OPTIMAL SHAPE DESIGN WITH DOMAIN DECOMPOSITION

by

Lena V. Sadtchikova

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfilment of the requirement for the degree of

MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

APPROVED:

Dr. Eugine M. Cliff, Chairman

Dr. John A. Burns

Dr. Frederick H. Lutze

July 1996
Blacksburg, Virginia

OPTIMAL SHAPE DESIGN WITH DOMAIN DECOMPOSITION

by

Lena V. Sadtchikova

Dr. Eugene M. Cliff, Chairman

Aerospace and Ocean Engineering

(ABSTRACT)

In this work, we considered an "inverse-design" problem, where we specified the flow distribution in the computational domain or on its subset and sought the geometrical configuration that produced this flow. Based on this idea, we formulated a control problem, in which the optimization procedure minimizes the error between the target flow and the actual flow through successive adjustment of the design parameters. We were interested in exploring the computational efficiency of the numerical solution of this problem, particularly the implementation of workstation cluster environment to the solution of the control problem by employing numerical algorithms, which would allow coarse-grained parallelization. These aspects were studied with an example of one-dimensional heat transfer in surfaces of non-uniform cross-sectional area and the optimal design of a two-dimensional nozzle. We compared the computational cost and convergence properties of the optimization procedure for two approaches: the "Black-Box" method and domain decomposition

method. When employing the "Black-Box" technique, the unconstrained control problem was solved in terms of the design variables and in the case of domain decomposition implementation, the constrained control problem was solved in terms of design variables and boundary data on interfaces. Also, we implemented the grid-embedding chimera technique to the solution of a one-dimensional heat transfer problem. The formulation of this scheme in terms of domain decomposition leads to an overlapping domain decomposition method.

It was concluded from one-dimensional results that domain decomposition methods can be successfully incorporated into the optimization-based design framework. The original analysis problem can be split into problems on subdomains, which can be solved in parallel with data transfer at each step limited to exchanging boundary information between the neighboring subdomains and between the "master" processor. In the case of a two-dimensional flow, the optimization was applied to supersonic flow and the discontinuous flow. It was concluded, that if one wishes to implement this algorithm in a parallel environment, the computations should be spread between the processors in such a way, that the number of processors is proportional to the number of control variables.

Acknowledgements

I would like to express my sincere appreciation and gratitude to my advisor, Dr. Eugene M. Cliff, for his support, encouragement, guidance and suggestions which made this work possible.

Thanks are also due to Dr. John A. Burns and Dr. Frederick H. Lutze for serving on my committee.

I would also like to thank the group at ICAM for their friendship and kindness.

Finally I would like to express my deepest feelings of gratitude towards my family and my closest and dearest friend David Tang for their support, endless love and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Modeling physical phenomena with scientific computing is an interdisciplinary effort involving engineers, mathematicians and computer scientists. Practical physical systems are often modeled by complicated systems of partial differential equations (PDE's). Their numerical solution requires high performance computers, large software systems and efficient algorithms. The design of numerical algorithms must balance many factors. From the numerical analysis point of view, one often focuses on good approximation, fast convergence and low arithmetic expense. In practical applications, one should be able to handle the complexity and generality of PDE problems. Modern technologies and concepts, such as the use of parallel computing, lead to issues of parallel algorithms, communication cost and

others. Obviously, many of these objectives contradict each other. The principal trade-off is programming effort versus execution time efficiency [**20**]. In this work, we examine various computational aspects of an "inverse-design" problem, arising in the context of optimization-based design, from the practical point of view. Among those are the choice of optimization procedure, the parameterization of design space and, mainly, the high computational cost of this problem. Our main goal was to investigate the possibilities of solving this control problem in workstation-cluster environment, where only coarse-grained parallelism is considered. We wanted to explore whether an inexpensive network of workstations is capable of simulation modeling that otherwise would require supercomputer resources. Therefore, we were interested in algorithms that lead to parallel computations that are essentially independent. These considerations led us to consider domain decomposition methods, which appear as natural means to exploit coarse-grained parallelism.

