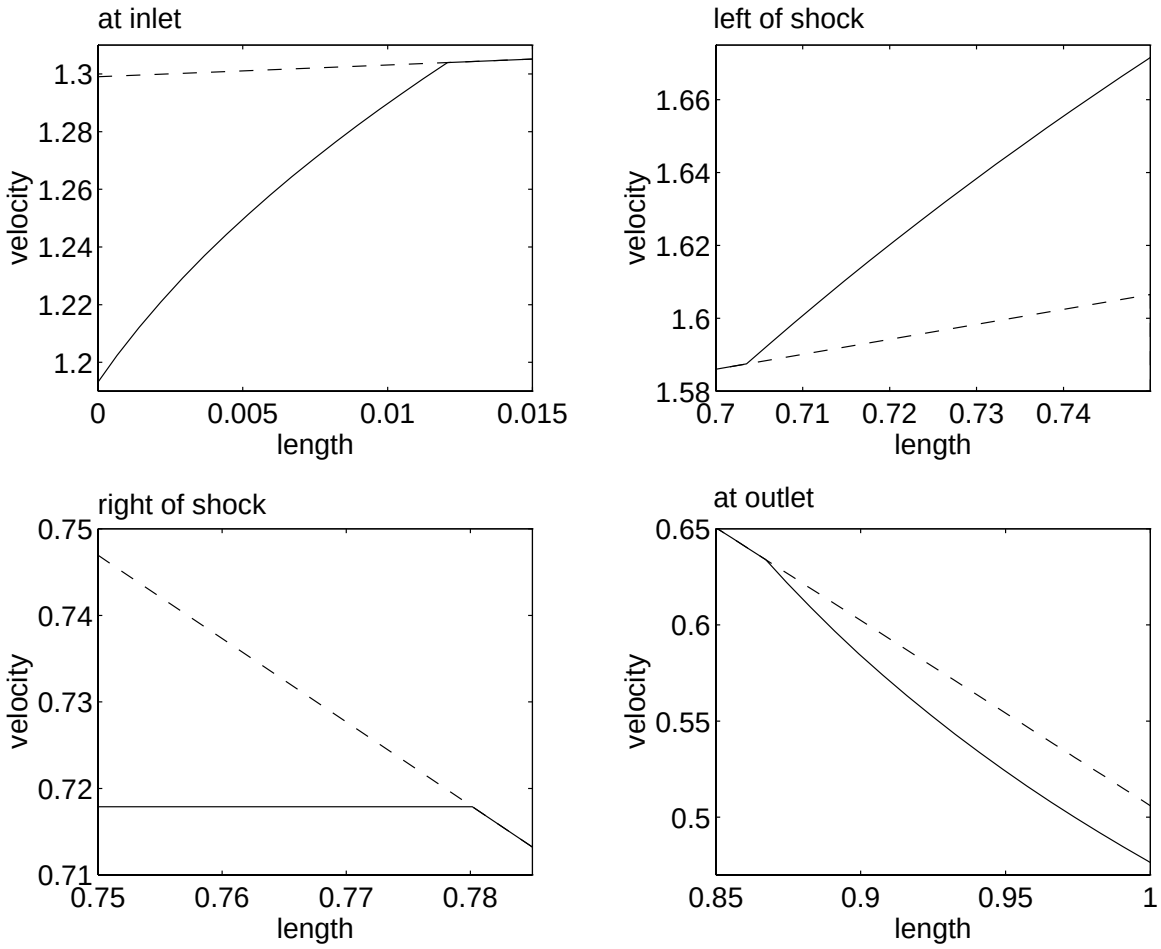


Figure 2.10: Close-up of the Velocity Profile for the Nonsingular Arcs, for $A_{\text{out}} = 2.0$

We have

$$\frac{\partial J}{\partial x_s} = \int_0^1 \left[u(x;x_s) - \hat{u}(x)\right] \frac{\partial u}{\partial x_s} dx + \frac{1}{2}\left[u(x_s^-;x_s) - \hat{u}(x_s^-)\right]^2 - \frac{1}{2}\left[u(x_s^+;x_s) - \hat{u}(x_s^+)\right]^2$$

Note that $u(x_s^-;x_s) \equiv u^-, u(x_s^+;x_s) \equiv u^+$ and $\hat{u}(x_s^-) = \hat{u}(x_s^+) = \hat{u}$, since $\hat{u}(x)$ is continuous at $x = x_s$. Hence, we have, upon simplification

$$\frac{\partial J}{\partial x_s} = \int_0^1 \left[u(x;x_s) - \hat{u}(x)\right] \frac{\partial u}{\partial x_s} dx + \left[u^- - u^+\right] \cdot \left[\frac{(u^- + u^+)}{2} - \hat{u}\right]$$

A similar analysis for $x_s < \hat{x}_s$ produces the same result. At $x_s = \hat{x}_s$ the $\hat{u}$ term is discontinuous. This produces a jump in the derivative. We have

$$\begin{aligned}\lim_{x_s \uparrow \hat{x}_s} \frac{\partial J}{\partial x_s} &= \int_0^1 \left[u(x;x_s) - \hat{u}(x)\right] \frac{\partial u}{\partial x_s} dx \Big|_{\hat{x}_s} \\ &\quad + \left[u(\hat{x}_s^-;\hat{x}_s) - u(\hat{x}_s^+;\hat{x}_s)\right] \cdot \left[\frac{u(\hat{x}_s^-;\hat{x}_s) + u(\hat{x}_s^+;\hat{x}_s)}{2} - \hat{u}(\hat{x}_s^-)\right]\end{aligned}$$

and

$$\begin{aligned}\lim_{x_s \downarrow \hat{x}_s} \frac{\partial J}{\partial x_s} &= \int_0^1 \left[u(x;x_s) - \hat{u}(x)\right] \frac{\partial u}{\partial x_s} dx \Big|_{\hat{x}_s} \\ &\quad + \left[u(\hat{x}_s^-;\hat{x}_s) - u(\hat{x}_s^+;\hat{x}_s)\right] \cdot \left[\frac{u(\hat{x}_s^-;\hat{x}_s) + u(\hat{x}_s^+;\hat{x}_s)}{2} - \hat{u}(\hat{x}_s^+)\right]\end{aligned}$$

Hence,

$$\frac{\partial J}{\partial x_s}\Big|_{\hat{x}_s^+} - \frac{\partial J}{\partial x_s}\Big|_{\hat{x}_s^-} = \left[u(\hat{x}_s^-) - u(\hat{x}_s^+)\right] \cdot \left[\hat{u}(\hat{x}_s^-) - \hat{u}(\hat{x}_s^+)\right]$$

that is, the jump in the derivative is the product of the jumps in the two flows. Note that this product is positive. Table 2.2 shows the jump in the derivative at the shock location, computed for various values of the outlet duct area. While we do not have a complete proof, in all the cases we have examined $J(\cdot)$ has a vee-shaped minimum at $\hat{x}_s$. A typical situation is illustrated in Figure 2.11, where the shock location x_s was varied for a given value of $A_{\text{out}} = 1.5$. As can be seen from the figure, the minimum value of the objective is attained for $x_s = \hat{x}_s$.

Our results indicate that, for the problem with a "free" shock location, the optimal location is at the shock in the "data", for the cases we have considered.

Table 2.2: Jump in Derivative of Objective Function for Various Values of A_{out}

A_{out}	$\frac{dJ}{dx_s}(\hat{x}_s^-)$	$\frac{dJ}{dx_s}(\hat{x}_s^+)$
2.180	-0.41316104	0.46022784
2.150	-0.41015240	0.45392157
2.100	-0.40516588	0.44412450
2.000	-0.39543183	0.42315496
1.900	-0.38552573	0.40161842
1.800	-0.37290991	0.38205581
1.750	-0.36998581	0.36780860
1.700	-0.36714114	0.36135269
1.600	-0.35814658	0.35269822
1.500	-0.35220891	0.33910304
1.400	-0.34497864	0.32309251
1.300	-0.33292888	0.30628595
1.200	-0.31998581	0.28717326
1.180	-0.31689717	0.28365459
1.170	-0.31631216	0.28161704

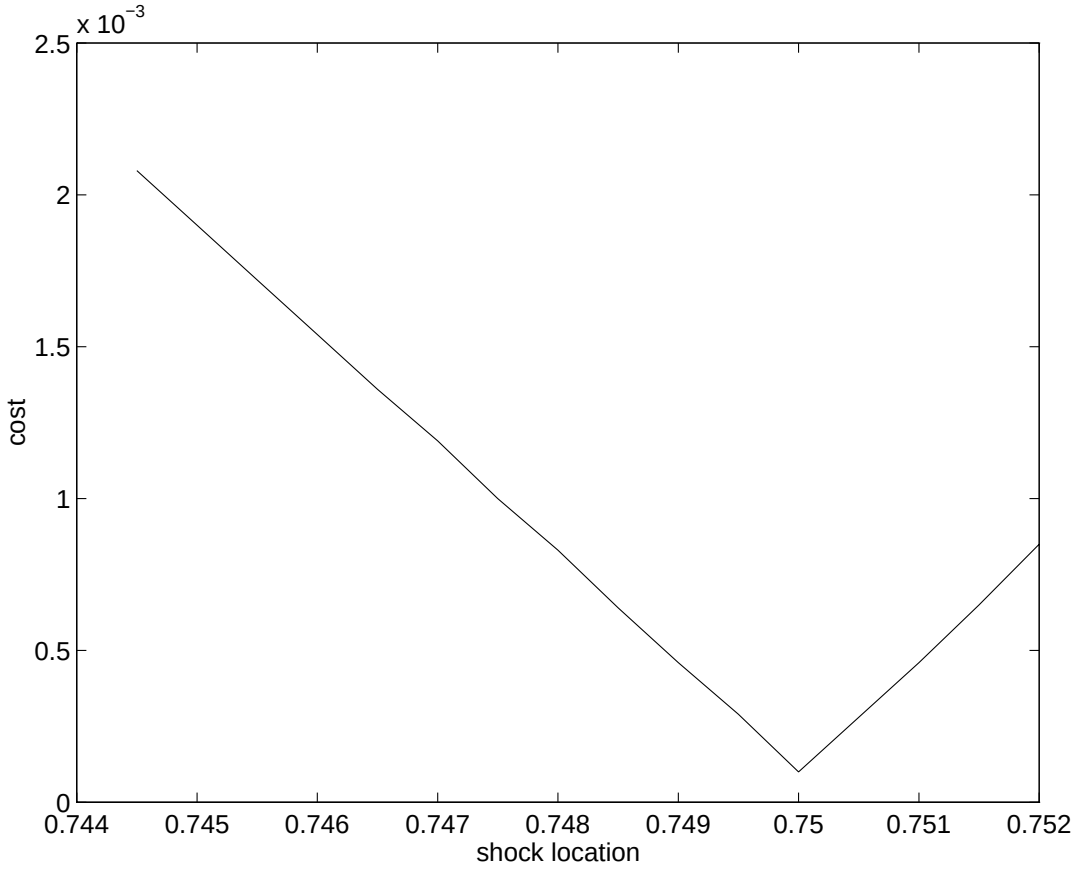


Figure 2.11: Variation of the Cost with the Shock Location, for $A_{\text{out}} = 1.5$

2.3.3 Remarks

At a certain level, our results are unexpected. Intuitively a best L_2 or "least-squares" fit-to data produces a solution wherein the local error is distributed—there is a small mis-match everywhere. Thus, the occurance of the singular arcs, with their interpretation as exact fit regions, is surprising. In this same vein the observation about the problem with free-shock location ($x_s^* = \hat{x}_s$) is perhaps counterintuitive.

In ending, one should note that while the above formulation proved to be an interesting mathematical exercise, from a practical standpoint, the solutions would be inadmissible due to the presence of the kinks in the area of the duct. The results do provide insight into the nature of the problem, though. In the following sections we will be using an alternate formulation where the area is parameterized by a few variables while the differential equation (2.4) is approximated by a finite-difference scheme to obtain a corresponding finite-dimensional optimization problem.

2.4 Using A Shock-Capturing Formulation

In this section we describe the area as a cubic Hermite polynomial. We take the inlet and outlet areas to be fixed, so that our parameters are identified with the slope at the inflow and outflow ends *i.e.*, $w_1 \equiv A_x(0)$ and $w_2 \equiv A_x(1)$. We use a Godunov-type scheme with finite differences to formulate the problem in finite dimensions.

2.4.1 Discretization

The design problem can, in the infinite dimensional case, be written as,

$$\text{Minimize } \int_0^1 (u(x) - \hat{u}(x))^2 \, dx \tag{2.27}$$

subject to constraints (2.4), (2.6) and (2.7).

In the finite dimensional case, we discretize the flow, and use the discrete values of the flow at various grid points as our state. The domain $(0, 1)$ is divided into N equispaced grid points, shown in Figure 2.12, yielding N subdomains, each of which must satisfy the state constraint. We have,

$$q = u = \{u_j\} \quad j = 1, \ldots, N$$

We use a cell-centered finite volume formulation. For the jth cell (subdomain), we have,

$$f_x \approx \frac{f_{j+1/2} - f_{j-1/2}}{\Delta x}$$

This relation yields the following constraints

$$C(j) = \frac{f_{j+1/2} - f_{j-1/2}}{\Delta x} + g_j \qquad j = 1, \ldots, N \tag{2.28}$$

Using first order accurate interpolation, we have, for the supersonic region

$$f_{j+1/2} \approx f_j; \text{ and } f_{j-1/2} = f_{j-1}, \qquad u_j > u_s$$

and, for the subsonic region

$$f_{j+1/2} = f_{j+1}; \text{ and } f_{j-1/2} = f_j, \qquad u_j < u_s$$

where, $u_s = \sqrt{\overline{H}}$ is the sonic velocity. We use a Godunov scheme to capture the shock. This scheme uses the following logic,

```
if      ( (u_j > u_s) & (u_{j+1} < u_s) )  then
            f_{j+1/2} = max(f_j, f_{j+1})
elseif  ( (u_{j-1} > u_s) & (u_j < u_s) )  then
            f_{j-1/2} = max(f_{j-1}, f_j)
endif
```

which implicitly forces the shock condition, $f_L = f_R$. The above system of equations yields a tridiagonal matrix for the Jacobian of the constraints. The constraints,

$$C = 0$$

can be solved using Newton's method, given some area distribution. The area profile is parameterized in terms of $A(0), A_x(0), A(1)$ and $A_x(1)$. We use a cubic hermite polynomial, defined by the afore-mentioned parameters, to represent the area distribution. The objective function in discrete form is given by,

$$F = \sum_{i=1}^{N} (u_i - \hat{u}_i)^2 \tag{2.29}$$

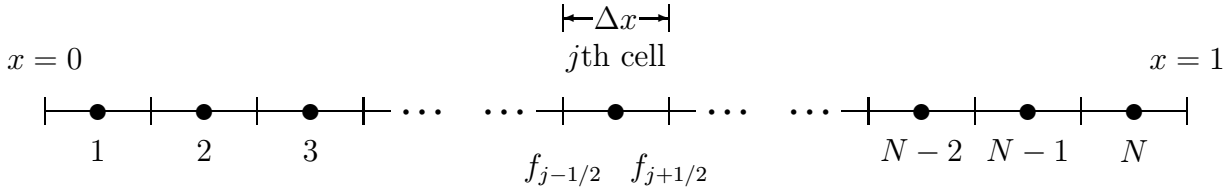


Figure 2.12: Grid Setup for Shock-Capturing Formulation

2.4.2 Numerical Results

We use the following values, to compute a target solution, $\hat{u}$:

$$\begin{aligned} u(0) &= 1.299 \\ A(0) &= 1.05 \\ A_x(0) &= 0.1 \\ u(1) &= 0.506 \\ A(1) &= 1.745 \\ A_x(1) &= 0.1 \end{aligned}$$

The target flow solution and the area profile are shown in Figure 2.13. Note that the area function A and the boundary values $u_{\text{in}}, u_{\text{out}}$ used in the computation of the target data are identical to the ones given in [55].

For the optimization problem, we use the following as our control variables:

$$w = [A_x(0) \quad A_x(1)]$$

with $A(0)$ and $A(1)$ fixed.

Results were primarily obtained using variants of the Sequential Quadratic Programming (SQP) method described in Section 1.1. As mentioned earlier, the underlying analysis problem is solved using a Newton method, wherein the linearized constraints are solved efficiently using a tridiagonal solver. The solution algorithm for the linearized state equations can be incorporated into the optimization algorithm, TRICE. The solver can also be used to compute solutions to the adjoint equation for the optimality system, as described in Section 1.1. We used both the *full* and *reduced* SQP schemes available in TRICE, to perform our computations.

Results, comparing the computational effort for the two methods, for a grid size of $N = 400$, are given in Table 2.3. An initial guess of $A_x(0) = A_x(1) = 0.09$ was used for the parameters, and the initial flow variables are estimated to be equal to the target data. Additionally, results obtained using a *Black-box* method are also shown, for comparison. The *Black-box* method uses NLPQL, an SQP-based algorithm [122], with finite central differencing used to obtain gradient information. It was found, overall that the number of discrete evaluations of the flux, f, made the biggest contribution to the overall computational effort,and this has been used as a benchmark in quantifying the computational effort.

It can be seen that the *Black-box* method is totally outperformed by the other methods. While the *reduced* and *full* SQP schemes consume about the same amount of computational

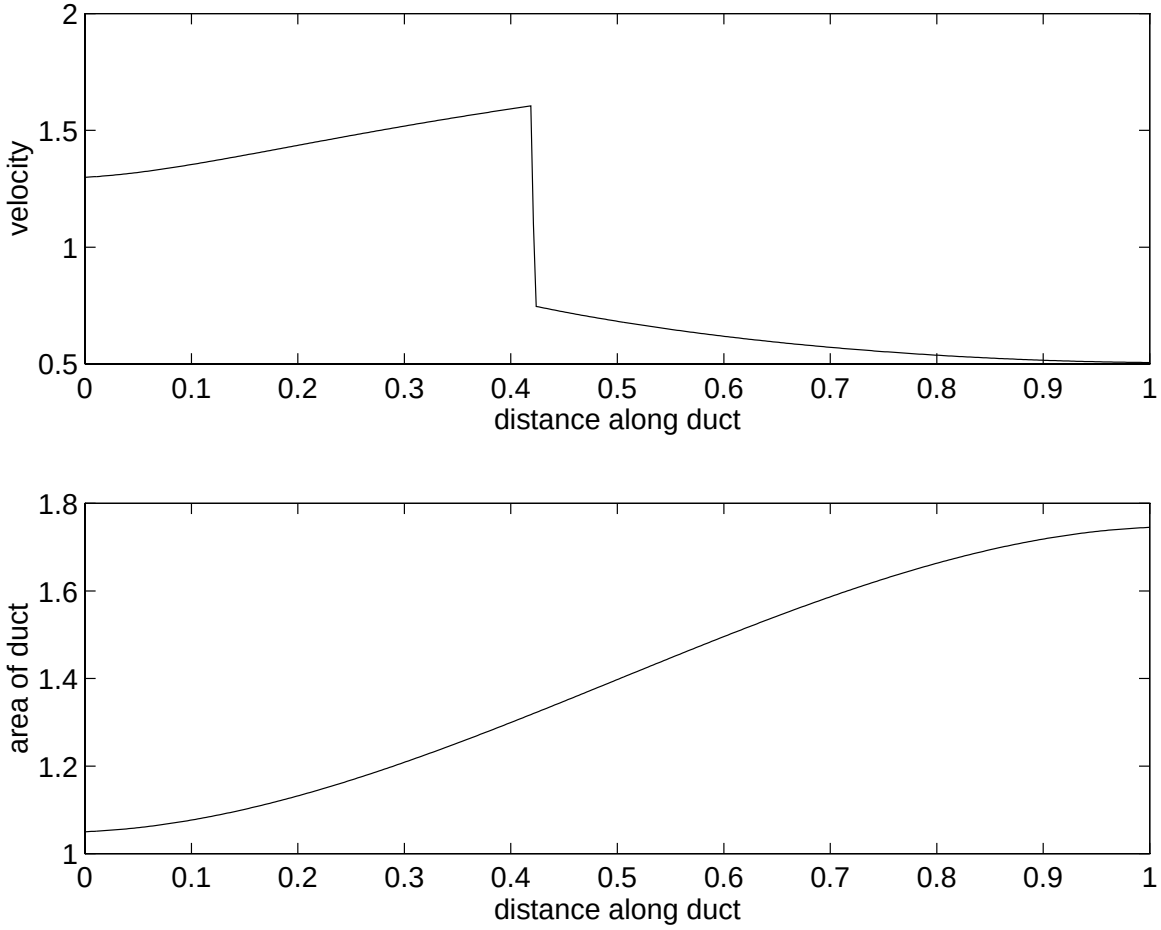


Figure 2.13: Target Data for Finite-Dimensional Formulations

Table 2.3: Comparison of Computational Efforts

	Blackbox	*Full SQP*	*Reduced SQP*
# QP Iterations	5	6	7
# Flux Evaluations	7897456	270505	212588
# Constraint Evaluations	2224	29	36
# Gradient Evaluations	2192	81	62

effort, the *Black-box* scheme requires about 10 times as much effort to produce a converged solution. While the *reduced* SQP method outperforms the *full* SQP procedure in this case, it was found that, as the initial guess for the parameters is moved further away from the optimal value, the *reduced* SQP method requires far more iterations to converge. This is not surprising, as analysis [68] shows that the local convergence properties are superior for the *full* method. One should remember that for the latter, we approximate the full Hessian, while for the former, we approximate the reduced Hessian, which has all the information of the full Hessian compressed to a reduced scale. Also, one may conjecture that the cross term $N(z)^T \nabla_{zz} L(z, \lambda) p(z)$ (*cf.* (1.9)), which has been neglected, may play a more significant role in this case.

2.5 Using A Shock-Fitting Formulation

Our experience in the previous section showed us that while the all-at-once approach was highly efficient, it was not very robust. This lack of robustness is mainly due to the numerical difficulties caused by the presence of the shock in the flow. The potential difficulty with shock-capturing schemes is that the constraints are not smooth. To overcome this we treat the shock location as an explicit state variable, and enforce a shock condition as an additional state constraint. This approach yields better smoothness properties for the constraints.

In the following we will denote the logarithmic derivative of A by w,

$$w(x) \equiv \frac{A_x(x)}{A(x)} = \frac{d}{dx} \ln(A(x)). \tag{2.30}$$

With this substitution the state equation is given by

$$(f(u))_x + g(u, w) = 0, \tag{2.31}$$

and (2.6), (2.8), where

$$g(u, w) \equiv w\,(\bar{\gamma} u - \bar{H}/u). \tag{2.32}$$

Note that instead of introducing another symbol we redefine g. Unless stated otherwise, in the following $g(u, w)$ is always given by (2.32).

Trivially, w is determined by A. On the other hand, if the area is known at some point, for example, if $A(0) > 0$ is given, then $A(x)$ can be computed from w by integrating (2.30):

$$A(x) = \exp\left(\ln(A(0)) + \int_0^x w(\xi)\, d\xi\right). \tag{2.33}$$

The function $A(x)$ defined by (2.33) is absolutely continuous and, therefore, differentiable almost everywhere. From now on, we assume that $A(0) > 0$ is given.

For a rigorous treatment of the dependence of the solution upon parameters, we have to transform the ODE, the shock condition, and the boundary conditions. As before, we denote the velocity left of the shock by u_L and the velocity right of the shock by u_R. We use a similar transformation of variables as in Section 2.3. The ODE (2.31), the shock condition (2.8), and the boundary conditions (2.6) become

$$\begin{aligned} (f(u_L))_\xi + x_s g(u_L, w_L) &= 0, \quad \xi \in [0,1], && (2.34)\\ (f(u_R))_\xi + (x_s - 1) g(u_R, w_R) &= 0, \quad \xi \in [0,1], && (2.35) \end{aligned}$$

$$f(u_L(1)) = f(u_R(1)), \tag{2.36}$$

and

$$u_L(0) = u_{\text{in}}, \quad u_R(0) = u_{\text{out}}. \tag{2.37}$$

The functions w_L and w_R in (2.34), (2.35) are defined by

$$\begin{aligned} w_L(\xi) &= w(x_s \xi), \qquad \text{and} \\ w_R(\xi) &= w(1 - (1 - x_s)\xi), \end{aligned}$$

respectively. In the following we view w_L and w_R as independent variables. To guarantee that the corresponding area function is monotonically increasing we have to impose the conditions

$$w_L(\xi) > 0, \quad w_R(\xi) > 0, \quad \xi \in [0,1].$$

2.5.1 The Design Problem

The control problem we are interested in is the design of an area function generating a flow that best approximates a desired velocity in the least squares sense. Given a desired velocity $\hat{u} \in L^2(0,1)$ and a point x_s we introduce

$$\hat{u}_L(x_s; \xi) \equiv \hat{u}(x_s \xi), \quad \hat{u}_R(x_s; \xi) \equiv \hat{u}(1 - (1 - x_s)\xi). \tag{2.38}$$

With the domain transformation described in Section 2.5 the objective function (2.3) is given by

$$\begin{aligned} \int_0^1 (u(x) - \hat{u}(x))^2 \, dx &= \int_0^{x_s} (u(x) - \hat{u}(x))^2 \, dx + \int_{x_s}^1 (u(x) - \hat{u}(x))^2 \, dx \\ &= x_s \int_0^1 (u_L(\xi) - \hat{u}_L(x_s; \xi))^2 \, d\xi \\ &\qquad + (1 - x_s) \int_0^1 (u_R(\xi) - \hat{u}_R(x_s; \xi))^2 \, d\xi. \end{aligned}$$

Thus, using the transformation described in Section 2.5, the control problem we have to solve is given as follows:

$$\min\ J(q,w) \equiv \frac{x_s}{2}\int_0^1 (u_L(\xi)-\hat{u}_L(x_s;\xi))^2\,d\xi + \frac{1-x_s}{2}\int_0^1 (u_R(\xi)-\hat{u}_R(x_s;\xi))^2\,d\xi \tag{2.39}$$

subject to the equality constraints

$$(f(u_L))_\xi + x_s g(u_L,w_L) = 0, \quad \xi\in[0,1], \tag{2.40}$$

$$(f(u_R))_\xi + (x_s-1) g(u_R,w_R) = 0, \quad \xi\in[0,1], \tag{2.41}$$

$$f(u_L(1)) = f(u_R(1)), \tag{2.42}$$

$$u_L(0) = u_{\text{in}}, \quad u_R(0) = u_{\text{out}}, \tag{2.43}$$

and to the inequality constraints

$$0 \le x_s \le 1, \tag{2.44}$$

$$0 \le w_{\min} \le w_L(\xi), w_R(\xi) \le w_{\max}, \quad \xi\in[0,1]. \tag{2.45}$$

Based on the existence results described in Section 2.2 we assume that the boundary conditions obey

$$u_{\text{in}} \in (\sqrt{\bar{H}}, \sqrt{2H}), \quad u_{\text{out}} \in (\sqrt{\bar{\gamma}\bar{H}}, \sqrt{\bar{H}}). \tag{2.46}$$

The states are given by the triple $q = (u_L, u_R, x_s)$ and the controls are $w = (w_L, w_R)$. The equations (2.40), (2.41), (2.42), (2.43) are the state equations.

As the control space we use

$$\mathcal{W} = L^\infty([0,1]) \times L^\infty([0,1])$$

and we denote the set of admissible controls by

$$\mathcal{W}_{ad} = \{(w_L,w_R)\in\mathcal{W} \mid 0 \le w_{\min} \le w_L(\xi), w_R(\xi) \le w_{\max} \ \text{ a.e. in } [0,1]\}.$$

The set of admissible controls is closed and convex. By a solution to (2.40) (or (2.41)) we mean an absolutely continuous function which satisfies (2.40) (or (2.41)) almost everywhere on $[0,1]$.

Using the arguments applied in the Section 2.2 we can establish the following result.

Lemma 2.5.1 *Suppose that* u_{in} *and* u_{out} *obey (2.46) and that* $(w_L, w_R) \in \mathcal{W}_{ad}$. *If* (u_L, u_R, x_s) *with* $x_s \in [0,1]$ *is a solution of (2.40)–(2.43), then*

$$0 < u_{\text{out}} \le u_R(\xi) \le \frac{\bar{H}}{u_{\text{in}}} < \sqrt{\bar{H}} < u_{\text{in}} \le u_L(\xi) \le \frac{\bar{H}}{u_{\text{out}}} < \sqrt{2H}, \quad \xi \in [0,1], \tag{2.47}$$

and there exists $c > 0$ *which depends only on* $u_{\text{in}}, u_{\text{out}}$ *and* $w_{\min}, w_{\max}$ *such that*

$$|(u_L)_\xi(\xi)| \le c, \quad |(u_R)_\xi(\xi)| \le c \quad \text{a.e. on } [0,1]. \tag{2.48}$$

Proof: See [36]. □

Thus, the state space appropriate for this design problem is given by

$$\mathcal{Q} = W^{1,\infty}([0,1]) \times W^{1,\infty}([0,1]) \times I\!R$$

In the following we simply write $W^{1,\infty}, L^\infty$ instead of $W^{1,\infty}([0,1]), L^\infty([0,1])$ and we set

$$||w||_\infty = \text{ ess sup}_{[0,1]}|w(\xi)|, \quad ||q||_{1,\infty} = ||q||_\infty + ||q_\xi||_\infty.$$

We note that the shock location x_s enters the design problem in the differential equations (2.40), (2.41) and in the objective function, see also (2.38). The functions u_L, u_R, w_L, and w_R do not depend explicitly on x_s, but are implicitly coupled with the shock location through the design problem, in particular through (2.40) and (2.41).

Theorem 2.5.2 *Suppose there exist* $x_s \in (0,1)$ *and* $\bar{u} \in \left(u_{\text{in}}, \bar{H}/u_{\text{out}}\right)$ *such that*

$$0 \le w_{\min} \le \min\left\{ \frac{\left(1 - \frac{\bar{H}}{u_{\text{in}}^2}\right)(\bar{u} - u_{\text{in}})}{x_s\left(\frac{\bar{H}}{u_{\text{in}}} - \bar{\gamma}u_{\text{in}}\right)} \, , \, \frac{\left(\frac{\bar{u}^2}{\bar{H}} - 1\right)\left(\frac{\bar{H}}{\bar{u}} - u_{\text{out}}\right)}{(1 - x_s)(\frac{\bar{H}}{u_{\text{out}}} - \bar{\gamma}u_{\text{out}})} \right\} \tag{2.49}$$

and

$$w_{\max} \ge \max\left\{ \frac{\left(1 - \bar{\gamma}\right)(\bar{u} - u_{\text{in}})}{x_s\left(\frac{\bar{H}}{\bar{u}} - \bar{\gamma}\bar{u}\right)} \, , \, \frac{\left(\frac{\bar{H}}{u_{\text{out}}} - 1\right)\left(\frac{\bar{H}}{\bar{u}} - u_{\text{out}}\right)}{(1 - x_s)\left(\bar{u} - \bar{\gamma}\frac{\bar{H}}{\bar{u}}\right)} \right\}. \tag{2.50}$$

Then there exists an optimal control $(w_L^*, w_R^*) \in \mathcal{W}$ *of (2.39)–(2.45).*

Proof: See [36]. □

2.5.2 Fréchet Differentiability

In the previous section, we showed that using reasonable assumptions based on existence results for the analysis problem, we can show existence of optimal solutions for the design problem in the transformed domain. We also chose appropriate spaces for the states and controls. This will enable us to address the issue of differentiability of the mappings associated with the problem.

In the following we view u_L, u_R, x_s and w_L, w_R as independent variables. Since the shock location is treated explicitly, Fréchet differentiability of the objective function and the function of constraints can be established. In this section we introduce the mathematical framework that permits us to prove Fréchet differentiability, derive the first derivatives and we prove the continuous invertibility of the partial Fréchet derivative of the constraints with respect to the state variables. The latter property is important to show that constraint qualifications hold and is essential in SQP methods, in which one has to solve linearized state equations.

We introduce the operator

$$C : \mathcal{Q} \times \mathcal{W} \to \mathcal{C}, \tag{2.51}$$

where

$$\mathcal{C} = L^\infty \times L^\infty \times I\!R^3.$$

The operator C is defined as follows: For $r = (r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}}) \in \mathcal{C}$ the equality

$$C(u_L, u_R, x_s, w_L, w_R) = r$$

holds if and only if

$$\begin{aligned} (f(u_L))_\xi + x_s g(u_L, w_L) &= r_L, \quad \xi \in [0,1], & (2.52)\\ (f(u_R))_\xi + (x_s - 1) g(u_R, w_R) &= r_R, \quad \xi \in [0,1], & (2.53) \end{aligned}$$

$$f(u_L(1)) - f(u_R(1)) = r_s, \tag{2.54}$$

and

$$u_L(0) - u_{\text{in}} = r_{\text{in}}, \quad u_R(0) - u_{\text{out}} = r_{\text{out}}. \tag{2.55}$$

The equation $C(u_L, u_R, x_s, w_L, w_R) = 0$ is equivalent to (2.34), (2.35), (2.36), (2.37).

To be able to evaluate (2.52) and (2.53) the velocities have to satisfy $u_L(x) \neq 0$, $u_R(x) \neq 0$ for all $x \in [0,1]$.

Theorem 2.5.3 *The nonlinear operator $C : \mathcal{Q} \times \mathcal{W} \to \mathcal{C}$ is Fréchet differentiable at any point $(u_L, u_R, x_s, w_L, w_R) \in \mathcal{Q} \times \mathcal{W}$ satisfying $u_L(x) \neq 0$, $u_R(x) \neq 0$ for all $x \in [0,1]$. The partial Fréchet derivatives are given by*

$$C_q(q,w)\,(\tilde{q}) = \begin{pmatrix} (f_u(u_L)\tilde{u}_L)_\xi + x_s g_u(u_L,w_L)\tilde{u}_L + \tilde{x}_s g(u_L,w_L) \\ (f_u(u_R)\tilde{u}_R)_\xi + (x_s-1) g_u(u_R,w_R)\tilde{u}_R + \tilde{x}_s g(u_R,w_R) \\ f_u(u_L(1))\tilde{u}_L(1) - f_u(u_R(1))\tilde{u}_R(1) \\ \tilde{u}_L(0) \\ \tilde{u}_R(0) \end{pmatrix}$$

and

$$C_w(q,w)\,(\tilde{w}) = \begin{pmatrix} x_s g_q(u_L,w_L)\tilde{w}_L \\ (x_s-1) g_q(u_R,w_R)\tilde{w}_R \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

where

$$\begin{aligned} \tilde{q} &= (\tilde{u}_L, \tilde{u}_R, \tilde{x}_s) \\ \tilde{w} &= (\tilde{w}_L, \tilde{w}_R) \end{aligned}$$

Proof: See *e.g.*, [100] for the definition of Fréchet differentiability. For the appropriately chosen control and state spaces, we can prove differentiability from the definition of the constraints (2.52)–(2.55), using standard estimates. For details see [36]. □

The following result concerns the invertibility of the partial Fréchet derivative $C_q(q,w)$.

For given $(u_L, u_R, x_s) \in \mathcal{Q}$, $(w_L, w_R) \in \mathcal{W}$, and $(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}}) \in \mathcal{C}$ we consider the system

$$(f_u(u_L)\tilde{u}_L)_\xi + x_s g_u(u_L,w_L)\tilde{u}_L + \tilde{x}_s g(u_L,w_L) = r_L, \quad \xi \in [0,1], \tag{2.56}$$

$$(f_u(u_R)\tilde{u}_R)_\xi + (x_s-1) g_u(u_R,w_R)\tilde{u}_R + \tilde{x}_s g(u_R,w_R) = r_R, \quad \xi \in [0,1], \tag{2.57}$$

$$f_u(u_L(1))\tilde{u}_L(1) - f_u(u_R(1))\tilde{u}_R(1) = r_s, \tag{2.58}$$

and

$$\tilde{u}_L(0) = r_{\text{in}}, \quad \tilde{u}_R(0) = r_{\text{out}}. \tag{2.59}$$

Theorem 2.5.4 *(i) Suppose that* $(u_L, u_R, x_s, w_L, w_R) \in \mathcal{Q} \times \mathcal{W}_{ad}$ *is a point satisfying*

$$u_i(\xi) \neq 0, \quad u_i(\xi) \neq \sqrt{\bar{H}}, \quad \forall\, \xi \in [0,1], \qquad i = L, R.$$

If

$$\begin{aligned} &\int_0^1 \exp\left(-\int_t^1 \frac{x_s g_u(u_L(s), w_L(s))}{f_u(u_L(s))}\, ds\right) g(u_L(t), w_L(t))\, dt \\ \neq\ &\int_0^1 \exp\left(-\int_t^1 \frac{(x_s - 1) g_u(u_R(s), w_R(s))}{f_u(u_R(s))}\, ds\right) g(u_R(t), w_R(t))\, dt, \end{aligned} \tag{2.60}$$

then for every $(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}}) \in \mathcal{C}$ *the system (2.56), (2.57), (2.58), (2.59) admits a unique solution* $(\tilde{u}_L, \tilde{u}_R, \tilde{x}_s) \in \mathcal{Q}$ *which depends continuously on* $(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}})$.

(ii) If, in addition, there exists constants δ, Δ *with* $0 < \delta < \Delta$ *such that the point* $(u_L, u_R, x_s, w_L, w_R)$ *obeys*

$$\delta \leq u_i(\xi) \leq \Delta, \quad |u_i(\xi) - \sqrt{\bar{H}}| \geq \delta, \quad \forall\, \xi \in [0,1], \qquad i = L, R,$$

and

$$\begin{aligned} \Big| &\int_0^1 \exp\left(-\int_t^1 \frac{x_s g_u(u_L, w_L)}{f_u(u_L)}\, ds\right)\, g(u_L, w_L) \\ &- \exp\left(-\int_t^1 \frac{(x_s - 1) g_u(u_R, w_R)}{f_u(u_R)}\, ds\right)\, g(u_R, w_R)\, dt \Big| \ > \ \delta, \end{aligned}$$

then there exists a constant K *dependent on* δ, Δ, *but independent of* $(u_L, u_R, x_s, w_L, w_R)$ *such that*

$$||(\tilde{u}_L, \tilde{u}_R, \tilde{x}_s)||_{\mathcal{Q}} \leq K ||(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}})||_{\mathcal{C}}.$$

Proof: (i) First we note that since $W^{1,\infty}(0,1) \subset C([0,1])$, there exists $\delta > 0$ such that $u_i(\xi) > \delta$, $|u_i(\xi) - \sqrt{\bar{H}}| > \delta$, for all $\xi \in [0,1]$, $i = L, R$.

The equation (2.56) is equivalent to

$$(f_u(u_L)\tilde{u}_L)_\xi + \frac{x_s g_u(u_L, w_L)}{f_u(u_L)}\, (f_u(u_L)\tilde{u}_L) = r_L - \tilde{x}_s g(u_L, w_L). \tag{2.61}$$

Using the integrating factor

$$\mu_L(\xi) = \exp\left(\int_0^\xi \frac{x_s g_u(u_L(t), w_L(t))}{f_u(u_L(t))}\, dt\right),$$

the solution of (2.61) with initial condition $\tilde{u}_L(0) = r_{\text{in}}$ is given by

$$\tilde{u}_L(\xi) = \frac{1}{\mu_L(\xi) f_u(u_L(\xi))} \left(r_{\text{in}} f_u(u_L(0)) + \int_0^\xi \mu_L(t)[r_L(t) - \tilde{x}_s g(u_L(t), w_L(t))]\, dt \right). \tag{2.62}$$

Similarly, one can show that the solution of (2.57) with initial condition $\tilde{u}_R(0) = r_{\text{out}}$ is given by

$$\tilde{u}_R(\xi) = \frac{1}{\mu_R(\xi) f_u(u_R(\xi))} \left(r_{\text{out}} f_u(u_R(0)) + \int_0^\xi \mu_R(t)[r_R(t) - \tilde{x}_s g(u_R(t), w_R(t))]\, dt \right), \tag{2.63}$$

where

$$\mu_R(\xi) = \exp\left(\int_0^\xi \frac{(x_s - 1) g_u(u_R(t), w_R(t))}{f_u(u_R(t))}\, dt \right).$$

Inserting (2.62), (2.63) into (2.58) yields

$$\begin{aligned} & r_{\text{in}} \tfrac{f_u(u_L(0))}{\mu_L(1)} + \int_0^1 \exp\left(- \int_t^1 \tfrac{x_s g_u(u_L(s), w_L(s))}{f_u(u_L(s))}\, ds \right) [r_L(t) - \tilde{x}_s g(u_L(t), w_L(t))]\, dt \\ = \; & r_{\text{out}} \tfrac{f_u(u_R(0))}{\mu_R(1)} + \int_0^1 \exp\left(- \int_t^1 \tfrac{(x_s - 1) g_u(u_R(s), w_R(s))}{f_u(u_R(s))}\, ds \right) [r_R(t) - \tilde{x}_s g(u_R(t), w_R(t))]\, dt. \end{aligned} \tag{2.64}$$

If the inequality (2.60) is valid, then (2.64) can be solved for $\tilde{x}_s$. This proves the existence and uniqueness of the solution.

The continuous dependence of $(\tilde{u}_L, \tilde{u}_R, \tilde{x}_s) \in \mathcal{Q}$ upon $(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}}) \in \mathcal{C}$ follows from the equations (2.62), (2.63), (2.64).

(ii) The assertion follows from equations (2.62), (2.63), and (2.64). □

Corollary 2.5.5 *If $(\bar{u}_L, \bar{u}_R, \bar{x}_s, \bar{q}_L, \bar{q}_R) \in \mathcal{Q} \times \mathcal{W}$ is feasible, i.e., it satisfies the constraints (2.40)–(2.45), and if there exists $\delta > 0$ with*

$$\begin{aligned} \Big| & \int_0^1 \exp\left(- \int_t^1 \frac{x_s g_u(\bar{u}_L, \bar{w}_L)}{f_u(\bar{u}_L)}\, ds \right) g(\bar{u}_L, \bar{w}_L) \\ & - \exp\left(- \int_t^1 \frac{(x_s - 1) g_u(\bar{u}_R, \bar{w}_R)}{f_u(\bar{u}_R)}\, ds \right) g(\bar{u}_R, \bar{w}_R)\, dt \Big| \;\; > \;\; \delta, \end{aligned}$$

then there exists $\epsilon > 0$ *such that for all* $(u_L, u_R, x_s) \in \mathcal{Q}$, $(w_L, w_R) \in \mathcal{W}$ *with*

$$||(u_L, u_R, x_s) - (\bar{u}_L, \bar{u}_R, \bar{x}_s)||_{\mathcal{Q}} < \epsilon, \quad ||(w_L, w_R) - (\bar{w}_L, \bar{w}_R)||_{\mathcal{W}} < \epsilon$$

and for all $(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}}) \in \mathcal{C}$ *the system (2.56), (2.57), (2.58), (2.59) admits a unique solution* $(\tilde{u}_L, \tilde{u}_R, \tilde{x}_s) \in \mathcal{Q}$. *Moreover, there exists a constant* K *independent of* $(\bar{u}_L, \bar{u}_R, \bar{x}_s, \bar{q}_L, \bar{q}_R)$ *such that*

$$||(\tilde{u}_L, \tilde{u}_R, \tilde{x}_s)||_{\mathcal{Q}} \leq K||(r_L, r_R, r_s, r_{\text{in}}, r_{\text{out}})||_{\mathcal{C}}.$$

Proof: The solutions u_L, u_R satisfy (2.47). Hence, the assertion follows from Theorem 2.5.4(ii).

□

From the definitions of f and g one can see that the Fréchet derivative of C is Lipschitz-continuous for all u_L, u_R with $u_L(\xi), u_R(\xi) \geq \underline{u} > 0$ for all $\xi \in [0, 1]$. Moreover, C is even twice Fréchet differentiable if $u_L, u_R > 0$.

To prove the Fréchet differentiability of the objective function we have to keep in mind that the desired velocity depends on x_s, *cf.* (2.38). Therefore differentiability with respect to x_s can only be guaranteed if the desired velocity $\hat{u}$ is sufficiently smooth, a fact that will be addressed again in Section 2.5.5.

Theorem 2.5.6 *If the desired velocity* $\hat{u}$ *is differentiable with absolutely continuous derivative, then the objective function* J *is Fréchet differentiable. The partial Fréchet -derivatives are given by*

$$\begin{aligned} J_{u_L}(u_L, u_R, x_s, w_L, w_R)\tilde{u}_L &= x_s \int_0^1 (u_L(\xi) - \hat{u}_L(x_s; \xi))\, \tilde{u}_L(\xi)\, d\xi, \\ J_{u_R}(u_L, u_R, x_s, w_L, w_R)\tilde{u}_R &= (1 - x_s) \int_0^1 (u_R(\xi) - \hat{u}_R(x_s; \xi))\, \tilde{u}_R(\xi)\, d\xi, \end{aligned}$$

and

$$\begin{aligned} &J_{x_s}(u_L, u_R, x_s, w_L, w_R) \\ &= \int_0^1 \frac{1}{2}(u_L(\xi) - \hat{u}_L(x_s; \xi))^2 - x_s\, (u_L(\xi) - \hat{u}_L(x_s; \xi))\, (\hat{u}_L)_x(\xi)\, \xi\, d\xi \\ &\quad - \int_0^1 \frac{1}{2}(u_R(\xi) - \hat{u}_R(x_s; \xi))^2 + (1 - x_s)\, (u_R(\xi) - \hat{u}_R(x_s; \xi))\, (\hat{u}_R)_x(\xi)\, \xi\, d\xi. \end{aligned} \tag{2.65}$$

The objective function J is twice Fréchet differentiable if the desired velocity $\hat{u}$ is twice differentiable with absolutely continuous second derivative.

Proof: The assertion follows from the definition of J using standard estimates. The proof is therefore omitted. □

We conclude this section with a brief discussion of the differentiability of the velocity function. Suppose that we have an area function $\bar{w} = (\bar{w}_L, \bar{w}_R)$ and corresponding velocities $\bar{u}_L, \bar{u}_R$ and shock location $\bar{x}_s$ that satisfy the state equations (2.40), (2.41), (2.42), (2.43) and are such that (2.60) is fulfilled. Then the implicit function theorem guarantees the differentiability of the function

$$L^\infty \times L^\infty \ni (w_L, w_R) \longrightarrow (u_L, u_R, x_s) \in W^{1,\infty} \times W^{1,\infty} \times I\!R$$

that maps the area into the solution of the state equation at this point. In fact, the derivative is given by

$$(q(\bar{w}))_w = -C_q(\bar{u}_L, \bar{u}_R, \bar{x}_s, \bar{w})^{-1} C_w(\bar{u}_L, \bar{u}_R, \bar{x}_s, \bar{w}).$$

Given u_L, u_R, x_s, the velocity of the original problem can be obtained as

$$u(x) = \begin{cases} u_L\left(\frac{x}{x_s}\right), & x \in [0, x_s), \\ u_R\left(\frac{1-x}{1-x_s}\right), & x \in [x_s, 1]. \end{cases} \tag{2.66}$$

If one considers the map

$$W^{1,\infty} \times W^{1,\infty} \times I\!R \ni (u_L, u_R, x_s) \longrightarrow u \in L^\infty$$

that is defined by (2.66), then it is easy to see that because of the presence of a shock this map is not Fréchet differentiable. In fact it is not even continuous. This shows that differentiability is only lost when the composite function

$$(w_L, w_R) \longrightarrow (u_L, u_R, x_s) \longrightarrow u$$

is considered. If left and right velocity and shock location are treated as independent variables, then, as shown in this section, differentiability can be guaranteed under suitable assumptions.

2.5.3 Optimality Conditions

We define the Lagrange function

$$\begin{aligned} & L(q,w,\lambda) \\ &= \frac{x_s}{2}\int_0^1 (u_L-\hat{u}_L)^2\,d\xi + \frac{1-x_s}{2}\int_0^1 (u_R-\hat{u}_R)^2\,d\xi + \int_0^1 \lambda_L[(f(u_L))_\xi + x_s g(u_L,w_L)]\,d\xi \\ &\quad + \int_0^1 \lambda_R[(f(u_R))_\xi + (x_s-1)g(u_R,w_R)]\,d\xi + \lambda_s[f(u_L(1)) - f(u_R(1))]. \end{aligned} \tag{2.67}$$

where

$$\lambda = (\lambda_L, \lambda_R, \lambda_s)$$

If the shock location at the optimum obeys $x_s \in (0,1)$, then the first order necessary optimality conditions are

$$\begin{aligned} 0 &= L_{(u_L,u_R,x_s)}(q,w,\lambda)(\tilde{u}_L,\tilde{u}_R,\tilde{x}_s), \\ 0 &\le L_{(w_L,w_R)}(q,w,\lambda)(\tilde{w}_L,\tilde{w}_R), \\ 0 &= L_{(\lambda_L,\lambda_R,\lambda_s)}(q,w,\lambda)(\tilde{\lambda}_L,\tilde{\lambda}_R,\tilde{\lambda}_s), \end{aligned} \tag{2.68}$$

for all $(\tilde{u}_L,\tilde{u}_R,\tilde{x}_s)$ with $\tilde{u}_L(0) = \tilde{u}_R(0) = 0$, for all $(\tilde{w}_L,\tilde{w}_R)$ with $(w_L+\tilde{w}_L, w_R+\tilde{w}_R) \in \mathcal{W}_{ad}$, and for all $(\tilde{\lambda}_L,\tilde{\lambda}_R,\tilde{\lambda}_s)$.

The third equation in (2.68) yields the state equation (2.40), (2.41), (2.42), (2.43). Using integration by parts we find that the first equation in (2.68) with $\tilde{u}_L(0) = \tilde{u}_R(0) = 0$ yields

$$\begin{aligned} 0 &= L_{(u_L,u_R,x_s)}(q,w,\lambda)(\tilde{u}_L,\tilde{u}_R,\tilde{x}_s) \\ &= x_s\int_0^1 (u_L-\hat{u}_L)\tilde{u}_L\,d\xi + (1-x_s)\int_0^1 (u_R-\hat{u}_R)\tilde{u}_R\,d\xi + J_{x_s}(q,w)\tilde{x}_s \\ &\quad + \int_0^1 -(\lambda_L)_\xi f_u(u_L)\tilde{u}_L + \lambda_L[x_s g_u(u_L,w_L)\tilde{u}_L + \tilde{x}_s g(u_L,w_L)]\,d\xi \\ &\quad + \int_0^1 -(\lambda_R)_\xi f_u(u_R)\tilde{u}_R + \lambda_R[(x_s-1)g_u(u_R,w_R)\tilde{u}_R + \tilde{x}_s g(u_R,w_R)]\,d\xi \\ &\quad + (\lambda_L(1)+\lambda_s)f_u(u_L(1))\tilde{u}_L(1) + (\lambda_R(1)-\lambda_s)f_u(u_R(1))\tilde{u}_R(1). \end{aligned} \tag{2.69}$$

If one sets $\tilde{x}_s = 0$ and varies over all $(\tilde{u}_L,\tilde{u}_R)$ with $\tilde{u}_L(0) = \tilde{u}_R(0) = \tilde{u}_L(1) = \tilde{u}_R(1) = 0$, then one obtains the adjoint equations

$$(\lambda_L)_\xi\ f_u(u_L) = x_s g_u(u_L,w_L)\ \lambda_L + x_s(u_L-\hat{u}_L), \tag{2.70}$$

$$(\lambda_R)_\xi\ f_u(u_R) = (x_s-1)g_u(u_R,w_R)\ \lambda_R + (1-x_s)(u_R-\hat{u}_R). \tag{2.71}$$

Allowing $\tilde{u}_L(1), \tilde{u}_R(1) \neq 0$ yields the conditions

$$\lambda_L(1) = -\lambda_s, \quad \lambda_R(1) = \lambda_s. \tag{2.72}$$

Finally, varying $\tilde{x}_s$ gives

$$\int_0^1 \lambda_L g(u_L, w_L) + \lambda_R g(u_R, w_R)\, d\xi + J_{x_s}(q, w) = 0. \tag{2.73}$$

Existence of Lagrange multipliers are guaranteed if the operator of linearized constraints is onto. Thus, existence of Lagrange multipliers is expected under the assumptions of Theorem 2.5.4(i). We provide a proof of this result, since the explicit form of the Lagrange multipliers derived in the proof are of interest in connection with the discretized problem.