The term Domain Decomposition generally refers to a class of methods for solving PDE's on a given domain by first decomposing it into smaller ones and then obtaining the overall solution by solving smaller problems on these subdomains. In this sense, the idea is rather old and can be traced to Schwarz's alternating procedure, in which existence of solution to boundary value problem was proven by an iteration involving solutions on overlapping subdomains. This idea is also widely used in many fields of scientific computing. In structural mechanics, these techniques are known as substructuring methods and are especially useful when the size of the complete system is too large for the main

memory of the computer. In computational fluid dynamics, it is common to decompose the physical domain into different regions and use different forms of governing equations in each subdomain. Domain decomposition methods received additional attention because of the advent of parallel computing and the obvious opportunity for parallelism in these methods.

In this work, we discuss the major aspects of solving the "inverse-design" problem with and without implementation of domain decomposition techniques. We explore the performance of a generic domain decomposition algorithm with an example of a one-dimensional and a two-dimensional problem. We use two approaches: our first approach is a so-called "Black-Box" method, which leads to the formulation of an unconstrained control problem in one domain. Our second approach is based on subdividing the computational domain into subdomains and solving a constrained control problem in terms of design variables and boundary data on the subdomain interfaces.

1.2 Inverse Design Problem

Design of a complex aerodynamic configuration, which needs to satisfy numerous requirements, has always been a challenging task. Recent advances in computational power and ability to accurately model physical phenomena have provided the design engineer with efficient and powerful tools. After it became possible to approach the analysis problem

with the aid of numerical methods, there is a growing need for tools that suit the needs of the designer. That is, beyond knowing the aerodynamic characteristics of a current design, the designer wants to know how to change it in order to improve the systems performance. Optimization-based design is one of the ways to approach this problem. The basic idea of this method is to treat the design problem, but only at the price of heavy computational expense, with numerical optimization methods. In essence, a numerical optimization procedure is directly coupled to an analysis algorithm. It attempts to extremize a chosen measure of merit which is evaluated by the given analysis code. The configuration is systematically modified through user-specified design variables.

Formalizing the process requires that one parameterizes the set of available choices. Additionally, one must provide a procedure for identifying feasible designs (*i.e.* constraints to be imposed). For a specified design, evaluation of the feasibility constraints and the performance attributes is performed by the relevant analysis code.

In view of this structure, it is natural to use the term control variable, (u) to describe the design choice, and the term state variable, (y) to describe those quantities implied by the control choice and the relevant physics. In most cases, the physical laws, modeled with complex systems of PDE's, can be stated as implicit relations:

$$r(y, u) = 0, \tag{1.1}$$

so that for fixed u, the system (1.1) can be solved for y.

With the states and controls in hand, it is possible to assess the design. In this setting, additional feasibility restrictions are imposed by the mathematical requirement

$$h(y,u) \geq 0 \in R^m. \tag{1.2}$$

There can be several ways to choose the measure of performance of a design configuration. We limit our attention to a narrow class of so-called "inverse-design" problems, for which there is a scalar-valued function J to be minimized. An example of this type of problem is to find the shape of a body that would have a specified flow configuration over its surface. This is essentially the inverse of a direct analysis calculation, in which the shape of a body is given and the flow around it is calculated. In this work, the inverse-design problem has been formulated as a control problem in which a control system minimizes the error between the actual and the desired state variables by making successive changes to the shape of the body. The cost function for this problem can be written as:

$$J(y,u) = \int_\Omega |y(x,u) - \hat{y}(x)|^2 d\Omega,$$

where Ω denotes the computational domain and $\hat{y}(x)$ denotes the target (flow) distribution.

Standard theory applied to this problem can be conveniently stated in terms of Lagrange function [**24**] :

$$\mathcal{L}(y,u,\lambda,\mu) = J(y,u) + < \lambda, r(y,u) > + < \mu, h(y,u) >, \tag{1.3}$$

where $y \in R^n, u \in R^m$. The Lagrange multipliers λ and μ can then be naturally identified as vectors in R^n and R^m, respectively.

Many of these concepts can be carried over to the case where y is infinite–dimensional and r involves a differential operator. This is the case for the examples we study. Of course, computer solutions will commonly require some finite-dimensional approximation. Nevertheless, it is not clear that one should approximate the physics before starting the optimization problem. Quite commonly, new possibilities for efficient solution are apparent when one optimizes and then approximates.

Necessary conditions for optimality of a given state/control pair (y^*, u^*) can be stated in terms of the Lagrangian function (1.3). If (y^*, u^*) is optimal for our problem then there exists multiplier vector λ and μ such that:

$$\nabla_y J(y^*, u^*) + \lambda \nabla_y r(y^*, u^*) + \mu \nabla_y h^a(y^*, u^*) = 0 \tag{1.4}$$

$$\nabla_u J(y^*, u^*) + \lambda \nabla_u r(y^*, u^*) + \mu \nabla_u h^a(y^*, u^*) = 0 \tag{1.5}$$

$$r(y^*, u^*) = 0 \tag{1.6}$$

$$h^a(y^*, u^*) = 0, \tag{1.7}$$

where h^a denotes the subset of active feasibility constraints.

In this work, we used an optimization code, based on a sequential quadratic programming algorithm. At each iteration step, a linearly constrained quadratic or linear least squares subproblem is formulated by approximating the Lagrange function quadratically and by linearizing the constraints. Subsequently, a one dimensional line search is performed with respect to an augmented Lagrange merit function to obtain the new iterate.[**25**]

The motivation of the present research was to explore some of the computational issues of the inverse design procedure. Primarily, incorporating parallel computation in the framework of optimization was investigated. This led us to consider domain decomposition methods as potential algorithms which would allow coarse-grained parallelization.

1.3 Overview of Domain Decomposition Methods

In recent years a considerable attention has been devoted to the use of domain decomposition techniques for numerical solution of partial differential equations. Among others, the following reasons underly the development of these techniques. The equations in the different subdomains may be of different type, or more simply, can contain different parameters. In addition, when dealing with complicated geometries, a subdivision of the entire domain by simply shaped subdomains, on which special solution techniques can be applied, may increase the overall efficiency of the numerical scheme. And most importantly, domain decomposition methods can be applied to the parallel solution of the often very large systems of linear and nonlinear algebraic equations that arise when problems in elasticity, fluid dynamics, and many other important applications are discretized by finite elements or finite differences.

Domain decomposition, in its simplest form, consists of dividing the domain of a problem into a number of (possibly overlapping) subdomains, and then interactively adjusting the

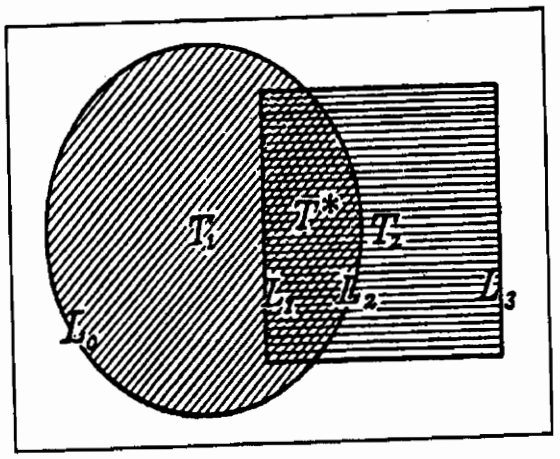


Figure 1.1: Diagram used by Schwarz in an early paper to illustrate his method

global solution using local subdomain solutions. This idea was first suggested by H.A. Schwarz [**27**] in the 1800's for analytical solution of Poisson's equation using overlapping subdomains. He proposed an iterative method for the solution of a classical boundary-value problem for harmonic equations: it consists of solving successively a similar problem in subdomains, going alternatively from one to the other. The convergence of this process was proven by the use of the maximum principle. Since then, many variants of this concept were proposed and studied both theoretically and numerically. We refer for instance to Q. V. Dinh, R. Glowinski, J. Periaux [**17**], M. Dryja [**15**], Yu. A. Kuznetsov, A. M. Matsokin, G. I. Marchuck [**19**] and others.

The methods can be regarded as "divide-and-conquer" algorithms — subproblems, corresponding to subregions of a model problem, can be solved by known techniques with

interactions between the subregions accounted for by an iteration that involves passing suitable information across the interfaces partitioning the original region.