Theorem 2.5.7 *If the assumptions of Theorem 2.5.4(i) are valid, then the adjoint system (2.70), (2.71), (2.72), (2.73) admits a unique solution.*

Proof: Equation (2.70) is equivalent to

$$(\lambda_L)_\xi - \frac{x_s g_u(u_L, w_L)}{f_u(u_L)}\, \lambda_L = x_s\, \frac{u_L - \hat{u}_L}{f_u(u_L)}.$$

Using the integrating factor

$$\nu_L(\xi) = \exp\left(\int_\xi^1 \frac{x_s g_u(u_L(t), w_L(t))}{f_u(u_L(t))}\, dt \right),$$

the solution of (2.70) with $\lambda_L(1) = -\lambda_s$ is given by

$$(\lambda_L)(\xi) = \frac{1}{\nu_L(\xi)} \left(-\lambda_s - \int_\xi^1 x_s\, \nu_L\, \frac{u_L - \hat{u}_L}{f_u(u_L)}\, dt \right). \tag{2.74}$$

Similarly, the solution of (2.71) with $\lambda_R(1) = \lambda_s$ is given by

$$(\lambda_R)(\xi) = \frac{1}{\nu_R(\xi)} \left(\lambda_s - \int_\xi^1 (1 - x_s)\, \nu_R\, \frac{u_R - \hat{u}_R}{f_u(u_R)}\, dt \right), \tag{2.75}$$

where

$$\nu_R(\xi) = \exp\left(\int_\xi^1 \frac{(x_s - 1) g_u(u_R(t), w_R(t))}{f_u(u_R(t))}\, dt \right).$$

Inserting the solutions into (2.73), we find that

$$\begin{aligned}\int_0^1 \exp\left(-\int_\xi^1 \tfrac{x_s g_u(u_L,w_L)}{f_u(u_L)}\,ds\right)\, g(u_L,w_L)\, - \exp\left(-\int_\xi^1 \tfrac{(x_s-1) g_u(u_R,w_R)}{f_u(u_R)}\,ds\right)\, g(u_R,w_R)\,d\xi\;\; \lambda_s \\ = J_{x_s}(q,w) - \int_0^1 \left\{ \int_\xi^1 (1-x_s) \exp\left(-\int_\xi^t \tfrac{(x_s-1) g_u(u_R,w_R)}{f_u(u_R)}\,ds\right) \tfrac{u_R-\hat{u}_R}{f_u(u_R)}\,dt \right. \\ \left. + \int_\xi^1 x_s \exp\left(-\int_\xi^t \tfrac{x_s g_u(u_L,w_L)}{f_u(u_L)}\,ds\right) \tfrac{u_L-\hat{u}_L}{f_u(u_L)}\,dt \right\} d\xi.\end{aligned} \tag{2.76}$$

Since (2.60) holds true, equation (2.76) has a unique solution λ_s. This concludes the proof of the theorem. □

The second equation in (2.68) is equivalent to

$$\lambda_L(\xi)\; x_s \left(\bar{\gamma} u_L(\xi) - \frac{\bar{H}}{u_L(\xi)}\right) \begin{cases} \geq 0 & \text{if } w_L(\xi) = w_{\min}, \\ = 0 & \text{if } w_L(\xi) \in (w_{\min}, w_{\max}), \\ \leq 0 & \text{if } w_L(\xi) = w_{\max}, \end{cases} \tag{2.77}$$

and

$$\lambda_R(\xi)\; (x_s - 1) \left(\bar{\gamma} u_R(\xi) - \frac{\bar{H}}{u_R(\xi)}\right) \begin{cases} \geq 0 & \text{if } w_R(\xi) = w_{\min}, \\ = 0 & \text{if } w_R(\xi) \in (w_{\min}, w_{\max}), \\ \leq 0 & \text{if } w_R(\xi) = w_{\max}. \end{cases} \tag{2.78}$$

We conclude this section with a brief discussion of the continuity properties of the Lagrange multiplier. The co-states λ_L and λ_R obey the shock condition (2.72). From the examination of the other equations it can be seen that $\lambda_L(1) = -\lambda_R(1) = 0$ can be guaranteed only if the desired velocities can be matched exactly, *i.e.*, the source terms in (2.70), (2.71), and the term $J_{x_s}(q,w)$ in (2.73) vanish. For a related design problem governed by the full Euler equations Iollo *et al.* [82] concluded that the co-state is continuous and zero at the shock location. Even though Iollo *et al.* [82] do not use a shock fitting method and even though the problem investigated in the present research is a reduction of the one-dimensional Euler equations and therefore a simplification of the problem studied in [82] our result indicates a different behavior of the co-states. The fact that for this problem the co-states are nonzero at the shock if the optimal value of the objective function is nonzero can also be observed in the numerical experiments. See Section 2.5.5. We have, in fact, generalized the results [35] obtained in this section to the full 1-D Euler equations (2.1), and our analysis does indicate nonzero and discontinuous behavior of the co-state at the shock location.

2.5.4 The Discrete Design Problem

As in the analysis of the continuous problem we divide the interval into two subintervals $[0, x_s]$ and $[x_s, 1]$. The shock location x_s is one of the state variables. The transformation onto the fixed domain as shown in Section 2.5, however, is not performed explicitly, but incorporated implicitly using moving grids on the left and on the right of the shock. As we will see later, this is equivalent to discretizing the fixed domain control problem (2.39)–(2.45).

For the discretization of the optimal control problem we use a cell centered grid. The subinterval $[0, x_s]$ left of the shock is subdivided into N_L equidistant subintervals of length $h_L = x_s/N_L$, the subinterval $[x_s, 1]$ right of the shock is subdivided into N_R equidistant subintervals of length $h_R = (1 - x_s)/N_R$, shown in figure 2.14. The point x_i denotes the midpoint of the ith cell:

$$\begin{aligned} x_i &= (i - \tfrac{1}{2})h_L, && i = 1, \ldots, N_L, \\ x_i &= x_s + (i - \tfrac{1}{2} - N_L)h_R, && i = N_L + 1, \ldots, N_L + N_R. \end{aligned} \tag{2.79}$$

On each cell the function w defining the area of the duct and the velocity u are approximated by constants w_i and u_i, respectively. Therefore the number of state variables is $N_L + N_R + 1$ and the number of control variables is $N_L + N_R$.

The objective function is discretized using the midpoint rule:

$$\frac{x_s}{2} \int_0^1 (u_L(\xi) - \hat{u}_L(x_s; \xi))^2 \, d\xi + \frac{1 - x_s}{2} \int_0^1 (u_R(\xi) - \hat{u}_R(x_s; \xi))^2 \, d\xi \;\approx\; J^h(u, x_s, w),$$

where

$$J^h(u, x_s, w) \equiv \frac{1}{2} \sum_{i=1}^{N_L} h_L \, (u_i - \hat{u}(x_i))^2 + \frac{1}{2} \sum_{i=N_L+1}^{N_L+N_R} h_R \, (u_i - \hat{u}(x_i))^2. \tag{2.80}$$

For the discretization of the differential equation (2.31) we use

$$\frac{f_{i+1/2} - f_{i-1/2}}{\Delta x_i} + g(u_i, w_i) = 0,$$

where Δx_i denotes the width of cell i. The fluxes at the cell boundaries are approximated as follows: In the supersonic region left of the shock we set $f_{i+1/2} = f(u_i)$ giving

$$\frac{f(u_i) - f(u_{i-1})}{h_L} + g(u_i, w_i) = 0, \quad i = 1, \ldots, N_L. \tag{2.81}$$

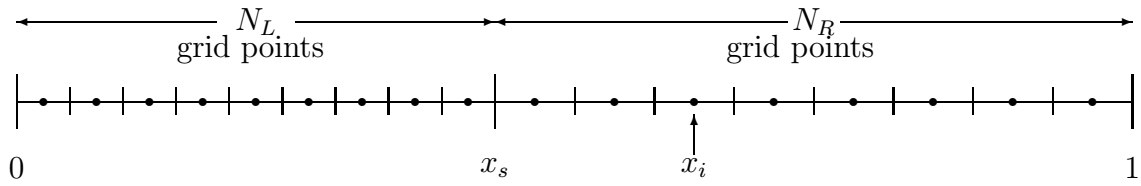


Figure 2.14: The Grid for Shock-Fitting Formulation

In the subsonic region right of the shock we set $f_{i+1/2} = f(u_{i+1})$ giving

$$\frac{f(u_{i+1}) - f(u_i)}{h_R} + g(u_i, w_i) = 0, \quad i = N_L + 1, \dots, N_L + N_R. \tag{2.82}$$

The Rankine-Hugoniot condition (2.7) is discretized as

$$f(u_{N_L}) - f(u_{N_L+1}) = 0. \tag{2.83}$$

The equations (2.81), (2.82) are also the ones used in the Godunov scheme for the supersonic and subsonic regions, respectively. See *e.g.*, [55].

If one multiplies (2.81) by x_s and (2.82) by $(x_s - 1)$, then one can see that the resulting equations are implicit discretization schemes for (2.40) and (2.41). For (2.40) the indices $i = 0$ and $i = N_L$ correspond to the boundaries $\xi = 0$ and $\xi = 1$, respectively, whereas for (2.41) the indices $i = N_L + N_R$ and $i = N_L + 1$ correspond to the boundaries $\xi = 0$ and $\xi = 1$, respectively.

The equations (2.81) and (2.82) multiplied by x_s and $x_s - 1$, respectively, and the equation (2.83) form the $N_L + N_R + 1$ state constraints

$$C_i^h(u, x_s, w) = 0, \quad i = 1, \dots, N_L + N_R + 1,$$

where

$$C_i^h(u, x_s, w) \equiv \begin{cases} N_L \left(f(u_i) - f(u_{i-1})\right) + x_s \, g(u_i, w_i) & i = 1, \dots, N_L, \\ N_R \left(-f(u_{i+1}) + f(u_i)\right) + (x_s - 1) \, g(u_i, w_i) & i = N_L + 1, \dots, N_L + N_R, \\ f(u_{N_L}) - f(u_{N_L+1}) & i = N_L + N_R + 1. \end{cases} \tag{2.84}$$

The scalars u_0 and $u_{N_L+N_R+1}$ are determined from the boundary conditions (2.43).

This leads to the finite dimensional optimal control problem

$$\begin{aligned} \min \quad & J^h(u, x_s, w) \\ \text{s.t.} \quad & C^h(u, x_s, w) = 0, \\ & 0 \le w_{\min} \le w \le w_{\max}. \end{aligned} \tag{2.85}$$

The state variables in the discrete problem are $(u_1, \dots, u_{N_L+N_R}, x_s)$ and the control variables are $(w_1, \dots, w_{N_L+N_R})$. One may add a state constraint $0 \le x_s \le 1$ to (2.85). However, in our numerical experiments the shock was always in the interior.

Under the assumptions of Theorem 2.5.6 the discretized objective function J^h is differentiable. In fact, due to the discretization, one can even relax the differentiability assumptions on $\hat{u}$. The objective function J^h is differentiable if $\hat{u}$ is differentiable at $x_i,\ i = 1, \ldots, N_L + N_R$. In particular, it holds that

$$\begin{aligned} J^h_{x_s}(u, x_s, w) &= \sum_{i=1}^{N_L} \frac{1}{2N_L} (u_i - \hat{u}(x_i))^2 - h_L \frac{i - \frac{1}{2}}{N_L} (u_i - \hat{u}(x_i))\ \hat{u}_x(x_i) \\ &\quad - \sum_{i=N_L+1}^{N_L+N_R} \frac{1}{2N_R} (u_i - \hat{u}(x_i))^2 + h_R \Big(1 - \frac{i - \frac{1}{2} - N_L}{N_R}\Big)(u_i - \hat{u}(x_i))\ \hat{u}_x(x_i). \end{aligned}$$

From the definition of f and g, it is easy to see that C^h is differentiable for all u, x_s, q with $u > 0$. The partial Jacobian $C^h_{(u,x_s)}$ of C^h is a bordered matrix given by

$$C^h_{(u,x_s)}(u, x_s, w) = \begin{pmatrix} B_L & 0 & e_L \\ 0 & B_R & e_R \\ d_L^T & d_R^T & 0 \end{pmatrix}, \tag{2.86}$$

where $B_L \in I\!R^{N_L \times N_L}$ is a lower bidiagonal matrix, $B_R \in I\!R^{N_R \times N_R}$ is a upper bidiagonal matrix, and $e_L, d_L \in I\!R^{N_L}$, $e_R, d_R \in I\!R^{N_R}$. The structure of the matrix reflects the left hand side of the system (2.56), (2.57), (2.58), (2.59). The partial Jacobian C^h_q of C^h is a $(N_L + N_R + 1) \times (N_L + N_R)$ 'diagonal' matrix with diagonal entries given by

$$\Big(C^h_q(u, x_s, w)\Big)_{ii} = \begin{cases} x_s\ g_q(u_i, w_i) & i = 1, \ldots, N_L, \\ (x_s - 1)\ g_q(u_i, w_i) & i = N_L + 1, \ldots, N_L + N_R. \end{cases}$$

Note that the function g depends linearly on w, *cf.* (2.32).

If $C^h_{(u,x_s)}(u, x_s, w), B_L, B_R$ are nonsingular, the linear system

$$\begin{pmatrix} B_L & 0 & e_L \\ 0 & B_R & e_R \\ d_L^T & d_R^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{u}_L \\ \tilde{u}_R \\ \tilde{x}_s \end{pmatrix} = \begin{pmatrix} r_L \\ r_R \\ r_s \end{pmatrix}$$

can be solved using

$$\tilde{u}_L = B_L^{-1}(r_L - e_L\, \tilde{x}_s), \tag{2.87}$$

$$\tilde{u}_R = B_R^{-1}(r_R - e_R\, \tilde{x}_s), \tag{2.88}$$

where

$$\tilde{x}_s = (d_L^T B_L^{-1} e_L + d_R^T B_R^{-1} e_R)^{-1}(d_L^T B_L^{-1} r_L + d_R^T B_R^{-1} r_R - r_s). \tag{2.89}$$

This solution procedure is the discrete version of the procedure applied in the proof of Theorem 2.5.4 to establish the existence of a unique solution of (2.56), (2.57), (2.58), (2.59). It is also this solution procedure that is used in our numerical examples.

Theorem 2.5.8 *If*

$$u_i > \sqrt{\bar{H}}, \quad i = 1, \ldots, N_L, \tag{2.90}$$

then B_L is nonsingular. If

$$u_i \in (0, \sqrt{\bar{H}}), \quad i = N_L + 1, \ldots, N_L + N_R, \tag{2.91}$$

then B_R is nonsingular, and if (2.90), (2.91), and

$$\begin{aligned} & \frac{1}{N_L} \sum_{j=1}^{N_L} \left(\prod_{k=j}^{N_L} \frac{N_L\, f_u(u_k) + x_s\, g_u(u_k, w_k)}{N_L\, f_u(u_k)} \right)^{-1} g(u_j, w_j) \\ \neq \; & \frac{1}{N_R} \sum_{j=N_L+1}^{N_L+N_R} \left(\prod_{k=N_L+1}^{j} \frac{N_R\, f_u(u_k) + (x_s - 1)\, g_u(u_k, w_k)}{N_R\, f_u(u_k)} \right)^{-1} g(u_j, w_j), \end{aligned} \tag{2.92}$$

then the matrix $C^h_{(u,x_s)}(u, x_s, w)$ is nonsingular.

Proof: The ith equation of the system $B_L \tilde{u}_L = r_L - e_L\, \tilde{x}_s$ is given by

$$\begin{aligned} \Big(N_L\, f_u(u_i) + x_s\, g_u(u_i, w_i) \Big) (\tilde{u}_L)_i - N_L\, f_u(u_{i-1}) (\tilde{u}_L)_{i-1} & \\ = -g(u_i, w_i)\tilde{x}_s + (r_L)_i, \quad i = 1, \ldots, N_L, & \end{aligned} \tag{2.93}$$

where $(\tilde{u}_L)_0 = 0$. Condition (2.90) guarantees that $N_L\, f_u(u_i) + x_s\, g_u(u_i, w_i) > 0$, $i = 1, \ldots, N_L$. With

$$\begin{aligned} z_i &= \frac{N_L\, f_u(u_{i-1})}{N_L\, f_u(u_i) + x_s\, g_u(u_i, w_i)}, \\ t_i &= \frac{g(u_i, w_i)}{N_L\, f_u(u_i) + x_s\, g_u(u_i, w_i)}, \\ v_i &= \frac{(r_L)_i}{N_L\, f_u(u_i) + x_s\, g_u(u_i, w_i)}, \end{aligned}$$

these equations can be written as

$$(\tilde{u}_L)_i - z_i (\tilde{u}_L)_{i-1} = -t_i \tilde{x}_s + v_i, \quad i = 1, \ldots, N_L,$$

If we multiply the last difference equation by $1/Z_i$, where $Z_i = \prod_{j=1}^{i} z_j,\ i = 1, \ldots, N_L$, $Z_0 = 1$, then we obtain

$$\frac{1}{Z_i}(\tilde{u}_L)_i - \frac{1}{Z_{i-1}}(\tilde{u}_L)_{i-1} = -\frac{t_i}{Z_i}\tilde{x}_s + \frac{v_i}{Z_i}, \quad i = 1, \ldots, N_L,$$

The solution of this difference equation is given by

$$(\tilde{u}_L)_i = Z_i \left(\sum_{j=1}^{i} -\frac{t_j}{Z_j}\tilde{x}_s + \frac{v_j}{Z_j} \right), \quad i = 1, \ldots, N_L.$$

In particular, it holds that

$$\begin{aligned} (\tilde{u}_L)_{N_L} &= \frac{\prod_{k=1}^{N_L-1} N_L\ f_u(u_k)}{\prod_{k=1}^{N_L} N_L\ f_u(u_k) + x_s\ g_u(u_k, w_k)} \times \\ &\Big(-\sum_{j=1}^{N_L} \prod_{k=1}^{j-1} \frac{N_L\ f_u(u_k) + x_s\ g_u(u_k, w_k)}{N_L\ f_u(u_k)}\ g(u_j, w_j)\ \tilde{x}_s\ + \sum_{j=1}^{N_L} \frac{v_j}{Z_j} \Big). \end{aligned} \tag{2.94}$$

The ith equation of the system $B_R \tilde{u}_R = r_R - e_R\,\tilde{x}_s$ is given by

$$\begin{aligned} \Big(N_R\ f_u(u_i) + (x_s - 1)\ g_u(u_i, w_i) \Big)(\tilde{u}_R)_i - N_R\ f_u(u_{i+1})(\tilde{u}_R)_{i+1} \\ = -g(u_i, w_i)\tilde{x}_s + (r_R)_i, \quad i = N_L + 1, \ldots, N_L + N_R, \end{aligned} \tag{2.95}$$

where $(\tilde{u}_R)_{N_L+N_R+1} = 0$. As before we can rewrite these equations in the form

$$(\tilde{u}_R)_i - z_i(\tilde{u}_R)_{i+1} = -t_i\tilde{x}_s + v_i, \quad i = N_L + 1, \ldots, N_L + N_R,$$

where

$$\begin{aligned} z_i &= \frac{N_R\ f_u(u_{i+1})}{N_R\ f_u(u_i) + (x_s - 1)\ g_u(u_i, w_i)}, \\ t_i &= \frac{g(u_i, w_i)}{N_R\ f_u(u_i) + (x_s - 1)\ g_u(u_i, w_i)}, \\ v_i &= \frac{(r_R)_i}{N_R\ f_u(u_i) + (x_s - 1)\ g_u(u_i, w_i)}. \end{aligned}$$

Note that condition (2.91) implies $N_R\ f_u(u_i) + (x_s - 1)\ g_u(u_i, w_i) < 0,\ i = N_L + 1, \ldots, N_L + N_R$. If we multiply the last difference equation by $1/Z_i$, where $Z_i = \prod_{j=i}^{N_L+N_R} z_j,\ i = N_L + 1, \ldots, N_L + N_R$, $Z_{N_L+N_R+1} = 1$, then we obtain

$$\frac{1}{Z_i}(\tilde{u}_R)_i - \frac{1}{Z_{i+1}}(\tilde{u}_R)_{i-1} = -\frac{t_i}{Z_i}\tilde{x}_s + \frac{v_i}{Z_i}, \quad i = N_L + 1, \ldots, N_L + N_R,$$

The solution of (2.95) is given by

$$(\tilde{u}_R)_i = Z_i \left(\sum_{j=i}^{N_L+N_R} -\frac{t_j}{Z_j}\hat{x}_s + \frac{v_j}{Z_j} \right), \quad i = N_L+1, \ldots, N_L+N_R. \tag{2.96}$$

In particular, it holds that

$$(\tilde{u}_R)_{N_L+1} = \frac{\prod_{k=N_L+2}^{N_L+N_R} N_R\ f_u(u_k)}{\prod_{k=N_L+1}^{N_L+N_R} N_R\ f_u(u_k) + (x_s-1)\ g_u(u_k,w_k)} \ \times$$

$$\Big(- \sum_{j=N_L+1}^{N_L+N_R} \prod_{k=j+1}^{N_L+N_R} \frac{N_R\ f_u(u_k) + (x_s-1)\ g_u(u_k,w_k)}{N_R\ f_u(u_k)}\ g(u_j,w_j)\ \hat{x}_s\ + \sum_{j=N_L+1}^{N_L+N_R} \frac{v_j}{Z_j} \Big).$$

The last equation $d_L^T\tilde{u}_L + d_R^T\tilde{u}_R = r_s$ of the system is equivalent to

$$f_u(u_{N_L})(\tilde{u}_L)_{N_L} - f_u(u_{N_L+1})(\tilde{u}_R)_{N_L+1} = r_s. \tag{2.97}$$

Using the expressions (2.94), (2.96) one can see that the equation (2.97) admits a unique solution $\tilde{x}_s$ if and only if

$$\begin{aligned} & \tfrac{1}{N_L}\prod_{k=1}^{N_L} \tfrac{N_L\ f_u(u_k)}{N_L\ f_u(u_k)+x_s\ g_u(u_k,w_k)} \ \sum_{j=1}^{N_L}\prod_{k=1}^{j-1} \tfrac{N_L\ f_u(u_k)+x_s\ g_u(u_k,w_k)}{N_L\ f_u(u_k)}\ g(u_j,w_j) \\ \neq\ & \tfrac{1}{N_R}\prod_{k=N_L+1}^{N_L+N_R} \tfrac{N_R\ f_u(u_k)}{N_R\ f_u(u_k)+(x_s-1)\ g_u(u_k,w_k)} \ \sum_{j=N_L+1}^{N_L+N_R}\prod_{k=j+1}^{N_L+N_R} \tfrac{N_R\ f_u(u_k)+(x_s-1)\ g_u(u_k,w_k)}{N_R\ f_u(u_k)}\ g(u_j,w_j). \end{aligned}$$

This condition is equivalent to (2.92). □

Remark 2.5.9 *Equation (2.92) is the discretized version of (2.60) with $e^x \approx 1+x$ and*

$$\begin{aligned} \int_{x_j-\frac{1}{2}h_L}^{1} \frac{x_s g_u(u_L(s),w_L(s))}{f_u(u_L(s))}\ ds &\approx \sum_{k=j}^{N_L} \frac{x_s\ g_u(u_k,w_k)}{N_L\ f_u(u_k)}, \\ \int_{x_j+\frac{1}{2}h_R}^{1} \frac{(x_s-1) g_u(u_R(s),w_R(s))}{f_u(u_R(s))}\ ds &\approx \sum_{k=N_L+1}^{j} \frac{(x_s-1)\ g_u(u_k,w_k)}{N_R\ f_u(u_k)}. \end{aligned}$$

The Lagrange function of the discretized design problem (2.85) is given by

$$\begin{aligned} L(u,x_s,w,\lambda) &= \frac{1}{2}\sum_{i=1}^{N_L} h_L\,(u_i-\hat{u}(x_i))^2+\frac{1}{2}\sum_{i=N_L+1}^{N_L+N_R} h_R\,(u_i-\hat{u}(x_i))^2 \\ &\quad+\sum_{i=1}^{N_L}\lambda_i\Big(N_L\,(f(u_i)-f(u_{i-1}))+x_s\,g(u_i,w_i)\Big) \\ &\quad+\sum_{i=N_L+1}^{N_L+N_R}\lambda_i\Big(N_R\,(-f(u_{i+1})+f(u_i))+(x_s-1)\,g(u_i,w_i)\Big) \\ &\quad+\lambda_{N_L+N_R+1}\Big(f(u_{N_L})-f(u_{N_L+1})\Big). \end{aligned} \tag{2.98}$$

The equations $L_{(u,x_s)}(u,x_s,w,\lambda)=0$ are equivalent to

$$\Big(N_L\,f_u(u_i)+x_s\,g_u(u_i,w_i)\Big)\lambda_i-N_L\,f_u(u_i)\lambda_{i+1} = -h_L\Big(u_i-\hat{u}(x_i)\Big), \quad i=1,\dots,N_L-1, \tag{2.99}$$

$$\Big(N_L\,f_u(u_i)+x_s\,g_u(u_i,w_i)\Big)\lambda_i+f_u(u_i)\lambda_{N_L+N_R+1} = -h_L\Big(u_i-\hat{u}(x_i)\Big), \quad i=N_L, \tag{2.100}$$

$$\Big(N_R\,f_u(u_i)+(x_s-1)\,g_u(u_i,w_i)\Big)\lambda_i-f_u(u_i)\lambda_{N_L+N_R+1} = -h_R\Big(u_i-\hat{u}(x_i)\Big), \quad i=N_L+1, \tag{2.101}$$

$$\Big(N_R\,f_u(u_i)+(x_s-1)\,g_u(u_i,w_i)\Big)\lambda_i-N_R\,f_u(u_i)\lambda_{i-1} = -h_R\Big(u_i-\hat{u}(x_i)\Big), \quad i=N_L+2,\dots,N_L+N_R, \tag{2.102}$$

and

$$J^h_{x_s}(u,x_s,w)+\sum_{i=1}^{N_L}\lambda_i\,g(u_i,w_i)+\sum_{i=N_L+1}^{N_L+N_R}\lambda_i\,g(u_i,w_i)=0. \tag{2.103}$$

The system (2.99)–(2.103) is given by

$$\begin{pmatrix} B_L^T & 0 & d_L \\ 0 & B_R^T & d_R \\ e_L^T & e_R^T & 0 \end{pmatrix}\begin{pmatrix} \lambda_L \\ \lambda_R \\ \lambda_s \end{pmatrix}=\begin{pmatrix} r_L \\ r_R \\ r_s \end{pmatrix}, \tag{2.104}$$

where we used the notation

$$\lambda_L=(\lambda_1,\dots,\lambda_{N_L}),\quad \lambda_R=(\lambda_{N_L+1},\dots,\lambda_{N_L+N_R}),\quad \lambda_s=\lambda_{N_L+N_R+1},$$

and

$$\begin{aligned} r_L &= \Big(- h_L(u_1 - \hat{u}(x_1)), \dots , -h_L(u_{N_L+1} - \hat{u}(x_{N_L+1}))\Big), \\ r_R &= \Big(- h_R(u_{N_L+1} - \hat{u}(x_{N_L+1})), \dots , -h_R(u_{N_L+N_R} - \hat{u}(x_{N_L+N_R}))\Big), \\ r_s &= -J^h_{x_s}(u, x_s, w). \end{aligned}$$

The system (2.99)–(2.103) are the adjoint equations of the discretized problem (2.85). However, the equations (2.99)–(2.103) are *not* consistent with the adjoint equations (2.70), (2.71), (2.73).

The inconsistency of the adjoint equations (2.99) to (2.103) of the discretized problem can be removed if we define

$$\begin{aligned} \breve{\lambda}_i &= N_L\lambda_i, \quad i = 1, \dots , N_L, \\ \breve{\lambda}_i &= N_R\lambda_i, \quad i = N_L + 1, \dots , N_L + N_R, \\ \breve{\lambda}_i &= \lambda_i, \qquad i = N_L + N_R + 1. \end{aligned} \tag{2.105}$$

In the scaled Lagrange multipliers, the equations (2.99)–(2.102) are equivalent to

$$- \frac{\breve{\lambda}_{i+1} - \breve{\lambda}_i}{1/N_L} f_u(u_i) + x_s \; g_u(u_i, w_i)\breve{\lambda}_i = -x_s\Big(u_i - \hat{u}(x_i)\Big), \quad i = 1, \dots , N_L - 1, \tag{2.106}$$

$$\Big(N_L \; f_u(u_i) + x_s \; g_u(u_i, w_i)\Big)\breve{\lambda}_i + N_L f_u(u_i)\breve{\lambda}_{N_L+N_R+1} = -x_s\Big(u_i - \hat{u}(x_i)\Big), \quad i = N_L, \tag{2.107}$$

$$\Big(N_R \; f_u(u_i) + (x_s - 1) \; g_u(u_i, w_i)\Big)\breve{\lambda}_i - N_R f_u(u_i)\breve{\lambda}_{N_L+N_R+1} = -(1 - x_s)\Big(u_i - \hat{u}(x_i)\Big), \quad i = N_L + 1, \tag{2.108}$$

$$-\frac{\breve{\lambda}_{i-1} - \breve{\lambda}_i}{1/N_R} f_u(u_i) + (x_s - 1) \; g_u(u_i, w_i)\breve{\lambda}_i = -(1 - x_s)\Big(u_i - \hat{u}(x_i)\Big), \quad i = N_L + 2, \dots , N_L + N_R. \tag{2.109}$$

The equations (2.106) and (2.109) are consistent with the infinite dimensional adjoint equations (2.70) and (2.71). Equation (2.106) is an implicit scheme for (2.70) starting at $x = x_s$ and marching towards $x = 0$, the equation (2.109) is an implicit scheme for (2.71) starting at $x = x_s$ and marching towards $x = 1$.

The equations (2.107) and (2.108) are equivalent to

$$f_u(u_i)\breve{\lambda}_i + f_u(u_i)\breve{\lambda}_{N_L+N_R+1} + \frac{x_s}{N_L}\, g_u(u_i,w_i)\breve{\lambda}_i = -h_L\Big(u_i - \hat{u}(x_i)\Big), \quad i = N_L, \tag{2.110}$$

$$f_u(u_i)\breve{\lambda}_i - f_u(u_i)\breve{\lambda}_{N_L+N_R+1} + \frac{x_s - 1}{N_R}\, g_u(u_i,w_i)\breve{\lambda}_i = -h_R\Big(u_i - \hat{u}(x_i)\Big), \quad i = N_L + 1. \tag{2.111}$$

These equations are consistent with the initial conditions (2.72).

In the scaled Lagrange multipliers, equation (2.103) is written as

$$\sum_{i=1}^{N_L} \breve{\lambda}_i\, \frac{1}{N_L}\, g(u_i,w_i) + \sum_{i=N_L+1}^{N_L+N_R} \breve{\lambda}_i\, \frac{1}{N_R}\, g(u_i,w_i) = -J^h_{x_s}(u,x_s,w) \tag{2.112}$$

which correspond to the equation (2.73).

The system (2.106)–(2.109) and (2.112) is given by

$$\begin{pmatrix} B_L^T & 0 & \breve{d}_L \\ 0 & B_R^T & \breve{d}_R \\ \breve{e}_L^T & \breve{e}_R^T & 0 \end{pmatrix} \begin{pmatrix} \breve{\lambda}_L \\ \breve{\lambda}_R \\ \breve{\lambda}_s \end{pmatrix} = \begin{pmatrix} \breve{r}_L \\ \breve{r}_R \\ \breve{r}_s \end{pmatrix}, \tag{2.113}$$

where we used the notation

$$\breve{\lambda}_L = (\breve{\lambda}_1, \ldots, \breve{\lambda}_{N_L}), \quad \breve{\lambda}_R = (\breve{\lambda}_{N_L+1}, \ldots, \breve{\lambda}_{N_L+N_R}), \quad \breve{\lambda}_s = \breve{\lambda}_{N_L+N_R+1},$$

and

$$\begin{aligned} \breve{r}_L &= \Big(- x_s(u_1 - \hat{u}(x_1)), \ldots, -x_s(u_{N_L} - \hat{u}(x_{N_L}))\Big), \\ \breve{r}_R &= \Big(- (1-x_s)(u_{N_L+1} - \hat{u}(x_{N_L+1})), \ldots, -(1-x_s)(u_{N_L+N_R} - \hat{u}(x_{N_L+N_R}))\Big), \\ \breve{r}_s &= -J^h_{x_s}(u,x_s,w). \end{aligned}$$

The entries in the system matrices in (2.104) and (2.113) are related as follows:

$$\breve{d}_L = N_L\, d_L, \quad \breve{d}_R = N_R\, d_R, \quad \breve{e}_L = \frac{1}{N_L}\, e_L, \quad \breve{e}_R = \frac{1}{N_R}\, e_R.$$

Notice that

$$\begin{pmatrix} B_L & 0 & \breve{e}_L \\ 0 & B_R & \breve{e}_R \\ \breve{d}_L^T & \breve{d}_R^T & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{N_L} I & 0 & 0 \\ 0 & \frac{1}{N_R} I & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} B_L & 0 & e_L \\ 0 & B_R & e_R \\ d_L^T & d_R^T & 0 \end{pmatrix} \begin{pmatrix} N_L I & 0 & 0 \\ 0 & N_R I & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.114}$$

In particular, this relation shows that the system (2.113) is uniquely solvable if and only if (2.104) is uniquely solvable.

The previous considerations raise the question of the "correct" Lagrange multipliers. The Lagrange multipliers λ appear to be the correct ones if one starts with the discrete system. On the other hand, the Lagrange multipliers $\breve{\lambda}$ appear to be the appropriate ones if one tries to establish a relation with the original, infinite dimensional problem. This discrepancy can be overcome, if one chooses the appropriate scalar product for the control space.

If we consider u, x_s as a function of the area function w defined by the discrete state equation $C^h(u, x_s, w) = 0$, then we can write the discrete optimal control problem (2.85) in the reduced form

$$\begin{aligned} \min \quad & \hat{J}^h(w) \equiv J^h(u(w), x_s(w), w) \\ \text{s.t.} \quad & 0 \leq w_{\min} \leq w \leq w_{\max}. \end{aligned} \tag{2.115}$$

For the sake of presentation, we assume that for all w with $0 \leq w_{\min} \leq w \leq w_{\max}$ the equation $C^h(u, x_s, w) = 0$ has a unique solution. Using the implicit function theorem, the derivative of the reduced objective can shown to be

$$\hat{J}^h_w(w)\ \delta w = \Big(\nabla_w J^h(u(w), x_s(w), w) + C^h_w(u(w), x_s(w), w)\lambda \Big)^T\ \delta w.$$

See for example [55], [67]. Hence, the gradient of the reduced objective function with respect to the Euclidean scalar product is given by

$$\nabla_w \hat{J}^h(w) = \nabla_w J^h(u(w), x_s(w), w) + C^h_w(u(w), x_s(w), w)\ \lambda.$$

If we define the scalar product in the discretized control space to be the weighted Euclidean scalar product

$$\langle w_1, w_2 \rangle_{\mathcal{W}_h} = \sum_{i=1}^{N_L} \frac{1}{N_L}\ (w_1)_i (w_2)_i + \sum_{i=N_L+1}^{N_L+N_R} \frac{1}{N_R}\ (w_1)_i (w_2)_i, \tag{2.116}$$

then we find that

$$\hat{J}^h_w(w)\ \delta w = \Big(\Big(C^h_w(u, x_s, w) \Big)^T \lambda \Big)^T\ \delta w = \langle \Big(C^h_w(u, x_s, w) \Big)^T \breve{\lambda}, \delta w \rangle_{\mathcal{W}_h}.$$

Thus, the gradient of the reduced objective function with respect to the weighted Euclidean scalar product is given by

$$\nabla_w \hat{J}^h(w) = \nabla_w J^h(u(w), x_s(w), w) + C^h_w(u(w), x_s(w), w)\ \breve{\lambda}.$$

Moreover, with (2.114) it is easy to see that the system matrix in (2.113) is the adjoint of the Jacobian $C^h_{(u,x_s)}(u, x_s, w)$ with respect to a weighted scalar product. In fact, if we define

$$\langle \lambda_1, \lambda_2 \rangle_{\Lambda_h} = \sum_{i=1}^{N_L} \frac{1}{N_L} (\lambda_1)_i (\lambda_2)_i + \sum_{i=N_L+1}^{N_L+N_R} \frac{1}{N_R} (\lambda_1)_i (\lambda_2)_i + (\lambda_1)_{N_L+N_R+1} (\lambda_2)_{N_L+N_R+1}, \tag{2.117}$$

then we can show that

$$\left\langle \begin{pmatrix} \breve{\lambda}_L \\ \breve{\lambda}_R \\ \breve{\lambda}_s \end{pmatrix}, \begin{pmatrix} B_L & 0 & e_L \\ 0 & B_R & e_R \\ d_L^T & d_R^T & 0 \end{pmatrix} \begin{pmatrix} \breve{u}_L \\ \breve{u}_R \\ \breve{u}_s \end{pmatrix} \right\rangle_{\Lambda_h} = \left\langle \begin{pmatrix} B_L^T & 0 & \breve{d}_L \\ 0 & B_R^T & \breve{d}_R \\ \breve{e}_L^T & \breve{e}_R^T & 0 \end{pmatrix} \begin{pmatrix} \breve{\lambda}_L \\ \breve{\lambda}_R \\ \breve{\lambda}_s \end{pmatrix}, \begin{pmatrix} \breve{u}_L \\ \breve{u}_R \\ \breve{u}_s \end{pmatrix} \right\rangle_{\Lambda_h}.$$

Note that $\langle w_1, w_2 \rangle_{\mathcal{W}_h}$ and $\langle \lambda_1, \lambda_2 \rangle_{\Lambda_h}$ are the appropriate discretizations of the scalar products on $L^2(0,1) \times L^2(0,1)$ and $L^2(0,1) \times L^2(0,1) \times I\!R$, respectively.

2.5.5 Numerical Results

In our numerical experiments the discrete optimal control problem (2.85) is solved using the SQP method, TRICE, described in Section 1.1. We apply a reduced SQP method which uses limited memory BFGS updates for the reduced Hessian. The initial Hessian was chosen to be the identity and the number of updates stored will be denoted by L. In all computations, the trust region was active in the first few iterations. We also point out that since we do not add a regularization term like $\rho(\int_0^1 w_L^2 + \int_0^1 w_R^2)$ in the objective function, the reduced Hessian for the infinite dimensional problem can only expected to be positive semidefinite. The discretization sometimes has a regularizing effect and in this case for a fixed discretization the reduced Hessian for the discretized problem may be positive definite. However, in this case the smallest eigenvalue converges towards zero as the discretization is refined. The lack of positive definiteness will of course effect the convergence behavior. We have made runs with a regularization term as shown above added to the objective function. As expected, the SQP algorithm required fewer iterations. However, to be compatible with the computations reported in Frank and Shubin [55], we have omitted the regularization term here.

One issue which will be emphasized in this section is the influence of the relation between the infinite dimensional problem and its discretization onto the performance of the optimization method. For the algorithms studied in [40] and [68] no convergence theory in infinite dimensional spaces is known yet. However, convergence results for SQP methods in

infinite dimensional spaces are given *e.g.*, in [5], [6], where Lagrange-Newton-SQP methods in Banach spaces using exact second order derivative information are investigated. A convergence analysis of SQP methods in Hilbert spaces allowing quasi–Newton approximations for the second derivative is given in [95]. Here, equality constrained problems are considered, but bound constraints are not included. In all these references, the convergence theory is local and it is assumed that the quadratic programming subproblems are solved exactly. Another reference that is important in this context is [78]. In these references the appropriate implementation of the optimization algorithms for the discretized problems is shown to be important. The underlying infinite dimensional problem dominates the discretized problems. If the discretized problems are treated as finite dimensional nonlinear programming problems, *i.e.*, if the underlying infinite dimensional problem structure is ignored, the performance of the optimization algorithms usually deteriorates as the discretization is refined. The use of weighted scalar products that are obtained from the discretization of the proper scalar products of the infinite dimensional problem emphasize the underlying infinite dimensional character of these problems. The implementation of the SQP algorithm with weighted scalar products as discussed in the previous section is the proper application of the frameworks used in the previous references. In our numerical experiments reported below this leads to a substantial improvement in the performance of the algorithms. We also point out that differentiability of the functions and continuous invertibility of the linearized constraints are important conditions that have to hold in order to formulate the SQP method and to prove its local convergence in the neighborhood of strict local minimizers. For the infinite and finite dimensional version of the design problem, these properties have been established in the previous sections.

The target velocity $\hat{u}$ used is the same as described in Section 2.4.2 and shown in Figure 2.13. The discretization scheme applied to solve (2.4), (2.6), (2.7) also treats the shock location as an explicit variable and approximates the ODE using a scheme corresponding to (2.84). We use 200 subintervals left of the shock and 200 subintervals right of the shock to compute the target velocity. Note that for the construction of the target velocity the area A and not its logarithmic derivative is used. For the formulation of our optimal design problem we need continuous data. These are obtained by using spline interpolations. First we compute two cubic splines using the points $(x_1, u_1), \ldots, (x_{200}, u_{200})$ and $(x_{201}, u_{201}), \ldots, (x_{400}, u_{400})$ and then we join these two cubic splines by constructing a cubic polynomial interpolating $(x_{200}, u_{200}), (x_{201}, u_{201})$ and the derivatives of the two previously constructed splines at x_{200} and x_{201}, respectively. The so computed resulting target velocity $\hat{u}$ is continuously differentiable. Unless stated otherwise, we use the bound constraints $w_{\min} = 0, w_{\max} = 1$.

The starting values for the SQP method are as follows: The initial logarithmic derivative w of the area is chosen to be $w = 0.5$. The initial shock location is computed from the target data and is chosen to be $x_s = \frac{1}{2}(x_{200} + x_{201})$. For the initial velocity we use a piecewise

linear function. On the left of the shock the initial velocity is a linear interpolation between u_{in} at $x = 0$ and $u_{x_s} = 1.7$ at the initial estimate x_s of the shock location. On the right of the shock we use the linear interpolation $u_{x_s} = \bar{H}/1.7$ and u_{out}. This interpolation scheme guarantees that $u \in (\sqrt{\bar{H}}, \sqrt{2\bar{H}})$ left of the shock and $u < \sqrt{\bar{H}}$ right of the shock. If we would simply use $u_i = \hat{u}(x_i)$, then these restrictions on u would not be satisfied if the initial shock location does not match the target shock location.

With these starting values and target data, and the discretization $N_L = N_R = 100$ the initial function value is $J^h(u, x_s, w) \approx 0.9 * 10^{-3}$ and the norm of the residual is $||C^h(u, x_s, w)||_{\Lambda_h} \approx 0.13$. Here, the residual is computed using the weighted norm induced by (2.117). The bound constraints were never active. The necessary optimality conditions show that in this case the Lagrange multipliers $\lambda_1, \ldots, \lambda_{N_L+N_R}$ at the optimum are zero. See also (2.77), (2.78). If the truncation criteria $||C_w^T \lambda||_{I\!R^{N_L+N_R}} < \epsilon$ is used, then the Lagrange multipliers $\lambda_1, \ldots, \lambda_{N_L+N_R}$ are of the order ϵ. Analogous statements hold for the Lagrange multipliers $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{N_L+N_R}$.

In the first set of computations we study the importance of the scalar products for the numerical computations. In these computations the bounds on q were inactive. We use the weighted scalar products introduced in Section 2.5.4 and their corresponding norms. The scalar product $\langle \cdot, \cdot \rangle_{\mathcal{W}_h}$ is used in the truncation criteria $||C_w^* \breve{\lambda}||_{\mathcal{W}_h} < 10^{-5}$ and, more importantly, for the computations of the BFGS updates. The scalar product $\langle \cdot, \cdot \rangle_{\Lambda_h}$ is used to compute quantities like $\langle \breve{\lambda}, C \rangle_{\Lambda_h}$. These computations are compared with the ones in which the discretized problem (2.85) is solved as a nonlinear programming problem in $I\!R^{N_L+N_R} \times I\!R^{N_L+N_R+1}$. The truncation criteria is $||C_w^T \lambda||_{I\!R^{N_L+N_R}} < 10^{-5}$.

First, we observe that the SQP method with weighted scalar products requires significantly fewer iterations to converge. The results are summarized in Table 2.4 in which we compare the two SQP versions for various choices of numbers of updates stored.

The superiority of the SQP method with weighted scalar products over the one with Euclidean scalar products, not only shows in the number of iterations, but also in the quality of the computed solution. Typical results are shown in Figures 2.15 and 2.16. The differences between computed velocity and target velocity and between computed area and the cubic area function are significantly larger for the computations using Euclidean scalar products. Moreover, the results computed using weighted scalar products and limited memory BFGS updates with $L = 30$ or $L = 40$ are virtually identical, whereas, the logarithmic derivatives of the area functions computed using Euclidean scalar products and $L = 30$ or $L = 40$ were significantly different. This shows that the relation between the infinite dimensional problem and its discretization is not only of theoretical interest, but also promises significant advantages from a computational point of view. As we have noted before, the reason for this behavior is that the underlying infinite dimensional problem dominates the dis-

Table 2.4: Number of SQP iterations versus number L of updates stored ($N_L = 100, N_R = 100$).

Using weighted scalar products

L	Iterations
10	68
20	54
30	45
40	45

Using Euclidean scalar products

L	Iterations
10	88
20	75
30	52
40	52

cretized problems. If the discretized problems are treated as finite dimensional nonlinear programming problems, *i.e.*, if the underlying infinite dimensional problem structure is ignored, then the problems often become artificially ill-conditioned and the performance of the optimization algorithms usually deteriorates as the discretization is refined. In our examples, an ill–conditioning is indicated by the oscillating parameter functions w_L, w_R shown in *e.g.*, Figure 2.16. The use of weighted scalar products that are obtained from the discretization of the proper scalar products of the infinite dimensional problem take the underlying infinite dimensional problem structure into account. The resulting implementation of the SQP method is consistent with the formulation of the SQP method in the infinite dimensional framework.

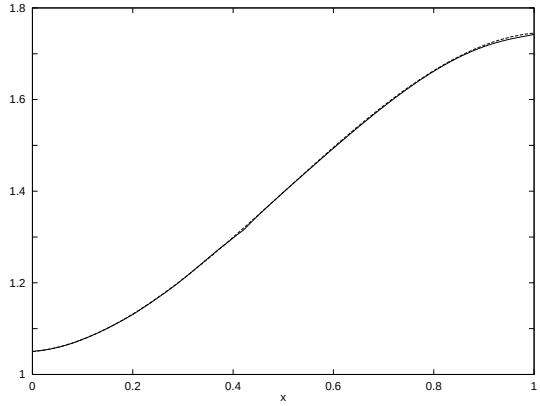
The next results concern target data with errors. Although the target data $\hat{u}$ was constructed using a different discretization for the area than the one used in the optimal design problem, the target data is almost feasible in the sense that we can find an area and a velocity profile such that $u \approx \hat{u}$. In the following we will use nonfeasible target data. This is done by modifying the procedure for the computation of the target data used previously. Now we compute two cubic splines using the points $(x_1, u_1), \ldots, (x_{200-s}, u_{200-s})$ and $(x_{201+s}, u_{201+s}), \ldots, (x_{400}, u_{400})$ and then we join these two cubic splines by constructing a cubic polynomial interpolating $(x_{200-s}, u_{200-s}), (x_{201+s}, u_{201+s})$ and the derivatives of the two previously constructed splines at x_{200+s} and x_{201+s}, respectively. This gives target data that are "smoother" around the shock.

Computations corresponding to the following figures were done using weighted scalar products and the parameters $L = 40$ and $N_L = 100, N_R = 100$. The target data were computed using $s = 10$. The bounds $w_{\min} = 0$ and $w_{\max} = 1$ were both active for some indices, as can be seen in Figure 2.17. The SQP method converged after 74 iterations. The function value at truncation was $J^h = 0.79 * 10^{-3}$, the norm of the constraints was $||C^h||_{\Lambda_h} = 0.75 * 10^{-7}$.

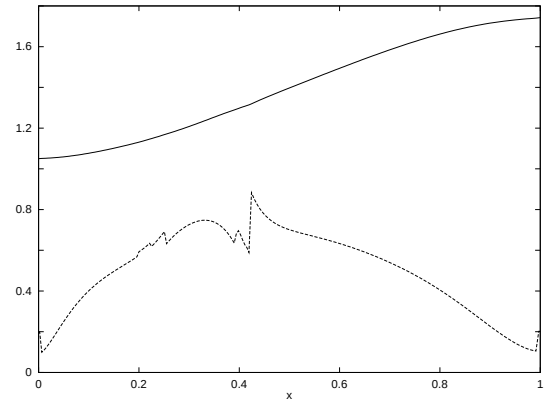
The fact that the logarithmic derivative is zero is due to the fact that the target velocity is not monotonically increasing left of the estimated shock location. In fact, if we consider the infinite dimensional problem, then the state equation (2.34) implies that

$$w_L = \frac{(1 - \bar{H}/u^2)u_x}{x_s(\bar{\gamma}u - \bar{H}/u)}.$$

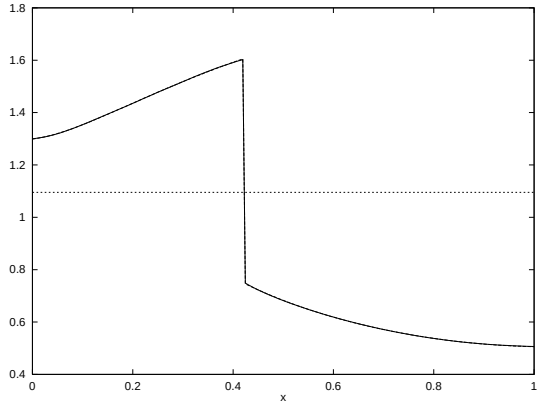
If the target velocity $\hat{u}$ is decreasing left of the computed shock, then, in order to be close to $\hat{u}$, the computed velocity tries to imitate the nature of the target flow and, hence, the logarithmic derivative w_L of the area tries to become negative. See the previous equation for w_L. Of course, the constraints prevent w_L from becoming negative. Similar reasoning can be used to explain why the logarithmic derivative of the area to the right of the computed shock, w_R is at its upper bound, $w_{\max}$.



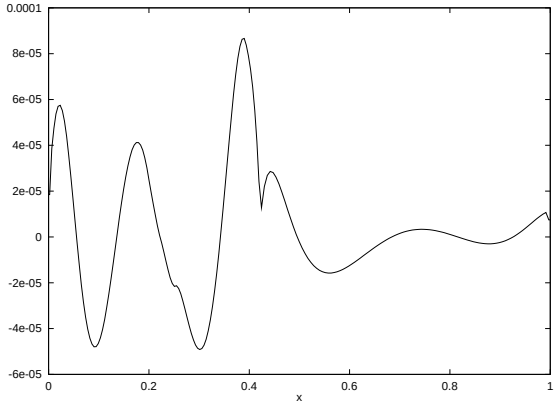
Computed area A.



Computed area A and its logarithmic derivative q.

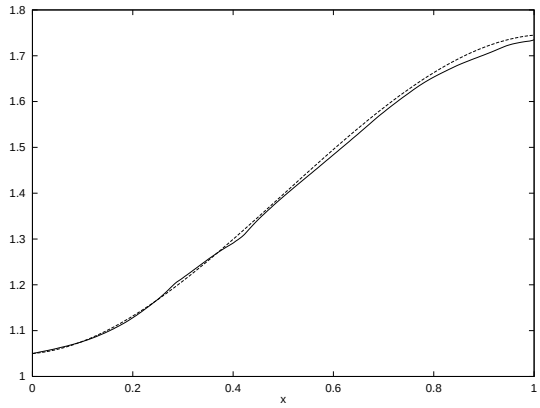


Computed velocity, and sonic speed $\sqrt{\bar{H}}$.

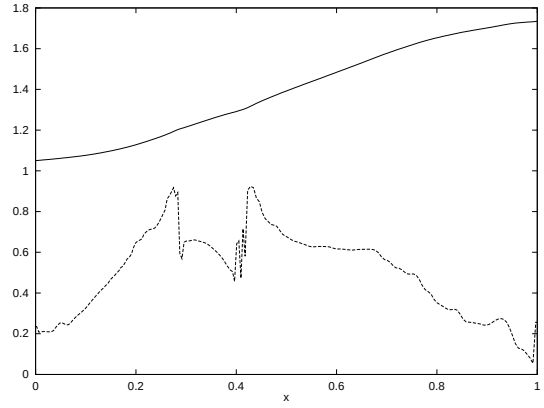


Adjoint

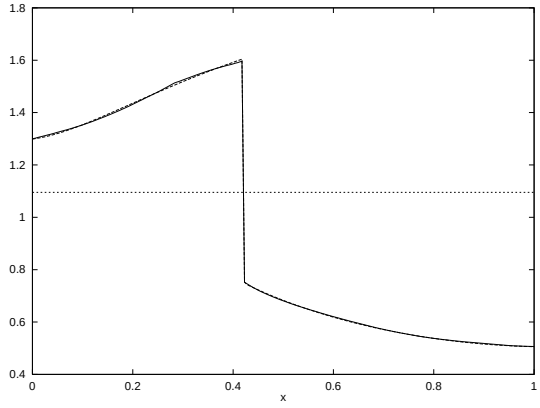
Figure 2.15: Computed area, velocity and adjoint using weighted scalar products and $L = 40, N_L = 100, N_R = 100$.



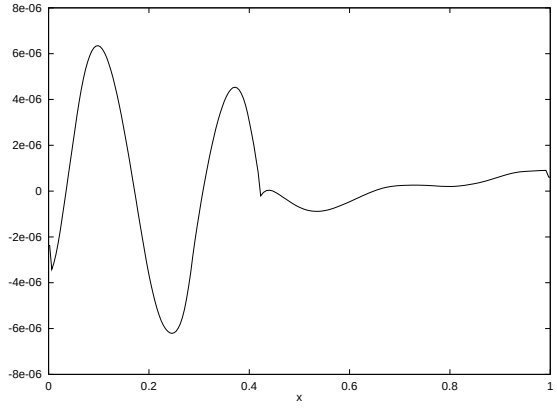
Computed area A.



Computed area A and its logarithmic derivative q.



Computed velocity, and sonic speed $\sqrt{\bar{H}}$.



Adjoint

Figure 2.16: Computed area, velocity and adjoint using Euclidean scalar products and $L = 40, N_L = 100, N_R = 100$.

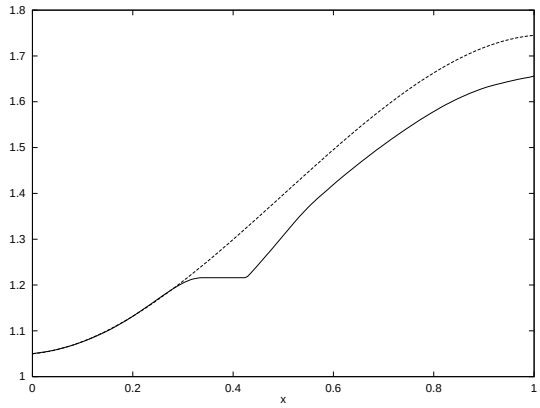
These results should not be surprising, as the Lagrangian (2.98) is linear in the design variable, (w_L, w_R), and optimal control theory tells us that this is a candidate for "bang-bang" control. In these cases, the bounds play an important role in the solution of the problem, as in most cases, a solution would not exist, without bounds. The region where the bound constraints on the design variables are inactive appears to correspond to a case of singular control, and we find the flows are perfectly matched in these regions. These results are similar to those obtained in Section 2.3. In this case the Lagrange multipliers $\breve{\lambda}_1, \ldots, \breve{\lambda}_{N_L+N_R}$ will generally not be zero in regions where the bounds are active, *cf.* (2.77), (2.78). This behavior can be observed in Figure 2.17.

The presence of the lower bound at $w_{\min} = 0$ is important in this case, for the results to make physical sense. The magnitude of the upper bound also seems to be important, as might be expected. We tried to run the same problem with $w_{\min} = 0$ and $w_{\max} = 10$. However, the SQP algorithm stopped because the maximum number of iterations 100 was exceeded. The reason is that a spike evolves in the function q right of the estimated shock.

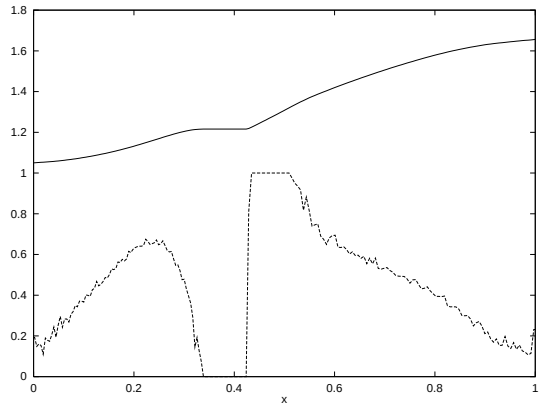
A similar situation prevails for the case where the discretization used for the computed solution, N_L, N_R, exceeds the number of discrete grid points used to represent the target data. For the numerical example discussed above, when N_L or N_R exceeds 200, we find that some subintervals exist in the region around the shock where the left and right target velocities are connected by a cubic spline. Since this cubic spline causes a smoothing, effects similar to those observed with the perturbed, smooth target data discussed previously were observed, for some cases. Thus, if a target velocity similar to the one used here has to be identified, then it seems to be important that the discretization of the problem is sufficiently coarse relative to the target data.

2.6 Conclusions

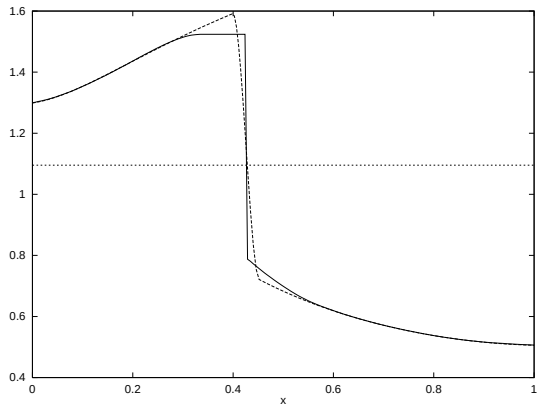
We have studied a design optimization problem involving a compressible flow with a shock. The initial results obtained in Section 2.3 provide good insight into the singular nature of the problem. Though this design problem is fairly simple it does exhibit the inherent characteristics of problems involving transonic flow, and hence is an invaluable asset for preliminary testing of optimization algorithms. Our experiences with shock-capturing formulations to discretize the problem, as reported in Section 2.5, showed poor convergence behavior for the Reduced Hessian method. The differentiability of the constraint functions and the formulation of the optimality conditions in the presence of shocks is a difficult issue. In Section 2.4 we show how this problem can be alleviated by treating the shock location explicitly. This amounts to a shock-fitting concept. We were able to sharply resolve



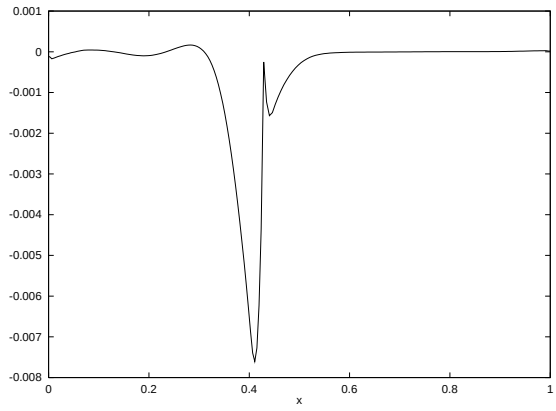
Computed versus 'exact' area A.



Computed area A and its logarithmic derivative q.



Computed velocity, and sonic speed $\sqrt{\bar{H}}$.



Adjoint

Figure 2.17: Computed area, velocity, and adjoint using smoothed data.

the discontinuity while preserving differentiability of the map from design parameters to flow solution. Moreover, under suitable conditions we have established that the linearization of this map is invertible, that this inverse is uniformly bounded in a neighborhood of feasible points and that the usual first order necessary optimality conditions are valid. Theorems 2.5.4 and 2.5.7 and their subsequent discretization in Theorem 2.5.8 and equation (2.113) allow us to justify the use of the Reduced Hessian method because we can guarantee the existence of solutions to the adjoint equation and the linearized state equation, which is required to solve the QP subproblem, as shown in the algorithm in Table 1.1, steps (a) and (d), respectively. This is possible only when the shock location is treated explicitly. An important finding of our study is that the co-state is discontinuous at the shock location, unless the target velocity can be matched perfectly.

The structure of the infinite dimensional problem is inherited by its discretization. However, important observations can be made concerning the numerical solution of the discretized optimal control problem. For the shock-fitting formulation in Section 2.5, one can view the discretized optimal control problem as a nonlinear programming problem in $I\!R^{N_L+N_R} \times I\!R^{N_L+N_R+1}$. On the other hand, one can establish the relation between the original, infinite dimensional problem and its discretization. This relation leads to slight reformulations of the optimality conditions and the introduction of weighted constraints that correspond to the infinite dimensional formulation of the problem. The reformulation has been proven valuable for the numerical performance of the SQP algorithm used for the solution of the optimization problem. The use of weighted scalar products, *i.e.*, the use of the infinite dimensional nature of the problem, reduced the number of iterations significantly and improved the quality of the computed solution.

Our results show that while a straightforward off-the-shelf application of SQP methods will likely fail, a careful analysis of the problem and an incorporation of the problem structure allows the successful application of these powerful methods.

Chapter 3

A Temperature Control Problem

We formulate a thermal-fluid control problem wherein the physics are described by a system of partial differential equations and the control enters through a thermal boundary condition. A finite-element approximation is used to transcribe this to a finite-dimensional Quadratic Programming problem. The finite-dimensional problem displays an expected sparsity pattern in the Jacobian of the constraints and the Hessian of the cost function. Three versions of the QP problem are considered—these differ in their treatment of certain control bounds. Numerical studies show that variants which faithfully reflect the structure of control bounds in the infinite-dimensional problem lead to well-behaved QP solutions, while variants that do not are troublesome for the QP algorithm. It is somewhat surprising that this behavior is apparent even when the finite-element grid is relatively coarse.

3.1 Problem Description

Given below is a coupled solid-fluid temperature control problem, as described by Gunzburger and Lee [63]. We have the governing equations representing the two-dimensional flow of a viscous, incompressible fluid within a solid container and the energy/heat transfer involved. The domain Ω in R^2 consists of the fluid subdomain Ω_2, and the solid subdomain Ω_1, separated by an interface $\Gamma_{\rm b}$, with the result that $\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma_{\rm b}$ (see Figure 3.1). The solid region is bounded by $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_{\rm b}$ and the fluid flow occupies a domain Ω_2 having a boundary $\Gamma_w \cup \Gamma_o \cup \Gamma_{\rm b} \cup \Gamma_4$. We have an inflow boundary $\Gamma_{\rm w}$, an outflow boundary $\Gamma_{\rm o}$, and a solid wall $\Gamma_{\rm b}$. The geometry of all these boundary segments is prescribed.

The problem is motivated by the desire to remove temperature peaks, *i.e.*, "hot spots"

along the bounding surfaces of containers of fluid flows. We desire, then, to regulate the temperature along Γ_b or a portion $\Gamma_r \subset \Gamma_b$. Control is to be effected by heating and cooling along the inflow boundary Γ_w. The heat equation for the solid domain is coupled to the energy equation for the fluid flow. Heat sources may be located in the solid body, the fluid, or both. The flow is assumed to be stationary, incompressible and convection driven, so that buoyancy effects can be neglected, and thus temperature effects on the mechanical properties of the flow, *i.e.*, the velocity and pressure, are negligible. The inflow velocity is prescribed, and "reasonable" boundary conditions may be imposed on the outflow boundary, Γ_o.

As a result of our assumptions about the flow, the state variables, *i.e.*, the velocity $\mathbf{u}$, pressure p, temperature T, and control w are required to satisfy the Navier-Stokes equations

$$-\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega_2, \tag{3.1}$$

the incompressibility constraint

$$\operatorname{div}\mathbf{u} = 0 \quad \text{in } \Omega_2, \tag{3.2}$$

and, for simplicity, the boundary condition

$$\begin{aligned} \mathbf{u} &= \mathbf{h} \quad \text{on } \Gamma_w, && (3.3)\\ \mathbf{u} &= \mathbf{0} \quad \text{on } \Gamma_b \cup \Gamma_4, && (3.4)\\ \frac{\partial\mathbf{u}}{\partial n} &= \mathbf{0} \quad \text{on } \Gamma_o, && (3.5) \end{aligned}$$

and the energy equations

$$-\kappa_1\Delta T = \bar{Q}_1 \quad \text{in } \Omega_1, \tag{3.6}$$

$$\begin{aligned} -\kappa_2\Delta T + (\mathbf{u}\cdot\nabla)T &= \bar{Q}_2 \\ +2\mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T) &: (\nabla\mathbf{u} + \nabla\mathbf{u}^T) \quad \text{in } \Omega_2, \end{aligned} \tag{3.7}$$

with the boundary conditions

$$\begin{aligned} T &= w \quad \text{on } \Gamma_w, && (3.8)\\ \frac{\partial T}{\partial n} &= 0 \quad \text{on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4 \cup \Gamma_o. && (3.9) \end{aligned}$$

The data functions $\mathbf{f}$, $\mathbf{h}$, $\bar{Q}_1$ and $\bar{Q}_2$ are assumed to be known. The constant ν is the kinematic viscosity coefficient of the fluid, and the constants κ_1, κ_2 and μ depend on the thermal conductivity coefficient, density, specific heat at constant volume, and viscosity coefficient of the fluid; see Serrin [125] for details.

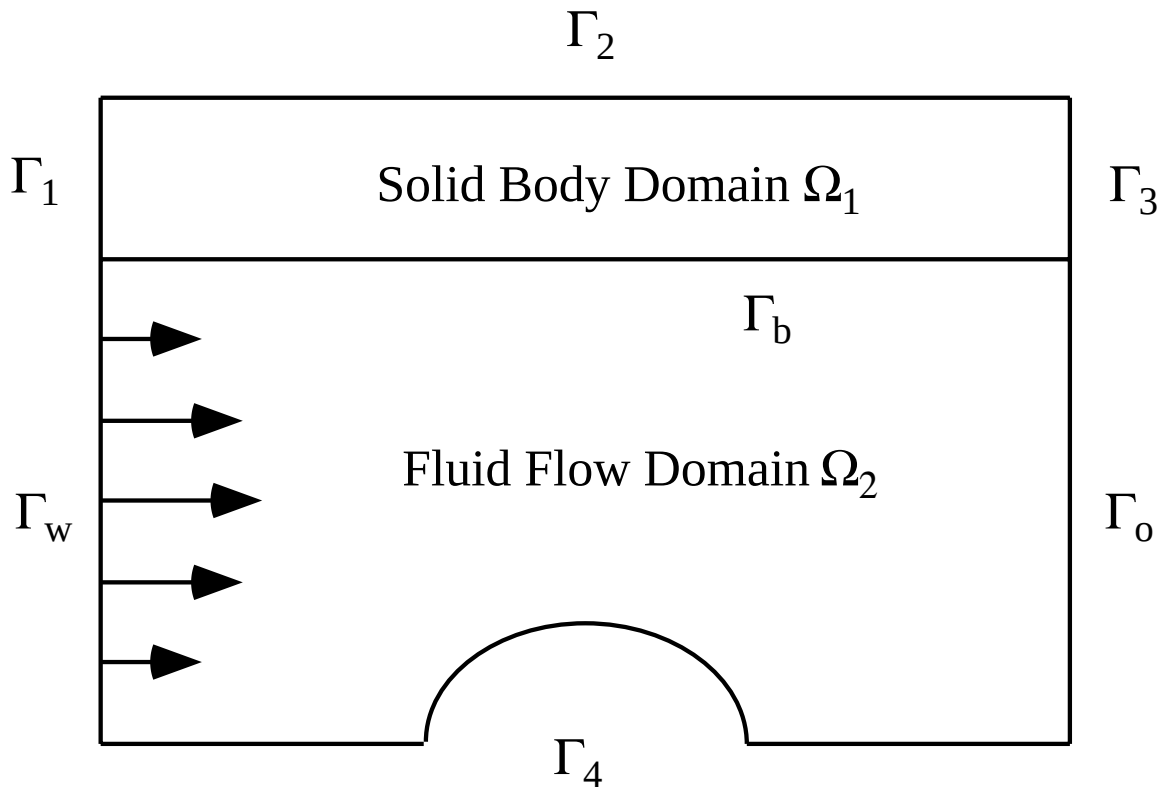


Figure 3.1: The Domain Ω

Note that as a result of our assumptions about the flow, the mechanical equations (3.1)–(3.5) uncouple from the thermal equations (3.6)–(3.9). Indeed, (3.1)–(3.5) may be solved for $\mathbf{u}$ and p without regard of the temperature T. Thus, in the present context, the velocity field $\mathbf{u}$, which is determined by solving (3.1)–(3.5), merely acts as a coefficient function and in the source term in (3.7).

Our goal is to find, for a given velocity field $\mathbf{u}$, a boundary control w such that on Γ_{r} the temperature field T given by (3.6)–(3.9) is as close to the desired temperature $\hat{T}(\cdot)$ as possible. We measure closeness in the least squares sense. This leads to the following objective function:

$$J(T,w) = \frac{1}{2}\int_{\Gamma_{\mathrm{r}}} |T-\hat{T}|^2 \, d\Gamma + \frac{\delta}{2}\int_{\Gamma_{\mathrm{w}}} (|w|^2 + |\nabla_s w|^2) \, d\Gamma. \tag{3.10}$$

Here ∇_s denotes the surface gradient operator. The non-negative parameter δ acts as a regularization parameter and can be used to change the relative importance of the two terms appearing in the definition of J.

3.2 The Optimal Control Problems

The first optimal control problem investigated in this work is given by

$$\begin{aligned} &\text{Minimize} \quad J(T,w) \\ &\text{subject to the equations (3.6)–(3.9).} \end{aligned} \tag{P1}$$

Here we assume that $\delta > 0$. This control problem has been studied by Gunzburger and Lee [63]. They have shown that if the flow satisfies

$$\mathbf{u}\cdot\mathbf{n} \;\geq\; 0 \quad \text{on } \Gamma_o, \tag{3.11}$$

then the state equation (3.6)–(3.9) has a unique solution $T \in H^1(\Omega)$ for all $w \in \mathcal{W}(\Gamma_{\mathrm{w}}), \bar{Q}_1 \in H^{-1}(\Omega_1)$, $\bar{Q}_2 \in H^{-1}(\Omega_2)$, which depends continuously on these data. The function spaces are given by

$$H^1(\Omega) = \{v \mid v \in L^2(\Omega), \frac{\partial}{\partial x_j} v \in L^2(\Omega), j = 1,2\}$$

and

$$\mathcal{W}(\Gamma_{\mathrm{w}}) = \{w \in H^1(\Gamma_{\mathrm{w}}) \mid w = 0 \text{ at } \bar{\Gamma}_{\mathrm{w}} \cap \bar{\Gamma}_1\}.$$

See *e.g.*, Adams [2] and Ciarlet [34] for more details on these Sobolev spaces. Under these conditions, the optimal control problem (P1) admits a unique solution. The presence of the

penalty term $\frac{\delta}{2}\int_{\Gamma_w}(|w|^2+|\nabla_s w|^2)\,d\Gamma$, $\delta > 0$, is crucial in the existence and uniqueness proof, since it implicitly imposes a bound on the controls. While the optimal control problem with the penalized objective function is relatively easy to solve, the penalty term does not allow a direct manipulation of the bound on the control. Therefore, it is often desirable to replace the penalty term in (P1) by a constraint on the controls. The immediate choice $|w(y)| \le \bar{w}$ for all $y \in \Gamma_w$ is not feasible in our framework, since it does not guarantee the boundary value to be in $H^{1/2}(\Gamma_w)$, a condition necessary to ensure solvability of the state equation (3.6)–(3.9) in $H^1(\Omega)$. We do not consider possible relaxations of the meaning of a solution to (3.6)–(3.9). Instead, we impose a bound on the control and its derivative, $|w(x)| \le \bar{w}^0$, $|w'(y)| \le \bar{w}^1$. Together with $w = 0$ at $\bar{\Gamma}_w \cap \bar{\Gamma}_1$ this implies $w \in \mathcal{W}(\Gamma_w)$. This leads to the following optimal control problem:

$$\begin{aligned} &\text{Minimize} \quad J(T, w) \\ &\text{subject to the equations (3.6)–(3.9) and} \\ &\text{subject to} \quad |w(y)| \le \bar{w}^0, \quad \forall y \in \Gamma_w, \\ &\qquad\qquad\quad |w'(y)| \le \bar{w}^1, \quad \forall y \in \Gamma_w. \end{aligned} \tag{P2}$$

In (P2) we assume that $\delta = 0$. The choice $\delta > 0$ would be feasible as well, but would contradict the motivation for this problem.

3.3 Problem Discretization

For the numerical solution of the optimal control problem we apply a finite element discretization. We use a standard finite element discretization with piecewise quadratic functions on triangles. In the following we describe some of the details necessary to present the implementation. More details can be found in any book on finite elements, such as the books by Ciarlet [34] and Zienkiewicz [161].

We subdivide the domain Ω uniformly into triangles as shown in Figure 3.2. In this discretization the width of triangle j is l_{xj} and its height is l_{yj}. The temperature is discretized using piecewise quadratic elements. We use N_e to denote the total number of element/triangles and N_g to denote the total number of nodes. For an $n_x \times n_y$ grid on domain Ω, this gives a total of $N_e = 2(n_x-1)(n_y-1)$ elements, with $N_g = (2n_x-1)(2n_y-1)$ global nodes. The number of nodes on the control boundary Γ_w is denoted by N_w.

The temperature is discretized using

$$T^h(x,y) = \sum_{i=1}^{N_n} \theta_i \tilde{\phi}_i(x,y),$$

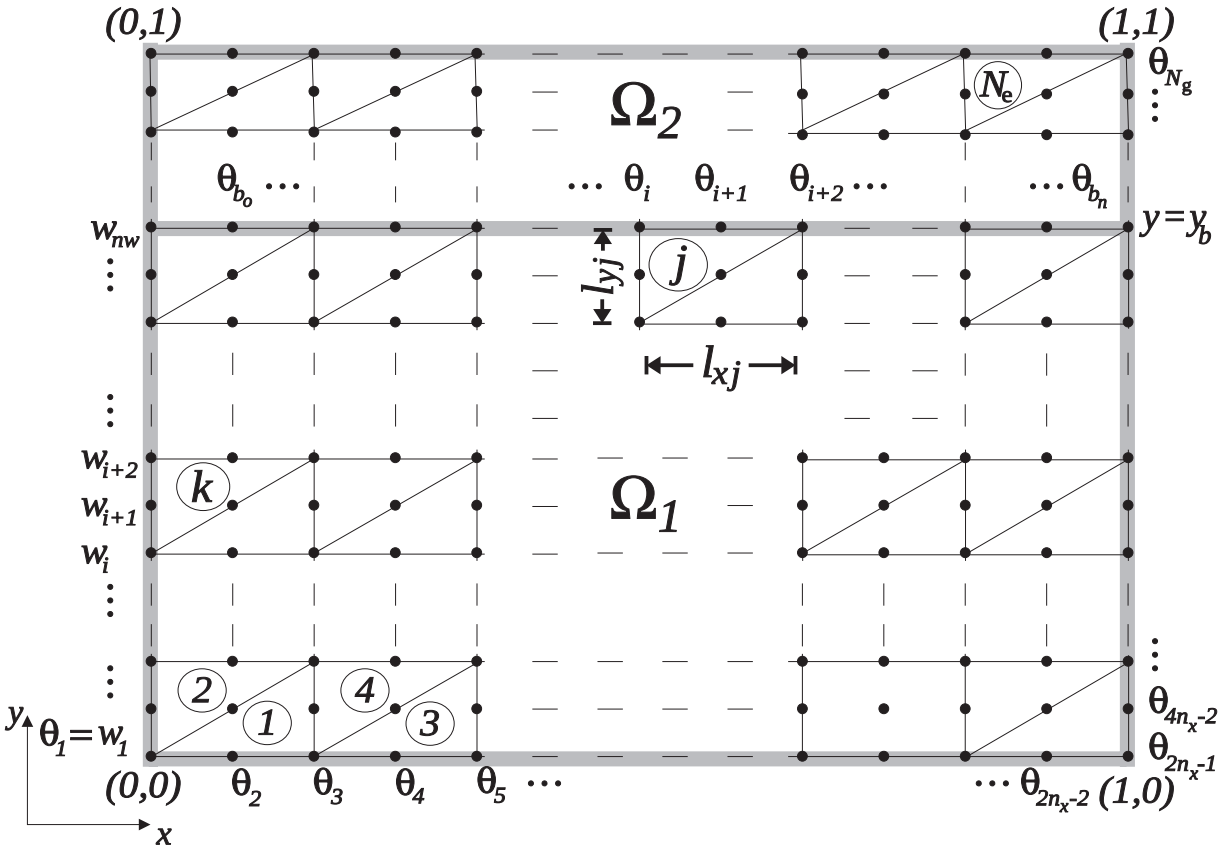


Figure 3.2: The Finite Element Mesh

where $\tilde{\phi}_i(x,y)$ denotes the piecewise quadratic function being one at node i and being zero at all other nodes. To describe some of the details of the implementation, it will be beneficial to use a local representation of the temperature. Let $i_1,\ldots,i_6$ be the numbers of the nodes in triangle i with i_1,i_2,i_3 being the numbers of vertices and i_4,i_5,i_6 being the numbers of the midpoints. Then the function

$$\Phi_i(\xi,\eta) = \begin{pmatrix} x_{i_1}(1-\xi-\eta) + x_{i_2}\xi + x_{i_3}\eta \\ y_{i_1}(1-\xi-\eta) + y_{i_2}\xi + y_{i_3}\eta \end{pmatrix}$$

maps the unit triangle with vertices $(0,0),(0,1),(1,0)$ into the triangle i with vertices (x_{i_1},y_{i_1}), (x_{i_2},y_{i_2}), (x_{i_3},y_{i_3}). The basis functions ϕ_j on the unit triangle and the basis functions $\tilde{\phi}_{i_j}$ on the given triangle i are related as follows:

$$\phi_j(\xi,\eta) = \tilde{\phi}_{i_j}(\Phi_j(\xi,\eta)).$$

With these settings, the local temperature in triangle i is given by

$$T_i^h(\xi,\eta,\vec{\theta}) = \sum_{j=1}^{6} \theta_{i_j}\phi_j(\xi,\eta).$$

Given some flow solution, $\mathbf{u}$ to equations (3.1)–(3.5), the heat equations (3.6) and (3.7) can be discretized over each element to obtain

$$A^i\theta^i = b^i,$$

where $\theta^i = (\theta_{i_1},\ldots,\theta_{i_6})^T$. The discretization of the system equations (3.6)–(3.9) is obtained by assembling the elements along with the boundary conditions (3.8) and (3.9). This gives a set of linear equations

$$\mathbf{A}\vec{\theta} = \mathbf{b}(\vec{w}). \tag{3.12}$$

We use $\vec{\cdot}$ to distinguish the vector of coefficients from the piecewise quadratic functions that the vector represents. The matrix $\mathbf{A}$ is a sparse, in this case banded square matrix. One can easily eliminate the components of q corresponding to nodes on the control boundary Γ_{w}. Since we have Dirichlet controls, these components of θ are equal to the corresponding components of w. This is done in our implementation.

To discretize the objective function J, we introduce the set of elements E_{r} adjacent to the fluid-solid interface Γ_{r} and the set of elements E_{w} adjacent to the control boundary, *i.e.*, the inlet to the duct Γ_{w}. Using the local representation of the temperature and the structure of our finite element grid, the discretized objective function is given by

$$J^h(\vec{\theta},\vec{w}) = \frac{1}{2}\sum_{i\in E_{\rm r}} l_{xi}\int_0^1 \left(T_i^h(\xi,\eta_b;\vec{\theta}) - \hat{T}_i(\xi)\right)^2 d\xi + \frac{\delta}{2}\sum_{i\in E_{\rm w}} l_{yi}\int_0^1 \mid T_i^h(\xi_0,\eta;\vec{\theta}) \mid^2 d\eta$$
$$+\frac{\delta}{2}\sum_{i\in E_{\rm w}} \frac{1}{l_{yi}}\int_0^1 \mid \frac{\partial T_i^h(\xi_0,\eta;\vec{\theta})}{\partial\eta} \mid^2 d\eta. \qquad (3.13)$$

In (3.13) we use ξ_0 to denote the local coordinate for the ith element at $x = x_0$, where x_0 represents the x-coordinate at the inlet to the duct, and η_b to denote the local coordinate for the ith element at $y = y_b$, where y_b is the y-coordinate at the fluid-solid interface. Note, that $\hat{T}_i(\xi)$ is the localized representation of the desired temperature distribution, $\hat{T}(\cdot)$, on the fluid-solid interface, $\Gamma_{\rm r}$, for the ith element.

Consider the first term in the objective function. For any element within $E_{\rm r}$, the contribution to the objective function can be seen to be

$$J_{1i}^h = \frac{l_{xi}}{2}\sum_{j=1}^{6}\int_0^1 \left(\theta_{i_j}\phi_j(\xi,\eta_b) - \hat{T}_i(\xi)\right)^2 d\xi \qquad (3.14)$$

This simplifies to,

$$J_{1i}^h = \frac{l_{xi}}{2}\sum_{j=1}^{6}\left(C_{2j}\theta_{i_j}^2 + C_{1j}\theta_{i_j} + C_{0j}\right)$$

where, C_{0j}, C_{1j} and C_{2j} are constants, given by

$$\begin{aligned} C_{2j} &= \int_0^1 \phi_j(\xi,\eta_b)^2 d\xi, \\ C_{1j} &= -2\int_0^1 \hat{T}_i(\xi)\phi_j(\xi,\eta_b)^2 d\xi, \quad \text{and} \\ C_{0j} &= \int_0^1 \hat{T}_i(\xi)^2 d\xi, \end{aligned}$$

respectively. The above term (3.14) is a quadratic expression involving θ^i. We can show, similarly, that the contributions from the elements within $E_{\rm w}$, for the penalty terms in the objective function, are also quadratic expressions. We have linear constraints comprising the discretized governing equations, and bounds on the variables, if any. Thus, the resulting problem can be seen to be a quadratic programming problem.

Now we are able to formulate the three discrete optimal control problems that we study in this chapter. The first discrete optimal control problem corresponds to (P1) and is given by

$$\begin{aligned} \text{Minimize} \quad & J^h(\vec{\theta}, \vec{w}) \\ \text{s.t.} \quad & \mathbf{A}\vec{\theta} = \mathbf{b}(\vec{w}), \end{aligned} \tag{DP1}$$

where $\delta > 0$. Under the condition (3.11) on $\mathbf{u}$, the matrix $\mathbf{A}$ can shown to be positive definite and it is not hard to show that the quadratic programming problem (DP1) has a unique solution. The fact that the penalty parameter δ is positive is also important in the discrete case. Gunzburger and Lee [63] have studied the convergence of the solution to the discretized problems (DP1) to the solution to (P1) as $h \to 0$.

In the discrete framework it is tempting to replace the penalty term by bound constraints. This leads to the problem

$$\begin{aligned} \text{Minimize} \quad & J^h(\vec{\theta}, \vec{w}) \\ \text{s.t.} \quad & \mathbf{A}\vec{\theta} = \mathbf{b}(\vec{w}) \\ & |\vec{w}| \le \bar{w}^0, \end{aligned} \tag{DP2}$$

where we assume that $\delta = 0$. Here $|\vec{w}| \le \bar{w}^0$ is understood component wise. Using standard arguments, it can be seen that the problem (DP2) has a solution. Notice, that, as discussed previously, the infinite dimensional version of the discrete problem (DP2) may not have a solution. To understand (DP2), we use inverse inequalities. It is well known, see *e.g.*, Ciarlet [34], that for quasi-uniform finite element discretizations there exists a constant c independent of h such that

$$\max_{\Gamma_w} |w_h'(y)| \le \frac{c}{h} \max_{\Gamma_w} |w_h(y)|. \tag{3.15}$$

Here w_h denotes the piecewise quadratic function corresponding to the vector of coefficients $\vec{w}$. Due to the inverse inequality (3.15) the bound $|\vec{w}| \le \bar{w}^0$ on the coefficients implies bounds on $\max_{\Gamma_w} |w_h(y)|$ and on $\max_{\Gamma_w} |w_h'(y)|$. However, these bounds depend on h and the second bound will grow towards infinity as h goes to zero. In fact, if i_1, i_2, i_3 are the indices of the nodes on the edge i, then the local control is given by

$$\begin{aligned} w_{hi}(y) = w_{i_1} \frac{(y - y_{i_2})(y - y_{i_3})}{(y_{i_1} - y_{i_2})(y_{i_1} - y_{i_3})} + w_{i_2} \frac{(y - y_{i_1})(y - y_{i_3})}{(y_{i_2} - y_{i_1})(y_{i_2} - y_{i_3})} \\ + w_{i_3} \frac{(y - y_{i_1})(y - y_{i_2})}{(y_{i_3} - y_{i_1})(y_{i_3} - y_{i_2})}. \end{aligned} \tag{3.16}$$

Thus,

$$|w_h(y)| \le c_0 \|\vec{w}\|_\infty \quad \forall y \in \Gamma_w$$

for some c_0 independent of h. By the inverse inequality,

$$|w_h'(y)| \leq \frac{c_1}{h}\|\vec{w}\|_\infty \quad \forall y \in \Gamma_{\rm w}$$

for some c_1 independent of h. Hence, for coarse discretizations, *i.e.*, large h, the bound $|\vec{w}| \leq \bar{w}^0$ should be sufficient, but as the mesh is refined, *i.e.*, h is decreased, the bound $|\vec{w}| \leq \bar{w}^0$ must be tightened to guarantee a reasonable bound on $|w_h'(y)|$. This behavior is typical if there is no well-posed infinite dimensional problem formulation corresponding to the discrete problem formulations. In this case, even though the discrete problem formulation may be well posed for arbitrary fine but fixed discretization levels, there is no convergence of the solutions for the discrete problems as h tends to zero. This will be demonstrated by our numerical results. Moreover, it is important to notice, that this effect is not only of theoretical interest, but as the discretizations are refined, optimization methods applied to solve (DP2) will perform extremely poorly. We will return to this issue later.

In view of the preceding discussion, we want to add a constraint to (DP2) which bounds the derivative of w_h. Using the local control (3.16) on the edge i, we find that

$$\begin{aligned} w_{hi}'(y) = w_{i_1}\frac{2y - y_{i_2} - y_{i_3}}{l_{yi}/2} + w_{i_2}\frac{2y - y_{i_1} - y_{i_3}}{-l_{yi}/4} \\ + w_{i_3}\frac{2y - y_{i_1} - y_{i_2}}{l_{yi}/2}. \end{aligned}$$

Here we assume that y_{i_1}, y_{i_3} are the endpoints of the edge and y_{i_2} is the midpoint and we used the fact that $|y_{i_1} - y_{i_3}| = l_{yi}$. Since $w_{hi}'(y)$ is linear, $|w_{hi}'(y)| \leq \bar{w}^1$ if and only if $|w_{hi}'(y_{i_1})| \leq \bar{w}^1$ and $|w_{hi}'(y_{i_3})| \leq \bar{w}^1$. This leads to

$$\begin{aligned} |w_{i_1}\frac{2y_{i_1} - y_{i_2} - y_{i_3}}{l_{yi}/2} + w_{i_2}\frac{y_{i_1} - y_{i_3}}{-l_{yi}/4} + w_{i_3}\frac{2y_{i_1} - y_{i_1} - y_{i_2}}{l_{yi}/2}| &\leq \bar{w}^1, \\ |w_{i_1}\frac{2y_{i_3} - y_{i_2} - y_{i_3}}{l_{yi}/2} + w_{i_2}\frac{y_{i_3} - y_{i_1}}{-l_{yi}/4} + w_{i_3}\frac{2y_{i_3} - y_{i_1} - y_{i_2}}{l_{yi}/2}| &\leq \bar{w}^1. \end{aligned}$$

Using the fact that $|y_{i_1} - y_{i_3}| = l_{yi}$ we arrive at the inequalities

$$\begin{aligned} \frac{1}{l_{yi}}|-3w_{i_1} + 4w_{i_2} - w_{i_3}| &\leq \bar{w}^1, \\ \frac{1}{l_{yi}}|-w_{i_1} + 4w_{i_2} - w_{i_3}| &\leq \bar{w}^1. \end{aligned}$$

If we impose this for all edges on the control boundary $\Gamma_{\rm w}$, then we arrive at the inequalities

$$|\mathbf{B}\vec{w}| \leq \bar{w}^1,$$

where $\mathbf{B} \in I\!R^{2N_w \times N_w}$ is composed of two square tridiagonal matrices. This leads to the following problem:

$$\begin{array}{ll} \text{Minimize} & J^h(\vec{\theta}, \vec{w}) \\ \text{s.t.} & \mathbf{A}\vec{\theta} = \mathbf{b}(\vec{w}) \\ & |\vec{w}| \leq \bar{w}^0 \\ & |\mathbf{B}\vec{w}| \leq \bar{w}^1. \end{array} \tag{DP3}$$

Again, (DP3) is well posed. Moreover, it corresponds to the infinite dimensional problem (P2).

3.4 Numerical Results

For the solution of the optimal control problems (DP1), (DP2), (DP3) any method for the solution of quadratic programming problems can used. We applied a sparse optimization code developed by Betts [14, 15, 77]. Betts' code is a sequential quadratic programming (SQP) based method for the solution of nonlinear problems. Therefore it is designed to solve much more general problems and most of its features are not needed when it is applied to solve (DP1), (DP2), or (DP3). However, since this research was performed in a larger context, including nonlinear phenomena and since the speed of convergence of the optimizer is not the focus of this study, we used this package. Gradient and Hessian information are computed analytically, and stored in sparse form, as described in [14].

As a test problem we have chosen one of the problems considered by Gunzburger and Lee [63]. The domain Ω is the unit square $(0,1) \times (0,1) \subset I\!R^2$ with sub-domains $\Omega_1 = (0,1) \times (0.75,1)$ and $\Omega_2 = (0,1) \times (0,0.75)$. Thus, the fluid–solid interface is the line $y_b = 0.75$. As the line on which the temperature is to be matched we choose $\Gamma_r = (0.075,1) \times \{0.75\} \subset \Gamma_b = (0,1) \times \{0.75\}$. The control boundary is $\Gamma_w = \{0\} \times (0,0.75)$. (See Figure 3.1 without the bump on the bottom boundary).

We consider the following problem

$$-\Delta T = 6.0 \quad \text{on} \quad \Omega_1, \tag{3.17}$$

$$-2\Delta T + (\mathbf{u} \cdot \nabla)T = 0 \quad \text{on} \quad \Omega_2, \tag{3.18}$$

$$T = 1 + \tilde{w} \quad \text{on} \quad \Gamma_w, \tag{3.19}$$

$$\frac{\partial T}{\partial n} = 0 \quad \text{on} \quad \partial\Omega \backslash \Gamma_w, \tag{3.20}$$

where the velocity $\mathbf{u}$ is the solution of the Navier-Stokes equations

$$-\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{0} \quad \text{in } \Omega_2, \tag{3.21}$$

the incompressibility constraint

$$\operatorname{div}\mathbf{u} = 0 \quad \text{in } \Omega_2, \tag{3.22}$$

and the boundary condition

$$\mathbf{u} = \mathbf{h} \qquad \text{on} \quad \Gamma_{\mathrm{w}}, \tag{3.23}$$

$$\mathbf{u} = \mathbf{0} \qquad \text{on} \quad \Gamma_{\mathrm{b}} \cup \Gamma_4, \tag{3.24}$$

$$\frac{\partial u_1}{\partial n} = 0 \quad \text{and} \quad u_2 = 0 \qquad \text{on } \Gamma_{\mathrm{o}}, \tag{3.25}$$

where $\mathbf{u} = (u_1, u_2)$ and $\mathbf{h} = (1.5y - 2y^2, 0)$. We have a simple solution, $\mathbf{u} = (1.5y - 2y^2, 0)$, for the above Navier-Stokes problem.

In keeping with the results that Lee and Gunzburger obtained, we also impose an additional condition, $\tilde{w}(y_b) = 0$, which in terms of the finite element problem corresponds to the condition, $w_{N_{\mathrm{w}}} = 1$. All cases were started from an initial point, corresponding to the solution to equations (3.17)–(3.20), with $\tilde{w} = 0$ in equation (3.19). This problem is referred to as the uncontrolled problem, and its numerical solution is shown in Figure 3.3 and Figure 3.4. The solution to the uncontrolled problem is used as an initial guess for the optimizer.

It can be seen from the above mentioned figures that the temperature is above 2.0 on $(0.3, 1) \times \{0.75\}$ and even higher in the domain $(0.3, 1) \times (0.75, 1)$. We try to regulate the temperature along Γ_{r}. Any reasonable target temperature distribution may be chosen. Following Gunzburger and Lee, we have chosen to target $\hat{T} = 1.2$ on Γ_{r}. Thus, we have

$$J(T, w) = \frac{1}{2}\int_{\Gamma_{\mathrm{r}}} |T - 1.2|^2 \, d\Gamma + \frac{1}{\delta}\int_{\Gamma_{\mathrm{w}}} (|w-1|^2 + |\nabla_s w|^2) \, d\Gamma. \tag{3.26}$$

Note that we penalize $\tilde{w}$. Therefore, in the control $w = 1 + \tilde{w}$ the penalty term is given by $\int_{\Gamma_{\mathrm{w}}} (|w-1|^2 + |\nabla_s w|^2) \, d\Gamma$.

First, we duplicate some of the results computed by Gunzburger and Lee [63], *i.e.*, we solve (DP1) with data and the modifications for w outlined above. We use the penalty parameters $\delta = 1$ and $\delta = 6 \times 10^{-5}$. The costs computed for a 13×13 grid are indicated in Table 3.1.

As can be expected, for $\delta = 2$, where we place a high weight on the size of the control, the optimal solution does not exhibit a significantly large control effort. The numerical results

Table 3.1: Objective Function Values (13×13 Grid)

	$\|\|T^h - 1.2\|\|^2_{0,\Gamma_r}$	$\|\|w^h\|\|^2_{1,\Gamma_w}$	$\mathcal{J}^h(T,w)$
uncontrolled	1.1548	0	0.5774
$\delta = 2$	1.0751	0.0196	0.5571
$\delta = 6 \times 10^{-5}$	0.00155	66.90	0.0028

for this case are shown in Figures 3.5 and 3.6 through a temperature surface plot, and a contour plot, respectively.

When the relative weight on the penalty term is reduced, $\delta = 6 \times 10^{-5}$, we find that the control exhibits much more dynamical behavior. Results for this case are shown in Figures 3.7 and 3.8 through temperature surface and contour plots, respectively.

The temperature distributions at the inlet to the duct, obtained in the two cases, are plotted in Figure 3.9. A comparison of the temperature distributions generated at the solid-fluid interface for the two cases is shown in Figure 3.10. As would be expected, the optimal control does a much better job at tracking the desired temperature profile for the second case.

We remark that the results we have obtained do not agree with those reported by Lee and Gunzburger. In fact, there seem to be some inconsistencies in their results. Personal communication with the authors indicated that they have used some scaling to accelerate convergence. This leads us to believe that they have reported inconsistent values of δ for the cases they have shown.

The costs for various grid sizes with $\delta = 6 \times 10^{-5}$ are shown in Table 3.3. Figures 3.11 and 3.12 show comparisons of the results obtained for each case, in terms of the optimal control distribution obtained and the temperature distribution at the interface generated, respectively.

The sparse optimization code does a good job of solving the problems. As one might expect, each problem is solved in a single QP iteration. A typical output is shown (for $\delta = 6 \times 10^{-5}$) in Table 3.4. Here `It` is the major iteration number (number of SQP iterations), `Qit` is the number of QP iterations within one major iteration, `Nkt` is the number of numeric factorizations of the symmetric indefinite Kuhn-Tucker system, `Ndof` is the number of degrees of freedom for the problem, `KT Cond` is the condition number of the symmetric indefinite Kuhn-Tucker system, `Step` is the length of the step taken in the current search direction, `Norm p` is the norm of the search direction vector, `Violtn` is a measure of the constraint violation, `Levnbrg` is the value of the Levenberg parameter, `Penalty` is the norm of the penalty vector used in the merit function, `Norm pg` is the norm of the projected gradient, and `Merit Function` is the value of the merit function. [14] The reader is referred to [52, 53, 60] for the meanings of some of the above terms.

Next we consider the optimal control problem (DP2) with explicit bounds on the control variables and with $\delta = 0$. As we have shown earlier, the bounds on the coefficients of the discretized control impose a crude bound on ∇w^h which, however, varies as $1/h$. The lack of penalization of the control is apparent in the computed controls, and also in the

Table 3.2: Grid Sizes Used to Solve Temperature Control Problem

grid size	N_g	N_w
7×7	169	8
13×13	625	18
25×25	2401	36
33×33	4225	48

Table 3.3: Grid Convergence Study for Problem (DP1) with $\delta = 6 \times 10^{-5}$

grid size	$\|\|T - 1.2\|\|^2_{0,\Gamma_r}$	$\|\|w - 1\|\|^2_{1,\Gamma_w}$	$\mathcal{J}^h(T, w)$
7×7	1.47×10^{-3}	67.66	2.76×10^{-1}
13×13	1.55×10^{-3}	66.90	2.78×10^{-1}
25×25	1.72×10^{-3}	74.47	3.10×10^{-1}
33×33	1.60×10^{-3}	69.28	2.88×10^{-1}

Table 3.4: A Typical Output from SNLP ($\delta = 6 \times 10^{-5}$)

```
                                    Optimization

It Qit Nkt Ndof  KT Cond    Step    Norm p   Violtn  Levnbrg  Penalty  Norm Pg  Merit Function
(1)  1   1   18  5.2E+03  1.0E+00  4.5E+01  2.0E-14  0.0E+00  1.5E-08  1.4E+00    5.774109E+01
(2)  0   0   18  5.2E+03  1.0E+00  4.5E+01  1.9E-12  0.0E+00  1.5E-08  1.9E-11    2.780404E-01
  ------------------------------------------------------------------------------------------
  |                                                                                        |
  |                   SPRNLP-2 ALGORITHM PERFORMANCE STATISTICS                            |
  |                                                                                        |
  |    TOTAL CPU TIME.....................................................+2.29694      |
  |                                                                                        |
  |    NUMBER OF FUNCTION CALLS.....................................................2     |
  |    NUMBER OF GRADIENT CALLS.....................................................3     |
  |    NUMBER OF HESSIAN CALLS......................................................1     |
  |    TOTAL NUMBER OF FUNCTION EVALUATIONS.........................................2     |
  |                                                                                        |
  |    EXCESS STORAGE IN HOLD ARRAY.....................................2503072          |
  |                                                                                        |
  ------------------------------------------------------------------------------------------
SOLUTION ...
```

performance of the optimizer. Unless a tight bound on $\vec{w}$ is enforced, the optimizer performs rather poorly. This can be attributed to a lack of regularity of the solution. While the penalized problem (DP1) has a unique solution which converges to the solution of the corresponding infinite dimensional problem, such a convergence property does in general not hold for the solutions of (DP2). Compare Figures 3.11 and 3.17. For the penalized problem (DP1) the properties of the infinite dimensional problem, in particular the strict convexity of the infinite dimensional problem, determine the properties of the discretized problem (DP1). Since there is no well-posed infinite dimensional problem corresponding to (DP2), such a behavior can not be expected in this case. In fact, our results indicate the strict convexity of the problem (DP2) is related to the grid size h. The larger h, the more strictly convex the problem is, *i.e.*, the larger the eigenvalues of the Hessian projected onto the null-space of the active constraints.

The optimal control obtained for this case is shown in Figure 3.13. The '$*$'s and 'o's indicate the positions of the nodes (and their temperature values) for the problem, using a 13×13 grid. The results show that most of the control nodes are at either of the two bounds. The algorithm seems to have trouble in determining the correct active set in this case. The problem worsens as the bounds are pushed out. The controls obtained for the cases when $|\vec{w}| \leq 4$ and $|\vec{w}| \leq 8$ are shown in Figures 3.14 and 3.15, respectively. These figures also confirm our previous analysis. If the bound on $|\vec{w}|$ becomes too large relative to h, then the derivative of w^h is essentially unbounded. This can be seen in Figures 3.13 to 3.15. As the bound on $|\vec{w}|$ is relaxed, the computed solution w^h tends to oscillate and the oscillations increase as the bound is further relaxed. It should also be noted that tight bounds on $|\vec{w}|$ seem to be necessary to guarantee that the function w^h roughly lies between the same bounds as the vector of coefficients $\vec{w}$. We also point out that the discretization of size 13×13 is not particularly fine. Therefore, the poor behavior at this rather coarse discretization level is somewhat surprising.