The principle of domain decomposition is quite general, so that many algorithms can be developed based on it. One can categorize the methods according to two criteria:

1) The level at which the decomposition is taken into account— a method based on a more internal level corresponds to substructuring of the linear systems [**22**], while an external level method involves the choice of the model for the physical phenomena [**1**]. Algebraic type of domain decomposition first discretizes a PDE problem on an entire domain and then partitions the discrete system according to the geometric decomposition. Then, by a block eliminating procedure, independent systems are derived for each subdomain. The remaining unknowns pertaining to the interface boundaries are coupled by the global system. The interactions between the two sets of unknowns are then handled by a suitable iterative method (for example, conjugate gradient method). At this step, the use of a properly designed preconditioner may remarkably reduce the number of iterations.[**10**]

At another extreme is the problem of partitioning on the continuous level so that PDE solution techniques in different regions may be independent, depending on local properties. One may use finite differences for one subdomain, finite elements or even analytic solution for another. The main concern of the latter approach is to find the correct boundary relations between the local solutions in order to enforce continuity and other appropriate conditions [**5**].

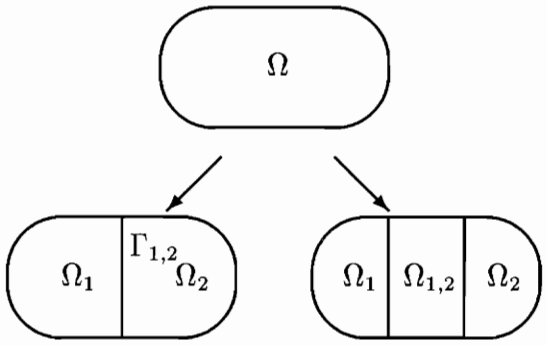


Figure 1.2: Non-overlapping and overlapping domain decomposition

2) The decomposition of the initial geometric domain either by partitioning the domain into non-overlapping subdomains [**6**],[**8**] or by overlapping subdomains [**9**], [**23**]. The general idea of a non-overlapping subdomain-iteration approach is presented below. Figure (1.2) shows the difference between these two decompositions.

Let Γ_i be an interface, that is, the common boundary piece of two neighboring subdomains Ω_i and Ω_j, *i.e.* $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$. Each subdomain obeys a physical law locally. Namely, there is a PDE L_l and a function U_l defined in each Ω_l so that

$$L_l U_l = f_l \quad \text{in } \Omega_l \quad \text{for } l = i, j. \tag{1.8}$$

The interface condition on Γ_{ij} can be usually specified in the form

$$g_{ij}(U_i, U_j, \frac{\partial U_i}{\partial n}, \frac{\partial U_j}{\partial n}) = 0. \tag{1.9}$$

For example, for the continuity condition of the global solution and its normal derivative

(1.9) takes the form:

$$(U_i - U_j)^2 + \left(\frac{\partial U_i}{\partial n} - \frac{\partial U_j}{\partial n}\right)^2 = 0. \tag{1.10}$$

The above expression can also be stated in terms of two equality constraints. In order to match continuity for both the global solution U and its normal derivative $\frac{\partial U}{\partial n}$ on interfaces, one starts with an initial guess for U and $\frac{\partial U}{\partial n}$ on the interfaces, takes them as boundary data to solve a Dirichlet or Neumann boundary-value problem on subdomains, then updates interface values using the new solution data and iterates until convergence.

When using the overlapping method, one defines subdomains with overlap and solves local problems in succession, propagating data from the solution on the interior of one subdomain to the boundary values of the next subdomain.

As parallel multi-processor computers are becoming indispensable for large-scale scientific computing, effective parallel algorithms are necessary. An efficient parallel strategy for a broad class of problems is domain decomposition. The geometric domain decomposition is intrinsically suited for the purposes of parallelization since such a decomposition is in agreement with the natural data dependencies of elliptic and time-parabolic problems.

The conceptually simple strategy of domain decomposition is very appealing from a standpoint of parallel processing, since each subdomain can be assigned to a different processor and computed independently. The partition of the whole region into individual subdomains leads to computational units of smaller size, which are to a large extent independent of each other. The original problem is solved independently in each processor,

then some patching procedure is employed to enforce continuity conditions on the interfaces. Such a patching step requires communication between processors. As a result, the computation can be performed in parallel, thus decreasing the overall time, provided that the communication-to-computation ratio is a favorable one. That is why the main objective of any domain decomposition method is to reduce the amount of data transfer and communication between subdomains. The number of subdomains, the size of subdomain grids and the connectivity of subdomains will influence the performance and efficiency of the method on a given parallel computer architecture.