A comparison of the temperature distributions generated for the three cases, shown in Figure 3.16, indicates that the optimal solution does a better job at regulating the temperature as the bounds on the control are relaxed.

The poor performance of the optimization algorithm can be clearly seen in Table 3.5, which shows the output from SNLP for the case, $|\vec{w}| \leq 8$. We do not have quadratic convergence behavior. The oscillatory behavior of the solution is apparent from the fact that the correct number of degrees of freedom `Ndof` is not obtained until the final iterations. It should be noted that this problem is a QP problem only when the correct active set is used. The poor behavior of the algorithm can be attributed to the fact that it struggles to find the correct active set of constraints due to the oscillatory nature of the solution. This can also be seen from the fact that the Levenberg parameter `Levnbrg` is nonzero till the correct active set is found in iteration (11), following which the algorithm does converge in one QP iteration.

Finally, we consider the optimal control problem (DP3) with explicit bounds on the control variables and with $\delta = 0$. As expected, the numerical results are closer to those for the penalized formulation (DP1). Figures 3.18 to 3.20 show the computed controls for the constraints $|\vec{w}| \leq 4$, and $|\mathbf{B} \cdot \vec{w}| \leq 2, 20, 2000$, respectively. For reasonable bounds on the derivative ($|\mathbf{B} \cdot \vec{w}| \leq 2, 20$), the computed results are close to those for the penalized problem with appropriately chose penalty parameter δ. This shows very clearly in the match between computed and desired temperature (see Figures 3.10 and 3.21). However, similarities can also be observed in the computed controls. Compare Figures 3.18 and 3.23 to Figures 3.19 and 3.24. Of course, if the bound on the derivative is too large, then the results compare with those for problem (DP2). See *e.g.*, Figures 3.14 and 3.20. Generally, we can see a qualitative improvement, *i.e.*, less oscillatory behavior in the solutions of (DP3) compared with (DP2). Along with this we could also observed a much improved convergence behavior of the optimizer, due to the regularizing effect of the bound constraints, similar to the addition of a penalty term. As for the penalized problem (DP1), convergence of the computed controls as the grid is refined can be observed. See Figure 3.22.

Table 3.6 shows the output from SNLP for the case, $|\mathbf{B} \cdot \vec{w}| \leq 20$. It can be seen that though the algorithm does not achieve quadratic performance, the behavior is still markedly improved from the previous case.

3.5 Remarks

We have considered a thermal-fluid control problem wherein the physics are described by a system of partial differential equations and control is implemented through a thermal boundary condition. A finite-element approximation was used to transcribe this to a finite-dimensional quadratic programming problem. Three versions of the discrete optimal control problem are considered. These differ in their treatment of certain control bounds. Two of these formulations correspond to infinite-dimensional optimal control problems. The numerical studies in this chapter show that variants which faithfully reflect the structure of control bounds in the infinite-dimensional problem lead to well-behaved QP solutions, while variants that do not are troublesome for the QP algorithm. In the first case, the optimization algorithm behaves well and one can observe convergence of the discrete solutions as the grid is refined. However, in the second case, when there is no corresponding infinite dimensional well-posed problem, the convergence behavior of the optimization algorithm deteriorates and the computed discretized solutions tend to oscillate as the discretization is refined. We have provided some explanations for this behavior. Other optimization algorithms, such as interior point methods, *e.g.*, Wright [157] for the linear case, and SQP interior point methods, *e.g.*, Dennis *et al.* [40] for the nonlinear case, would also provide efficient methodologies for solution of this problem.

Table 3.5: Output from SNLP for the case: $|\vec{w}| \leq 8$

Optimization

It	Qit	Nkt	Ndof	KT Cond	Step	Norm p	Violtn	Levnbrg	Penalty	Norm Pg	Merit Function
(1)	25	6	6	1.3E+08	1.0E+00	6.3E+01	2.0E-14	1.5E-07	1.5E-08	1.4E+00	5.773721E+01
(2)	15	3	6	1.3E+08	1.0E+00	4.5E-01	1.4E-09	1.5E-07	1.5E-08	1.5E-06	1.794529E-03
(3)	15	3	6	1.3E+08	1.0E+00	4.5E-01	5.5E-09	1.5E-07	1.5E-08	5.4E-08	1.794471E-03
(4)	3	1	6	2.6E+08	1.0E+00	9.0E-01	4.8E-09	7.5E-08	1.5E-08	5.5E-08	1.794438E-03
(5)	1	1	6	5.3E+08	1.0E+00	1.8E+00	3.2E-14	3.7E-08	1.5E-08	5.5E-08	1.794372E-03
(6)	1	1	6	1.1E+09	1.0E+00	3.6E+00	4.6E-14	1.9E-08	1.5E-08	5.5E-08	1.794239E-03
(7)	13	3	6	1.3E+08	1.0E+00	4.5E-01	4.4E-14	1.5E-07	1.5E-08	5.5E-08	1.793974E-03
(8)	3	1	6	2.6E+08	1.0E+00	9.0E-01	2.8E-09	7.5E-08	1.5E-08	5.5E-08	1.793941E-03
(9)	2	1	5	5.3E+08	1.0E+00	1.2E+00	1.1E-14	3.7E-08	1.5E-08	5.5E-08	1.793875E-03
(10)	2	1	4	1.1E+09	1.0E+00	1.5E+00	3.2E-14	1.9E-08	1.5E-08	4.1E-08	1.793794E-03
(11)	1	1	4	7.6E+05	1.0E+00	3.0E-05	8.0E-14	0.0E+00	1.5E-08	2.8E-08	1.793730E-03
(12)	0	0	4	7.6E+05	1.0E+00	3.0E-05	7.8E-14	0.0E+00	1.5E-08	7.0E-15	1.793730E-03

SPRNLP-2 ALGORITHM PERFORMANCE STATISTICS

TOTAL CPU TIME.....+33.0683

NUMBER OF FUNCTION CALLS.....12
NUMBER OF GRADIENT CALLS.....13
NUMBER OF HESSIAN CALLS.....11
TOTAL NUMBER OF FUNCTION EVALUATIONS.....12

EXCESS STORAGE IN HOLD ARRAY.....590626

SOLUTION...

Table 3.6: Output from SNLP for the case: $|\mathbf{B} \cdot \vec{w}| \leq 20$

```
                                    Optimization

It Qit Nkt Ndof  KT Cond    Step    Norm p   Violtn  Levnbrg  Penalty  Norm Pg  Merit Function
(1) 45   3    2  1.1E+04  1.0E+00  4.8E+01  2.0E-14  1.5E-07  1.5E-08  1.4E-02    5.762901E-01
(2)  3   1    2  3.4E+05  1.0E+00  2.8E-02  2.4E-10  0.0E+00  1.5E-08  7.5E-07    2.836980E-04
(3)  0   0    2  3.4E+05  1.0E+00  2.8E-02  6.4E-15  0.0E+00  1.5E-08  6.9E-16    2.836298E-04
  ------------------------------------------------------------------------------------------
  |                                                                                        |
  |                   SPRNLP-2 ALGORITHM PERFORMANCE STATISTICS                            |
  |                                                                                        |
  |    TOTAL CPU TIME.....................................................+9.28956       |
  |                                                                                        |
  |    NUMBER OF FUNCTION CALLS.....................................................3     |
  |    NUMBER OF GRADIENT CALLS.....................................................4     |
  |    NUMBER OF HESSIAN CALLS......................................................2     |
  |    TOTAL NUMBER OF FUNCTION EVALUATIONS.........................................3     |
  |                                                                                        |
  |    EXCESS STORAGE IN HOLD ARRAY.....................................2496316           |
  |                                                                                        |
  ------------------------------------------------------------------------------------------
SOLUTION ...
```

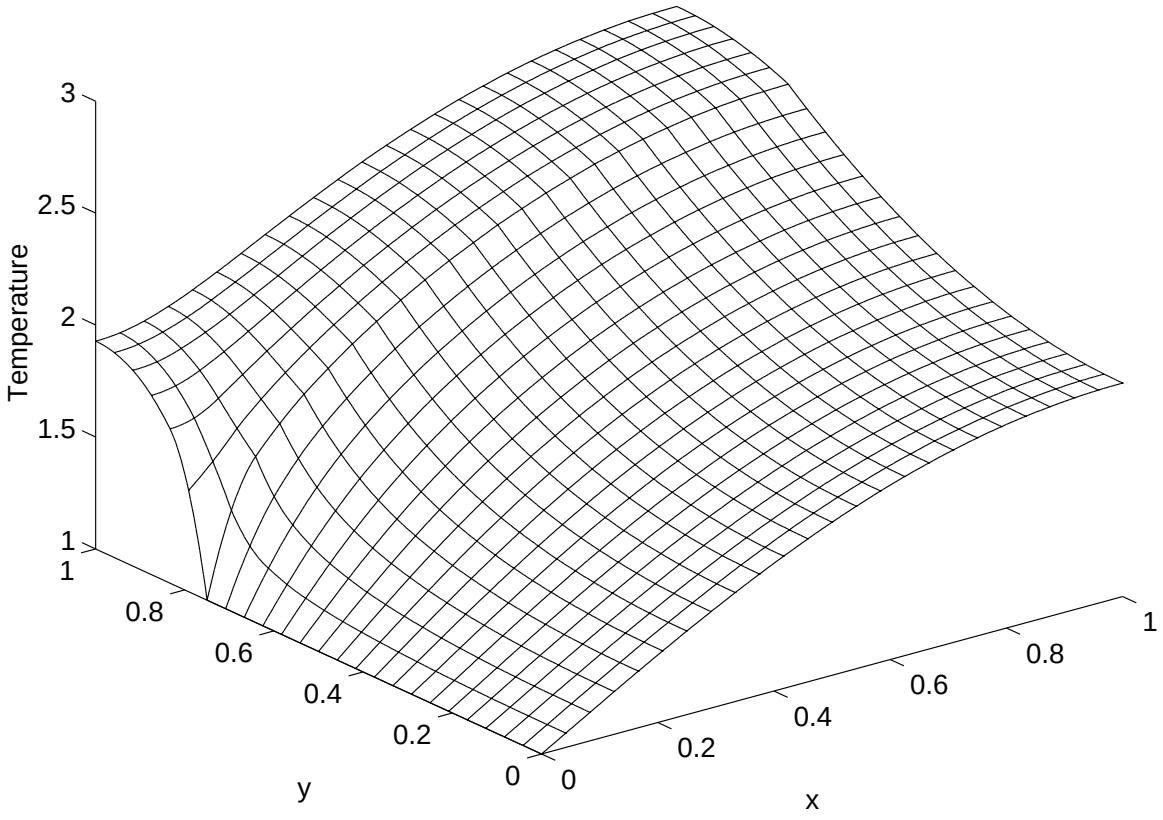


Figure 3.3: The temperature surface plot for the uncontrolled problem

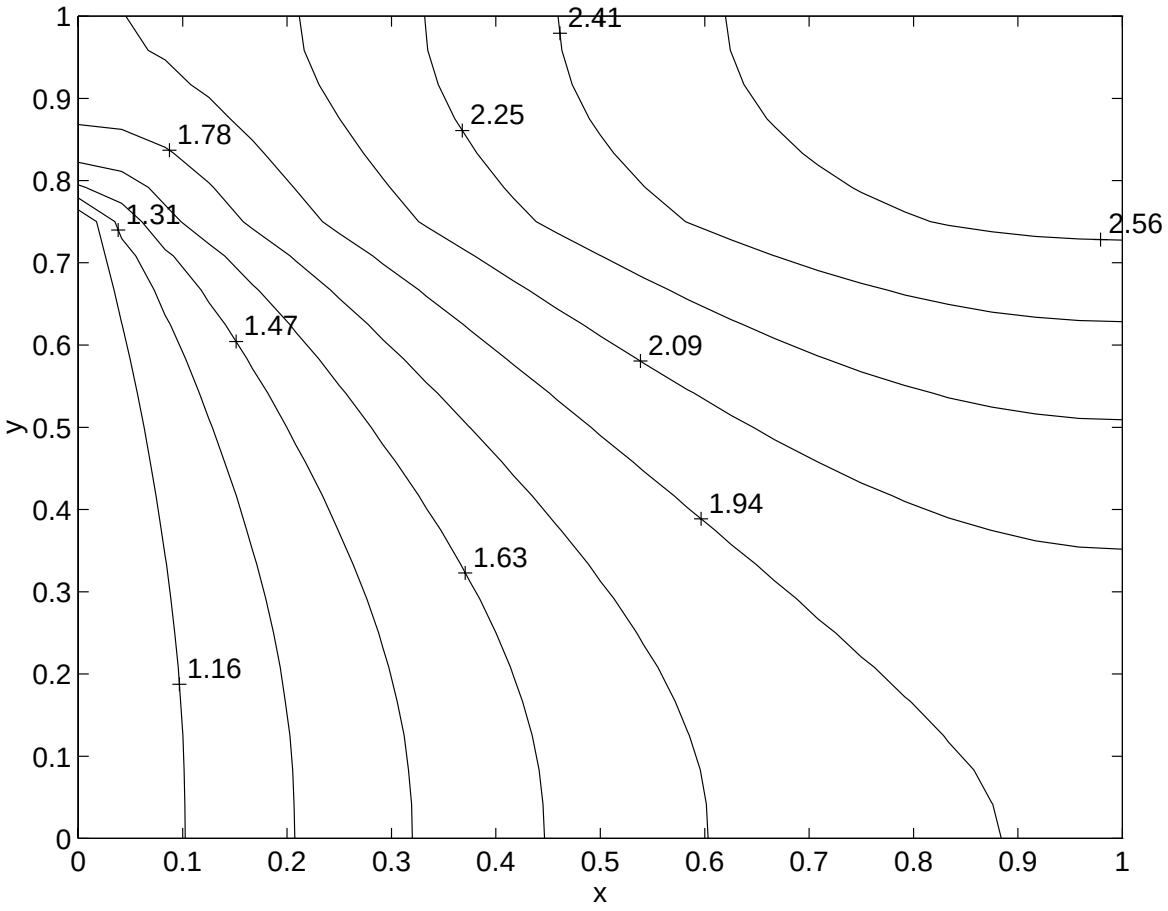


Figure 3.4: The temperature contour plot for the uncontrolled problem

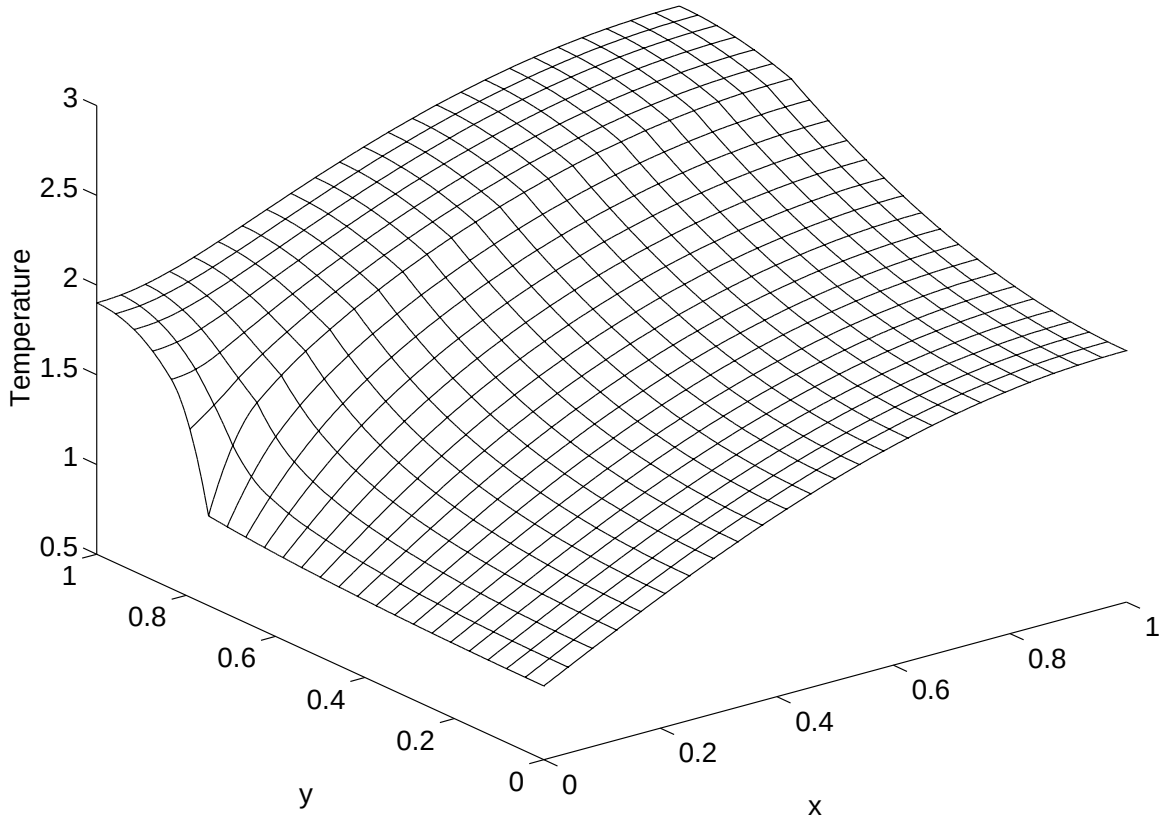


Figure 3.5: The temperature surface plot for Problem (DP1) with $\delta = 2$

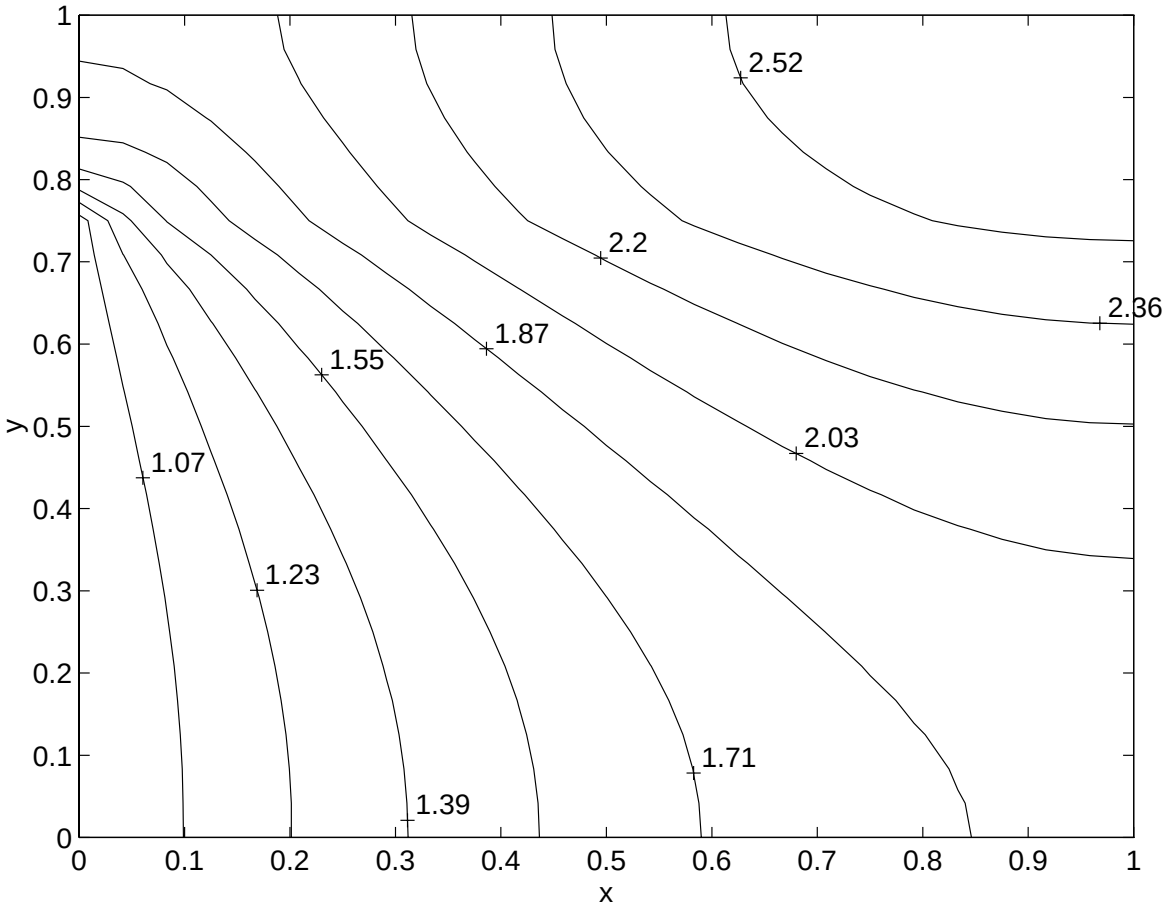


Figure 3.6: The temperature contour plot for Problem (DP1) with $\delta = 2$

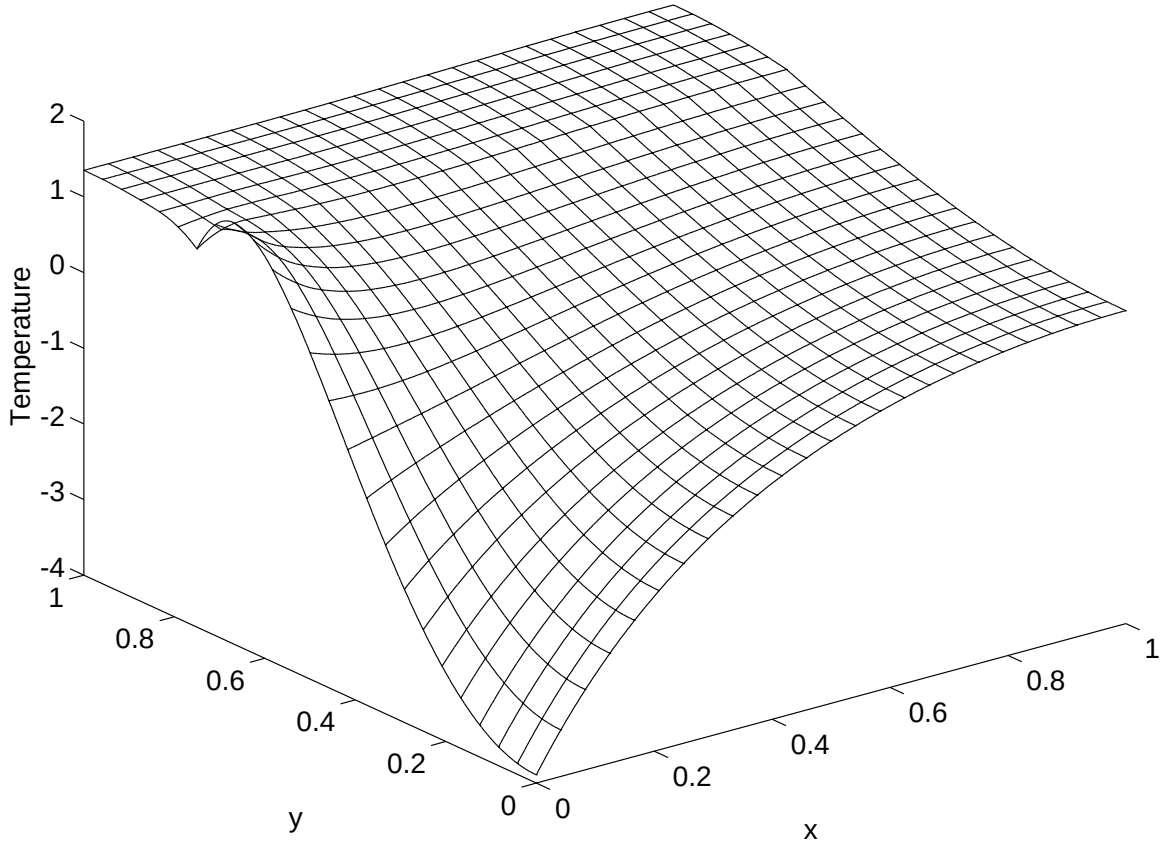


Figure 3.7: The temperature surface plot for Problem (DP1) with $\delta = 6 \times 10^{-5}$

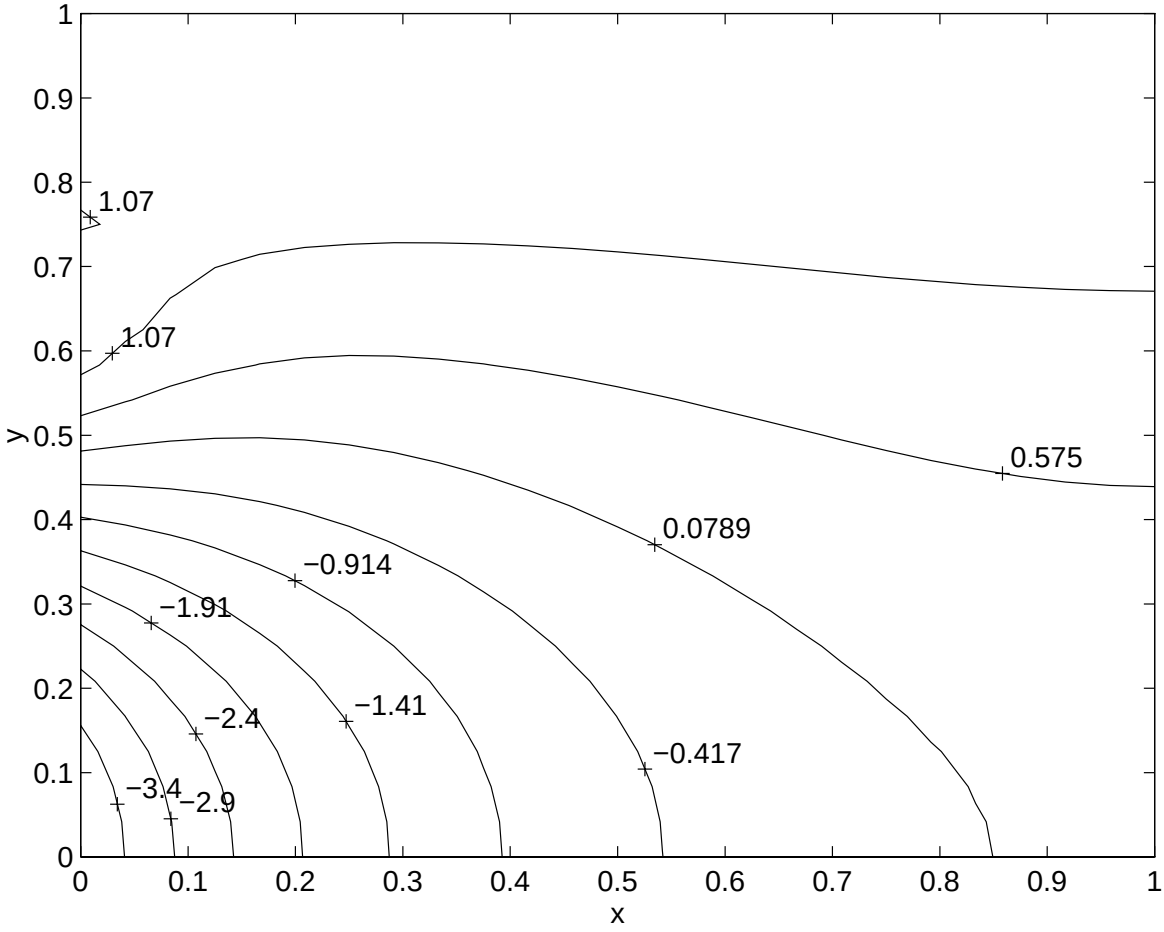


Figure 3.8: The temperature contour plot for Problem (DP1) with $\delta = 6 \times 10^{-5}$

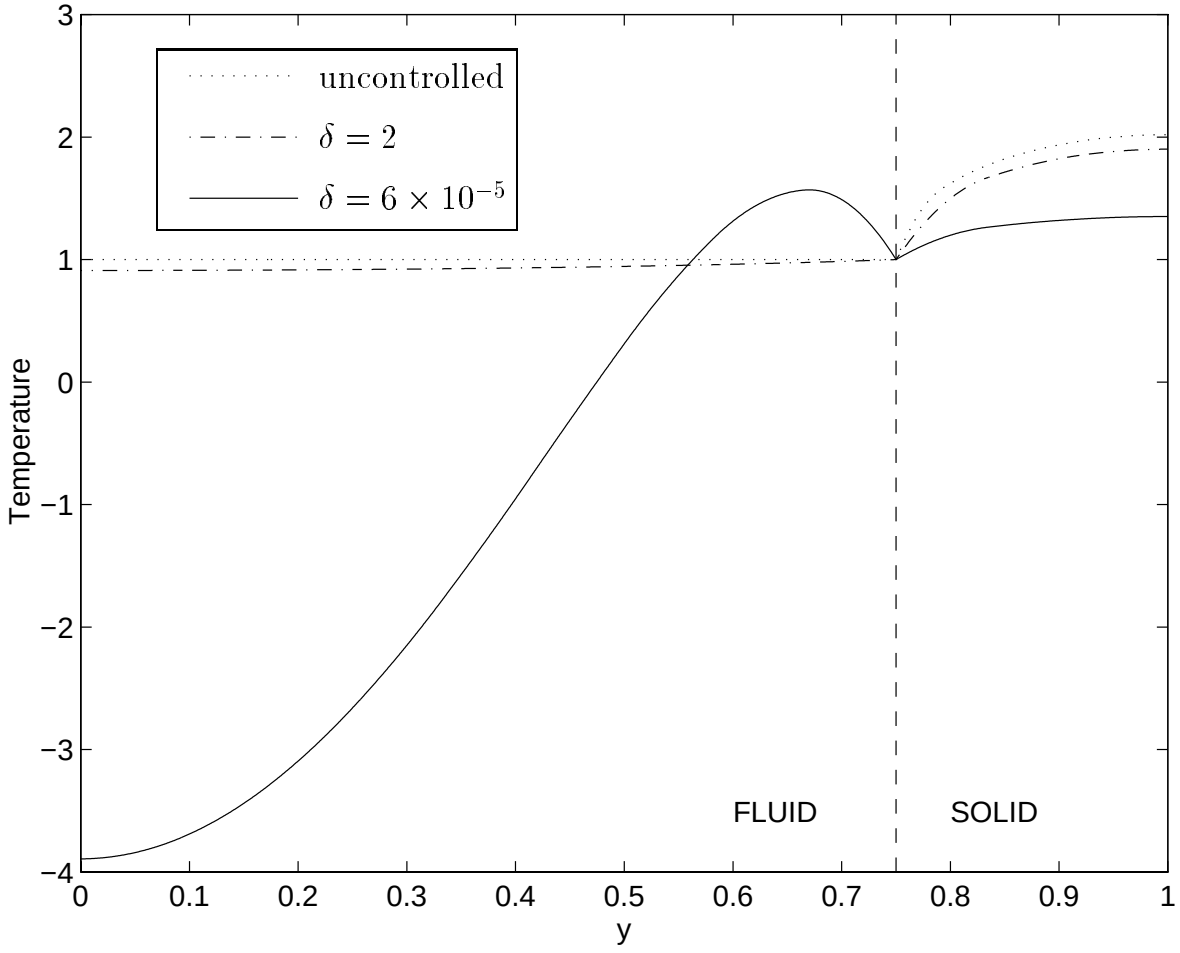


Figure 3.9: The Optimal Controls on Γ_w for Problem (DP1)

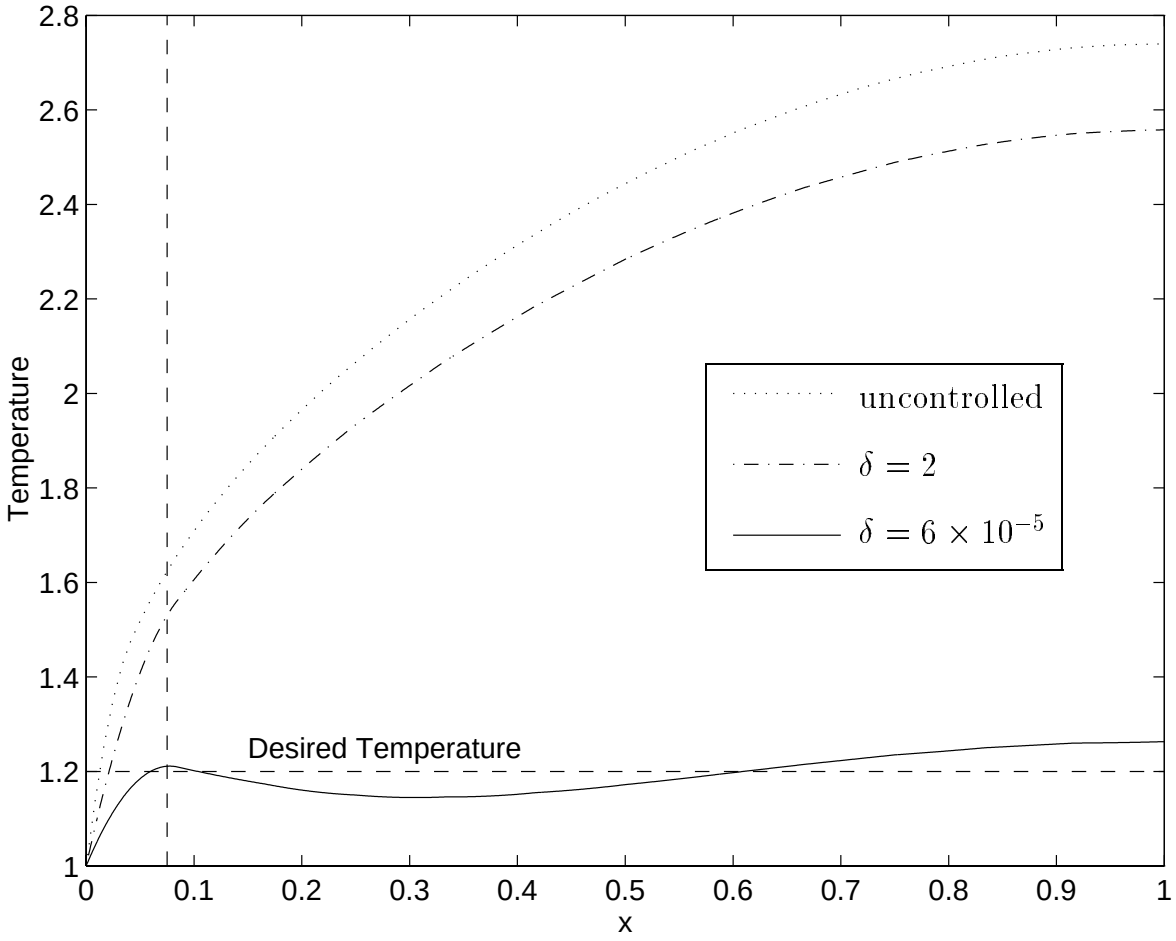


Figure 3.10: The Temperature Distributions on Γ_r for Problem (DP1)

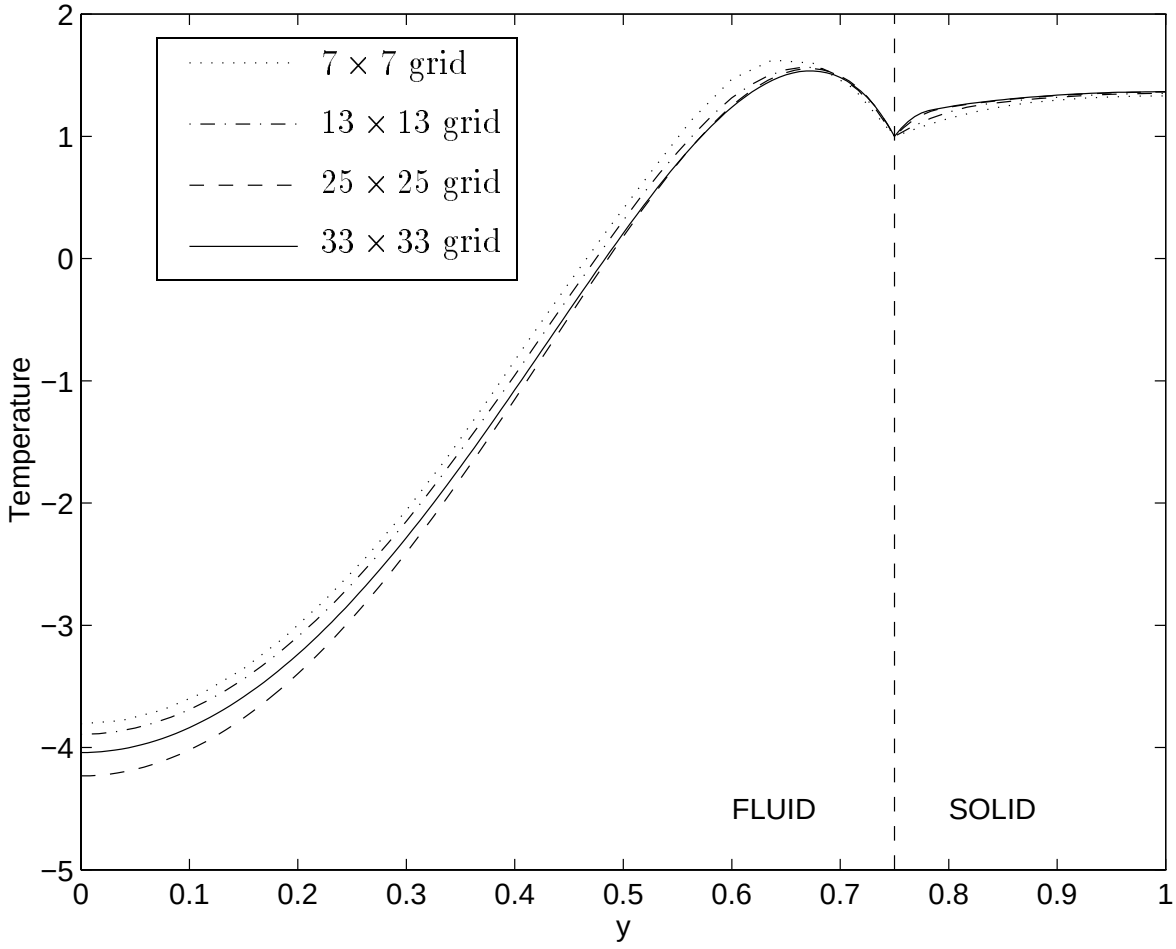


Figure 3.11: The Optimal Controls on Γ_w for Problem (DP1): Grid Study

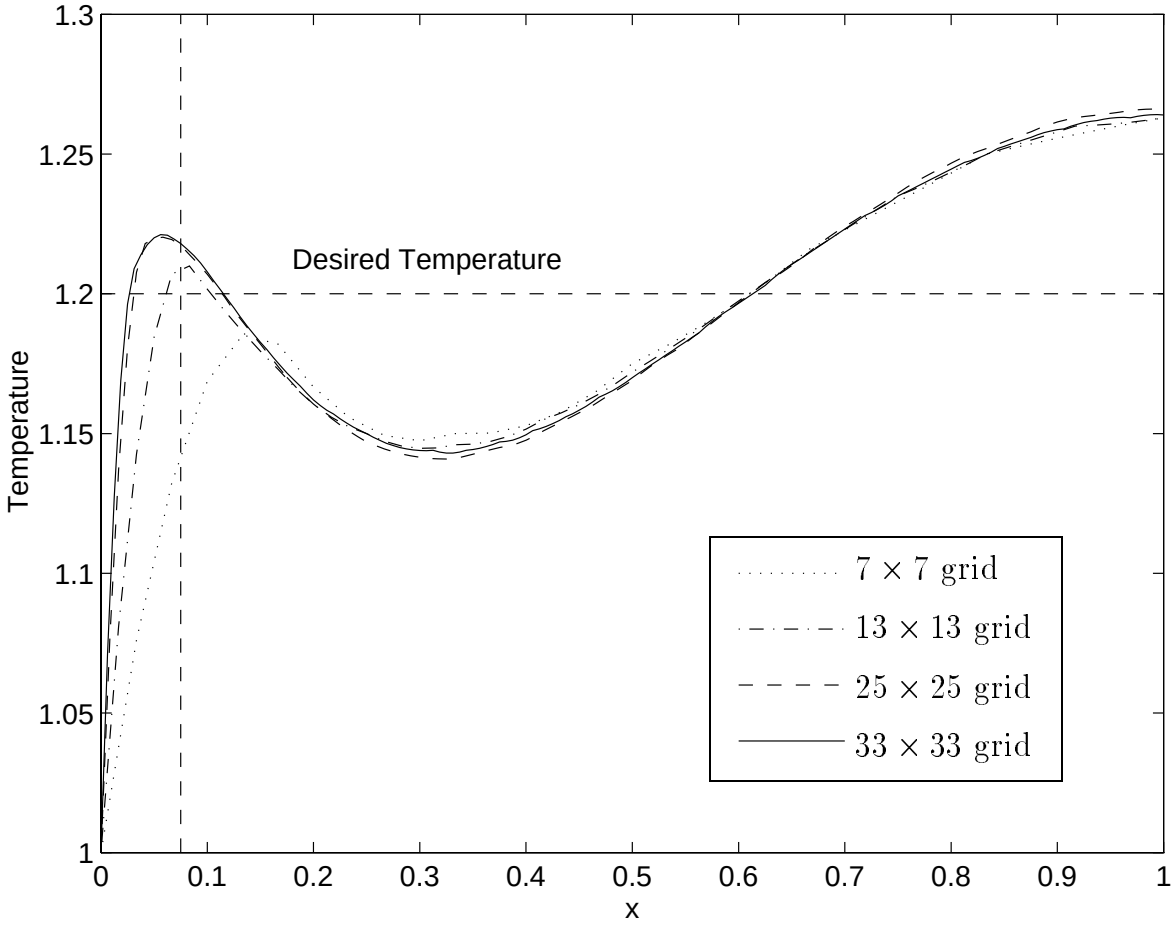


Figure 3.12: The Temperature Distributions on Γ_r for Problem (DP1): Grid Study

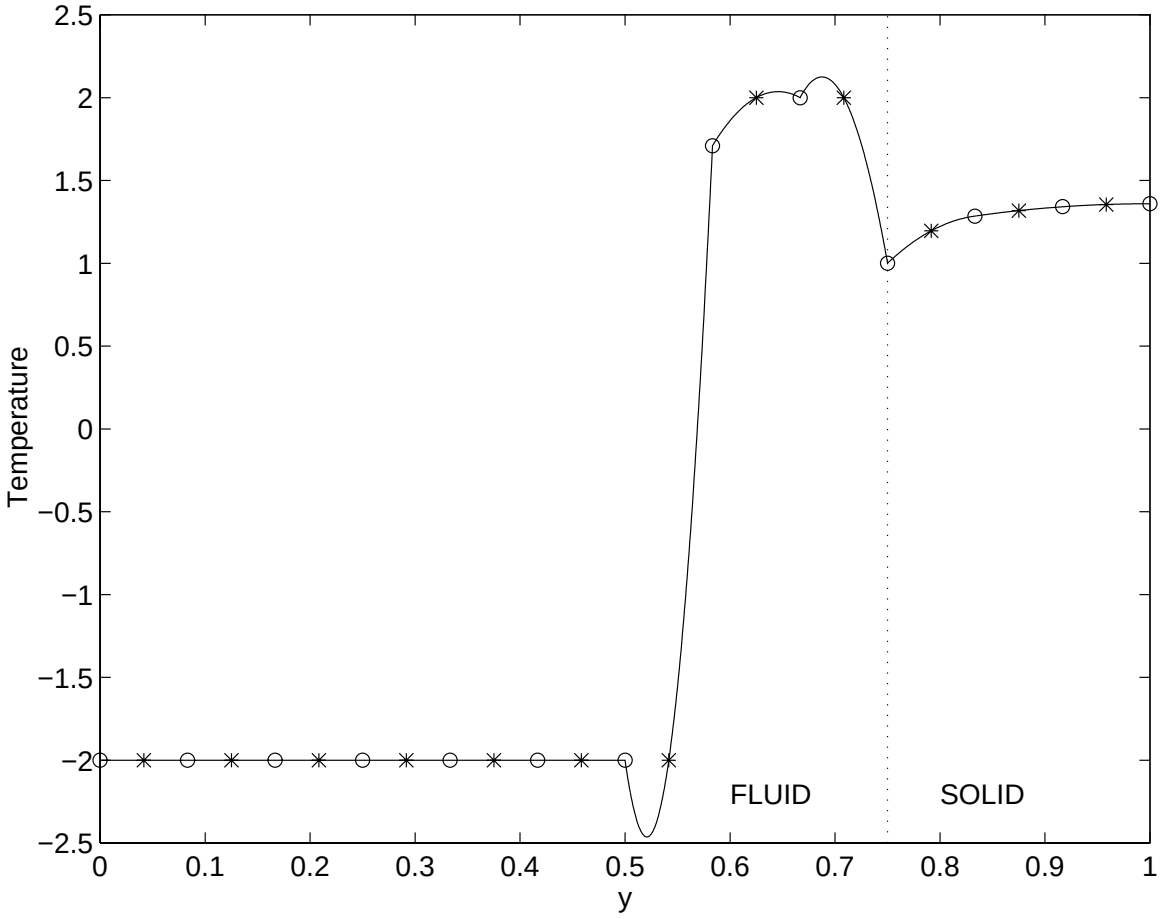


Figure 3.13: The Optimal Controls on Γ_w for Problem (DP2) with $|\vec{w}| \leq 2$

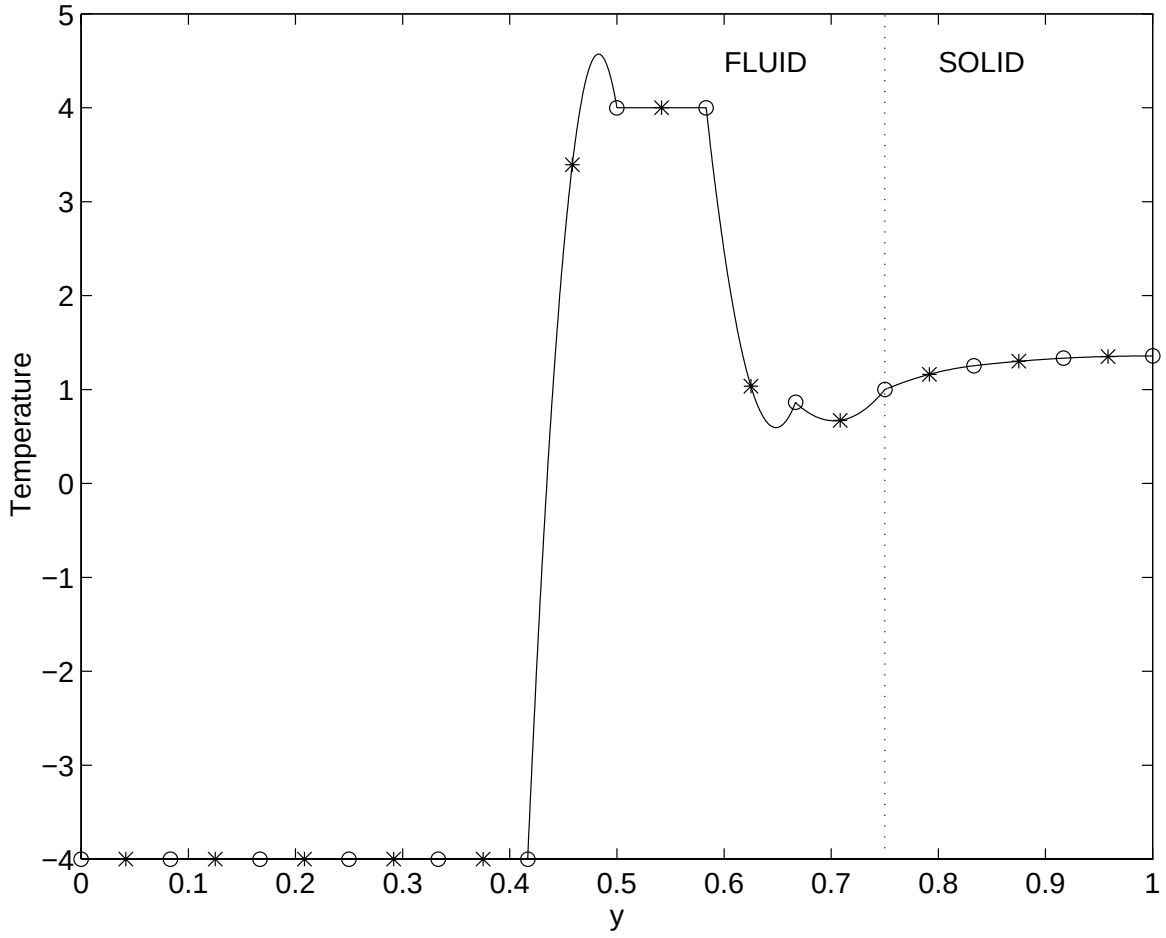


Figure 3.14: The Optimal Controls on Γ_w for Problem (DP2) with $|\vec{w}| \leq 4$

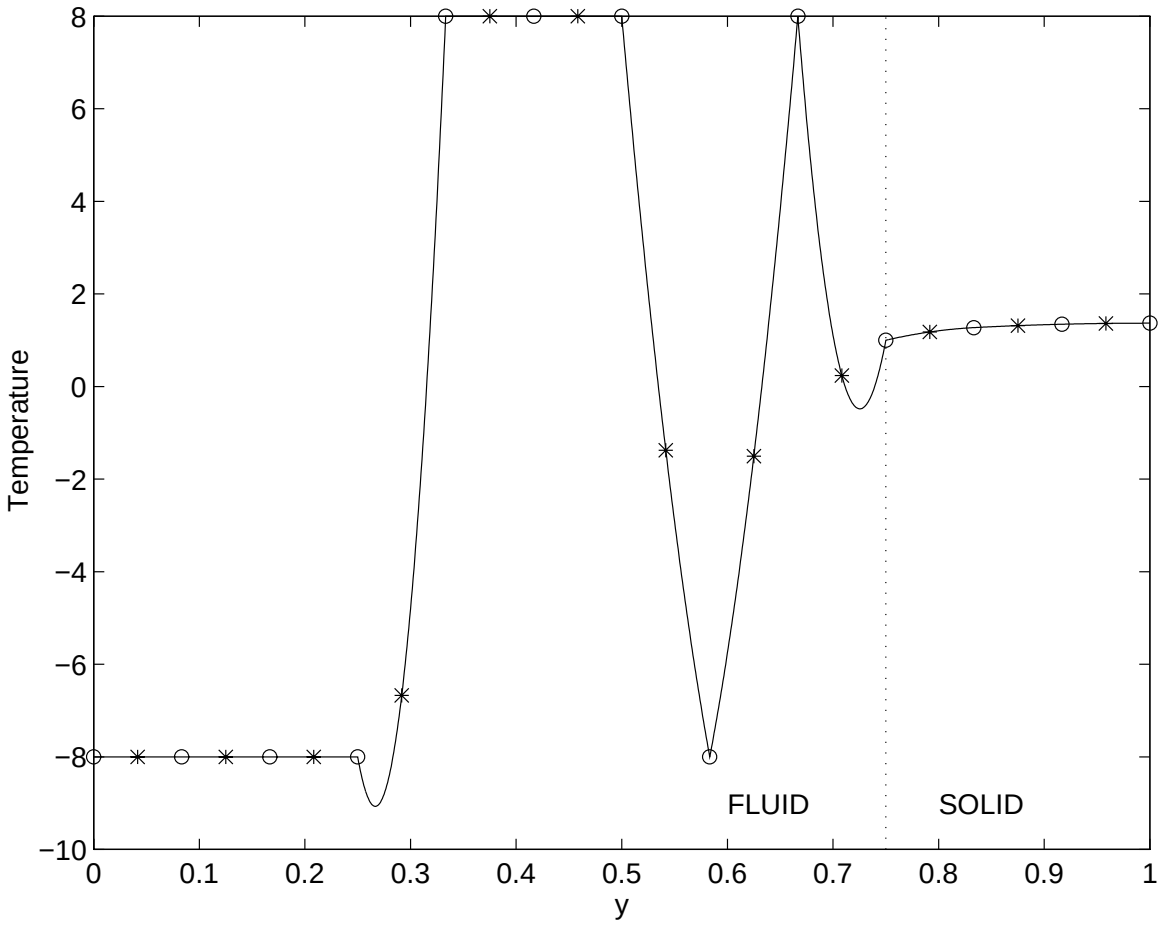


Figure 3.15: The Optimal Controls on Γ_{w} for Problem (DP2) with $|\vec{w}| \leq 8$

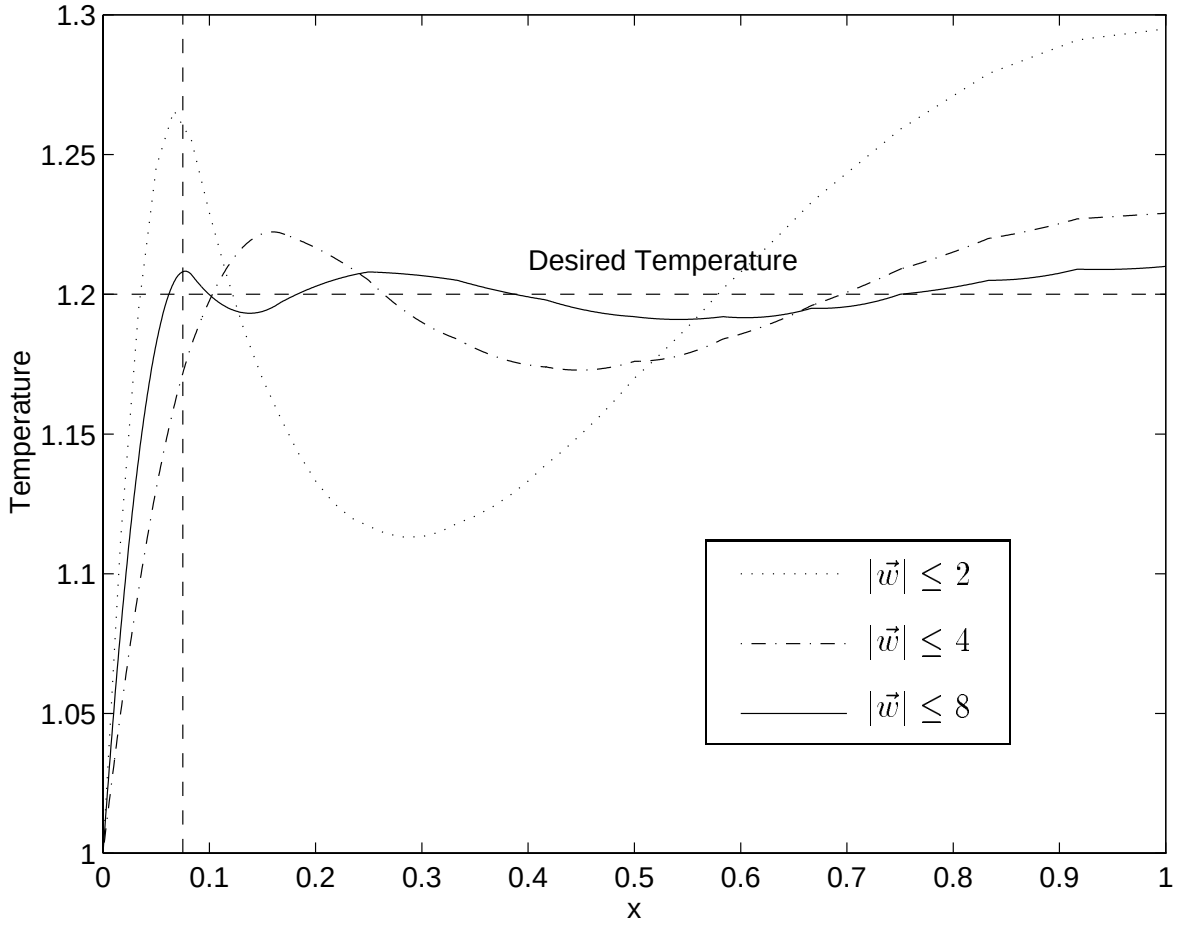


Figure 3.16: The Temperature Distributions on Γ_r for Problem (DP2)

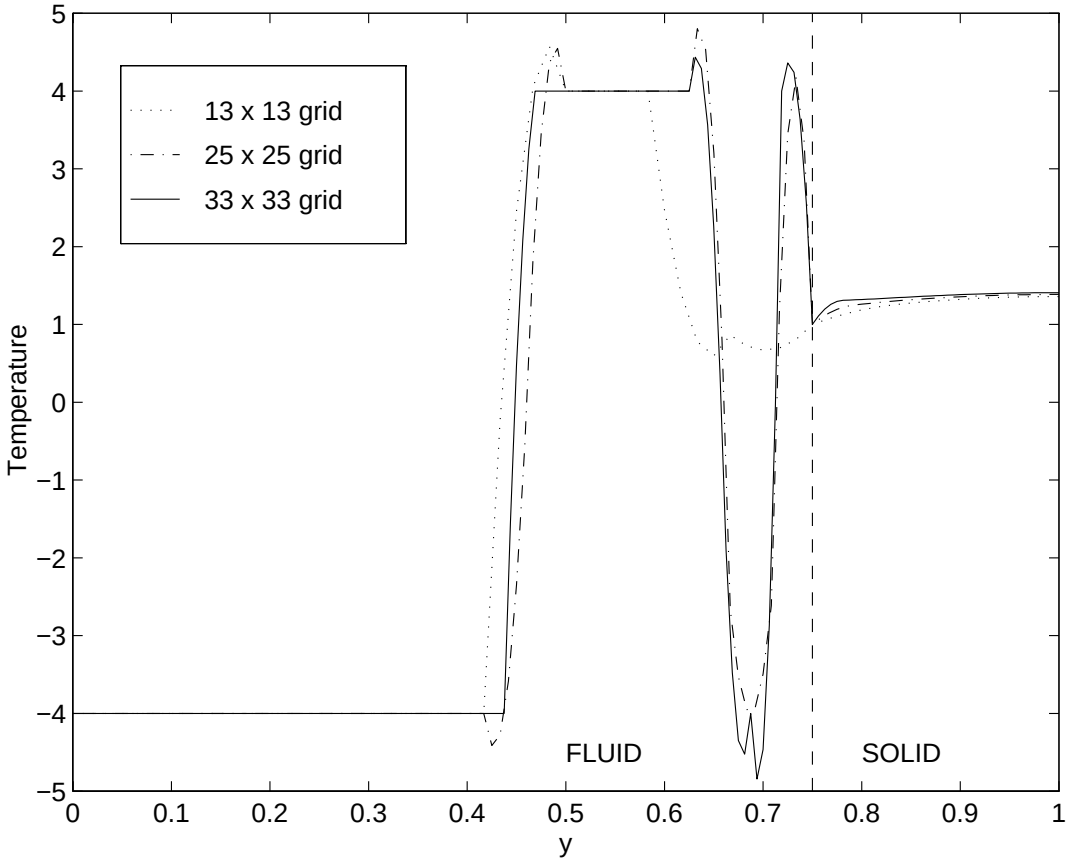


Figure 3.17: The Optimal Controls on Γ_w for Problem (DP2) with $|\vec{w}| \leq 4$: Grid Study

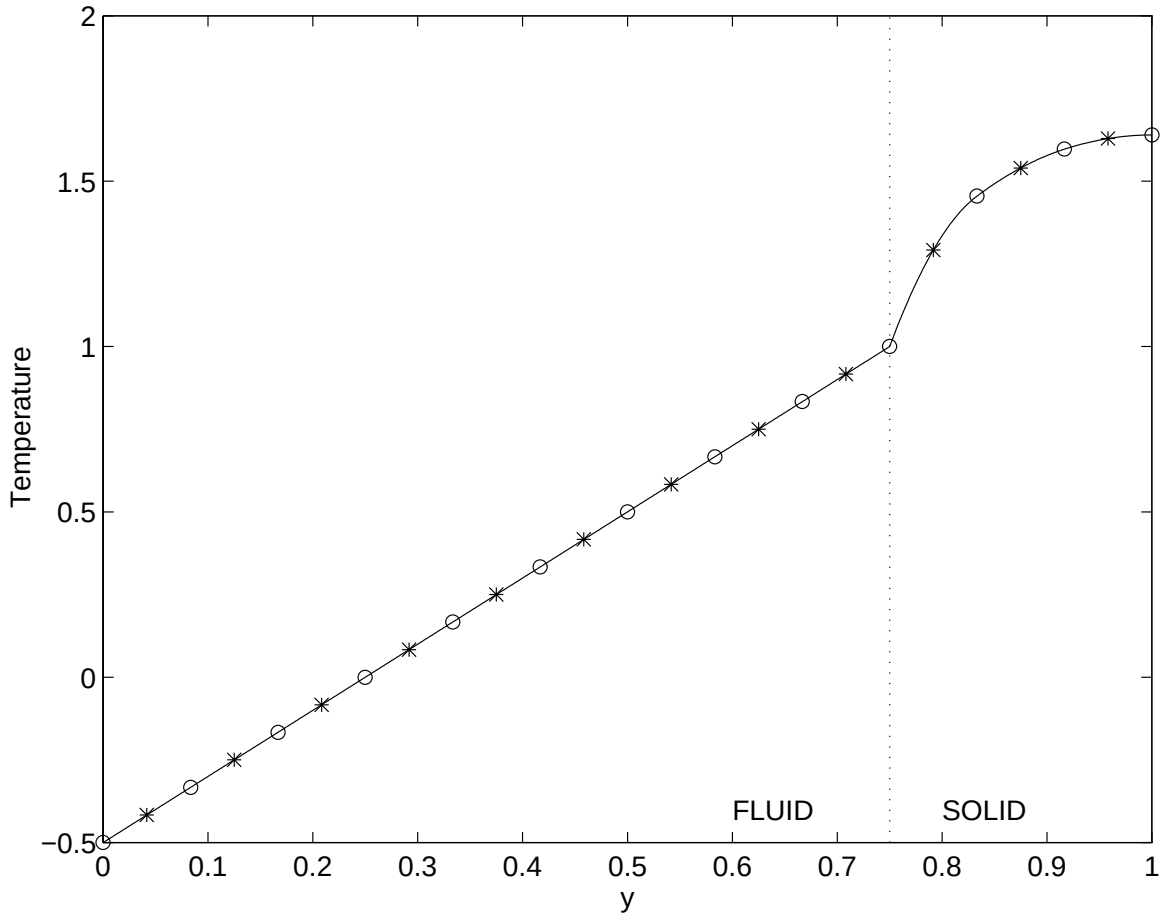


Figure 3.18: The Optimal Controls on Γ_{w} for Problem (DP3) with $|\vec{w}| \leq 4$, $|\mathbf{B} \cdot \vec{w}| \leq 2$

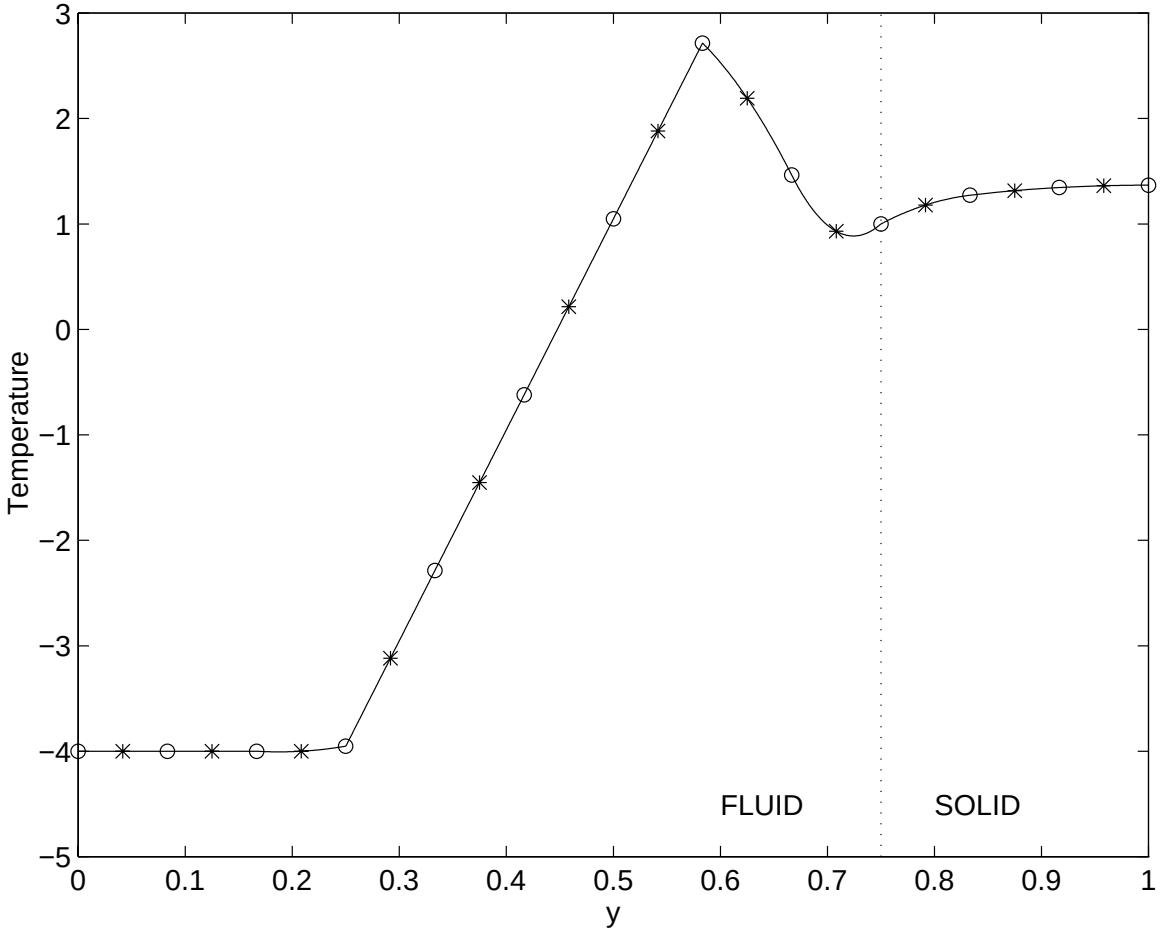


Figure 3.19: The Optimal Controls on Γ_w for Problem (DP3) with $|\vec{w}| \leq 4$, $|\mathbf{B} \cdot \vec{w}| \leq 20$

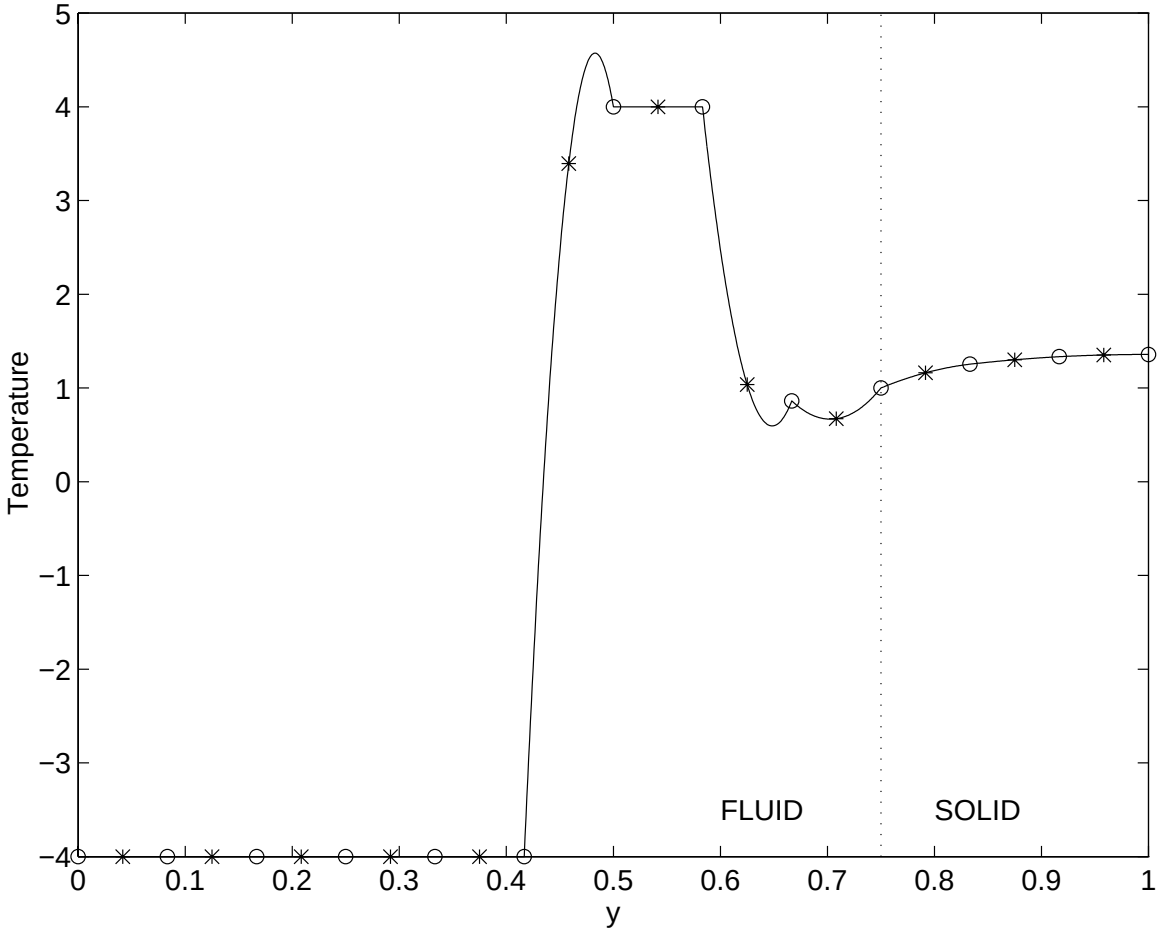


Figure 3.20: The Optimal Controls on Γ_w for Problem (DP3) with $|\vec{w}| \leq 4$, $|\mathbf{B} \cdot \vec{w}| \leq 2000$

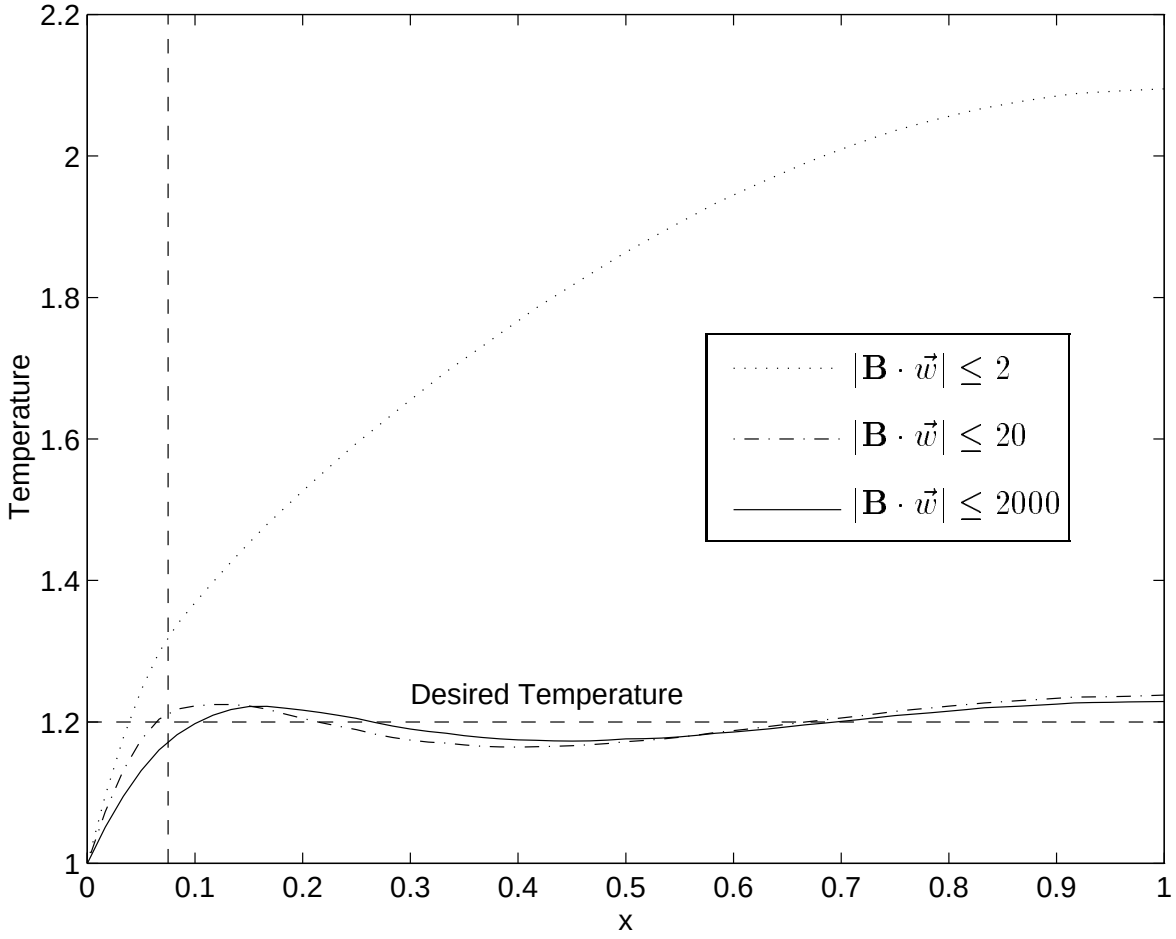


Figure 3.21: The Temperature Distributions on Γ_{r} for Problem (DP3)

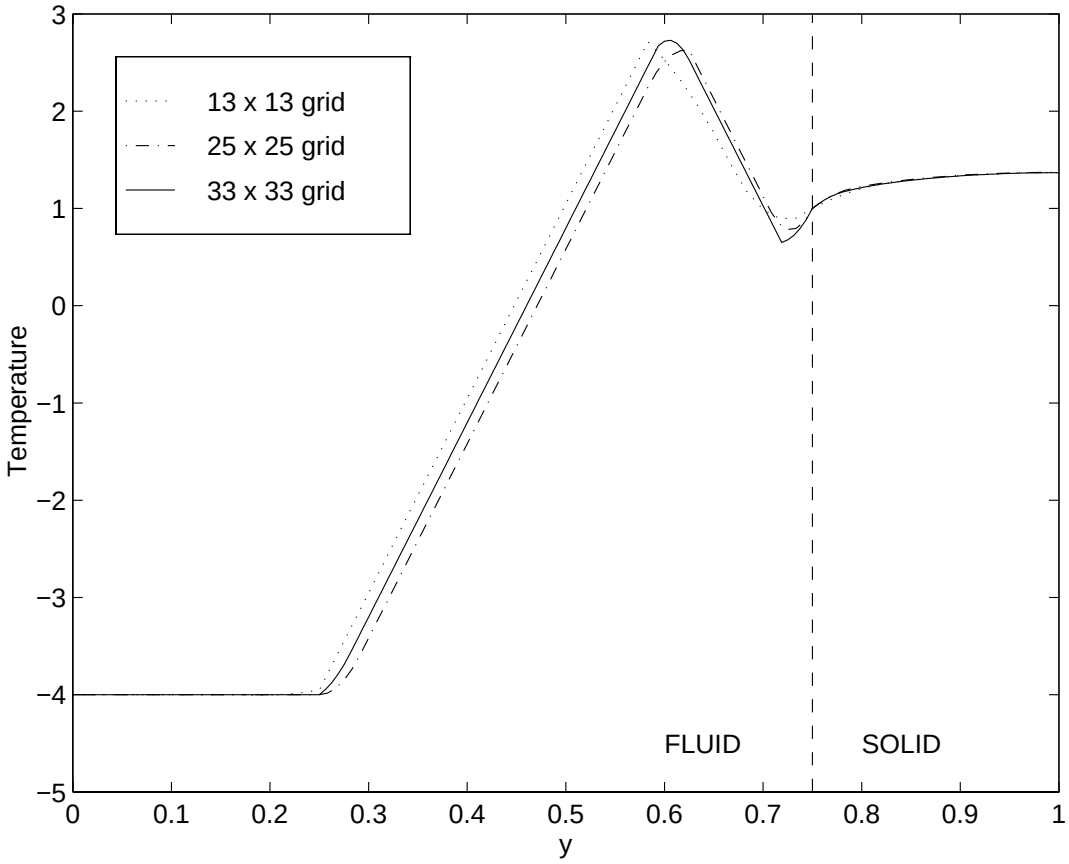


Figure 3.22: The Optimal Controls on Γ_w for Problem (DP3) with $|\vec{w}| \leq 4$, $|\mathbf{B} \cdot \vec{w}| \leq 20$: Grid Study

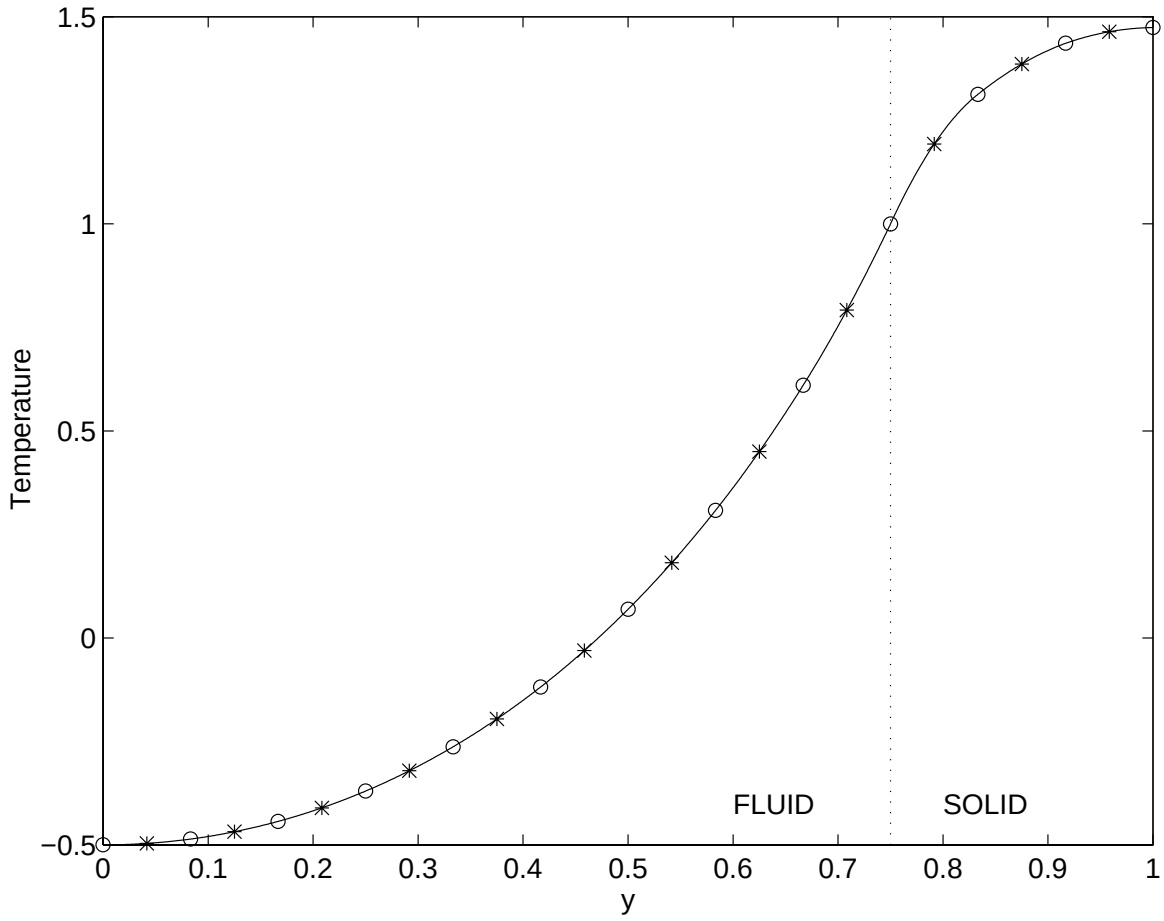


Figure 3.23: The Optimal Controls on Γ_w for Problem (DP1) with $\delta = 0.05$

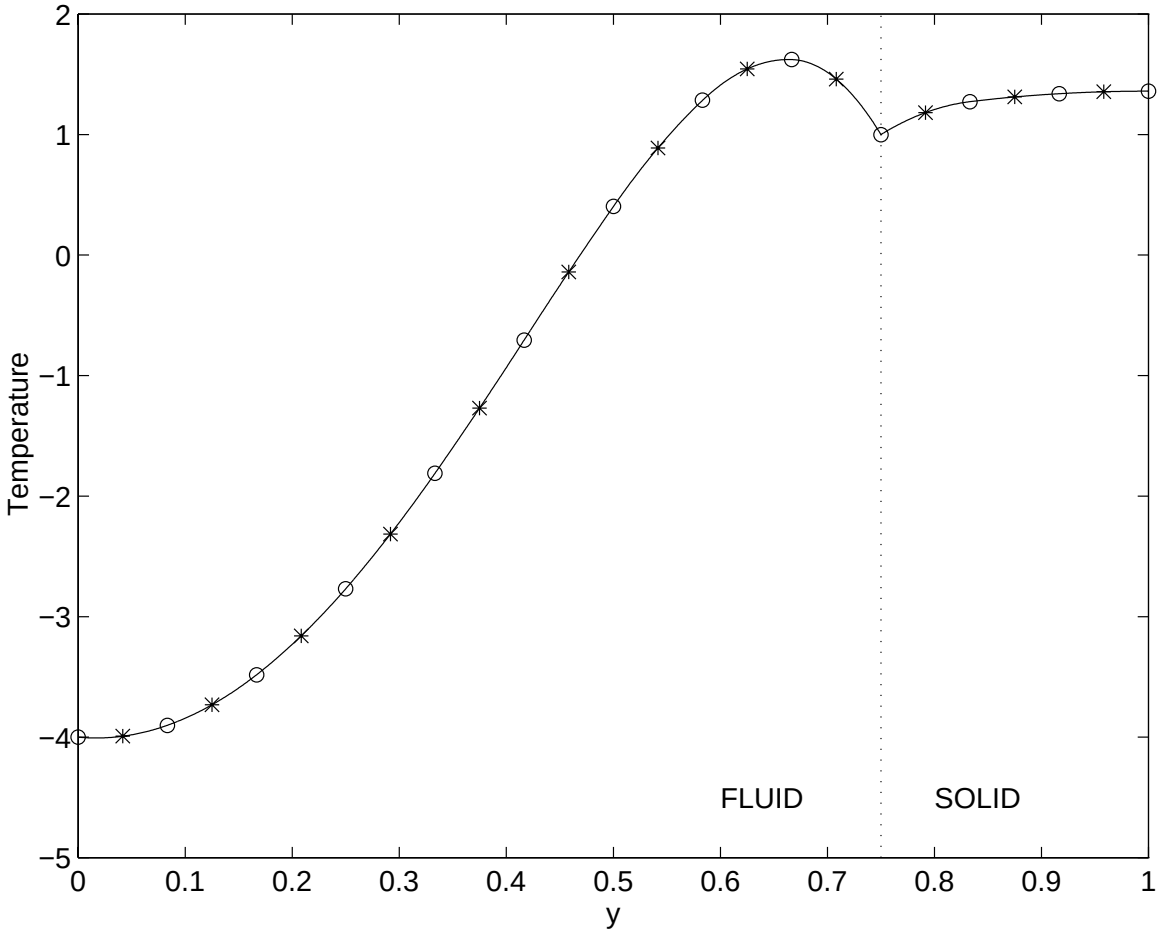


Figure 3.24: The Optimal Controls on Γ_w for Problem (DP1) with $\delta = 4 \times 10^{-5}$

Chapter 4

An Airfoil Design Problem

Transonic airfoil design has practical merit in developing wings which can fly more efficiently. Whitcomb, for instance, was responsible for developing supercritical wing sections, which allow subsonic aircraft to cruise at high Mach speeds [7]. Attempts to improve the aerodynamic efficiency of airfoils has been studied for years. In inverse design, pioneered by Lighthill [98], a flow property is prescribed on the surface of the airfoil, which would yield desirable flow characteristics under given conditions. A good overview is given by Drela [41]. The direct design methodology is based on trying to optimize a particular aerodynamic characteristic of the airfoil, and has been the focus of considerable research. Progress in the field is reviewed by Ta'asan [140]. In this chapter, we study a direct airfoil design problem using an all-at-once method.

Consider the following airfoil shape parameterization, as formulated by Vanderplaats [149] and implemented by Joh *et al.* [90] and Narducci *et al.* [106, 107]. The airfoil geometry is represented as the weighted combination of six shape functions

$$y(x/c) = \sum_{i=1}^{6} \omega_i y_i(x/c) \tag{4.1}$$

Four of the shape functions are pre-existing airfoils, namely, NACA 2412, NACA 64_1-412, NACA 65_2-415 and NACA 64_2-A215. The shape functions y_1–y_4 may be referenced from Abbott and von Doenhoff [1]. The two additional shape functions are used to impose certain geometric closure conditions at the trailing edge of the airfoil. We have

$$y_5 = \begin{cases} x/c, & \text{on upper surface} \\ 0, & \text{on lower surface} \end{cases} \tag{4.2}$$

$$y_6 = \begin{cases} 0, & \text{on upper surface} \\ -x/c, & \text{on lower surface} \end{cases} \tag{4.3}$$

Since these functions are used to close the airfoil at the trailing edge, the weights ω_5 and ω_6 are fixed in terms of ω_1–ω_4. We set

$$y_{us}(1) = y_{ls}(1) = 0$$

which yield the following relations [107]

$$\omega_5 = -[y_{1_{us}}(1)\omega_1 + y_{2_{us}}(1)\omega_2 + y_{3_{us}}(1)\omega_3 + y_{4_{us}}(1)\omega_4] \tag{4.4}$$

$$\omega_6 = [y_{1_{ls}}(1)\omega_1 + y_{2_{ls}}(1)\omega_2 + y_{3_{ls}}(1)\omega_3 + y_{4_{ls}}(1)\omega_4] \tag{4.5}$$

where subscripts us and ls refer to the upper and lower surfaces, respectively. The shape functions y_1–y_6 are shown in Figures 4.1 and 4.2.

4.1 Formulation of the Design Problem

Given $\{\omega_i\}$, the flow, q, around the airfoil is governed by the Euler equations for a perfect gas. The design problem is formulated as follows:

$$\max_{\omega} C_L(q, \omega) \tag{4.6}$$

such that

$$R(q, \omega) = 0 \tag{4.7}$$

$$C_D(q, \omega) \leq C_{D_{max}} \tag{4.8}$$

$$S_{min} \leq S(\omega) \leq S_{max} \tag{4.9}$$

where $\omega = \{\omega_i\}$, and $q = [\rho\ u\ v\ p]^T$ denote the primitive variables of flow, with the usual notation. Equation (4.7) refers to the steady state Euler equations of flow, given by (A.1). The drag, C_D, in this case, is the wave drag. The lower limit on the area, S, is imposed so that the airfoil does not reduce to a flat plate, thus maintaining structural integrity. The upper limit is imposed so that we do not end up with thick, unrealistic airfoils. The freestream conditions are based on Mach number $M = 0.75$ flow at angle of attack $\alpha = 0$.

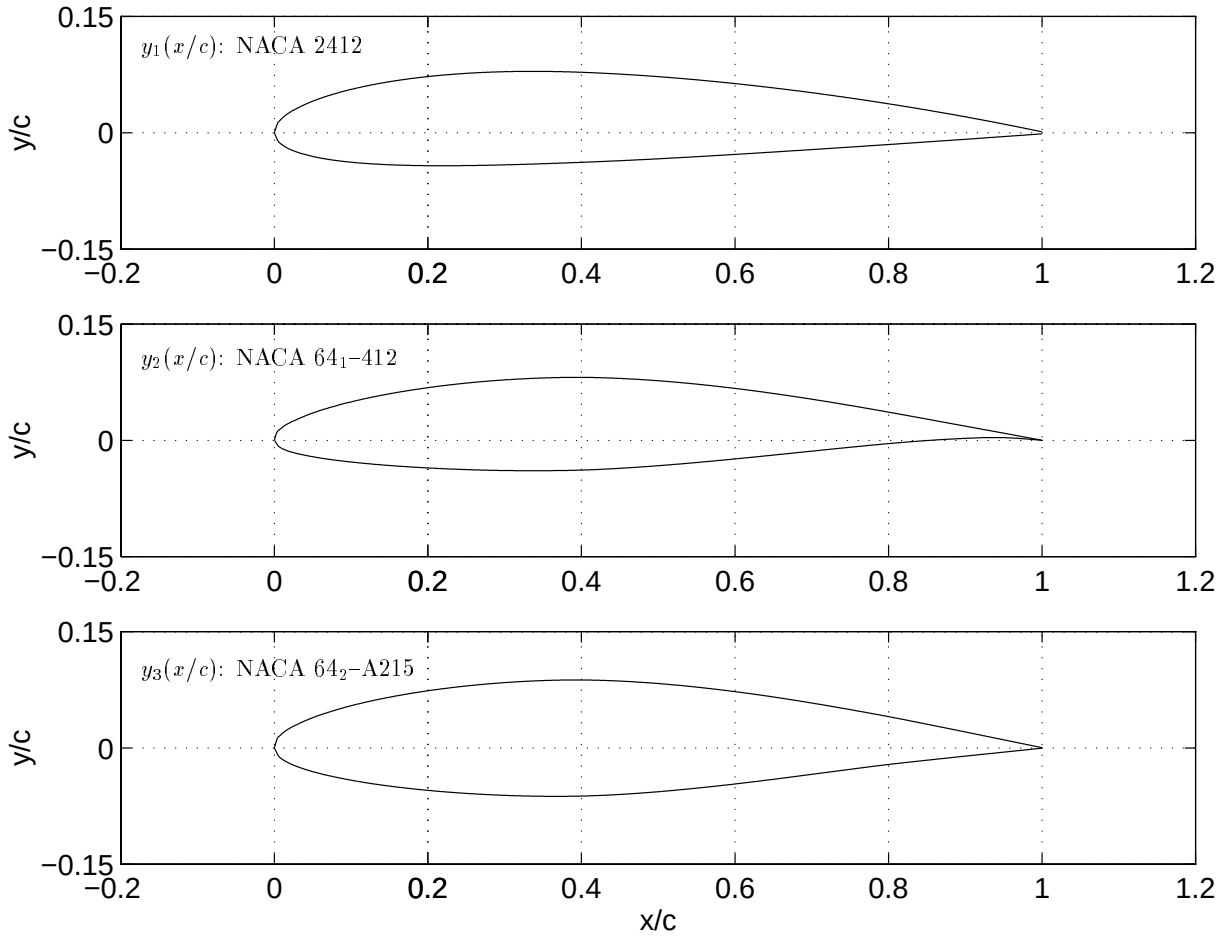


Figure 4.1: Shape Functions Used to represent the airfoil

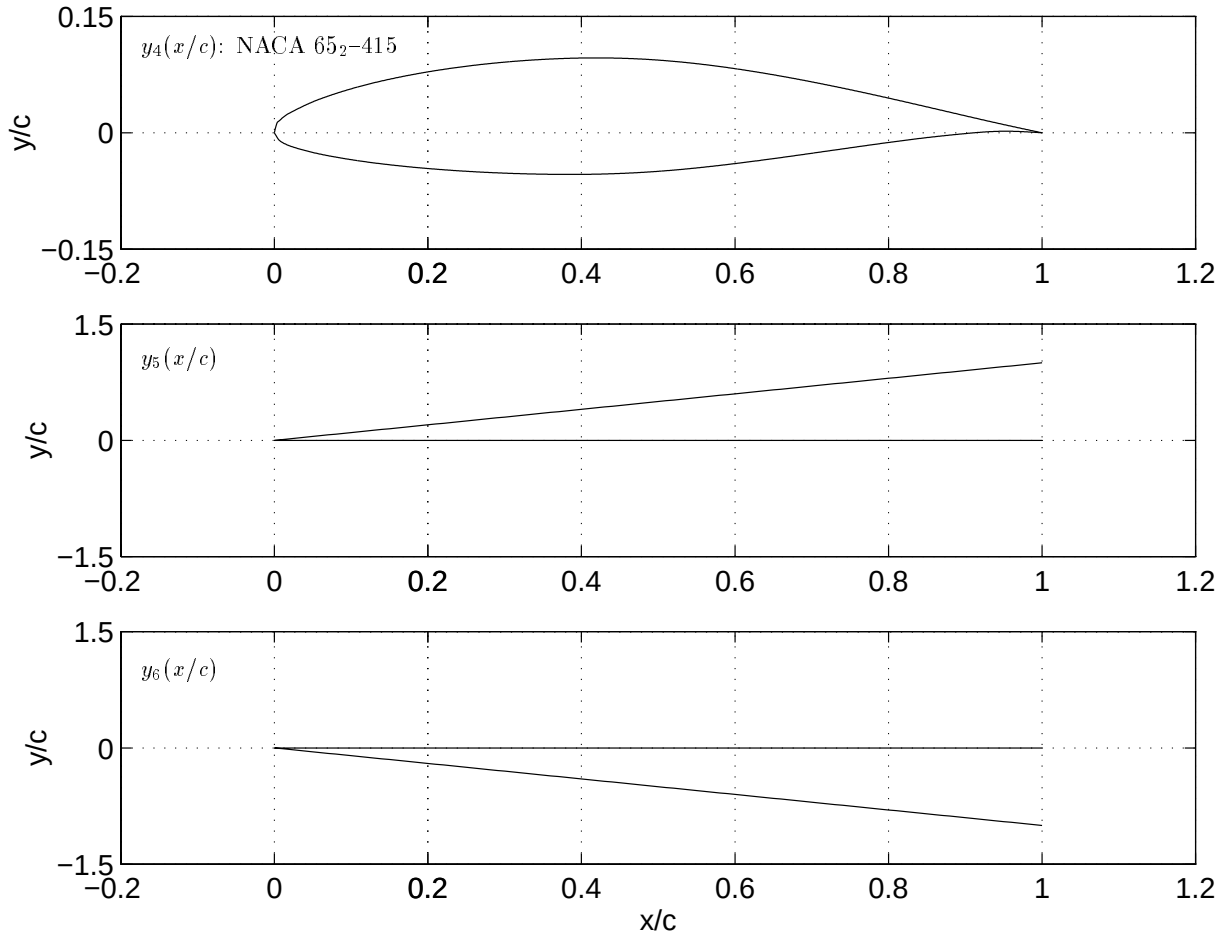


Figure 4.2: Shape Functions Used to represent the airfoil

4.2 Solution of the Analysis Problem

The analysis problem corresponds to the solution of the Euler equations for a given airfoil configuration. A detailed description of the algorithm used to obtain steady solutions is given in Appendix A. Following [107], the domain is discretized using a C–grid. Flow solutions are obtained using the EuleR Inviscid Code for Aerodynamics (ErICA) developed by Narducci [107]. Curvature corrected boundary conditions formulated by Dadone and Grossman [38] are used to enforce tangency on the airfoil surface. As a simplification in the current study, the curvature corrections are neglected. With the far-field boundary conditions of Thomas and Salas [145], the computational far field is placed roughly 10 chords away. A schematic of the algorithm used by ErICA to solve the governing Euler flow equations, $R(q, \omega) = 0$ for given ω, is given in Figure 4.3. The procedure describes an Euler implicit time integration scheme, with approximate factorization, also known as the Alternating Direction Implicit (ADI) method.

4.2.1 Airfoil Shape

As mentioned earlier, the airfoil is represented as the weighted combination of six shape functions (4.1),

$$y(x/c) = \sum_{i=1}^{6} \omega_i y_i(x/c)$$

The first four shape functions are pre-existing NACA series airfoils. While analytical formulae for computing the shapes exist [1], for the present purposes, it is sufficient to use curve fits to the pointwise data given for the airfoils, in Appendix III of Abbot and von Doenhoff [1]. We fit separate cubic splines to represent the upper and lower surfaces of the airfoils. The airfoils generated by curve-fitting are shown in Figures 4.1 and 4.2, respectively. The remaining two shape functions are given by equations (4.2) and (4.3), respectively, as shown in Figure 4.2. These functions are used to close the airfoil at the trailing edge. Given design parameters ω_1–ω_4, weights ω_5 and ω_6 can be obtained from equations (4.4) and (4.5), respectively. Our required airfoil shape can then be obtained as a weighted sum, using (4.1).

An efficient approach to implement the above is to close each airfoil y_1–y_4 individually to obtain $\hat{y}_1$–$\hat{y}_4$, and use these as our design bases. This eliminates ω_5 and ω_6. Note that though Figure 4.1 and 4.2 seem to indicate to the naked eye that the airfoils are closed, there *is* a finite trailing edge thickness for the NACA 2412 and NACA 64_2-$A215$ airfoils.

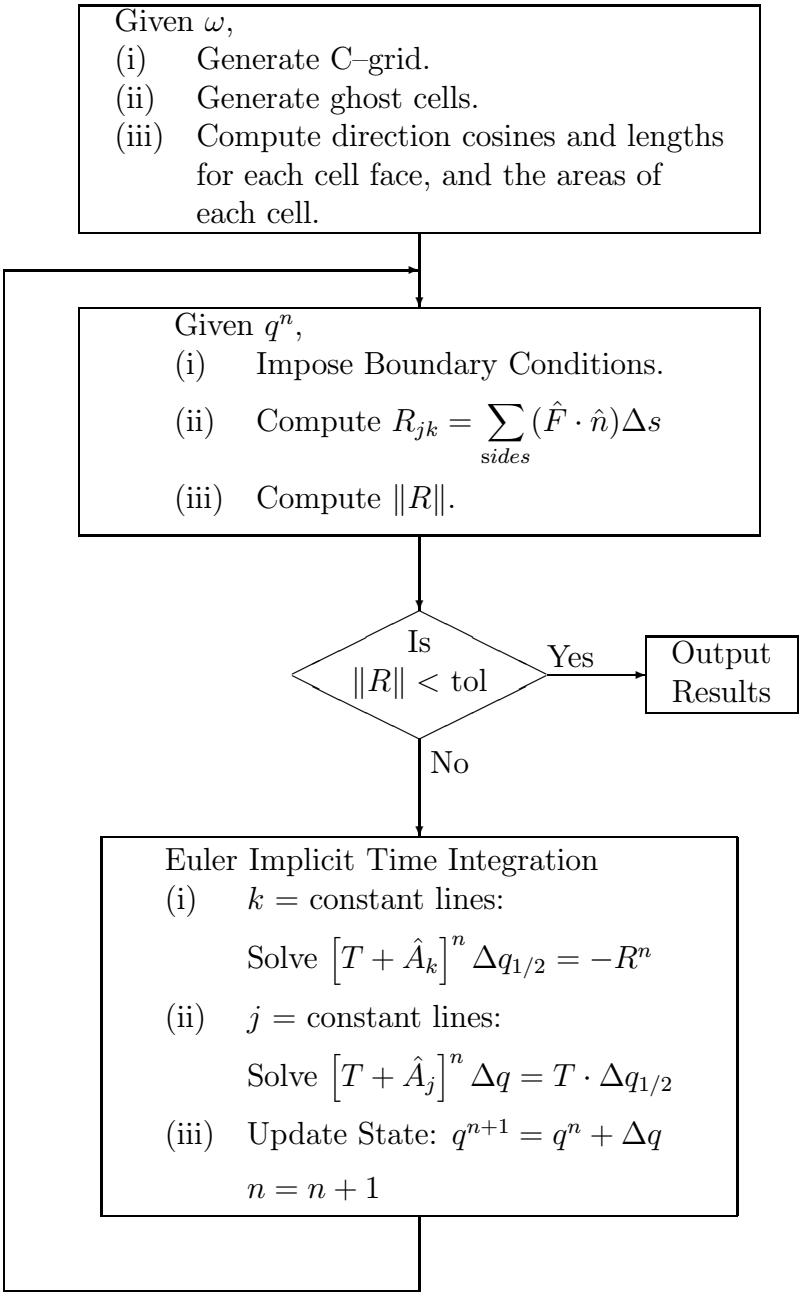


Figure 4.3: Schematic of ErICA Algorithm

We have

$$y(x/c) = \sum_{i=1}^{4} \omega_i \hat{y}_i(x/c) \tag{4.10}$$

4.2.2 Grid Generation

The grid is generated algebraically, by the following procedure: we first distribute points on bottom boundary, corresponding to the airfoil surface and the trailing edge wake, and on the top boundary of the computational grid, corresponding to the far-field boundary. Once the boundaries are defined, we connect corresponding pairs of points on the top and bottom boundaries using straight lines. While more sophisticated grid generation schemes such as elliptic schemes and algebraic schemes using splines exist, we prefer to use this simple grid because of the relative ease of generating the grid, and of obtaining grid sensitivities. The procedure is briefly described in the following sections.