One potentially attractive approach to building parallel processors is to use workstations connected on a local area network, often called a workstation cluster. This approach, however, has proved suitable only for applications where parallel computations are so coarse-grained as to be essentially independent. The domain-iterative based approach, described above, best fits into the coarse-grained network framework and is promising from a practical point of view, since the data exchange between the processors is reduced to passing boundary information.

1.4 Preview of the Thesis

The outline of the thesis is as follows. A brief description of the background and mathematical structure of the "inverse-design" problem and the overview if the domain decom-

position methods are provided in Chapter 1. This chapter is followed by an investigation of the control problem, arising in the "inverse-design" procedure, on a one-dimensional example of the inhomogeneous heat equation. In Section 4 of this chapter we introduce a non-overlapping domain decomposition method and compare the computational cost of this procedure to the "Black-Box" approach. Chapter 3 describes the implementation of the chimera scheme, which can be viewed as an overlapping domain decomposition. Chapter 4 is devoted to the optimal design of a two-dimensional nozzle and includes a description of the numerical algorithm used to solve for the flow in the nozzle. Results of optimization for supersonic and discontinuous flows are presented in Section 5 and the implementation of domain decomposition is presented in Section 6. Finally, conclusions are presented in Chapter 5.

Chapter 2

Heat Equation: Domain Decomposition

2.1 Problem Formulation

In this chapter we consider a problem of a heat transfer in a circular cylinder with varying cross-section area. We will confine our analysis to surfaces so proportioned that heat conduction within may be treated as one-dimensional. Conservation of energy, combined with Fourier's Law, leads to a boundary value problem formulation for the steady temperature in the rod:

$$\frac{d}{dx}\left(A(x,u)\frac{dT(x)}{dx}\right) - \frac{h}{k}\frac{dS(x,u)}{dx}T(x) = 0, \quad 0 < x < 1 \tag{2.1}$$

$$T(0) = T_0 \qquad T(1) = T_1, \tag{2.2}$$

Here, $A(x,u)$ is the cross-sectional area of the solid cylinder, $S(x,u)$ is the cylinder surface area, exposed to convection, $h \geq 0$ is convective film coefficient, $k > 0$ is the thermal conductivity. This equation may be applied generally to all surface configurations, for which the one-dimensional assumption is valid. For example, if $A = constant$ and $S = Cx$, equation (2.1) becomes:

$$\frac{d^2T}{dx^2} - \frac{hC}{kA}T = 0. \tag{2.3}$$

We will make use of the analytical solution for this boundary-value problem with the boundary data given by (2.2):

$$T(x) = \frac{T_1 e^m - T_0}{e^{2m} - 1} e^{mx} + \frac{T_1 e^{-m} - T_0}{e^{-2m} - 1} e^{-mx},$$

where $m^2 = \frac{hC}{kA}$.

Our model describes the radius as a parabolic function of the axial location, that is

$$R(x,u) = u_3 + u_2 x + u_1 x^2,$$

and our control parameters are the coefficients u_1, u_2 and u_3.

We consider an "inverse design" problem, wherein $\hat{T}(x)$ is a desired temperature profile and the task is to adjust the parameters, $u_i, i = 1, \ldots, 3$ to minimize the error between the target temperature distribution and the temperature obtained from the solution of (2.1)-(2.2) with these parameters. *Given* $\hat{T}(x)$ *find* u_1, u_2 *and* u_3 *such that*

$$J(T,u) = \int_0^1 \left[T(x,u) - \hat{T}(x)\right]^2 dx \tag{2.4}$$

is minimized.

Discrete versions of (2.1) can be constructed in various ways. We employ a finite-element model based on cubic elements and a uniform grid. By a standard procedure, we transform to a problem with homogeneous boundary data and find that a partition into $N+1$ intervals leads to a system of N dependent variables, thus $n = N$ in the Lagrange function (1.3). Furthermore, we have three control parameters, so that the dimension of the control space is $m = 3$. The components of the finite-dimensional state y are identified with nodal temperatures and in this case the system (1.1) is linear with a banded matrix. Appendix A contains the detailed description of the numerical solution of (2.1.)

The control problem (2.4) is solved in 2 ways. First, we consider the so-called "Black-Box approach", and second, we employ Domain Decomposition techniques, which assume the subdivision of the computational domain into smaller subdomains. We introduce the following notation to denote the discrete approximation to the solution of (2.1): y_M^N is the approximation on the interval [0,1], constructed with N finite elements on each of M subintervals of [0,1].