Distribution of Points on the Airfoil

Consider a $(N_j \times N_k)$ grid, with N_a points on the airfoil. To maintain symmetry, we stipulate that N_j and N_a are odd integers. We assume unit chord length. The leading edge of the airfoil is placed at the origin $(0, 0)$ and the trailing edge is at $(1, 0)$. Thus, we have,

$$\begin{array}{lllllll}
x(N_c, 1) & = & 0; & y(N_c, 1) & = & 0; & \text{at the leading edge} \\
x(N_c + N_{a/2}, 1) & = & 1; & y(N_c + N_{a/2}, 1) & = & 0; & \text{at the trailing edge, upper surface} \\
x(N_c - N_{a/2}, 1) & = & 1; & y(N_c - N_{a/2}, 1) & = & 0; & \text{at the trailing edge, lower surface}
\end{array}$$

where, $N_c = \frac{N_j - 1}{2} + 1$ and $N_{a/2} = \frac{N_a - 1}{2}$. We use a Hermite cubic distribution of points along the arclength for the upper and lower surfaces of the airfoil:

$$d_j = (3p^2 - 2p^3) + 0.25(p - 2p^2 + p^3) + 0.75(p^3 - p^2) \qquad j = 0, \ldots, N_{a/2}$$

where, $p = \frac{j}{N_{a/2}}$. See Figure 4.4. The slopes, $d' = 0.25$ at $p = 0$ and $d' = 0.75$ at $p = 1$ were chosen via numerical experimentation to improve quality of the solution and to improve the convergence behavior of the flow solver. This ensures a greater concentration of points near the leading edge and, to a lesser extent, the trailing edge of the airfoil. Distributing the points along the arclength entails the following subproblem

For given ω_i with airfoil representation (4.10),

$$\begin{array}{l} \text{Find:} \\ \qquad x_j \qquad\qquad j = 0, \ldots, N_{a/2} \end{array}$$

such that

$$\begin{aligned} L_m &= d_m L \qquad m = 1, \ldots, N_{a/2} - 1 \\ x_0 &= 0 \\ x_{N_{a/2}} &= 1 \end{aligned}$$

where,

$$\begin{aligned} l_n^2 &= (x_n - x_{n-1})^2 + (y(x_n) - y(x_{n-1}))^2 \\ L_m &= \sum_{n=1}^{m} l_n \\ L &= L_{N_{a/2}} \end{aligned}$$

The above problem can be solved using Newton's method applied to the system, $C = 0$, where

$$C_m(x, \omega) \equiv L_m(x, y(x, \omega)) - d_m L(x, y(x, \omega))$$

Grid sensitivities for the abscissas with respect to the design variables, ω can then be obtained using the implicit function theorem leading to:

$$\frac{\partial C}{\partial \omega} + \left[\frac{\partial C}{\partial x}\right] \frac{\partial x}{\partial \omega} = 0$$

Grid sensitivities for the ordinates on the airfoil surface can be obtained from (4.10). The points on the trailing edge line aft of the airfoil to the far-field are distributed using a hyperbolic tangent distribution, so that there is a concentration of points close to the trailing edge of the airfoil. Note that these points are independent of the design variables.

Distribution of Points on the Far-field Boundary

The far-field boundary is given by:

$$y = \begin{cases} \pm\sqrt{r^2 - x^2}, & -r \le x \le 0; \\ \pm r, & 0 \le x \le r + 1. \end{cases}$$

where r is the far-field boundary. Refer Figure 4.6. For our problem, we use $r = 10$. Points are distributed along the boundary using a piecewise hyperbolic tangent distribution such

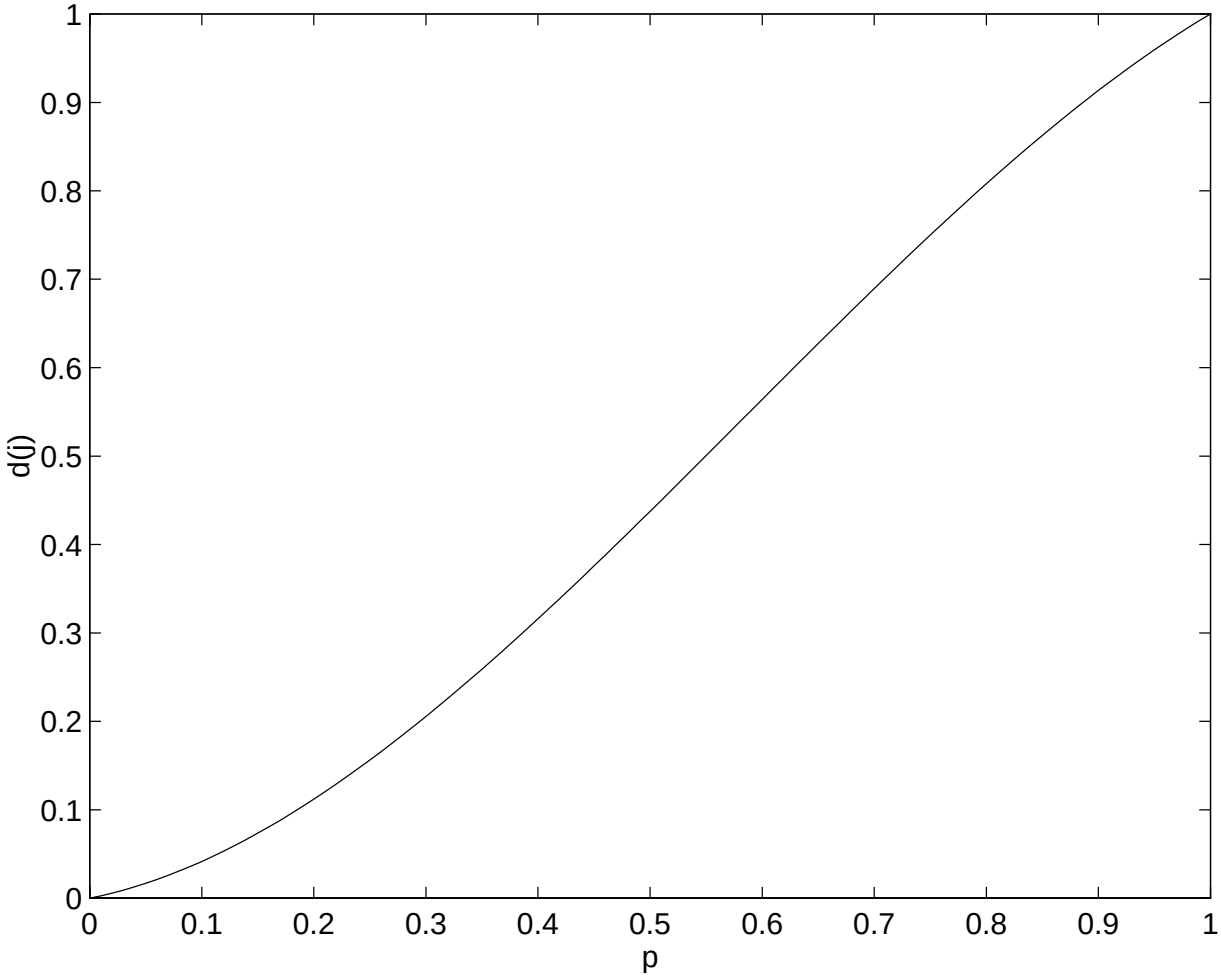


Figure 4.4: Hermite Cubic Distribution used for Airfoil Surface

that seventy five percent of the number of points used to represent the airfoil, N_a, are placed on the semicircular portion. The hyperbolic tangent distribution is given by

$$d_j = 1 - \frac{\tanh\left(\beta\left(1 - \frac{j}{m}\right)\right)}{\tanh(\beta)} \qquad j = 0, \ldots, m \tag{4.11}$$

where, β is a compression parameter. Figure 4.5 shows a typical tanh distribution, with $\beta = 4$.

First, we distribute points corresponding to the points on the airfoil surface. We use (4.11) with $m = N_{a/2}$. We have,

$$\theta_j = (1 - d_j)\delta \qquad j = 0, \ldots, N_{a/2}$$

where $\delta = \frac{\pi}{2} + \tan^{-1}(0.2)$. On the semicircular portion of the far-field, we have,

$$\left.\begin{array}{rcl} x(N_c \pm j, N_k) & = & -r\cos\theta_j \\ y(N_c \pm j, N_k) & = & r\sin\theta_j \end{array}\right\} \qquad j = 0, \ldots, \frac{3N_{a/2}}{4}$$

while on the horizontal portion,

$$\left.\begin{array}{rcl} x(N_c \pm j, N_k) & = & r\tan(\theta_j - \frac{\pi}{2}) \\ y(N_c \pm j, N_k) & = & \pm r \end{array}\right\} \qquad j = \frac{3N_{a/2}}{4} + 1, \ldots, N_{a/2}$$

The compression parameter is chosen such that 75% of the points are placed on the semicircular portion, *i.e.*, $\theta_j = \frac{\pi}{2}$ for $j = \frac{3N_{a/2}}{4}$. The rest of the points are distributed using (4.11) with $m = N_c - N_{a/2}$. We have

$$\left.\begin{array}{rcl} x(N_c \pm N_{a/2} \pm j, N_k) & = & 0.2r + d_j(r - 0.2) \\ y(N_c \pm N_{a/2} \pm j, N_k) & = & \pm r \end{array}\right\} \qquad j = 1, \ldots, m$$

The compression parameter β is chosen here so as to maintain continuity with the previously described distribution.

Distribution of Grid Points in the Interior

Given the distribution of the points on the top and bottom boundaries, a point on the $k = 1$ line, corresponding to the airfoil surface and the wake, is now connected to a corresponding point on the $k = N_k$ line, corresponding to the far-field, using a straight line. Points are distributed along the so-formed $j =$ constant lines using a hyperbolic tangent distribution. We use a higher compression parameter in the hyperbolic tangent distribution at the leading

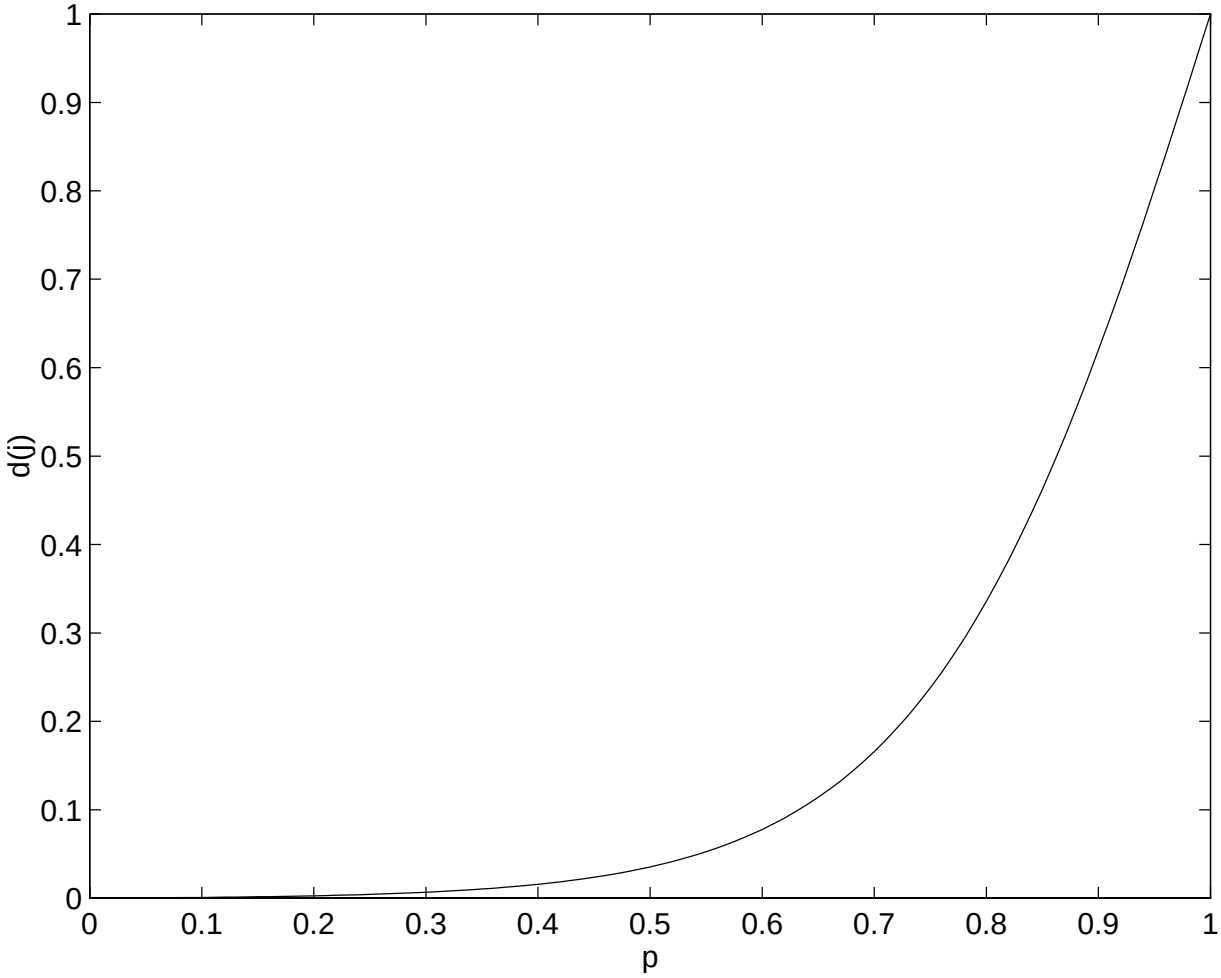


Figure 4.5: Tanh Distribution with $\beta = 4$

edge, as compared to the distribution near the trailing edge. This concentrates more points at the nose of the airfoil, where pronounced dynamical behavior is expected.

A typical 201×53 grid is shown in Figure 4.6, with a closer look at the distribution of points near the airfoil in Figure 4.7. We have 121 points on the airfoil surface, for the 201×53 C-grid.

4.3 Solution of the Design Problem by an SQP Method

We wish to solve the design problem (4.6)–(4.9) using the reduced SQP formulation described in Section 1.1. Refer Table 1.1. Basic requirements for executing this algorithm include being able to solve the linearized state equation (d), and the adjoint equation (a) at a given design point. Note that in the current chapter we use $R(\cdot)$ to represent the state constraints rather than $C(\cdot)$.

4.3.1 Solution of the Linearized State and Adjoint Equations

To solve the design problem using the Reduced SQP procedure described in Table 1.1, we need to be able to do the following:

- Provide an update, s_q, in the state variable, given an update in the control variable, s_w, by solving the linearized state constraint.
- Solve the adjoint equation at a given point.

We wish to achieve these tasks using the same problem-solving structure used in the flow solver, ErICA. A brief review of the methodology is given here. For more details, refer Appendix A. We have, in terms of the primitive variables, the time-dependent Euler equations in (semi-)discrete form are given by

$$SM\frac{\partial q}{\partial t} + R(q, \omega) = 0$$

where $M = \frac{\partial Q}{\partial q}$ is the Jacobian of the mapping between the conserved and the primitive variables, and $R = \sum_{sides}(\hat{F} \cdot \hat{n})\Delta s$ is the residual, where $\hat{F}$ is the inviscid flux. The residual is computed using an upwind scheme, with Van Leer Flux Vector Splitting. In order to

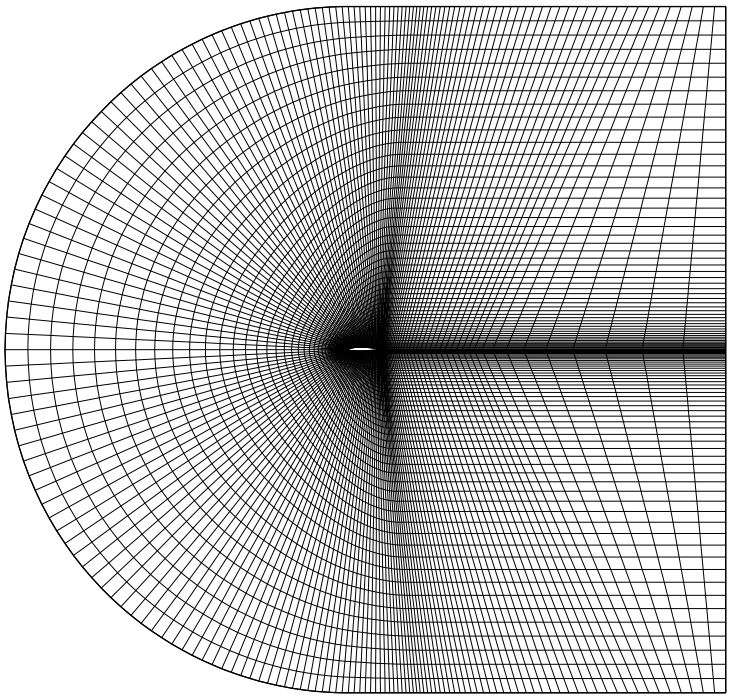


Figure 4.6: A typical sample of the 201×53 grid used

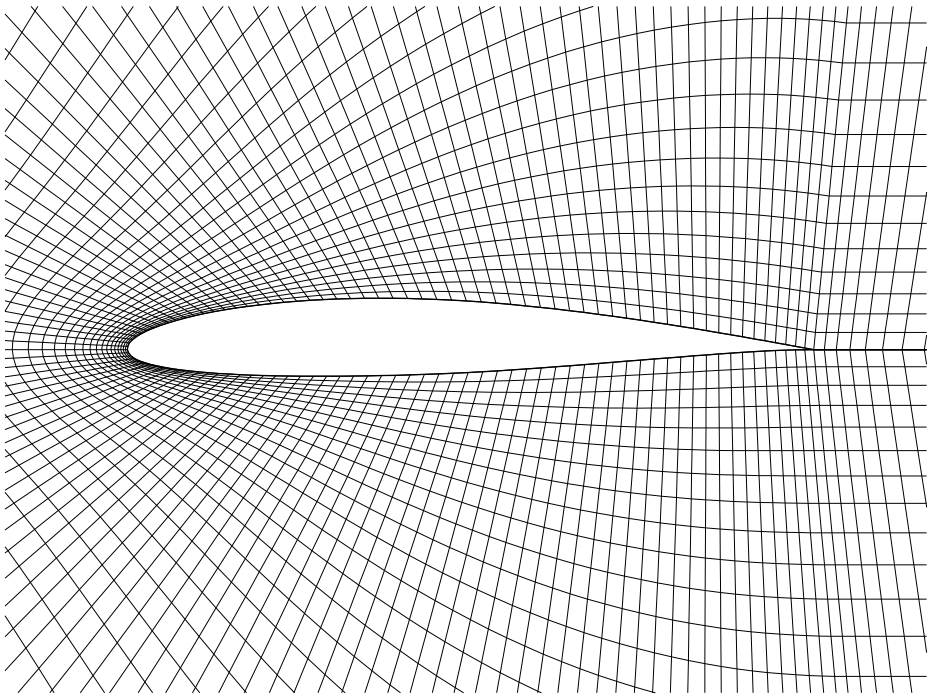


Figure 4.7: A Closeup View of the 201×53 grid

fully capture the shock, we use third order interpolation of the fluxes (see Section A.3.1). An Euler implicit scheme to march the residual to zero is given by,

$$\left[\frac{S}{\Delta t}M + \left(\frac{\partial R}{\partial q}\right)^n\right]\Delta q = -R^n$$

We solve this using an Alternating Direction Implicit (ADI) scheme. The idea is to factor the above problem so that we can solve it as a set of separate subproblems in each spatial direction. For the (j,k)th cell, we have

$$\begin{aligned}\left\{\frac{S}{\Delta t}M + \left(\left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j+1/2}\right.\right.\\ \left.\left. + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k+1/2}\right)^n\right\}\Delta q_{jk} &= -R(q^n_{jk})\end{aligned}$$

Rewriting, and factoring according to spatial directions, we have

$$\left[T + \hat{A}_k\right]^n T^{-1}\left[T + \hat{A}_j\right]^n \Delta q = -R^n \tag{4.12}$$

where

$$\begin{aligned} T &= \frac{S}{\Delta t}M \\ \hat{A}_k &= \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k+1/2} \\ \hat{A}_j &= \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j+1/2}\end{aligned}$$

We solve using the sequence

$$\begin{aligned}\left[T + \hat{A}_k\right]^n \Delta q_{1/2} &= -R^n \\ \left[T + \hat{A}_j\right]^n \Delta q &= T\cdot\Delta q_{1/2}\end{aligned} \tag{4.13}$$

$$q^{n+1} = q^n + \Delta q \tag{4.14}$$

The algorithm is shown in Figure 4.3. We use first order interpolation to compute the Jacobian terms, which yields a block tridiagonal structure along the ξ (k = constant) and η (j = constant) directions. Thus, each subproblem in (4.13) requires a block tridiagonal matrix inversion, which involves a block LU factorization and a block matrix solve; the latter consists of forward and backward substitutions.

Consider the algorithm given in Figure 4.8 for computing a solution to linearized state equation. This method entails driving an *unsteady* form of the linearized Euler equations towards steady state:

$$SM\frac{\partial s_q}{\partial t} = -\frac{\partial R}{\partial q}(q,\omega)s_q - \frac{\partial R}{\partial \omega}(q,\omega)s_\omega - R(q,\omega) \equiv -\bar{R}(s_q, s_\omega, q, \omega) = 0, \tag{4.15}$$

for given q, ω and s_ω. Note that the factor SM is added to the transient term in order to make the above equation consistent with the discretized Euler equations (See Section A.4). Note that the scheme used is identical to the one used in marching the nonlinear Euler equations, described above in equations (4.12)–(4.14), with the nonlinear residual, R replaced by the linearized residual, $\bar{R}$. We can view this algorithm as simply an iterative method for solving the linearized state equation. Note that a relaxation factor β is used to update the solution in the iterative process. Our numerical experiments showed that using $\beta = 1.25$ yielded improved convergence rates. Also, note that there is an external loop in the iterative process monitoring the residual. This is in order to ensure that the residual does not diverge. If the norm of the residual is greater than some predetermined value, $\bar{R}_{\max}$, then we restart the iterative process, with a reduced time step, Δt. This is necessitated by the fact that the Jacobians can be ill-conditioned if the solution is far from feasible, resulting in a divergent iteration. Reducing the time step had the effect of alleviating the ill-conditioning. Using $\bar{R}_{\max} = 12\|\bar{R}^0\|$ seemed adequate for our purposes. A factor of 12 is used because in some instances, the iterative process did start out by increasing the residual but managed to recover.

Note that the iteration shown in Figure 4.8 only involves one Jacobian evaluation and one nonlinear residual evaluation. The Jacobian undergoes one block LU factorization and the iterative loop only involves block matrix solves, and evaluation of the linearized residual, which simply requires block matrix multiplications and additions, and these are relatively cheap. Also, note that while computing the residual of the linearized state equation, we include Jacobian terms corresponding to the boundary conditions, $\hat{A}_g$.

It should be noted that the linearized state equation (4.15) used in these computations is inconsistent, because we compute the Jacobian of the residual using first order interpolation whereas third order interpolation is used in the evaluation of the residual.

Similarly, for the solution of the adjoint, consider a "pseudo" time dependent adjoint equation,

$$SM^T\frac{\partial \lambda}{\partial t} = -\left(\left(\frac{\partial R}{\partial q}\right)^T \lambda + \nabla_q J(z_k)\right) \equiv -\Lambda$$

where J is the objective. We use the approximate factorization algorithm used in ErICA to iterate this equation in time, until the residual of the adjoint equation, $\Lambda = 0$. The

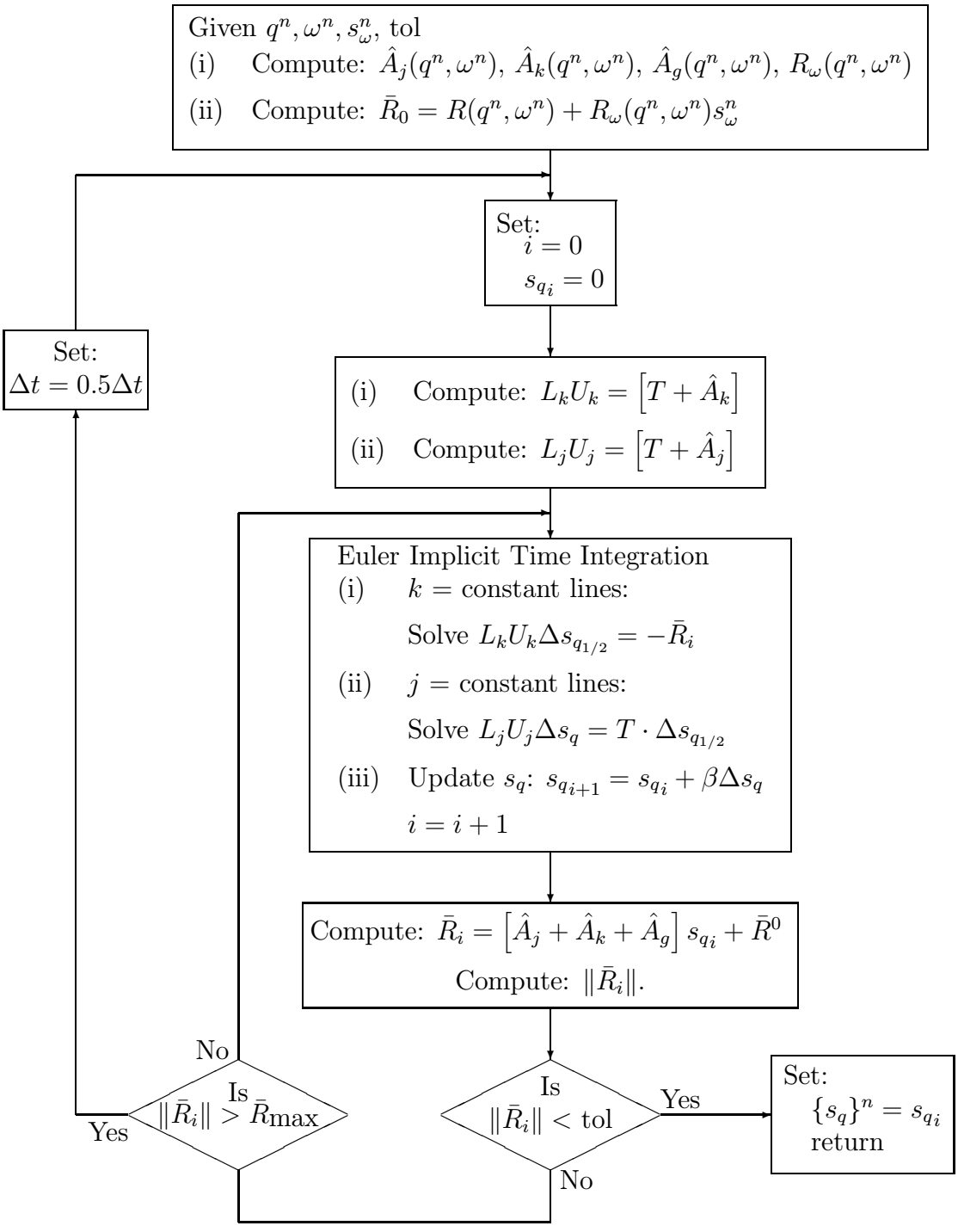


Figure 4.8: Solving the Linearized State Equation at kth step.

procedure is as follows. We have,

$$\left[T + \hat{A}_j + \hat{A}_k\right]^T \Delta\lambda = -\Lambda^n$$

where $\hat{A}_j$ and $\hat{A}_k$ are Jacobian terms. See Section A.4. The matrix of the left is factored approximately according to spatial directions

$$\left[T + \hat{A}_j\right]^T T^{-T} \left[T + \hat{A}_k\right]^T \Delta\lambda = -\Lambda^n$$

This system is solved using the sequence

$$\begin{aligned} \left[T + \hat{A}_j\right]^T \Delta\lambda_{1/2} &= -\Lambda^n \\ \left[T + \hat{A}_k\right]^T \Delta\lambda &= T^T \Delta\lambda_{1/2} \end{aligned}$$

$$\lambda^{n+1} = \lambda^n + \beta\Delta\lambda$$

The above iteration is performed until the residual of the adjoint equation, Λ, is reduced to zero. The algorithm is illustrated in Figure 4.9. Note, that while computing the residual of the adjoint equation, and the linearized state equation, we use (A.21), *i.e.*, we include the terms corresponding to the boundary conditions. Note that we could have just as well reversed the sequence above, *i.e.*, solve along $k =$ constant lines first and then solve along $j =$ constant lines. This would give us an algorithm which is exactly the same as that used in the linearized state algorithm, except the matrices would be transposed. However, philosophically speaking, the correct approach is to transpose the entire solution process, as is done above. Numerical experiments did show that the above algorithm was slightly superior in terms of convergence rate, and robustness, as compared to an algorithm with the sequence reversed.

Once again we have an outer loop monitoring divergence of the residual. We use $\Lambda_{\max} = 12\|\Lambda^0\|$. Numerical experiments showed that the computation of the adjoint was more susceptible to producing divergent results, and hence care has to be taken in choosing the value of the relaxation factor, β. We choose $\beta = \min(1.25, 1 - 0.12\log(10\|\Lambda^i\|))$, which has the desired effect of underrelaxing when the solution is crude, in order to reduce the possibility of divergence, and overrelaxing when the solution is refined in order to increase speed of convergence.

Also, note that we do not start with an estimate of $\lambda = 0$. Rather, we start from the previously computed estimate of the adjoint variable. Our experiments showed that this yielded significant savings in terms of the number of iterations. However, if the iteration proves to be divergent, then we reset λ to zero.

As with the procedure for the linearized state equation, this iteration only involves a single Jacobian evaluation. Note that we have introduced an inconsistency in the computation of the Jacobian, as before.

Note that the algorithms described are written in terms of the design weights ω and not in terms of the control variable w. We would like to distinguish between the two, for reasons which will be apparent in subsequent sections.

4.3.2 Computing Aerodynamic Forces

The aerodynamic forces are computed by numerically integrating the pressure over the surface of the airfoil. The normalized forces normal and tangential to the airfoil surface are respectively given by,

$$\begin{aligned} C_N &= - \frac{2}{\rho_\infty V_\infty} \sum_{j=j_o}^{j_f} p_{jk} \Delta x_{jk} \\ C_T &= \frac{2}{\rho_\infty V_\infty} \sum_{j=j_o}^{j_f} p_{jk} \Delta y_{jk} \end{aligned}$$

where $k = 1$ corresponds to the airfoil surface, with $j_o = N_c - N_{a/2}$ and $j_f = N_c + N_{a/2} - 1$. Refer to Section 4.2.2 for details regarding the domain discretization. We have

$$\begin{aligned} \Delta x_{jk} &= x_{j+1,k} - x_{j,k} \\ \Delta y_{jk} &= y_{j+1,k} - y_{j,k} \end{aligned}$$

We compute the lift and drag forces respectively as,

$$\begin{aligned} C_L &= C_N \cos\alpha - C_T \sin\alpha \\ C_D &= C_N \sin\alpha + C_T \cos\alpha \end{aligned} \tag{4.16}$$

The dependence of the lift and drag coefficients on the state q and the design ω can be determined from (4.16).

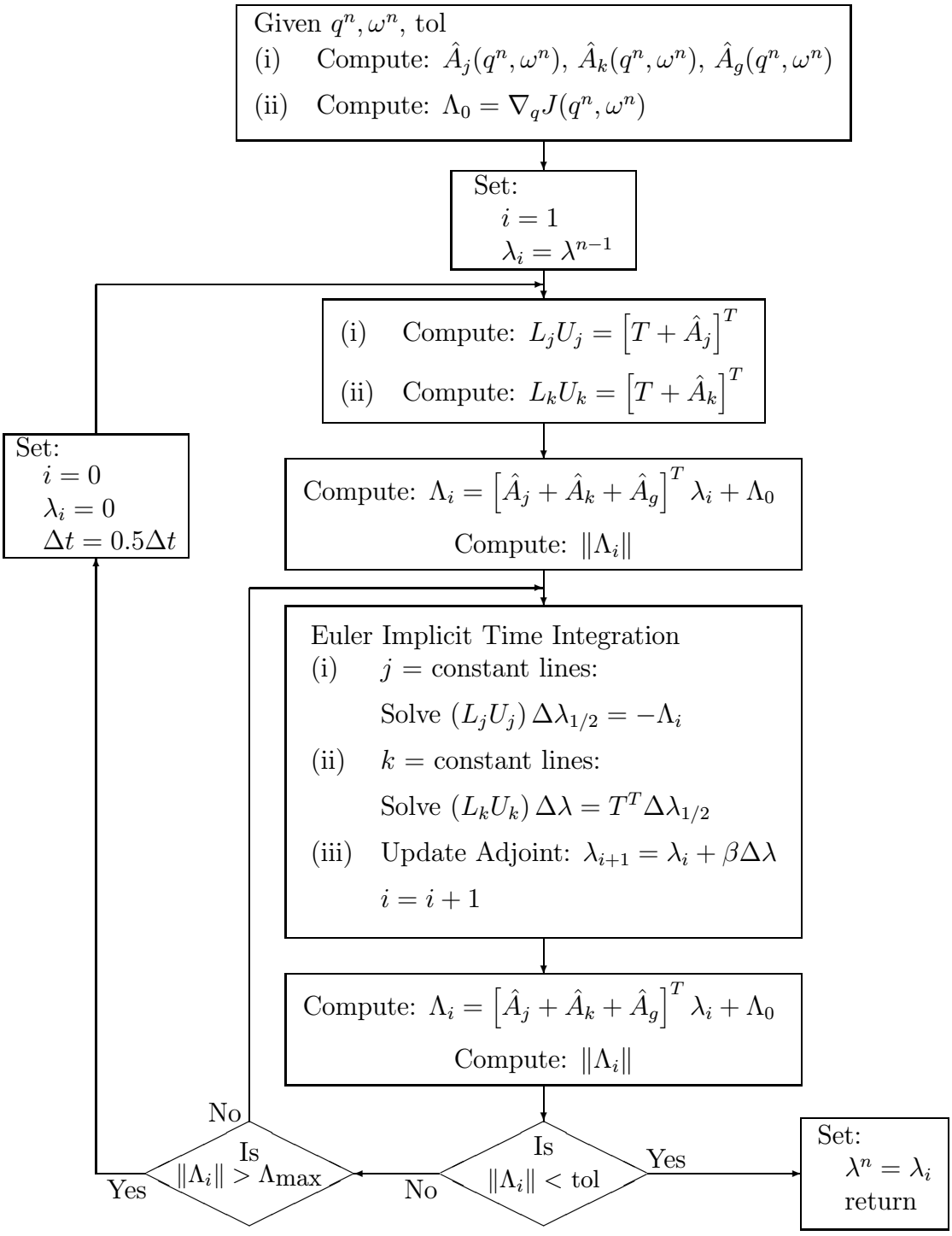


Figure 4.9: Solving the Adjoint Equation at kth step

4.3.3 Computation of the Area of the Airfoil

The area of the airfoil (for unit chordlength) is given by

$$\begin{aligned} S &= \int_0^1 y dx \\ &= \int_0^1 y_{\text{us}} dx - \int_0^1 y_{\text{ls}} dx \end{aligned}$$

Representing the airfoil in terms of the basic (closed) airfoils (4.10), we have

$$\begin{aligned} S &= \int_0^1 \sum_{i=1}^4 \omega_i \hat{y}_{i_{\text{us}}} dx - \int_0^1 \sum_{i=1}^4 \omega_i \hat{y}_{i_{\text{ls}}} dx \\ &= \sum_{i=1}^4 \omega_i \left(\int_0^1 \hat{y}_{i_{\text{us}}} dx - \int_0^1 \hat{y}_{i_{\text{ls}}} dx \right) \\ &= \sum_{i=1}^4 \omega_i S_i \end{aligned} \tag{4.17}$$

where S_i correspond to the areas of the individual airfoils. The areas of the individual airfoils can be computed at the beginning of the design cycle. For given ω the area is then computed as simply the weighted sum of the areas of the given (closed) airfoils.

4.3.4 Physical Compatibility

Our parametric representation of the airfoil (4.10) allows for situations where the upper surface can go below the lower surface of the airfoil. Such situations were actually encountered in our preliminary attempts at optimization. Hence, we need to impose an additional constraint to prevent such physically incompatible configurations to arise. This is done by constraining the trailing edge angle of the airfoil. The trailing edge angle is given by,

$$\delta_{\text{T}E} = \tan^{-1}(y'_{\text{ls}}(1)) - \tan^{-1}(y'_{\text{us}}(1))$$

which is approximated by

$$\delta_{\text{T}E} \approx y'_{\text{ls}}(1) - y'_{\text{us}}(1)$$

which yields

$$\delta_{\text{T}E} = \sum_{i=1}^4 \omega_i \delta_{\text{T}E_i} \tag{4.18}$$

that is, we can compute the trailing edge angle as simply the weighted sum of the (approximate) trailing edge angles of the four bases airfoils. It was found sufficient to impose the requirement

$$\delta_{\mathrm{T}E} \geq \delta_{\mathrm{m}in} \tag{4.19}$$

to ensure that the upper surface does not go below the lower surface. Note that while the above approximation of the trailing edge angle is fairly crude, it does yield a constraint that is easy to compute and achieves the desired effect.

4.3.5 Handling the Inequality Constraints and Reformulation of the Optimization Problem

To the author's knowledge, TRICE is the only SQP-based algorithm that allows for an interface with the flow-solving methodology of ErICA. At this point, however, TRICE, does not address general inequality constraints. We can only solve problems of the form

$$\min J(q, w)$$

such that

$$R(q, w) = 0$$
$$w_{\min} \leq w \leq w_{\max}$$

Hence we need to consider ways to improvise in order to be able to solve the specified design problem within the framework of the algorithm.

Reformulation of the Drag Constraint

One approach is to incorporate the inequality constraints on the drag by using slack variables. Consider, for example,

$$C_D(q, \omega) \leq C_{D_{max}}$$

We can formulate a constraint,

$$\mathcal{S}^2 = C_{D_{\max}} - C_d(q, \omega)$$

where $\mathcal{S}$ is a slack variable. For this constraint to be feasible, we must have

$$C_{D_{\max}} - C_d(q, \omega) \geq 0,$$

which satisfies our requirement. However, this does lead to a problem in that the SQP method leads to a feasible solution only at convergence. There would be no constrictions on these bounds being violated within the design process. Also, the addition of the slack variable constraint could cause some scaling problems, and overall, it does not yield good results.

An alternative procedure, described here, is to add a penalty function on the drag to the objective function. Consider the following function, illustrated in Figure 4.10,

$$P(z) = \begin{cases} 0 & z \leq 0 \\ z^2 & z > 0 \end{cases} \tag{4.20}$$

Define,

$$z = \mathcal{P} \cdot \left(\frac{C_D}{C_{D_{max}}} - 1 \right) \tag{4.21}$$

where $\mathcal{P}$ is a (scalar) penalty constant which can be used to increase emphasis on the drag violation. The addition of $P(z)$ to the objective function J has the desired effect of penalizing the objective when the drag constraint is violated. Note that this is a "soft" constraint, in the sense that the optimizer will allow the drag constraint to be violated as long as the penalty term added is not too large. This can be addressed to some extent by controlling the penalty factor $\mathcal{P}$.

Constraints on the Area and Trailing Edge Angle

The remaining constraints can be addressed using a mapping between the control variable, w, and the design weights, ω. Rather than use the design weights as our control variables, we use the area of the airfoil and its trailing edge angle as control variables, as shown below.

In Section 4.3.1 we had stressed on the fact that we wish to distinguish between the control variables w and the design weights ω in the airfoil representation (4.10). We would like to represent the weights as functions of the control variables. This enables us to address the issue of ensuring that the lower bound on the area and the trailing edge angle remain strictly enforced, by making use of the fact that we can place bounds on the control variables. We use, as our control variables,

$$w = W \begin{Bmatrix} 2 \cdot (\omega_1 - \omega_3) \\ 2 \cdot (\omega_2 - \omega_4) \\ 2 \cdot \bar{S} \\ 0.5 \cdot \bar{\delta}_{TE} \end{Bmatrix} \tag{4.22}$$

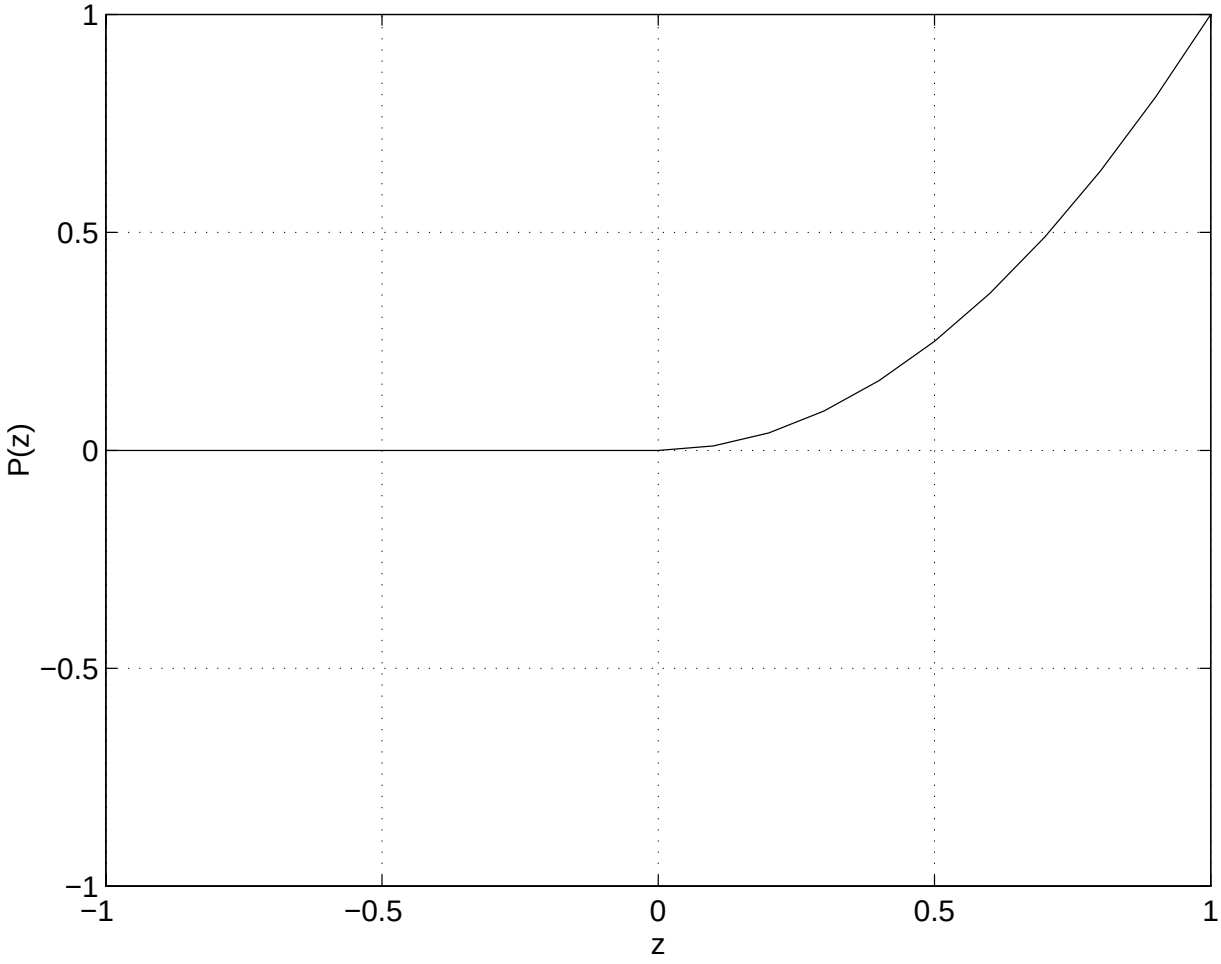


Figure 4.10: Proposed Penalty Function

where

$$\bar{S} = \frac{S}{S_{min}}; \qquad \bar{\delta}_{TE} = \frac{\delta_{TE}}{\delta_{min}},$$

and the scalar factor W has been added in order to be able to experiment with the scaling. Note that this results in a control space that is simply a result of a linear combination of the design weights, as should be evident from equations (4.17) and (4.18). Now, enforcing bounds

$$2W \leq w_3 \leq 2W \cdot \frac{S_{max}}{S_{min}}$$

strictly enforces the bounds on the area of the airfoil (4.9). Similarly, the bound,

$$0.5W \leq w_4$$

imposes the constraint (4.19). The mapping (4.22) can be used to translate between w and ω. Note that the above mapping was chosen so as to yield a low condition number for the transformation matrix providing the mapping between the weights and the controls. This was done with the philosophy that a change in the controls should produce roughly the same amount of change in the weights. This choice did yield improved performance in the optimization algorithm.

Reformulated Problem

The original design problem in Section 4.1 is now recast as

$$\min_{q,w} J(q,w) = -C_L(q,w) + P(\mathcal{P} \cdot (\tfrac{C_D(q,w)}{C_{D_{max}}} - 1))$$

such that

$$R(q,w) = 0$$
$$w_{\min} \leq w \leq w_{\max}$$

Note that for the linearized state equation, we have

$$R_\omega s_\omega = R_w s_w$$

As long as we apply the mapping (4.22) consistently, we do not need to make any changes to the solution algorithms described in Section 4.3.1. However, in computing the reduced gradient,

$$g = \nabla_w J(z) + R_w(z)^T \lambda$$

we must compute,

$$\nabla_w J(z) = \nabla_\omega J(z) \cdot \omega_w$$

and

$$R_w(z) = R_\omega(z) \cdot \omega_w$$

4.4 Numerical Results and Discussion

Numerical results were obtained using the TRICE algorithm. We choose the (closed) NACA 2412 airfoil as our baseline design and a grid convergence study was initially performed for this airfoil. We used five different grid dimensions, namely

1. 51×14 grid with 31 points on the airfoil surface.
2. 101×27 grid with 61 points on the airfoil surface.
3. 151×40 grid with 91 points on the airfoil surface.
4. 201×53 grid with 121 points on the airfoil surface.
5. 301×79 grid with 181 points on the airfoil surface.

The results are shown in Figure 4.11. As can be seen for the coarser grids the analysis code, ErICA, overestimates the drag, C_D, and underestimates the lift, C_L. Based on these results, we develop a scheme to solve the design problem (4.6)–(4.9) by sequentially solving the problem first on a coarse grid, and then progressively refining the grid. We start out with loose bound on the drag, which is tightened gradually as we refine the grid. We use $\mathcal{P} = 10$ in (4.21). The optimization is started with feasible initial data for the given geometric configuration. The criterion for optimality was that the norm of the reduced gradient be reduced to a tolerance of 10^{-2}. Note that the optimizer struggled to diminish the reduced gradient much below this. For the 51×14 grid, a scale factor of $W = 1$ was used, which produced results that had reduced the norm of the reduced gradient to about 1.5×10^{-2}, at which point the iteration was terminated due to lack of progress. The algorithm was implemented so that if the optimization stalled, *i.e.*, the algorithm terminated because it failed to make further progress, we restart the optimization with a feasible solution for the given configuration. Several restarts were required for the 51×14 grid. In particular, the algorithm struggled if we move too far away from feasibility, and it would be worthwhile trying to prevent the optimization process from allowing this to happen. For the other grids we used a scale factor of $W = 10$ (*cf.* (4.22)). The optimization converged with a single restart in these cases. The results are shown in Table 4.1. Note that we did not run the optimizer for the final grid, as we had a nearly converged solution. We use $S_{\mathrm{min}} = 0.075$ and $S_{\mathrm{min}} = 0.15$. The area of the optimized airfoil is at the lower bound, as one might expect.

Figure 4.12 shows the final results obtained for the 301×79 grid. Note that the shock has moved downstream. The optimization does manage to improve the performance of

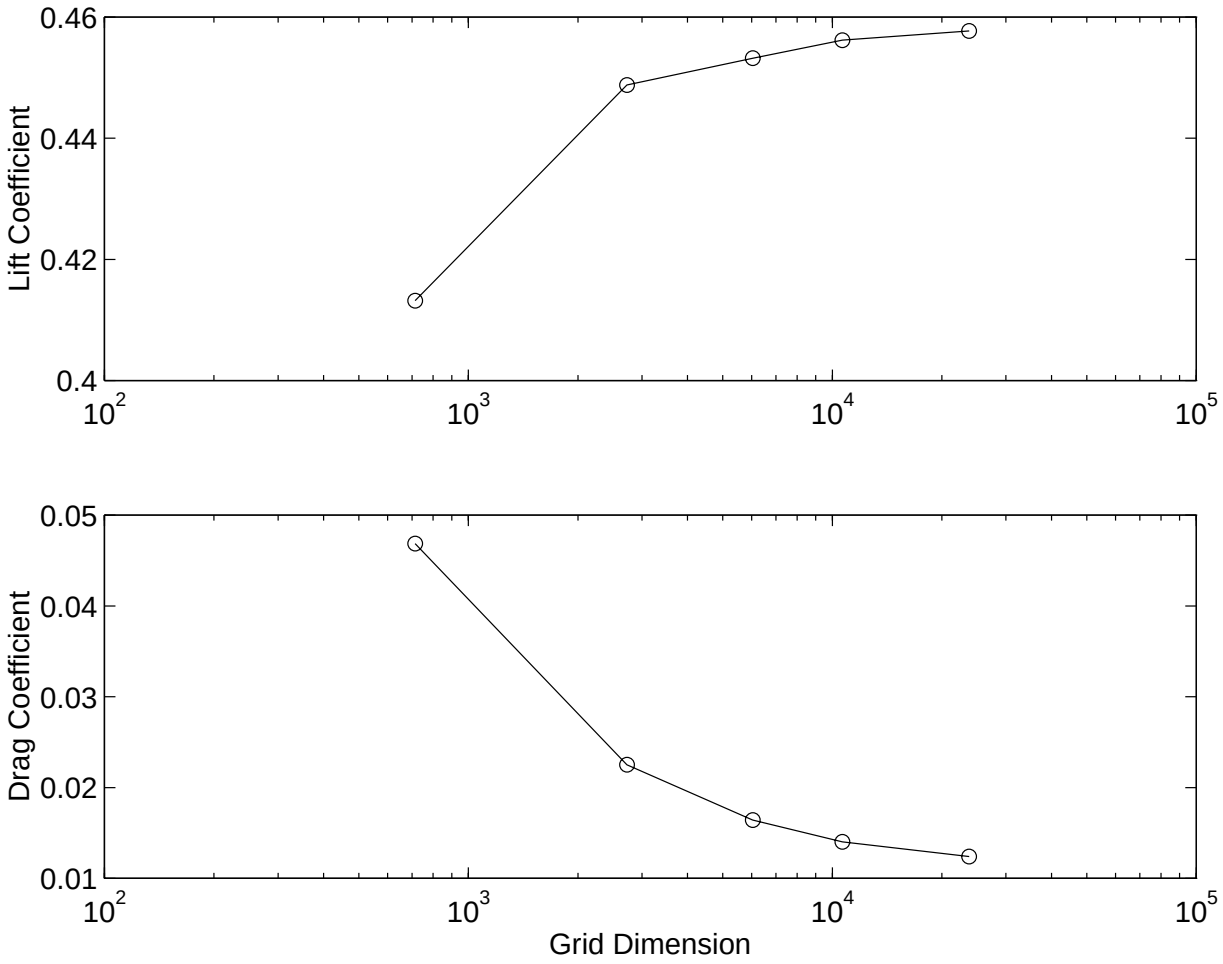


Figure 4.11: Grid Convergence Study for NACA 2412

Table 4.1: Numerical Results for Airfoil Design Problem

Grid Size	$C_{D_{max}}$	Data	ω_1	ω_2	ω_3	ω_4	C_L	C_D	S
51×14	0.04	Initial	1.0000	0.0000	0.0000	0.0000	0.4132	0.0469	0.0823
		Final	0.2538	0.1793	-0.1369	0.5793	0.5263	0.0403	0.0770
101×27	0.02	Initial	0.2538	0.1793	-0.1369	0.5793	0.5722	0.0251	0.0770
		Final	0.2812	0.1049	-0.0760	0.5314	0.5227	0.0201	0.0750
151×40	0.014	Initial	0.2812	0.1049	-0.0760	0.5314	0.5266	0.0159	0.0750
		Final	0.3059	0.0593	-0.0313	0.5006	0.5037	0.0142	0.0750
201×53	0.012	Initial	0.3059	0.0593	-0.0313	0.5006	0.5044	0.0126	0.0750
		Final	0.3153	0.0429	-0.0154	0.4893	0.4955	0.0120	0.0750
301×79	0.010		0.3153	0.0429	-0.0154	0.4893	0.4959	0.0103	0.0750

the airfoil. As compared to the baseline NACA 2412 airfoil, which had a lift coefficient, $C_L = 0.4577$ with $C_D = 0.01241$, the final design has $C_L = 0.4959$ and $C_D = 0.01028$, which means the lift coefficient has increased by approximately 8.5%, while the drag coefficient has been reduced by 17%. The pressure contours obtained for the given conditions for the optimized airfoil are shown in Figure 4.13

Table 4.2 gives an account of the computational effort required at each step to produce a "converged" solution. Here the number of "successful" iterations corresponds to those iterates where the computed step is accepted by the Trust Region algorithm, whereas the total iterations includes in addition those iterates which are rejected. It should be noted that most of the computational effort (in terms of number of iterations) occurs at the coarsest level, where the computations are fairly cheap. Though there is room for improvement the TRICE algorithm is relatively efficient in finding the given solutions. Consider, for example, the computational effort required for the 51×14 grid. We require 3786 residual evaluations, 5004 Jacobian evaluations, 10008 block LU factorizations and 454948 block matrix solves. Compare this to the effort required to obtain a single analysis solution. We require approximately 1000 Euler Implicit time integration steps to produce a converged solution which would mean 1000 residual evaluations, 1000 Jacobian evaluations, 2000 block LU factorizations and 2000 block matrix solves. Discounting the discrepancy in the number of solves, the computational effort required by TRICE is roughly equal to the effort required to perform 4–5 flow analyses, which is very cheap. It should be noted that though we require a large number of block matrix solves, this just involves forward and backward substitutions which is fairly cheap. Note that computational efficiency was not a prime concern while writing the interface subroutines between TRICE and ErICA. For example, we recompute the Jacobian information everytime that we require to solve the linearized state equation or the adjoint equation. A more efficient implementation would require only one evaluation to suffice for both the adjoint equation and the linearized state equation. Similar instances of implementing the code in a more efficient manner should yield significant savings from the elimination of redundant computations.

4.4.1 Computational Issues

One possible reason for the poor convergence behavior of the TRICE algorithm is that we do not compute the Jacobian of the state constraints consistently. To be precise, the residual of the governing equations is computed using a third order interpolation of the fluxes. The true Jacobian corresponding to this residual evaluation would have block pentadiagonal structure along the $j =$ constant and $k =$ constant lines. The Jacobian that is computed in ErICA, which is the one we use, is based on first order interpolation of the fluxes and yields a block tridiagonal structure, which one can invert more efficiently. Since the Jacobian we

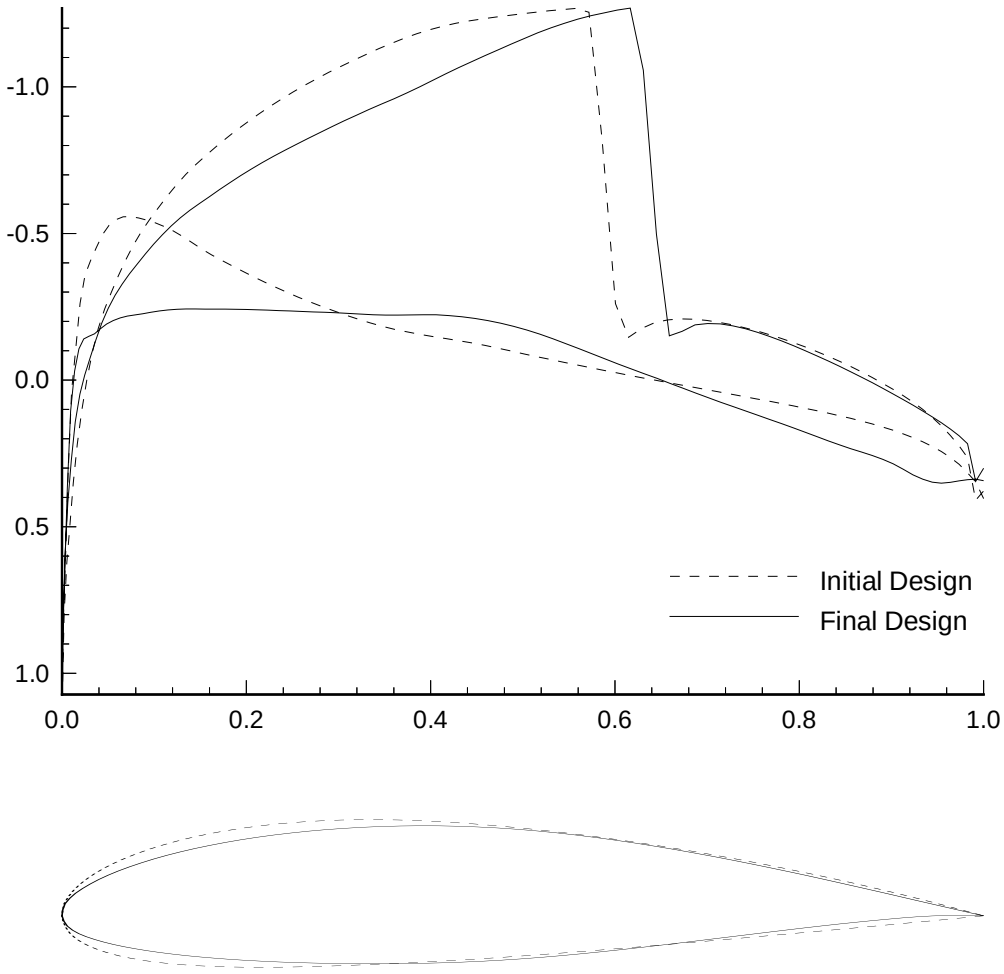


Figure 4.12: Results obtained using TRICE for Airfoil Design Problem

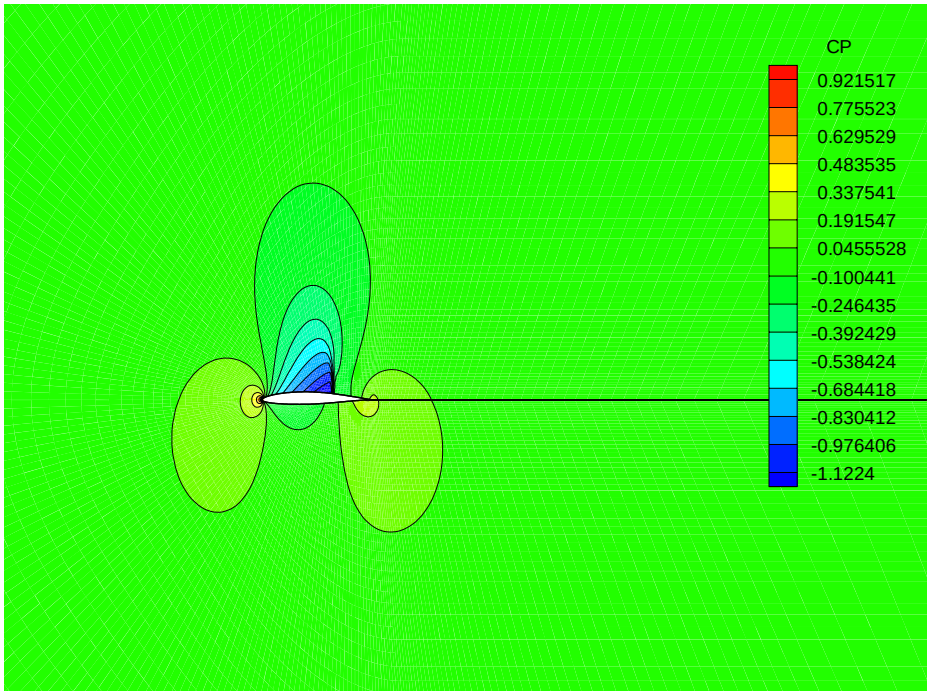


Figure 4.13: Pressure Contours for Optimized Airfoil

Table 4.2: Computational History for Airfoil Design Problem using TRICE

Grid Size	Whether Converged	Total Iterations	Successful Iterations	Number of Restarts	# Residual Evaluations	# Jacobian Evaluations	# LU Factorizations	# Solves
51×14	No	388	211	12	3786	5004	10008	454948
101×27	Yes	44	26	1	313	437	874	73588
151×40	Yes	27	11	1	506	518	1036	81953
201×53	Yes	19	5	1	386	425	850	137768

are using is inexact, the resulting numerical errors might corrupt the solution process. In particular, the solution of the adjoint equation in this case leads to an "inexact" projected gradient of the Lagrangian. See Section 1.1. The computation of the full Jacobian is further complicated by the need to differentiate the limiters, described in Section A.3.1. This matter should be investigated in depth in future research.

Also it should be noted that no literature seems to be available for existence or uniqueness results for the Euler equations. As a matter of fact Jameson [85] has shown the existence of cases where nonunique solutions of the Euler equations can be obtained for certain airfoils. In contrast to the 1-D nozzle problem of Chapter 2, in the present case we are unable to guarantee existence of solutions for the linearized state equations.

Another possible reason for the poor behavior is the fact that the simple algebraic grid that we use may be ill-conditioned. Since the primary purpose of the present study was to demonstrate the concept we have developed for solving the design problem, we have not investigated the effect of the grid on the solution process. As a result, no attempt has been made to ascertain the quality of the grid. It was, in fact, observed that the convergence behavior exhibited by the flow solver deteriorates as the grid is refined, which could certainly be an indication of ill-conditioning. It should be noted that several grid generation techniques exist, both algebraic and elliptic, and there is a vast literature available on these techniques and the evaluation of grid quality [44, 133, 134, 9, 32, 92, 119]. This is certainly an aspect which should be closely studied in future work. It may be worthwhile to pursue more complicated grid generation techniques since the overall savings that might be obtained from having better grid quality may more than make up for an extra computational effort required to compute the grid. Also, in light of the fact that the original formulation of the airfoil representation by Vanderplaats [149], Joh *et al.* [90] and Narducci [107] does not yield consistent grid sensitivities, it might be worthwhile to pursue an alternate formulation such as using Bezier curves to represent the airfoil surface [120] which should provide more flexibility than the formulation used in the present study. The scale factor W also seems to be a critical issue and should be studied more closely.

A high percentage of the number of block matrix solves required to solve the design problem (refer Table 4.2) can be attributed to the large number of iterations required to obtain a converged solution for the adjoint equation. This is especially true for the finer grids, which may be caused by ill-conditioning in the grid. The high number of solves can also be partly attributed to the penalty function approach we use to address the drag constraint (4.20), (4.21). When the drag constraint is violated the gradient of the objective function with respect to the state variables becomes very large, which in turn means that the residual of the adjoint equation is very large and requires a large number of iterations to converge. One way to reduce this effect might be to drive the value of the drag further down at the coarsest levels before proceeding to refine the grid. Though this might make the problem

at the coarse level more difficult to solve, the iterations are far cheaper at this level, and could lead to significant improvements in overall computational effort by reducing the number of solves required at the finer grid levels. It also might be worthwhile to consider alternate methods of computing solutions to the adjoint equation. The penalty function approach also has the effect of causing a large number of unsuccessful iterations. This is because the quadratic model built at a given iterate would not possess gradient information corresponding to the effect of the drag, unless the drag constraint is violated. Thus, when we violate the drag constraint, the model fails because of an abrupt change in the objective function, which the model is unable to predict. An alternate algorithm being developed within TRICE would also enable us to use inequality constraints on the state variables which should alleviate this problem.

Also, it is the author's opinion that we need to improve the technique used to update the penalty parameter, ϱ, used in the computation of the merit function (*cf.* 1.10). Specifically, the monotonic scheme used can slow down the convergence, once ϱ is increased to a large number, as the resulting emphasis on feasibility can inhibit the march to optimality. Often one finds that steps that seem reasonable to the observer are rejected by the optimizer because ϱ is too small or too large. This problem might be connected to the problems associated with the computation of an inexact Jacobian, as discussed earlier. In the present scheme the penalty parameter ϱ is updated so as to ensure that the decrease predicted in the quadratic model subproblem is positive. A scheme which is predicated on trying to achieve a positive reduction in the actual nonlinear problem might be more viable, especially if used in conjunction with a nonmonotonic scheme like the one proposed by El-Alem [46]. The effect of ϱ on the solution process is important and this needs to be studied. In this regard it should be noted that the algorithm, TRICE, is still in the developmental stages, and one expects further improvements in the algorithm in the future.

It should be kept in mind while reviewing these results that TRICE is a work in progress, and further improvements in the algorithm should yield better results.

4.4.2 Comparison of Results

Table 4.3 compares the results obtained in the present study with those obtained by Joh *et al.* [90] and Narducci *et al.* [106, 107]. As is evident, the optimal results obtained in the current study are not as good as those obtained by Joh *et al.* [90] and Narducci *et al.* [106, 107]. Here, one should note that the flow solver used by Joh is different than the one used by Narducci and in this present study. Also, we do not use the curvature correction in imposing flow tangency at the airfoil surface (see Section A.3.2) as a simplification. Our studies showed that adding the curvature corrections yielded much better results. Also, the

grid used in the present study is fairly crude, especially when compared to the elliptic grids obtained in the previous studies, using GRIDGEN [138]. However, one should note that the implementation used here is slightly different from the one used by Joh *et al.* [90] and Narducci *et al.* [106, 107]. In our treatment, the upper and lower surfaces are treated as separate entitities for the bases airfoils, and the airfoil upper and lower surfaces are obtained as linear combinations of the upper and lower surfaces of the bases airfoils, respectively. Thus, we have,

$$\begin{aligned} y_{\mathrm{us}}(x/c) &= \sum_{i=1}^{6} \omega_i y_{\mathrm{us}_i}(x/c) \\ y_{\mathrm{ls}}(x/c) &= \sum_{i=1}^{6} \omega_i y_{\mathrm{ls}_i}(x/c) \end{aligned}$$

This is not so in the case of the formulation used by Joh *et al.* [90] and Narducci [107]. The distinction between the upper and lower surfaces in their implementation depends upon the weights, ω_i. To understand this more clearly, consider the following example. An airfoil described by $\omega = [-1 \quad 0 \quad 0 \quad 0]$ would correspond to an inverted NACA 2412 airfoil, that is, they would have

$$\begin{aligned} y_{\mathrm{us}}(x/c) &= y_{\mathrm{ls}_1}(x/c) \\ y_{\mathrm{ls}}(x/c) &= y_{\mathrm{us}_1}(x/c) \end{aligned}$$

However, in our formulation, the above combination would yield,

$$\begin{aligned} y_{\mathrm{us}}(x/c) &= -y_{\mathrm{us}_1}(x/c) \\ y_{\mathrm{ls}}(x/c) &= -y_{\mathrm{ls}_1}(x/c) \end{aligned}$$

While the shape obtained is the same, our interpretation yields a configuration where the lower surface is above the upper surface, which is not a valid configuration and is disallowed, because it violates the constraint on the trailing edge angle (4.19).

Thus, the formulation used by Joh *et al.* [90] and Narducci [107] is more general in nature. Note, however, that the "absolute value" nature of their description means that the design sensitivities $\frac{\partial y}{\partial \omega_i}$ are not defined at $\omega_i = 0$. Hence, we prefer to use the current formulation because it yields meaningful derivatives. Note that this provides for a more restricted design space, and we are unable to obtain as much flexibility in terms of shapes for the airfoil as with their formulation. In this context, it might be worthwhile to pursue alternate formulations, such as using Chebychev polynomials [150] or Bezier curves [120] to represent the airfoil surface which should provide more flexibility than the formulation used in the present study.

Table 4.3: Comparison of Results

Method	C_L	C_D
Joh	0.7144	0.0100
Narducci	0.6207	0.0102
Present	0.4959	0.0103

4.5 Remarks

Though several issues remain to be resolved, the preliminary results obtained for the airfoil design are encouraging and merit further study. Of particular interest is the way the flow solver ErICA is interfaced with the optimizer, TRICE. The solution refinement methodology used, wherein we use optimized results obtained on coarser discretizations to obtain good starting points for finer discretizations is also of interest as it can provide significant savings in terms of overall computational effort.

Chapter 5

Concluding Remarks

In the current research, we have shown how the use of optimization which exploit the structure of a particular design problem can yield substantial savings in terms of the computational effort required to solve the problem.

In Chapter 2 we have solved a shape design problem for a nozzle housing one-dimensional transonic fluid flow governed by the Euler equations. The design problem has been solved using two approaches. The first one, a shock-capturing technique yields constraints that are not smooth. When this method is used in conjunction with the reduced Hessian SQP algorithm, TRICE, described in Section 1.1, we find that the behavior is not very robust. This is because the SQP method is based on the assumption of sufficiently smooth constraints which does not hold true for the case when we use a shock-capturing scheme. Our numerical studies, however, did show that when the SQP algorithm does converge it is far superior in terms of computational efficiency as compared to a conventional black-box method. The problem of lack of smoothness can be addressed by treating the shock location as an explicit variable, which amounts to a shock-fitting method. We can prove existence and uniqueness of solutions to the design and analysis problem for the shock-fitting scheme. The algorithm behaves much more robustly in this case, producing converged results even for very poor initial guesses. A careful analysis of the underlying infinite-dimensional problem provided significant insight into the behavior of the discretized problem. Making the discrete problem consistent with the infinite-dimensional problem by incorporating weighted scalar products yielded significant savings and also improved the quality of the design solutions produced by the optimization.

In Chapter 3 we use a different approach to solving the design problem, *i.e.*, we try to exploit the sparse structure that exists for a temperature control problem, using an SQP

scheme based on sparsity, SNLP. The problem was shown to be a perfect QP problem, which is solved very efficiently by the algorithm. Several variants of the problem were also tried. However, poor results were obtained when we attempted to use the optimization algorithm to solve some of these problems. A careful analysis showed that the cases where the optimizer did not perform well corresponded to cases where the underlying infinite-dimensional problem was not well posed.

Chapter 4 addresses a transonic airfoil design problem involving two-dimensional Euler flow. Here, we incorporate the reduced Hessian formulation by developing a scheme which is based on the implicit Euler time integration method using approximate factorization to solve the linearized state constraints and the adjoint equations. The advantage of using the above scheme is that we only need to "solve" the constraints to within a specified tolerance. When we are far from the solution, for instance, a crude solution of the constraints will suffice. The CFL number used in the scheme can be monitored to prevent divergent solutions. While the initial results obtained are promising, several issues need to be resolved including the effect of the inconsistency resulting from the computation of an inexact Jacobian on the overall behavior of the optimization scheme, as well as certain aspects of the optimization algorithm itself. This is an ongoing process.

A particularly attractive aspect of the proposed scheme is that the approximate factorization algorithm is highly amenable to parallelization, and algorithms have been developed that can solve this problem very efficiently on distributed memory architectures [65, 94]. In the present context these parallel algorithms can be incorporated very easily within the optimization algorithm which should yield significant savings. Also, the method can easily be expanded to a three-dimensional problem.

The use of all-at-once methods like the ones described in this research can lead to highly efficient solution finding for design problems. However, the methods do not always exhibit robust behavior, and naive applications of these methods will likely fail. An underlying theme in the current research is that discrete problems often inherit the nature of the underlying infinite-dimensional problem. A careful analysis can yield significant insight into the expected nature of solutions, which in turn can enable the successful application of all-at-once methods.

In ending the author would like to state his belief that the design process must be integrated with other disciplines to yield more efficient streamlining in product development in the industry. The author hopes that the present research is a step in that direction.

Appendix A

Flow Solutions for the 2D Euler Equations

The following describes some of the computational procedures implemented in the solver ErICA (EuleR Inviscid Code for Aerodynamics) to obtain approximate solutions for 2–D flow problems governed by the Euler equations [107]. We are specifically interested in the steady state flow around an airfoil. The description provided only addresses the relevant areas regarding the implicit time integration scheme used. For more details, see [107].

A.1 Governing Equations

The flow, q, around the airfoil is governed by the Euler equations for a perfect gas, which may be written in integral conservation law form in Cartesian coordinates as

$$\frac{\partial}{\partial t} \iint_{\Omega} Q \, dS + \oint_{d\Omega} \hat{F} \cdot \hat{n} \, ds = 0 \tag{A.1}$$

where

$$\hat{F} = F\hat{j} + G\hat{k}$$

and

$$Q = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_o \end{Bmatrix}, \; F = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho h_o) u \end{Bmatrix}, \; G = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho h_o) v \end{Bmatrix}$$

with velocity components u, v, density ρ, total energy per unit mass $e_o = e + (u^2 + v^2)/2$, with e being the internal energy per unit mass and pressure p, which for a perfect gas may be expressed by the relation, $p = (\gamma - 1)\rho e$. The total enthalpy per unit mass is given by $h_o = \frac{a^2}{\gamma - 1} + \frac{u^2}{2} + \frac{v^2}{2} = e_o + \frac{p}{\rho}$, where $a = \sqrt{\frac{\gamma p}{\rho}}$ is the sonic velocity. Here, Q represents the conserved variables with $q = [\rho \ u \ v \ p]^T$ denoting the primitive variables, and, F and G represent the inviscid fluxes. The problem domain is denoted by Ω and $d\Omega$ represents the boundary of the domain. For a detailed treatment of the Euler equations refer [43].

A.2 Problem Discretization

The solution of the above is obtained on a discrete computational grid. In this case, we use a C–type grid which is wrapped around the airfoil. An illustrative example is given in the form of a simple (8×3) C–type grid, for a diamond shaped airfoil, in figure A.1. Consider the (j, k)th cell, as shown in figure A.2.

We use a cell-centered finite volume formulation to rewrite the governing equations in (semi-)discrete form. In a finite volume setting, the computational domain is divided into several sub-domains. The state variables are assumed to be constant, within each sub-domain. Within a given cell, in terms of local coordinates, the Euler equations are approximated by,

$$\frac{\partial (QS)_{j,k}}{\partial t} + \sum_{sides} (\hat{F} \cdot \hat{n}) \Delta s = 0 \tag{A.2}$$

where S_{jk} is the area of the (j, k)th cell, and the unit normal, $\hat{n}$, is positive outward.

A.3 Computation of the Residual

For the (j, k)th cell (subdomain), the (semi-discretized) residual is given by,

$$S_{jk} \frac{\partial Q_{jk}}{\partial t} = - \sum_{sides} (\hat{F} \cdot \hat{n}) \Delta s \equiv -R_{jk}(Q) \tag{A.3}$$

The residual can be computed as,

$$R_{jk}(Q) = \left\{ \left[(\hat{F} \cdot \hat{n}) \Delta s \right]_{j-1/2} + \left[(\hat{F} \cdot \hat{n}) \Delta s \right]_{j+1/2} + \left[(\hat{F} \cdot \hat{n}) \Delta s \right]_{k-1/2} + \left[(\hat{F} \cdot \hat{n}) \Delta s \right]_{k+1/2} \right\}$$

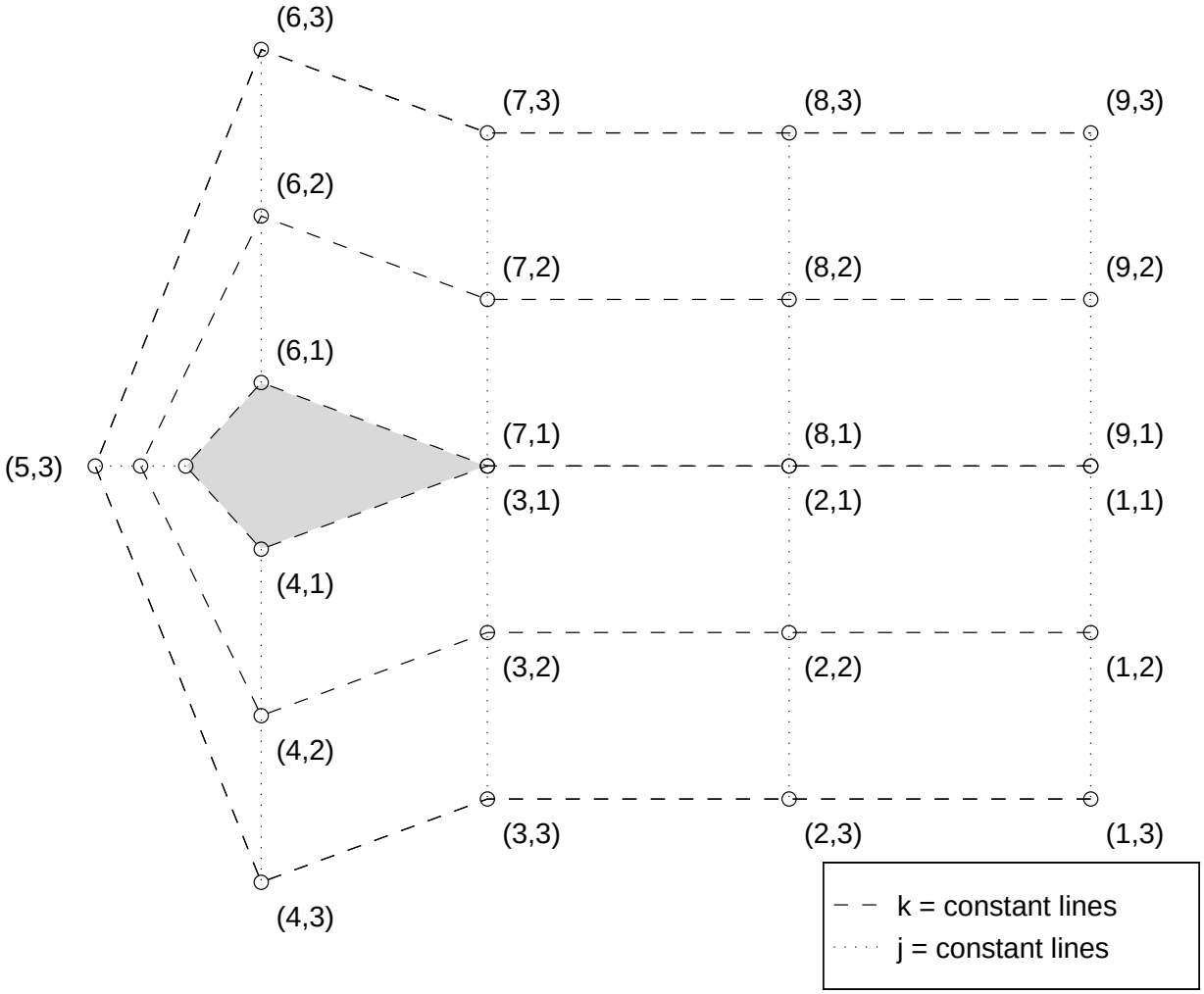


Figure A.1: An Illustrative Example of a C–grid

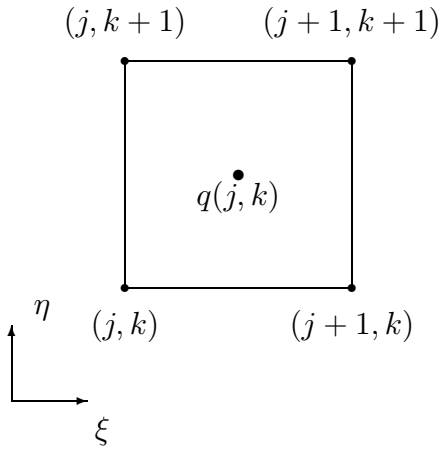


Figure A.2: Computational Grid: (j, k)th cell

where, $\hat{F}_{j+1/2}$ corresponds to the inviscid flux, $\hat{F}$, along the vertical cell face corresponding to index $(j+1)$, $\hat{F}_{j-1/2}$ corresponds to the flux along the vertical cell face corresponding to index j, $\hat{F}_{k+1/2}$ corresponds to the flux along the horizontal cell face corresponding to index $(k+1)$, and $\hat{F}_{k-1/2}$ corresponds to the flux along the horizontal cell face corresponding to index k. For a given cell face, we have

$$\hat{F} \cdot \hat{n} = \left\{ \begin{array}{c} \rho U \\ \rho u U + \hat{n}_x p \\ \rho U v + \hat{n}_y p \\ (\rho h_0) U \end{array} \right\},$$

where $U(= \hat{n}_x u + \hat{n}_y v)$ is the velocity normal to the cell face, and $\hat{n}_x$ and $\hat{n}_y$ are the cartesian components of the normal to the cell face.

A.3.1 Upwind Differencing

Upwind schemes originate from taking into account the physical properties of the flow equations in the discretized formulation. Information propogates along the characteristic lines of the governing system of equations. Upwind schemes ensure that information is acquired in the direction from which the characteristic is traveling, and this enhances the stability of the algorithm. ErICA provides the option of using either Van Leer's flux-vector splitting method or Roe's flux difference splitting method. While flux difference splitting yields more accurate results, the flux-vector splitting approach is more robust, and provides sufficiently accurate results for the airfoil design problem.

Spatial Accuracy

The computation of the discretized residual requires computing the fluxes at the cell faces. In an upwind scheme, where information is obtained from the direction of the characteristics, this requires computing the fluxes twice; once for left running characteristics and once for right running characteristics. Information from right running characteristics are stored as left-sided quantities as the information comes from the left. Likewise, right-sided quantities store information from left running characteristics. In CFD, fluxes at the cell faces may be computed in one of two ways. The flux quantities can be interpolated from computed values at cell centers, or the state variables may be interpolated from cell-centers to the cell faces, and fluxes computed from these values. The latter is called MUSCL (Monotone Upstream-centered Scheme for Conservation Laws) differencing, and is the method used in

ErICA. Up to third order accurate interpolation is available. The state variables, at the cell centers, are interpolated to the cell faces using,

$$\begin{aligned} q^{-}_{j+1/2} &= q_{jk} + \frac{\phi}{4}\left[(1-\kappa)\Delta^{-}_{\xi} + (1+\kappa)\Delta^{+}_{\xi}\right] q_{jk} \\ q^{+}_{j-1/2} &= q_{jk} - \frac{\phi}{4}\left[(1-\kappa)\Delta^{+}_{\xi} + (1+\kappa)\Delta^{-}_{\xi}\right] q_{jk} \end{aligned}$$

and

$$\begin{aligned} q^{-}_{k+1/2} &= q_{jk} + \frac{\phi}{4}\left[(1-\kappa)\Delta^{-}_{\eta} + (1+\kappa)\Delta^{+}_{\eta}\right] q_{jk} \\ q^{+}_{k-1/2} &= q_{jk} - \frac{\phi}{4}\left[(1-\kappa)\Delta^{+}_{\eta} + (1+\kappa)\Delta^{-}_{\eta}\right] q_{jk} \end{aligned}$$

and so on, where, $q^{-}_{j+1/2}$ corresponds to the left value of the state at cell face $(j+1)$, $q^{+}_{j-1/2}$ corresponds to the right value of the state at cell face j, $q^{-}_{k+1/2}$ corresponds to the lower value of the state at cell face $(k+1)$, $q^{+}_{k-1/2}$ corresponds to the upper value of the state at cell face k, etc, and

$$\begin{aligned} \Delta^{+}_{\xi} q_{jk} &= q_{j+1,k} - q_{jk} \\ \Delta^{-}_{\xi} q_{jk} &= q_{jk} - q_{j-1,k} \\ \Delta^{+}_{\eta} q_{jk} &= q_{j,k+1} - q_{jk} \\ \Delta^{-}_{\eta} q_{jk} &= q_{jk} - q_{j,k-1} \end{aligned} \tag{A.4}$$

respectively. Note that for $\phi = 0$, we have first order interpolation. The gradients listed above in (A.4) can be limited using limiters, such as *Van Albada's* limiter, or the *minmod* limiter. These have the effect of suppressing numerical oscillations in the computed solution, which can occur near regions of steep gradients for higher order interpolations. The minmod limiter modifies the gradients appearing in the above interpolations using the minmod function

$$\begin{aligned} \bar{\Delta} &= \text{minmod}[\Delta, \beta\nabla] \\ \bar{\nabla} &= \text{minmod}[\nabla, \beta\Delta] \end{aligned}$$

where the minmod function is defined by

$$\text{minmod}(a,b) = \begin{cases} 0 & \text{if } \; ab < 0 \\ \text{sign}(a)\min(|a|,|b|) & \text{if } \; ab > 0 \end{cases}$$

Here, β is a variable related to the order of interpolation and is defined by

$$\beta = \frac{3-\kappa}{1-\kappa}$$

Characteristic variables tend not to vary as much as the primitive variables. ErICA has the option to limit these variables instead of the primitive variables. The Van Albada limiter uses

$$\begin{aligned} \tilde{\phi} &= s\phi \\ \tilde{\kappa} &= s\kappa \end{aligned}$$

in the above interpolation formulae, where factor s is given by

$$s = \frac{2\nabla q_{jk}\Delta q_{jk} + \epsilon}{(\nabla q_{jk})^2 + (\Delta q_{jk})^2 + \epsilon}$$

ErICA uses ϵ on the order of Δx^2 or Δy^2.

Van Leer Flux Vector Splitting

The information about the propogative characteristics can be introduced into the discretization process based on the sign of the eigenvalues, whereby the flux terms are split and discretized directionally according to the sign of the associated propogative speeds. This leads to flux vector splitting. Consider, as an example, the one–dimensional semi–discrete form

$$\frac{\partial Q}{\partial t} = -\frac{F_{j+1/2} - F_{j-1/2}}{\Delta x} = -R_j$$

Flux Splitting gives

$$F_{j+1/2} = F^+_{j+1/2} + F^-_{j+1/2}; \quad F_{j-1/2} = F^+_{j-1/2} + F^-_{j-1/2};$$

Consider

$$\begin{aligned} F &= AQ \\ &= (T\Lambda T^{-1})Q \end{aligned}$$

where, $\Lambda = diag(u - a, u, u + a)$. Steger and Warming split the diagonal matrix into two parts, according to,

$$\Lambda = \Lambda^+ + \Lambda^-$$

where

$$\Lambda^\pm = \frac{\Lambda \pm |\Lambda|}{2}$$

Substituting into the flux vector, we have

$$\begin{aligned} F &= (T(\Lambda^+ + \Lambda^-)T^{-1})Q \\ &= (T\Lambda^+T^{-1})Q + (T\Lambda^-T^{-1})Q \\ &= A^+Q + A^-Q \\ &= F^+ + F^- \end{aligned}$$

Van Leer proposed a splitting similar to the Steger-Warming technique, that is smooth at the stagnation and sonic points, but imposed the additional requirement that the elements of the split flux vector be a smooth function of the Mach number. We have, extending to the two–dimensional case, at a given cell face, the flux normal to the cell face is given by

$$\hat{F}^{\pm} = f^{\pm} \begin{Bmatrix} 1 \\ u + \dfrac{\hat{n}_x(-U \pm 2a)}{\gamma} \\ v + \dfrac{\hat{n}_y(-U \pm 2a)}{\gamma} \\ \dfrac{1}{2}(u^2 + v^2 - U^2) + \dfrac{[(\gamma - 1)U \pm 2a]^2}{2(\gamma^2 - 1)} \end{Bmatrix}$$

where,

$$f^{\pm} = \pm \frac{\rho}{4a}(U \pm a)^2$$

Here, a is the sonic velocity, and $U(= \hat{n}_x u + \hat{n}_y v)$ is the velocity normal to the cell face, where $\hat{n}_x$ and $\hat{n}_y$ are the cartesian components of the normal to the cell face. Note that for the case, $U > a$, we have $\hat{F}^- = 0$, since $\Lambda^- = 0$, and $\hat{F}^+ = \hat{F}$.

Thus, we have, for the (j,k)th cell, the fluxes are given by,

$$\hat{F}_{j\pm1/2} = \hat{F}^+_{j\pm1/2}(Q^-_{j\pm1/2}) + \hat{F}^-_{j\pm1/2}(Q^+_{j\pm1/2}) \tag{A.5}$$

The residual for the (j,k)th cell is computed using (A.3).

Flux Difference Splitting

An alternative to flux-vector splitting are Godunov-type schemes, also known as flux difference splitting methods. Here, the conservative variables are considered as piecewise constant over the mesh cells at each time step, and the time evolution is determined by the exact/approximate solution of the Riemann problem at the inter-cell boundaries. Hence,

properties derived from the exact local solution of the Euler equations are introduced in the discretization. Roe developed an approximate Riemann solver based on an extension of the characteristic decomposition of the flux difference vector [73]. Roe proposed,

$$\hat{F}_{j+1/2} = \frac{1}{2}\left(\hat{F}^{+} + \hat{F}^{-} - |\tilde{A}| \cdot (Q^{+} - Q^{-})\right)$$

where

$$|\tilde{A}| = \tilde{T}|\tilde{\Lambda}|\tilde{T}^{-1}$$

and the subscripts $-$ and $+$ denote the left and right values at the cell interface, respectively. The matrix $\tilde{T}$ corresponds to the matrix of right eigenvectors of $\tilde{A}$ written in columns

$$\tilde{T} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ \tilde{u} & \hat{n}_y & \tilde{u} + \tilde{a}\hat{n}_x & \tilde{u} - \tilde{a}\hat{n}_x \\ \tilde{v} & -\hat{n}_x & \tilde{v} + \tilde{a}\hat{n}_y & \tilde{v} - \tilde{a}\hat{n}_y \\ \frac{\tilde{u}^2}{2} + \frac{\tilde{v}^2}{2} & \tilde{V} & \tilde{h}_o + \tilde{a}\tilde{U} & \tilde{h}_o - \tilde{a}\tilde{U} \end{bmatrix}$$

where h_o is the total enthalpy, and U and V are the contravariant velocities defined as

$$\begin{aligned} U &= \hat{n}_x u + \hat{n}_y v \\ V &= \hat{n}_y u - \hat{n}_x v \end{aligned}$$

The matrix of eigenvalues along the diagonal, $\tilde{\Lambda}$, is

$$\tilde{\Lambda} = \begin{bmatrix} \tilde{u} & 0 & 0 & 0 \\ 0 & \tilde{u} & 0 & 0 \\ 0 & 0 & \tilde{u} + \tilde{a} & 0 \\ 0 & 0 & 0 & \tilde{u} - \tilde{a} \end{bmatrix}$$

The $\sim$ notation refers to the Roe-averaged variables, defined according to

$$\begin{aligned} \tilde{\rho} &= \sqrt{\rho^{+}\rho^{-}} \\ \tilde{u} &= \Psi^{-}u^{-} + \Psi^{+}u^{+} \\ \tilde{v} &= \Psi^{-}v^{-} + \Psi^{+}v^{+} \\ \tilde{h}_o &= \Psi^{-}h_o^{-} + \Psi^{+}h_o^{+} \\ \tilde{a}^2 &= (\gamma - 1)\left[\tilde{h}_o - \frac{1}{2}(\tilde{u}^2 + \tilde{v}^2)\right] \end{aligned}$$

where

$$\Psi^{\pm} = \frac{\sqrt{\rho^{\pm}}}{\sqrt{\rho^{-}} + \sqrt{\rho^{+}}}$$

A.3.2 Boundary Conditions

Boundary conditions are imposed on a set of "ghost" cells constructed around the computational domain, given by

$$q_g = B(q), \tag{A.6}$$

where, q_g refers to the state variables in the ghost cells. For the $j = 0$, $j = jdim$ and $k = kdim$ edges, we impose the far-field boundary conditions proposed by Thomas and Salas [145], and for the $k = 0$ edge, corresponding to the airfoil surface, we use the curvature corrected boundary conditions formulated by Dadone and Grossman [38] to enforce tangency.

Surface Boundary Conditions

The approach proposed by Dadone and Grossman [38] involves extrapolating the surface pressure such that the normal momentum is satisfied. At the surface, the normal momentum equation is given by,

$$\frac{\partial p}{\partial n} = -\rho \frac{u^2 + v^2}{R}$$

where, n is the surface normal, and R is the radius of curvature of the surface. The symmetry technique implemented involves constructing an image cell to a given cell on the surface, outside the computational domain. The curvature of the surface is accounted for by using the momentum equation given above. We have,

$$p_{j,0} = p_{j,1} - \rho_{j,1} \frac{u_{j,1}^2 + v_{j,1}^2}{R} \Delta n_{j,1} \tag{A.7}$$

where, subscript $j, 1$ refers to the cell on the surface, subscript $j, 0$ refers to the image cell, and $\Delta n_{j,1}$ represents the distance between the cell centers of the surface and image cell. The other quantities may be evaluated by using conditions of constant entropy and total enthalpy. This yields,

$$\rho_{j,0} = \rho_{j,1} \left(\frac{p_{j,0}}{p_{j,1}} \right)^{1/\gamma} \tag{A.8}$$

$$U_{j,0} = -U_{j,1} \tag{A.9}$$

$$V_{j,0}^2 = V_{j,1}^2 + \frac{2\gamma}{\gamma - 1} \left(\frac{p_{j,1}}{\rho_{j,1}} - \frac{p_{j,0}}{\rho_{j,0}} \right) \tag{A.10}$$

Here, U and V represent the velocity normal to and along the cell face at the surface, respectively. The condition (A.9) stems from the impermeability condition. The Cartesian velocities are then given by,

$$u_{j,0} = \hat{n}_x U_{j,0} - \hat{n}_y V_{j,0} \tag{A.11}$$
$$v_{j,0} = \hat{n}_y U_{j,0} + \hat{n}_x V_{j,0} \tag{A.12}$$

Far-Field Boundary Conditions

The far-field boundary conditions used are those derived by Thomas and Salas [145] for transonic lifting flow over an airfoil. The conditions are derived based upon a point vortex representation of the airfoil, using linearized small-disturbance theory. The total velocities $\bar{u}$ and $\bar{v}$ at a given point in the far-field can be obtained as [145],

$$\begin{aligned} \bar{u}/q_\infty &= \cos\alpha + \bar{F}\sin\theta \\ \bar{v}/q_\infty &= \sin\alpha - \bar{F}\cos\theta \end{aligned} \tag{A.13}$$

where

$$\bar{F} \equiv \frac{C_L c}{4\pi r} \frac{\sqrt{1-M_\infty^2}}{(1-M_\infty^2 \sin^2(\theta-\alpha))}$$

Here, q_∞ is the freestream velocity, α is the angle of attack on the airfoil, C_L is the lift coefficient of the airfoil, c is the chord length, M_∞ is the freestream Mach number, and (r,θ) represents the location of the given point, in polar co-ordinates, with the origin fixed at the quarter-chord point of the airfoil.

The following characteristic equations can be developed along the far-field boundary,

$$\frac{ds}{dt} = 0 \quad \text{along} \quad C^0 : \frac{d\xi}{dt} = U \tag{A.14}$$
$$\frac{dV}{dt} = 0 \quad \text{along} \quad C^0 : \frac{d\xi}{dt} = U \tag{A.15}$$
$$\frac{d}{dt}(R^\pm) = 0 \quad \text{along} \quad C^\pm : \frac{d\xi}{dt} = U \pm a \tag{A.16}$$

where the local coordinate system (ξ,η) is constructed orthogonal to the outer boundary, with ξ normal to and directed outward from the boundary, U, V are the local velocities, and s is the entropy. The Riemann variables are given by

$$R^\pm \equiv U \pm (2a/\gamma - 1)$$

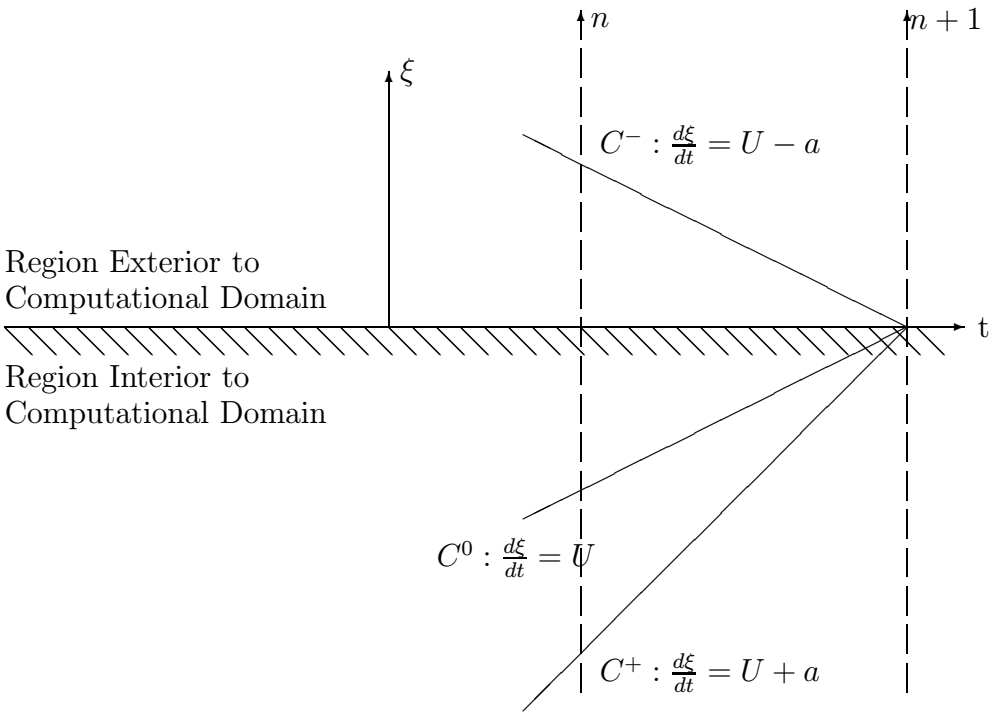


Figure A.3: Characteristics at Far-Field Boundary: $U > 0$

The characteristic paths are sketched in Figure A.3 for the case when there is subsonic outflow (*cf.* [145]). The characteristic equations are used to update the state variables on the boundary (ghost cells) at a new time level $n+1$. For the case shown in Figure A.3, R^- can be evaluated locally from the far-field representation (A.13), corresponding to conditions outside the computational domain, and R^+ can be evaluated locally from the interior of the computational domain. In this case, we use the conditions at the cell which the given ghost cell "mirrors". We have, for example, at the $j = 0$ edge,

$$\begin{aligned} R^- &= \bar{u} - \frac{2a_\infty}{\gamma - 1} \\ R^+ &= U_{1,k} - \frac{2a_{1,k}}{\gamma - 1} \end{aligned}$$

where $\bar{u}$ is computed from (A.13), evaluated at the cell interface along the boundary. The two Riemann variables can then be added and subtracted to give determine a local normal velocity and speed of sound at the boundary.

$$\begin{aligned} U_{0,k} &= \frac{1}{2}\left(R^- + R^+\right) \\ a_{0,k} &= \frac{\gamma - 1}{4}\left(R^+ - R^-\right) \end{aligned}$$

Since we have outflow conditions in this case, the tangential velocity is extrapolated from the interior of the domain. We have,

$$V_{0,k} = \hat{n}_x v_{1,k} - \hat{n}_y u_{1,k}$$

where $(\hat{n}_x, \hat{n}_y)$ are the cartesian components of the normal to the cell. For inflow conditions, we would extrapolate $\bar{v}$ computed from (A.13). The Cartesian velocities are then given by,

$$u_{0,k} = \hat{n}_x U_{0,k} - \hat{n}_y V_{0,k} \tag{A.17}$$

$$v_{0,k} = \hat{n}_y U_{0,k} + \hat{n}_x V_{0,k} \tag{A.18}$$

For outflow conditions, entropy is extrapolated from the interior, corresponding to (A.14). We obtain,

$$\rho_{0,k} = \rho_{1,k}\left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{2}{\gamma-1}} \tag{A.19}$$

$$p_{0,k} = p_{1,k}\left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{2\gamma}{\gamma-1}} \tag{A.20}$$

Note that for inflow conditions, we would extrapolate the entropy from the exterior, using freestream conditions.

A.4 Euler Implicit Time Integration

Consider the 2-D Euler equation

$$S\frac{\partial Q}{\partial t} + R(Q) = 0$$

We can rewrite the above, in terms of the primitive variables, to get

$$SM\frac{\partial q}{\partial t} + R(q) = 0$$

where

$$M = \frac{\partial Q}{\partial q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ \frac{u^2}{2} + \frac{v^2}{2} & \rho u & \rho v & \frac{1}{\gamma - 1} \end{bmatrix}$$

Euler implicit time integration and linearization yields

$$SM\frac{\Delta q}{\Delta t} = -R^{n+1} = -\left(R^n + \left(\frac{\partial R}{\partial q}\right)^n \Delta q + \cdots\right)$$

where $\Delta q = q^{n+1} - q^n$; $q^n = q(n\Delta t)$. In operator notation

$$\left[\frac{S}{\Delta t}M + \left(\frac{\partial R}{\partial q}\right)^n\right]\Delta q = -R^n$$

We have,

$$\left\{\frac{S}{\Delta t}M + \sum_{sides}\left[(\hat{A}\cdot\hat{n})\Delta s\right]^n\right\}\Delta q = -R(q^n)$$

where $\hat{A} = \frac{\partial \hat{F}}{\partial q}$ is the Jacobian. Here, the Jacobian is computed numerically using Van Leer's Flux Vector Splitting scheme (A.5) to evaluate the residual. The Jacobian terms are derived analytically from (A.5).

A.4.1 Approximate Factorization

For the (j,k)th cell, we have

$$\begin{aligned}\Big\{\frac{S}{\Delta t}M + \Big(\left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{j+1/2} \\ + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k-1/2} + \left[(\hat{A}\cdot\hat{n})\Delta s\right]_{k+1/2}\Big)^n\Big\}\Delta q_{jk} &= -R(q^n_{jk})\end{aligned}$$

Typically, the structure of the above linear problem is block pentadiagonal. Rewriting, and factoring according to spatial directions, we have

$$\left[T + \hat{A}_k\right]^n T^{-1} \left[T + \hat{A}_j\right]^n \Delta q = -R^n$$

where

$$\begin{aligned} T &= \frac{S}{\Delta t} M \\ \hat{A}_k &= \left[(\hat{A} \cdot \hat{n}) \Delta s\right]_{k-1/2} + \left[(\hat{A} \cdot \hat{n}) \Delta s\right]_{k+1/2} \\ \hat{A}_j &= \left[(\hat{A} \cdot \hat{n}) \Delta s\right]_{j-1/2} + \left[(\hat{A} \cdot \hat{n}) \Delta s\right]_{j+1/2} \end{aligned}$$

This is only an approximation to the previous equation. We solve using the sequence

$$\begin{aligned} \left[T + \hat{A}_k\right]^n \Delta q_{1/2} &= -R^n \\ \left[T + \hat{A}_j\right]^n \Delta q &= T \cdot \Delta q_{1/2} \end{aligned}$$

$$q^{n+1} = q^n + \Delta q$$

Provided that the spatial operators on the left hand side are three point operators, this algorithm only requires block tridiagonal inversions on ξ (k = constant) and η (j = constant) lines. It should also be noted that in performing this iteration, one can solve along each line independent of the other, which means the algorithm can be efficiently implemented in parallel. The above iteration is performed until the (normalized) residual is reduced within an acceptable tolerance. The Jacobian terms are computed using Van Leer Flux Vector Splitting (A.5), using analytical derivatives. The computation of the Jacobian is discussed in Section A.5.

Note: In practice, the Jacobian terms are not evaluated exactly. In reality, the Jacobian is given by,

$$\frac{\partial R}{\partial q} = \frac{\partial R}{\partial q} + \frac{\partial R}{\partial q_g} \frac{\partial q_g}{\partial q} \tag{A.21}$$

where $\frac{\partial q_g}{\partial q}$ is computed from the boundary conditions (A.6). This last term is neglected in practice so as to maintain the block tridiagonal structure. While this should reduce the rate of convergence of the algorithm, it does result in significant savings in terms of the ease of invertibility of the factored Jacobian, and "parallelizability".

A.5 Computation of the Gradients

Here, we consider the evaluation of the gradient terms, with respect to the state variables, as well as the control variables.

A.5.1 Jacobian of the Residual

Consider, for the (j,k)th cell, the residual is given by

$$R_{jk}(q,\beta) = \sum_{\text{sides}} (\hat{F}\cdot\hat{n})\Delta s$$

For a given cell face, the residual term is given by

$$R^{(i)}_{jk}(q,\beta) = (\hat{F}\cdot\hat{n}_i)\Delta s_i$$

We use Van Leer's Flux Vector Splitting Scheme (A.5). Consider, for example, a given term,

$$R^{(i)}_{jk}(q,\beta) = (\hat{F}^{+} + \hat{F}^{-})\Delta s_i$$

where

$$\hat{F}^{\pm} = f^{\pm}\begin{Bmatrix} 1 \\ u + \dfrac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \dfrac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \dfrac{1}{2}(u^2+v^2-U^2) + \dfrac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix}$$

with,

$$f^{\pm} = \pm\frac{\rho}{4a}(U \pm a)^2$$

and

$$U = u\cdot\hat{n}_{ix} + v\cdot\hat{n}_{iy}$$

is the normal velocity to the cell face.

Variation with respect to the State

We have,

$$\frac{\partial R_{jk}^{(i)}}{\partial q_i} = \frac{\partial}{\partial q_i}(\hat{F}^+ + \hat{F}^-)\Delta s_i$$

Differentiating $\hat{F}^\pm$ with respect to the state variables yields, with respect to ρ:

$$\frac{\partial \hat{F}^\pm}{\partial \rho} = \frac{\partial f^\pm}{\partial \rho}\begin{Bmatrix} 1 \\ u + \frac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \frac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \frac{1}{2}(u^2+v^2-U^2) + \frac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix} + f^\pm \begin{Bmatrix} 0 \\ \pm\frac{2\hat{n}_{ix}}{\gamma}\frac{\partial a}{\partial \rho} \\ \pm\frac{2\hat{n}_{iy}}{\gamma}\frac{\partial a}{\partial \rho} \\ \pm\frac{2[(\gamma-1)U \pm 2a]}{\gamma^2-1}\frac{\partial a}{\partial \rho} \end{Bmatrix},$$

where,

$$\begin{aligned} \frac{\partial f^\pm}{\partial \rho} &= \pm\frac{(U \pm a)^2}{4a} \mp \frac{\rho}{4a^2}(U \pm a)^2\frac{\partial a}{\partial \rho} + \frac{\rho}{2a}(U \pm a)\frac{\partial a}{\partial \rho}, \quad \text{and} \\ \frac{\partial a}{\partial \rho} &= -\frac{a}{2\rho}, \end{aligned}$$

with respect to u:

$$\frac{\partial \hat{F}^\pm}{\partial u} = \frac{\partial f^\pm}{\partial u}\begin{Bmatrix} 1 \\ u + \frac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \frac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \frac{1}{2}(u^2+v^2-U^2) + \frac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix} + f^\pm \begin{Bmatrix} 0 \\ 1 - \frac{\hat{n}_{ix}^2}{\gamma} \\ -\frac{\hat{n}_{ix}\hat{n}_{iy}}{\gamma} \\ u - U\hat{n}_{ix} + \frac{[(\gamma-1)U \pm 2a]}{(\gamma^2-1)}(\gamma-1)\hat{n}_{ix} \end{Bmatrix},$$

where

$$\frac{\partial f^\pm}{\partial u} = \pm\frac{\rho}{2a}(U \pm a)\hat{n}_{ix},$$

with respect to v:

$$\frac{\partial \hat{F}^\pm}{\partial v} = \frac{\partial f^\pm}{\partial v}\begin{Bmatrix} 1 \\ u + \frac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \frac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \frac{1}{2}(u^2+v^2-U^2) + \frac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix} + f^\pm \begin{Bmatrix} 0 \\ -\frac{\hat{n}_{ix}\hat{n}_{iy}}{\gamma} \\ 1 - \frac{\hat{n}_{iy}^2}{\gamma} \\ v - U\hat{n}_{iy} + \frac{[(\gamma-1)U \pm 2a]}{(\gamma^2-1)}(\gamma-1)\hat{n}_{iy} \end{Bmatrix},$$

where

$$\frac{\partial f^{\pm}}{\partial v} = \pm\frac{\rho}{2a}(U \pm a)\hat{n}_{iy}, \quad \text{and}$$

with respect to p:

$$\frac{\partial \hat{F}^{\pm}}{\partial p} = \frac{\partial f^{\pm}}{\partial p}\begin{Bmatrix} 1 \\ u + \dfrac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \dfrac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \dfrac{1}{2}(u^2 + v^2 - U^2) + \dfrac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix} + f^{\pm}\begin{Bmatrix} 0 \\ \pm\dfrac{2\hat{n}_{ix}}{\gamma}\dfrac{\partial a}{\partial p} \\ \pm\dfrac{2\hat{n}_{iy}}{\gamma}\dfrac{\partial a}{\partial p} \\ \pm\dfrac{2[(\gamma-1)U \pm 2a]}{\gamma^2-1}\dfrac{\partial a}{\partial p} \end{Bmatrix},$$

where,

$$\begin{aligned} \frac{\partial f^{\pm}}{\partial p} &= \mp\frac{\rho}{4a^2}(U \pm a)^2\frac{\partial a}{\partial p} + \frac{\rho}{2a}(U \pm a)\frac{\partial a}{\partial p}, \quad \text{and} \\ \frac{\partial a}{\partial p} &= \frac{a}{2p}. \end{aligned}$$

This yields a block 4×4 matrix, which is evaluated using linear interpolation.

Boundary Conditions

The effect of the boundary conditions is neglected when we perform approximate factorization to obtain state updates. This preserves a block tridiagonal structure. However, when we are required to compute the residual of the linearized state equation, as in the algorithm described in figure 4.8, we must account for the boundary conditions, using (A.21). This is effected by differentiating the boundary conditions (A.6). Note that the term $\frac{\partial R}{\partial q_g}$ is of similar form as the block matrices obtained above. We need to compute $\frac{\partial q_g}{\partial q}$ in order to compute the chain-rule terms. The terms corresponding to the surface boundary conditions can be derived by differentiating (A.7)–(A.12). We obtain, for the pressure:

$$\frac{\partial p_{j,0}}{\partial q_{j,1}} = \begin{Bmatrix} -\dfrac{u_{j,1}^2 + v_{j,1}^2}{R}\Delta n_{j,1} \\ -\rho_{j,1}\dfrac{2u_{j,1}}{R}\Delta n_{j,1} \\ -\rho_{j,1}\dfrac{2u_{j,1}}{R}\Delta n_{j,1} \\ 1 \end{Bmatrix}^T,$$

for the density:

$$\frac{\partial \rho_{j,0}}{\partial q_{j,1}} = \left\{ \begin{array}{c} \left(\frac{p_{j,0}}{p_{j,1}}\right)^{1/\gamma} + \rho_{j,1}\frac{1}{\gamma}\left(\frac{p_{j,0}}{p_{j,1}}\right)^{(1-\gamma)/\gamma}\frac{\partial p_{j,0}}{\partial \rho_{j,1}} \\ \rho_{j,1}\frac{1}{\gamma}\left(\frac{p_{j,0}}{p_{j,1}}\right)^{(1-\gamma)/\gamma}\frac{\partial p_{j,0}}{\partial u_{j,1}} \\ \rho_{j,1}\frac{1}{\gamma}\left(\frac{p_{j,0}}{p_{j,1}}\right)^{(1-\gamma)/\gamma}\frac{\partial p_{j,0}}{\partial v_{j,1}} \\ \rho_{j,1}\frac{1}{\gamma}\left(\frac{p_{j,0}}{p_{j,1}}\right)^{(1-\gamma)/\gamma}\left(\frac{\partial p_{j,0}}{\partial v_{j,1}} - \frac{p_{j,0}}{p_{j,1}^2}\right) \end{array} \right\}^T,$$

for the velocities:

$$\frac{\partial u_{j,0}}{\partial q_{j,1}} = \hat{n}_x \frac{\partial U_{j,0}}{\partial q_{j,1}} - \hat{n}_y \frac{\partial V_{j,0}}{\partial q_{j,1}}, \quad \text{and}$$

$$\frac{\partial v_{j,0}}{\partial q_{j,1}} = \hat{n}_y \frac{\partial U_{j,0}}{\partial q_{j,1}} - \hat{n}_x \frac{\partial V_{j,0}}{\partial q_{j,1}},$$

where,

$$\begin{aligned} \frac{\partial U_{j,0}}{\partial q_{j,1}} &= -\frac{\partial U_{j,1}}{\partial q_{j,1}} \\ 2V_{j,0}\frac{\partial V_{j,0}}{\partial q_{j,1}} &= 2V_{j,1}\frac{\partial V_{j,1}}{\partial q_{j,1}} + \frac{2\gamma}{\gamma-1}\left(\frac{1}{\rho_{j,1}}\frac{\partial p_{j,1}}{\partial q_{j,1}} - \frac{p_{j,1}}{\rho_{j,1}^2}\frac{\partial \rho_{j,1}}{\partial q_{j,1}} - \frac{1}{\rho_{j,0}}\frac{\partial p_{j,0}}{\partial q_{j,1}} + \frac{p_{j,0}}{\rho_{j,0}^2}\frac{\partial \rho_{j,0}}{\partial q_{j,1}}\right) \end{aligned}$$

where,

$$\frac{\partial \rho_{j,1}}{\partial q_{j,1}} = \left\{ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array} \right\}, \quad \frac{\partial U_{j,1}}{\partial q_{j,1}} = \left\{ \begin{array}{c} 0 \\ \hat{n}_x \\ \hat{n}_y \\ 0 \end{array} \right\}, \quad \frac{\partial V_{j,1}}{\partial q_{j,1}} = \left\{ \begin{array}{c} 0 \\ \hat{n}_y \\ -\hat{n}_x \\ 0 \end{array} \right\}, \quad \frac{\partial p_{j,1}}{\partial q_{j,1}} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} \right\},$$

respectively.

The terms corresponding to the far-field boundary conditions can be obtained by differentiating (A.13)–(A.20). As an example, consider the $j = 0$ edge for subsonic outflow, as described in Section A.3.2. We obtain, for the density,

$$\frac{\partial \rho_{0,k}}{\partial q_{1,k}} = \frac{\partial \rho_{1,k}}{\partial q_{1,k}}\left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{2}{\gamma-1}} + \frac{2\rho_{1,k}}{\gamma-1}\left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{3-\gamma}{\gamma-1}}\left(\frac{1}{a_{1,k}}\frac{\partial a_{0,k}}{\partial q_{1,k}} - \frac{a_{0,k}}{a_{1,k}^2}\frac{\partial a_{1,k}}{\partial q_{1,k}}\right)$$

and for the pressure,

$$\frac{\partial p_{0,k}}{\partial q_{1,k}} = \frac{\partial p_{1,k}}{\partial q_{1,k}} \left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{2\gamma}{\gamma-1}} + \frac{2\gamma p_{1,k}}{\gamma-1} \left(\frac{a_{0,k}}{a_{1,k}}\right)^{\frac{\gamma+1}{\gamma-1}} \left(\frac{1}{a_{1,k}}\frac{\partial a_{0,k}}{\partial q_{1,k}} - \frac{a_{0,k}}{a_{1,k}^2}\frac{\partial a_{1,k}}{\partial q_{1,k}}\right)$$

where,

$$\frac{\partial \rho_{1,k}}{\partial q_{1,k}} = \left\{ \begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array} \right\}, \quad \frac{\partial p_{1,k}}{\partial q_{1,k}} = \left\{ \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} \right\}, \quad \frac{\partial a_{1,k}}{\partial q_{1,k}} = \left\{ \begin{array}{c} -\frac{a_{1,k}}{2\rho_{1,k}} \\ 0 \\ 0 \\ \frac{a_{1,k}}{2p_{1,k}} \end{array} \right\},$$

and

$$\frac{\partial a_{0,k}}{\partial q_{1,k}} = \frac{\gamma-1}{4}\left(\frac{\partial R^+}{\partial q_{1,k}} - \frac{\partial R^-}{\partial q_{1,k}}\right),$$

where,

$$\begin{aligned} \frac{\partial R^+}{\partial q_{1,k}} &= \frac{\partial U_{1,k}}{\partial q_{1,k}} - \frac{2}{\gamma-1}\frac{\partial a_{1,k}}{\partial q_{1,k}}, \\ \frac{\partial R^-}{\partial q_{1,k}} &= 0, \end{aligned}$$

where,

$$\frac{\partial U_{1,k}}{\partial q_{1,k}} = \left\{ \begin{array}{c} 0 \\ \hat{n}_x \\ \hat{n}_y \\ 0 \end{array} \right\}$$

For the velocities, we obtain,

$$\frac{\partial u_{0,k}}{\partial q_{1,k}} = \hat{n}_x \frac{\partial U_{0,k}}{\partial q_{1,k}} - \hat{n}_y \frac{\partial V_{0,k}}{\partial q_{1,k}}, \quad \text{and}$$

$$\frac{\partial v_{0,k}}{\partial q_{1,k}} = \hat{n}_y \frac{\partial U_{0,k}}{\partial q_{1,k}} - \hat{n}_x \frac{\partial V_{0,k}}{\partial q_{1,k}},$$

where,

$$\frac{\partial U_{0,k}}{\partial q_{1,k}} = \frac{1}{2}\left(\frac{\partial R^+}{\partial q_{1,k}} + \frac{\partial R^-}{\partial q_{1,k}}\right),$$

and

$$\frac{\partial V_{0,k}}{\partial q_{1,k}} = \left\{ \begin{array}{c} 0 \\ -\hat{n}_y \\ \hat{n}_x \\ 0 \end{array} \right\}$$

Note that in this case, $q_{0,k}$ depends on not only $q_{1,k}$, but *also on the state variables at the airfoil surface*. This is because of the dependence of the far-field velocities (A.13) on the lift coefficient. We obtain the term,

$$\frac{\partial R^-}{\partial q_s} = \frac{\partial \bar{u}}{\partial q_s}$$

where, q_s refers to the state variables at the airfoil surface. We have,

$$\frac{\partial \bar{u}}{\partial q_s} = q_\infty \frac{\partial \bar{F}}{\partial q_s} \sin\theta,$$

where,

$$\frac{\partial \bar{F}}{\partial q_s} = \frac{c}{4\pi r} \frac{\beta}{1 - M_\infty^2 \sin^2(\theta - \alpha)} \frac{\partial C_L}{\partial q_s}$$

The dependence of the lift coefficient on the airfoil surface data will be discussed later. Since R^- affects $U_{0,k}$ and $a_{0,k}$, which in turn affects the state variables on the boundary (A.17)–(A.20), we obtain non-zero terms for $\frac{\partial q_g}{\partial q_s}$, which must be taken into account. We can obtain similar results for the case when we have subsonic inflow. The block 4×4 matrices, which comprise $\frac{\partial q_g}{\partial q}$ can be multiplied to the corresponding block 4×4 matrices, which comprise $\frac{\partial R}{\partial q_g}$ to obtain the contribution from the chain-rule term in (A.21). It should be noted that the terms corresponding to the dependence of the boundary (ghost cell) data on the adjacent cell data can be incorporated into $\hat{A}$ without affecting the block-diagonal nature of the system of equations. This has not been done in ErICA. However, in our modified algorithm, we do take this into account. The remaining terms, $\hat{A}_g$, *i.e.*, the terms corresponding to the dependence of the far-field data on the airfoil surface data would affect the structure of the problem. For these data, we only take into account $\hat{A}_g$ while computing the right hand side, *i.e.*, the residual of the linearized state equation, in the algorithm described in Figure 4.8.

Variation with respect to the Grid

Suppose, the computational grid is defined by,

$$(x, y) = \mathcal{G}(\omega) \tag{A.22}$$

where, ω could be some design parameter which defines the shape of the airfoil. We have,

$$\frac{\partial R_{jk}^{(i)}}{\partial \omega} = \frac{\partial}{\partial \omega}(\hat{F}^+ + \hat{F}^-)\Delta s_i$$

Consider

$$\frac{\partial}{\partial\omega}(\hat{F}^{\pm})\Delta s_i = \Delta s_i \frac{\partial \hat{F}^{\pm}}{\partial\omega} + \hat{F}^{\pm}\frac{\partial \Delta s_i}{\partial\omega}$$

Using the chain-rule we have,

$$\frac{\partial \hat{F}^{\pm}}{\partial\omega} = \frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{ix}}\frac{\partial \hat{n}_{ix}}{\partial\omega} + \frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{iy}}\frac{\partial \hat{n}_{iy}}{\partial\omega}$$

Consider

$$\frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{ix}} = \frac{\partial \hat{F}^{\pm}}{\partial U}\frac{\partial U}{\partial \hat{n}_{ix}} + \frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{ix}}$$

where, $\frac{\partial U}{\partial \hat{n}_{ix}} = u$. We have,

$$\frac{\partial \hat{F}^{\pm}}{\partial U} = \frac{\partial f^{\pm}}{\partial U}\begin{Bmatrix} 1 \\ u + \dfrac{\hat{n}_{ix}(-U \pm 2a)}{\gamma} \\ v + \dfrac{\hat{n}_{iy}(-U \pm 2a)}{\gamma} \\ \dfrac{1}{2}(u^2+v^2-U^2) + \dfrac{[(\gamma-1)U \pm 2a]^2}{2(\gamma^2-1)} \end{Bmatrix} + f^{\pm}\begin{Bmatrix} 0 \\ -\dfrac{\hat{n}_{ix}}{\gamma} \\ -\dfrac{\hat{n}_{iy}}{\gamma} \\ -U + \dfrac{[(\gamma-1)U \pm 2a]}{\gamma+1)} \end{Bmatrix}$$

where

$$\frac{\partial \hat{F}^{\pm}}{\partial U} = \pm\frac{\rho}{2a}(U \pm a)$$

and

$$\frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{ix}} = f^{\pm}\begin{Bmatrix} 0 \\ -\dfrac{-U \pm 2a}{\gamma} \\ 0 \\ 0 \end{Bmatrix}$$

Similarly, we can obtain an expression for $\frac{\partial \hat{F}^{\pm}}{\partial \hat{n}_{iy}}$. The terms $\frac{\partial \hat{n}_{iy}}{\partial\omega}$, $\frac{\partial \hat{n}_{ix}}{\partial\omega}$ and $\frac{\partial \Delta s_i}{\partial\omega}$ can be obtained from grid mapping, given by equation A.22. Note that the design, *i.e.*, the airfoil shape also affects the boundary conditions, because of the corrections accounting for curvature, used in enforcing the boundary conditions at the surface, and because of the dependence on the lift coefficient C_L in (A.13). These must also be taken into account while computing the sensitivity.

Bibliography

[1] Ira H. Abbott and Albert E. Von Doenhoff. *Theory of Wing Sections*. Dover Publications, Inc., New York, 1959.

[2] R. A. Adams. *Sobolev Spaces*. Academic Press, New York, 1975.

[3] AGARD. *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel*, Loen, Norway, May 22–23 1989.

[4] AIAA. *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference*, San Diego, CA, June 19–22 1995.

[5] W. Alt. The Lagrange-Newton Method for Infinite Dimensional Optimization Problems. *Numerical Functional Analysis and Optimization*, 11:201–224, 1990.

[6] W. Alt and K. Malanowski. The Lagrange-Newton Method for Nonlinear Optimal Control. *Computational Optimization and Applications*, 2:77–100, 1993.

[7] John D. Anderson Jr. *Introduction to Flight*. Mc-Graw Hill, Inc., New York, 1985.

[8] John D. Anderson Jr. *Modern Compressible Flow: With Historical Perspective*. McGraw-Hill, New York, second edition, 1990.

[9] A. R. Arcilla, J. Hauser, P. R. Eiseman, and J. F. Thompson (Editors). *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*. Elsevier Science Publishers B. V., North-Holland, 1991.

[10] Richard Barrett, Michael Berry, Tony Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1993.

[11] Wolf Bartelheimer. An Improved Integral Equation Method for the Design of Transonic Airfoils and Wings. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 463–473. AIAA Paper 95-1688.

[12] F. Bauer, P. Garabedian, and D. Korn. *Supercritical Wing Section III.* Springer Verlag, 1977.

[13] J. T. Betts and P. D. Frank. A Sparse Nonlinear Optimization Algorithm. *Journal of Optimization Theory and Applications*, 82(3):519–542, September 1994.

[14] John T. Betts. Software for Sparse Nonlinear Programming. BCSTECH-93-054, Boeing Computer Services, Seattle, WA, December 1993.

[15] John T. Betts, Samuel K. Eldersveld, and William P. Huffman. Sparse Nonlinear Programming Test Problems (Release 1.0). BCSTECH-93-047, Boeing Computer Services, Seattle, WA, December 1993.

[16] Lorenz T. Biegler. Convergence Analysis for a Multiplier-Free Reduced Hessian Method. EDRC 06-203-95, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, June 1995.

[17] Lorenz T. Biegler, Jorge Nocedal, and Claudia Schmid. A Reduced Hessian Method for Large-Scale Constrained Optimization. *SIAM Journal of Optimization*, 5(2):314–347, May 1995.

[18] C. H. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR: Generating Derivative Codes from FORTRAN Programs. *Scientific Programming*, 1(1):11–29, 1992.

[19] A. J. Bocci. Aerofoil Design Techniques. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 2.

[20] K.-W. Bock. Aerodynamic Design by Optimization. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 20.

[21] J. W. Boerlstoel and G. H. Huizing. Transonic Airfoil Design by an Analytic Hodograph Method. AIAA Paper 74-539, 1974.

[22] Jeffery T. Borggaard. *The Sensitivity Equation Method for Optimal Design.* PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1994.

[23] J. Borggard and D. Pelletier. Computing Design Sensitivities using an Adaptive Finite Element Method. In *AIAA 14th Applied Aerodynamics Conference*, New Orleans, June 18–20 1996. AIAA Paper 96-1938.

[24] D. Bos. Multidisciplinary Design Optimization of a Supersonic Transport Aircraft using a Hybrid Genetic/Gradient-Based Algorithm. In *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, September 4–6 1996. AIAA Paper 96-4055.

[25] J. C. Brock and W. F. Ng. A Consistent Direct-Iterative Inverse Design Method for High Speed Flows. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0326.

[26] A. E. Bryson and Y. C. Ho. *Applied Optimal Control*. Hemisphere Publishing Co., New York, 1975.

[27] Richard H. Byrd, Jean Charles Gilbert, and Jorge Nocedal. A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming. OTC 96/02, Optimization Technology Center, Northwestern University, Evanston, Illinois, 1996.

[28] Richard L. Campbell. An Approach to Constrained Aerodynamic Design with Application to Airfoils. NASA TP 3620, NASA Langley Research Center, Hampton, VA, November 1992.

[29] H. V. Cao and G. A. Blom. Navier-stokes/genetic optimization of multi-element airfoils. In *AIAA 14th Applied Aerodynamics Conference*, New Orleans, June 18–20 1996. AIAA Paper 96-2487.

[30] A. Carle, L. L. Green, C. H. Bischof, and P. A. Newman. Applications of Automatic Differentiation in CFD. AIAA Paper 94-2197, June 1994.

[31] L. A. Carlson. Transonic Airfoil Analysis & Design Using Cartesian Coordinates. *Journal of Aircraft*, 13, 1976.

[32] José E. Castillo (Editor). *Mathematical Aspects of Numerical Grid Generation*, volume 8 of *Frontiers in Applied Mathematics*. SIAM, Philadeliphia, 1991.

[33] Samson Cheung. Aerodynamic Design: Parallel CFD and Optimization Routines. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 1180–1188. AIAA Paper 95-1748.

[34] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*, volume 4 of *Studies in Mathematics and its Applied Problems*. North-Holland, New York, 1979.

[35] E. M. Cliff, M. Heinkenschloss, and A. Shenoy. On the Optimality System for a 1-D Euler Flow Problem. In *6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization*, Bellevue, September 4–6 1996. AIAA. AIAA Paper 96-3933.

[36] E. M. Cliff, M. Heinkenschloss, and A. Shenoy. Optimal Control for Flows with Discontinuities. *Journal of Optimization Theory and Applications*, 94, August 1997. to appear.

[37] Edwin Z. Crues. *Software User Manual: XALTOS X-Windows User Interface*. Institute for Flight System Dynamics, German Aerospace Research Establishment, DLR, Oberpfaffenhofen, September 1991. Documentation for the Advanced Launcher Trajectory Optimization Software (ALTOS), ESA Contract No. 8046/88/NL/MAC.

[38] A. Dadone and B. Grossman. Surface Boundary Conditions for the Numerical Solution of the Euler Equations. *AIAA Journal*, 32(2):285–293, February 1994.

[39] Sergio De Ponte, Maurizio Boffadossi, and Claudio Mantegazza. A Fast Collocation Method for Transonic Airfoil Design. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 8.

[40] J. E. Dennis, Matthias Heinkenschloss, and Luis N. Vicente. Trust–Region Interior–Point Algorithms for a Class of Nonlinear Programming Problems. ICAM Report 94-12-01, Interdisciplinary Center for Applied Mathematics, Virginia Tech, Blacksburg, VA, December 1994.

[41] Mark Drela. Elements of Airfoil Design Methodology. In P. A. Henne, editor, *Applied Computational Aerodynamics*, volume 125 of *Progress in Astronautics and Aeronautics*, chapter 6, pages 167–189. American Institute of Aeronautics and Astronautics, Inc., Washington, D.C., 1990.

[42] G. S. Dulikravich. Aerodynamic Shape Design. AGARD Report No. 780, 1990.

[43] Albrecht Eberle, Arthur Rizzi, and Ernst Heinrich Hirschel. *Numerical Solutions of the Euler Equations for Steady Flow Problems*, volume 34 of *Notes on Numerical Fluid Mechanics*. Vieweg, Wiesbaden, 1992.

[44] Peter R. Eiseman. Grid Generation for Fluid Mechanics Computations. *Annual Review of Fluid Mechanics*, 17:487–522, 1985.

[45] Mahmoud El-Alem. A Global Convergence Theory for the Celis-Dennis-Tapia Trust-Region Algorithm for Constrained Optimization. *SIAM Journal of Numerical Analysis*, 28(1):266–290, February 1991.

[46] Mahmoud El-Alem. A Robust Trust-Region Algorithm with a Nonmonotonic Penalty Parameter Scheme for Constrained Optimization. *SIAM Journal of Optimization*, 5(2):348–378, May 1995.

[47] S. Eyi, K. K. Chand, K. D. Lee, S. E. Rogers, and D. Kwak. Multi-Element High-Lift Design using the Navier-Stokes Equations. In *AIAA 14th Applied Aerodynamics Conference*, New Orleans, June 18–20 1996. AIAA Paper 96-1943.

[48] Dan Feng and Thomas Pulliam. Aerodynamic Design Optimization via Reduced Hessian SQP with Solution Refining. NASA-CR-201054, December 1995.

[49] Dan Feng and Thomas Pulliam. An all-at-once reduced Hessian SQP scheme for Aerodynamic Design Optimization. NASA-CR-201090, October 1995.

[50] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume 1: Fundamentals and General Techniques. Springer-Verlag, Berlin, 2nd edition, 1991.

[51] C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume 2: Specific Techniques for Different Flow Categories. Springer-Verlag, Berlin, 2nd edition, 1991.

[52] R. Fletcher. *Practical Methods of Optimization*, volume 2: Unconstrained Optimization. John Wiley & Sons, New York, 1981.

[53] R. Fletcher. *Practical Methods of Optimization*, volume 2: Constrained Optimization. John Wiley & Sons, New York, 1981.

[54] Norman Foster, George S. Dulikravich, and Jeff Bowles. Three-Dimensional Aerodynamic Shape Optimization Using Genetic Evolution and Gradient Search Algorithms. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0555.

[55] 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.

[56] Omar Ghattas and Xiaogang Li. A Variational Finite ELement Method for Nonlinear Fluid-Solid Interaction and its Sensitivity Analysis. AIAA Paper 94-4399.

[57] L. Ghielmi, R. Marazzi, and A. Baron. A Tool for Automatic Design of Airfoils in Different Operating Conditions. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 18.

[58] M. Giles, M. Drela, and W.T. Thompkins Jr. Newton Solution of Direct and Inverse Transonic Euler Equations. In *Proceedings of the 7th AIAA Computational Fluid Dynamics Conference*, pages 394–402, Cincinatti, Ohio, July 15–17 1985. AIAA. AIAA Paper 85-1530.

[59] Michael B. Giles and Mark Drela. Two-Dimensional Transonic Aerodynamic Design Method. *AIAA Journal*, 25(8):1199–1206, September 1987.

[60] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[61] A. Giunta, V. Balabanov, M. Kaufman, B. Grossman, W. Mason, and L. Watson. Wing Design for a High-Speed Civil Transport Using a Design of Experiments Methodology. In *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, September 4–6 1996. AIAA. AIAA Paper 96-4001.

[62] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.

[63] Max D. Gunzburger and Hyung C. Lee. Analysis, Approximation, and Computation of a Coupled Solid/Fluid Temperature Control Problem. *Computational Methods in Applied Mechanics Engineering*, 118:133–152, 1994.

[64] W. Hallman. Optimal Scaling Strategies for the Nonlinear Programming Problem. AIAA Paper 94-4417.

[65] K. A. Hawick, E. A. Bogucz, A. T. Degani, G. C. Fox, and G. Robinson. CFD Algorithms in High Performance FORTRAN. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 1233–1243. AIAA Paper 95-1752.

[66] B. He, O. Ghattas, and J. Antake. Shape Optimization of Time-Dependent Navier Stokes Flows. In *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, September 4–6 1996. AIAA. AIAA Paper 96-4187.

[67] Matthias Heinkenschloss. Projected Sequential Quadratic Programming Methods. *SIAM Journal of Optimization*, 6:373–417, 1996.

[68] Matthias Heinkenschloss and Luis N. Vicente. Analysis of Inexact Trust-Region Interior-Point SQP Algorithms. ICAM Report 95-06-12, Interdisciplinary Center for Applied Mathematics, Virginia Tech, Blacksburg, VA, June 1995.

[69] Preston A. Henne and Robert D. Gregg III. New Airfoil Design Concept. *Journal of Aircraft*, 28(5):300–311, May 1991.

[70] R. M. Hicks and G. N. Vanderplaats. Design of Low-Speed Airfoils by Numerical Optimization. In *SAE Business Aircraft Meeting*, Wichita, April 1975.

[71] N. Hirose, S. Takanashi, and N. Kawai. Transonic Airfoil Design Procedure Utilizing a Navier-Stokes Analysis Code. *AIAA Journal*, 25(3):353–359, 1987.

[72] Charles Hirsch. *Numerical Computation of Internal and External Flows*, volume 1: Fundamentals of Numerical Discretization. John Wiley & Sons, West Sussex, 1991.

[73] Charles Hirsch. *Numerical Computation of Internal and External Flows*, volume 2: Computational Methods for Inviscid and Viscous Flows. John Wiley & Sons, West Sussex, 1991.

[74] Gene J.-W. Hou, Venkateshwarlu Maroju, Arthur C. Taylor, Vamshi M. Korivi, and Perry A. Newman. Transonic Turbulent Airfoil Design Optimization with Automatic Differentiation in Incremental Iterative Forms. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 512–526. AIAA Paper 95-1692.

[75] J. Huan and V. Modi. Optimum Design of Minimum Drag Bodies in Incompressible Laminar Flow Using a Control Theory Approach. *Inverse Problems in Engineering*, 1:1–25, 1994.

[76] D. H. Huddleston and C. W. Mastin. Optimization of Aerodynamic Designs using Computational Fluid Dynamics. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 23.

[77] W. Huffman and M. Carter. Software for Sparse Finite Difference Derivatives. AMS-LR-71, Boeing Computer Services, Seattle, WA, January 1992.

[78] D. M. Hwang and C. T. Kelley. Convergence of Broyden's Method in Banach Spaces. *SIAM Journal of Optimization*, 2:505–532, 1992.

[79] A. H. Ibrahim, G. J.-W. Hoy, S. N. Tiwari, and R. E. Smith. Stability Analysis and Aerodynamic Design of Euler Equations using Variational Methods. In *AIAA 14th Applied Aerodynamics Conference*, New Orleans, June 18–20 1996. AIAA Paper 96-2439.

[80] Angelo Iollo, Geojoe Kuruvila, and Shlomo Ta'asan. Pseudo-Time Method for Optimal Shape Design using the Euler Equations. ICASE Report 95-59, Institution for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, August 1995. NASA CR 198205.

[81] Angelo Iollo and Manuel D. Salas. Contribution to the Optimal Shape Design of Two-Dimensional Internal Flows with Embedded Shocks. ICASE Report 95-20, Institution for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, March 1995. NASA CR 195062.

[82] Angelo Iollo, Manuel D. Salas, and Shlomo Ta'asan. Shape Optimization Governed by the Euler Equations Using an Adjoint Method. ICASE Report 93-78, Institution for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, November 1993. NASA CR 191555.

[83] A. Jameson and J. J. Alonso. Automatic Aerodynamic Optimization on Distributed Memory Architectures. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0409.

[84] A. Jameson and J. Reuther. Control Theory Based Airfoil Design using the Euler Equations. AIAA Paper 94-4272.

[85] Antony Jameson. Airfoils Admitting Non-Unique Solutions of the Euler Equations. AIAA Paper 91-1625.

[86] Antony Jameson. Optimum Aerodynamic Design using CFD and Control Theory. AIAA Paper 95-1729.

[87] Antony Jameson. Full-Potential, Euler and Navier-Stokes Schemes. In P. A. Henne, editor, *Applied Computational Aerodynamics*, volume 125 of *Progress in Astronautics and Aeronautics*, chapter 3, pages 39–88. American Institute of Aeronautics and Astronautics, Inc., Washington, D. C., 1990.

[88] Antony Jameson. Computational Algorithms for Aerodynamic Analysis and Design. MAE Report 1966, Princeton University, December, 1992.

[89] Chris Jänsch. *Software User Manual: TROPIC Optimization Program*. Institute for Flight System Dynamics, German Aerospace Research Establishment, DLR, Oberpfaffenhofen, February 1991. Documentation for the Advanced Launcher Trajectory Optimization Software (ALTOS), ESA Contract No. 8046/88/NL/MAC.

[90] Joh, C.-Y. and Grossman, B. and Haftka, R. T. Design optimization of transonic airfoils. *Engineering Optimization*, 21(1):1–20, 1993.

[91] W. H. Jou, W. P. Huffman, D. P. Young, R. G. Melvin, M. B. Bieterman, C. L. Hilems, and F. T. Johnson. Practical Considerations in Aerodynamic Design Optimization. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 950–960. AIAA Paper 95-1730.

[92] Patrick Knupp and Stanly Steinberg. *Fundamentals of Grid Generation*. CRC Press, Boca Raton, 1993.

[93] H. Köster, C.-H. Rohardt, K.-H. Horstmann, and R. Radespiel. Aerodynamic Design Techniques at DLR Institute for Design Aerodynamics. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 3.

[94] R. R. Krishnan and T. M. Eidson. An Efficient, Parallel Space-Marching Euler Solver for HCST Research. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 1189–1199. AIAA Paper 95-1749.

[95] F.-S. Kupfer. An Infinite-Dimensional Convergence Theory for Reduced SQP Methods in Hilbert Space. *SIAM Journal of Optimization*, 6(1):126–163, February 1996.

[96] F.-S. Kupfer and E. W. Sachs. Numerical Solution of a Nonlinear Parabolic Control Problem by a Reduced SQP Method. *Computational Optimization and Applications*, 1:113–135, 1992.

[97] G. Kuruvila, Shlomo Ta'asan, and M. D. Salas. Airfoil Optimization by the One-Shot Method. In *AGARD FDP/VKI Special Course on Optimum Design Methods for Aerodynamics*, pages 7.1–7.21, Rhode-Saint-Genese, Belgium, November 1994. von Karman Institute for Fluid Dynamics. AGARD Report 803.

[98] M. J. Lighthill. A New Method of Two-Dimensional Aerodynamic Design. ARC R&M 2112, 1945.

[99] M. E. Lores and P. R. Smith. Supercritical Wing Design Using Numerical Optimization and Comparisons with Experiment. AIAA Paper 79-0065, January 1979.

[100] David G. Luenberger. *Optimization by vector space methods*. John Wiley & Sons, New York, 1969.

[101] J. B. Malone, J. C. Narramore, and L. N. Sankar. Airfoil Design Method Using the Navier-Stokes Equations. *Journal of Aircraft*, 28(3):216–224, March 1991.

[102] K. K. Mani. Design Using Euler Equations. AIAA Paper 84-2166, August 1984.

[103] G. Mosetti and C. Poloni. Aerodynamic Shape Optimization by means of a Genetic Algorithm. In *5th International Symposium on Computational Fluid Dynamics*, volume II, pages 279–284, Sendai, 1993.

[104] Walter Murray and Francisco J. Prieto. A Sequential Quadratic Programming Algorithm using an Incomplete Solution of the Subproblem. *SIAM Journal of Optimization*, 5(3):590–640, August 1995.

[105] R. Narducci, B. Grossman, and R. T. Haftka. Sensitivity Algorithms for an Inverse Design Problem involving a Shock Wave. *Inverse Problems in Engineering*, 2:49–83, 1995.

[106] R. Narducci, B. Grossman, M. Valorani, A. Dadone, and R. T. Haftka. Optimization Methods for Non-Smooth or Noisy Objective Functions in Fluid Design Problems. In *AIAA 12th Applied Aerodynamics Conference*, pages 21–32, New Orleans, June 19–22 1996. AIAA Paper 95-1648.

[107] Robert P. Narducci. *Selected Optimization Procedures for CFD-Based Shape Design involving Shock Waves or Computational Noise*. PhD thesis, Virgina Tech, Blacksburg, Virginia, May 1995.

[108] Eric J. Nielson, W. Kyle Anderson, Robert W. Walters, and David E. classes. Application of Newton-Krylov Methodology to a Three-Dimensional Unstructured Euler Code. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 981–990. AIAA Paper 95-1733.

[109] David Nixon. Perturbation of a Discontinuous Transonic Flow. *AIAA Journal*, 16(1):47–52, January 1978.

[110] H. J. Oberle and W. Grimm. *BNDSCO: A Program for the Numerical Solution of Optimal Control Problems*. Institute for Flight System Dynamics, German Aerospace Research Establishment, DLR, Oberpfaffenhofen, 1987. English Translation of DFVLR-Mitt. 85-05.

[111] M. J. Pandya and O. Baysal. Gradient-Based Aerodynamic Shape Optimization using ADI Method for Large-Scale Problems. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0091.

[112] O. Pironneau. *Optimal Shape Design for Aerodynamics*. AGARD/VKI Lecture Series on Optimum Design Methods in Aerodynamics. von Karman Institute for Fluid Dynamics, 1994.

[113] Thomas H. Pulliam, Stuart Rogers, and Timothy J. Barth. Practical Aspects of Krylov-Subspace Iterative Methods in CFD. In *AGARD 77th Fluid Dynamics Panel Symposium on Progress and Challenges in CFD Methods and Algorithms*, Seville, Spain, October 2–5 1995.

[114] J. Reuther and A. Jameson. Control Theory Based Airfoil Design for Potential Flow and a Finite Volume Discretization. AIAA Paper 94-0499.

[115] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0094.

[116] James John Reuther. *Aerodynamic Shape Optimization using Control Theory*. PhD thesis, University of California, Davis, California, 1996.

[117] M. H. Rizk. Aerodynamic Optimization by Simultaneously Updating Flow Variables and Design Parameters. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 15.

[118] E. Sadchikova, A. Shenoy, and E. M. Cliff. Computational Issues in Optimization-Based Design. In *34th IEEE Conference on Decision and Control*, pages 445–450, New Orleans, December 1995.

[119] I. Sadrehaghighi and S. N. Tiwari. Grid sensitivity for aerodynamic optimization and flow analysis, April 1993. NASA-CR-192980.

[120] Ideen Sadrehaghighi, Robert E. Smith, and Surendra N. Tiwari. Grid Sensitivity and Aerodynamic Optimization of Generic Airfoils. *Journal of Aircraft*, 32(6):1234–1239, 1995.

[121] Stephen Scherr. Portable Parallelization of Block-Structured Flow Solvers. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 1210–1217. AIAA Paper 95-1760.

[122] K. Schittkowski. NLPQL: A FORTRAN Subroutine for solving Constrained Nonlinear Programming Problems. *Annals of Operations Research*, 5:485–500, 1985.

[123] K. Schnepper. *Software User Manual: PROMIS Optimization Program*. Institute for Flight System Dynamics, German Aerospace Research Establishment, DLR, Oberpfaffenhofen, February 1992. Documentation for the Advanced Launcher Trajectory Optimization Software (ALTOS), ESA Contract No. 8046/88/NL/MAC.

[124] S. Schrier. *Compressible Flow*. John Wiley & Sons, New York, 1982.

[125] J. Serrin. Mathematical Principles of Classical Fluid Mechanics. In *Handbuch der Physik*, volume VIII/1, pages 125–263. Springer, Berlin, 1959.

[126] V. Shankar. A Full Potential Inverse Method Based on a Density Linearization Scheme for Airfoil/Wing Design. AIAA Paper 81-1234.

[127] A. Shenoy and E. M. Cliff. An Optimal Control Formulation of a Flow Matching Problem. In *5th AIAA/USAF/NASA/ISSMO Symposium of Multidisciplinary Analysis and Optimization*, pages 520–528, Panama City, September 1994. AIAA. AIAA Paper 94-4306.

[128] Aly Sherif, Frank Marconi, Madara Ogot, Richard Pelz, and Mike Siclari. Stochastic Optimization applied to CFD Shape Design. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 11–20. AIAA Paper 95-1647.

[129] G. R. Shubin. Obtaining "Cheap" Optimization Gradients from Computational Aerodynamic Codes. Technical Report AMS-TR-164, Boeing Computer Services, Seattle, WA, June 1991.

[130] G. R. Shubin and P. D. Frank. A comparison of the implicit gradient approach and the variational approach to aerodynamic design optimization. Technical Report AMS-TR-163, Boeing Computer Services, Seattle, WA, June 1991.

[131] M. J. Siclari, W. van Nostrand, and F. Austin. The Design of Transonic Airfoil Sections for an Adaptive Wing Concept using a Stochastic Optimization Method. In *AIAA 34th Aerospace Sciences Meeting & Exhibit*, Reno, January 15–18 1996. AIAA Paper 96-0329.

[132] J. W. Sloof. Computational methods for subsonic and transonic aerodynamic design. AGARD Report 712, 1983.

[133] R. E. Smith and M. R. Wiese. Interactive Algebraic Grid-Generation Technique. NASA TP 2533, NASA Langley Research Center, Hampton, VA, March 1986.

[134] Robert E. Smith and Lars-Erik Eriksson. Algebraic Grid Generation. *Computer Methods in Applied Mechanics and Engineering*, 64:285–300, 1987.

[135] H. Sobieczky, K. Y. Fung, and A. R. Seebass. A New Method for Designing Shock-free Transonic Configurations. AIAA Paper 78-1114.

[136] Helmut Sobieczky. Progress in Inverse Design and Optimization in Aerodynamics. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 1.

[137] Stephen S. Stahara. Rapid Approximate Description of Nonlinear Solutions: Application to Aerodynamic Flows and Design/Optimization Problems. In David Nixon, editor, *Transonic Aerodynamics*, volume 81 of *Progress in Astronautics and Aeronautics*, chapter 18, pages 637–661. American Institute of Aeronautics and Astronautics, Inc., New York, 1982.

[138] John P. Steinbrenner and John R. Chawner. The GRIDGEN Version 8 Multiple Block Grid Generation Software. MDA Engineering Report 92-01, MDA Engineering, Inc., Arlington, Texas, May 1993.

[139] T. Strand. Exact Method of Designing Airfoils with Given Velocity Distribution in Incompressible Flow. *Journal of Aircraft*, 10:651–659, 1973.

[140] Shlomo Ta'asan. Pseudo Time Methods for Constrained Optimization Problems Governed by PDE. ICASE Report 95-32, Institution for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, May 1995. NASA CR 195081.

[141] Shlomo Ta'asan. Trends in Aerodynamics Design and Optimization: A Mathematical Viewpoint. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 961–970. AIAA Paper 95-1731.

[142] S. Takanashi. Iterative Three-Dimensional Transonic Wing Design Using Integral Equations. *Journal of Aircraft*, 22(8):655–660, 1985.

[143] A. Taylor III, G. Hou, and V. Korivi. Sensitivity Analysis, Approximate Analysis and Design Optimization for Internal and External Viscous Flows. In *AIAA Aircraft Design Systems and Operations Meeting*, Baltimore, September 1991. AIAA. AIAA Paper 91-3083.

[144] A. C. Taylor III, P. A. Newman, G. J. Hou, and H. E. Jones. Recent Advances in Steady Compressible Aerodynamic Sensitivity Analysis. In M. D. Gunzberger, editor, *Flow Control*, volume 68 of *Volumes in Mathematics and its Applications*, pages 341–356. Springer-Verlag, 1995.

[145] James L. Thomas and M. D. Salas. Far–Field Boundary Conditions for Transonic Lifting Solutions to the Euler Equations. *AIAA Journal*, 24(7):1074–1080, July 1986.

[146] T. L. Tranen. A Rapid Computer Aided Transonic Airfoil Design Method. AIAA Paper 74-501.

[147] R. Unal, R. Lepsch, W. Engelund, and D. Stanley. Approximation Model Building and MDO Using Response Surface Methods. In *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, September 4–6 1996. AIAA Paper 96-4044.

[148] J. A. van Egmond. Numerical Optimization of Target Pressure Distributions for Subsonic and Transonic Airfoil Design. In *Computational Methods for Aerodynamic Design (Inverse) and Optimization, papers presented and discussions held at the Specialists's Meeting of the Fluid Dynamics Panel* [3]. Paper No. 17.

[149] Garret N. Vanderplaats. Approximation Concepts for Numerical Airfoil Optimization. NASA TP 1370, NASA Ames Research Center, 1979.

[150] A. Verhoff, D. Stookesberry, and A. B. Cain. An efficient approach to optimal aerodynamic design. i–analytic geometry and aerodynamic sensitivities. In *AIAA 31st Aerospace Sciences Meeting & Exhibit*, Reno, January 11–14 1993. AIAA Paper 93-0099.

[151] Luis N. Vicente. *Trust-Region Interior-Point Algorithms for a Class of Nonlinear Programming Problems*. PhD thesis, Rice University, Houston, Texas, March 1996.

[152] G. Volpe. Inverse Airfoil Design: A Classical Approach Updated for Transonic Applications. In P. A. Henne, editor, *Applied Computational Aerodynamics*, volume 125 of *Progress in Astronautics and Aeronautics*, chapter 7, pages 191–220. American Institute of Aeronautics and Astronautics, Inc., Washington, D. C., 1990.

[153] G. Volpe and R. E. Melnik. The Role of Constraints in the Inverse Design Problem for Transonic Airfoils. AIAA Paper 81-1233, June 1981.

[154] R. A. Weed, W. K. Anderson, and L. A. Carlson. A Direct-Inverse Three-Dimensional Transonic Wing Design Method for Vector Computers. AIAA Paper 84-2156.

[155] K.-H. Well and K. Ebert. Trajectory Optimization Techniques and Software Implementation. In *IFAC Symposium on Aerospace Control*, Ottobran, September 7–9 1992.

[156] L. C. Woods. Airfoil design in two-dimensional subsonic flow. R&M 2845, Aeronautical Research Council, London, 1952.

[157] M. H. Wright. Interior Point Methods for Constrained Optimization. In A. Iserles, editor, *Acta Numerica 1992*, pages 341–407. Cambridge University Press, 1992.

[158] X. Wu, E. M. Cliff, and M. D. Gunzburger. An Optimal Design Problem for a Two-Dimensional Flow in a Duct. *Optimal Control Applications & Methods*, 17:329–339, 1996.

[159] Kenji Yamamoto and Osamu Inoue. Applications of Genetic Algorithm to Aerodynamic Shape Optimization. In *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference* [4], pages 43–51. AIAA Paper 95-1650.

[160] D. P. Young, W. P. Huffman, R. G. Melvin, M. B. Bieterman, C. L. Hilmes, and F. T. Johnson. Inexactness and Global Convergence in Design Optimization. AIAA Paper 94-4386.

[161] O. C. Zienkiewicz. *The Finite Element Method*. McGraw-Hill, London, 1977.

Vita

The author, Ajit Shenoy, was born in Bombay, India on June 7, 1968. After completing his secondary education in 1985, Ajit's fascination with airplanes culminated in his attending the Indian Institute of Technology, Bombay, where he majored in Aeronautical Engineering. He graduated in 1989, with a senior project entitled "Microprocessor-Based Speed Control of a Hydraulic Motor" following which he worked for a period of one year as a Junior Research Assistant at the Aerodynamics Laboratory in the Aerospace Engineering Department at the Indian Institute of Technology, Bombay. His responsibilities included calibration of an existing newly built low speed wind tunnel facility, and design and fabrication of a stability wind tunnel for the experimental study of aircraft longitudinal stability. In 1990, Ajit entered graduate school at the Indian Institute of Technology, Bombay, continuing to work in the area of experimental aerodynamics. His Master's Thesis was based upon "An Experimental Study of Aircraft Longitudinal Stability". During this period, Ajit's interests shifted to computational research, and in 1992, after graduating with a Master of Technology degree in Aerospace Engineering, he moved to the state of Virginia in the United States, where he enrolled in the Ph.D. program in the Aerospace Engineering Department at Virginia Tech. On July 23, 1993, Ajit married Uma Talwalkar. While at Virginia Tech, Ajit's research efforts have been focused on the study of various optimization techniques, with applications to aerodynamic design problems. He completed his Doctorate of Philosophy in the spring of 1997.