Since the temperature $T(x)$ is represented by a sum of piece-wise cubic functions (*i.e.* cubic on each element) and a linear function, (see Appendix A), the numerical quadrature, based on Gauss-Legendre points, $\zeta_i \in [0,1]$, performed on each element, yields the exact

value of the integral (2.4).

2.2 Black-Box Method

Necessary conditions for optimality of a given state/control pair (y^*, u^*) are expressed by (1.4)—(1.5). Based on the observation that the analysis code will "solve" the system (1.1) for y given u, it is natural to substitute this implicitly defined $\hat{y}(u)$ into the cost function and constraints. This substitution leads to a problem in terms of the control variables u. We define $\hat{f}$ as $\hat{f}(u) = \hat{f}(y(u), u))$ and similarly for $\hat{h}(u)$. The implicit function theorem will guarantee the existence of the function $\hat{y}$ and its Jacobian $\nabla_u \hat{y}$. For the reduced form the necessary conditions can be written as:

$$\begin{aligned} \nabla_u \hat{f}(u^*) + \mu \nabla_u \hat{h}(u^*) &= 0 \\ \hat{h}(y^*, u^*) &= 0. \end{aligned} \tag{2.5}$$

The method, which is based on the reduced form of necessary conditions is commonly referred to as the Black-Box approach [**26**], since the optimization procedure is based on the structure of the problem with respect to control parameters u and not the state vector y. The analysis code, which solves for y_M^N appears as a "Black-Box" to the optimizer, which, once the control parameters are passed to it, simply produces the information necessary for the next control parameter update. For a given set of parameter values u_i the discrete approximation to the solution of (2.1) is computed and the integral (2.4) is evaluated. This

information is sent to an optimizer, which updates the parameters u_i and sends them back to the analysis code. This process is repeated until the optimal criteria is met. The number of active constraints in this setting is zero, *i.e.* number of h^a is zero.

The example of the Black-Box approach can be constructed in the following way. We attempt to find a pair (y^*, u^*), minimizing the integral (2.4) and satisfying the boundary-value problem (2.1), where $\hat{T}(x)$ is given by the exact solution of the reduced heat equation (2.3), and where the radius of a cylinder is independent of the $x-$location, *i.e.*, $R = C$, where C is taken to be unity. Then we know that the optimal set of control parameters is given by $(0, 0, C)$. Some sample runs contained in Table (2.2) demonstrate that the resulting solutions for u_i found by the optimization code [**25**] are reasonably close to the expected solution $u^* = (0, 0, 1)$. The results represented here were obtained on SGI 4D/340 machine.

As it was mentioned above, the size of y_M^N ($M = 1$ in this setting) is determined by the discretization scheme, and in our case it depends on the number of finite elements used to partition the computational domain. The accuracy of the optimization procedure and its computational cost depend on this discretization. One expects that for small N the discrete system may not adequately approximate the physics so that the optimization results would be poor. For large N, the physics are well described, but each evaluation of y_1^N is costly. Figure (2.1) shows the results of optimization for different number of finite elements. The target set of control parameters was $\hat{u} = (0, 0, 1)$ and the target temperature distribution was given by the analytical solution of (2.3) with boundary data $T(0) = 1, T(1) = 0$. The

initial guess for control parameters was $(u_1^0, u_2^0, u_3^0) = (-1, 7, 5)$. As seen in Figure (2.1), the number of QP iterations required for a solution increases slightly with problem size. Also, as expected, for large N evaluation of y_1^N consumes nearly all the computational effort.

Table 2.2 : Results of Optimization for the Black-Box Approach

$u_{target} = (0, 0, 1)$, N=8

initial guess	approximation of solution	cost function value	number of iterations	elapsed time
2 -1 4	-0.61813×10^{-5} 0.10580×10^{-3} 1.00017	0.49770×10^{-7}	56	14.42 sec
3 -1 5	0.36759×10^{-5} -0.21793×10^{-4} 0.99997	0.41425×10^{-7}	52	17.82 sec
1 -1 3	0.36338×10^{-5} -0.11270×10^{-4} 0.99999	0.42693×10^{-7}	35	11.63 sec

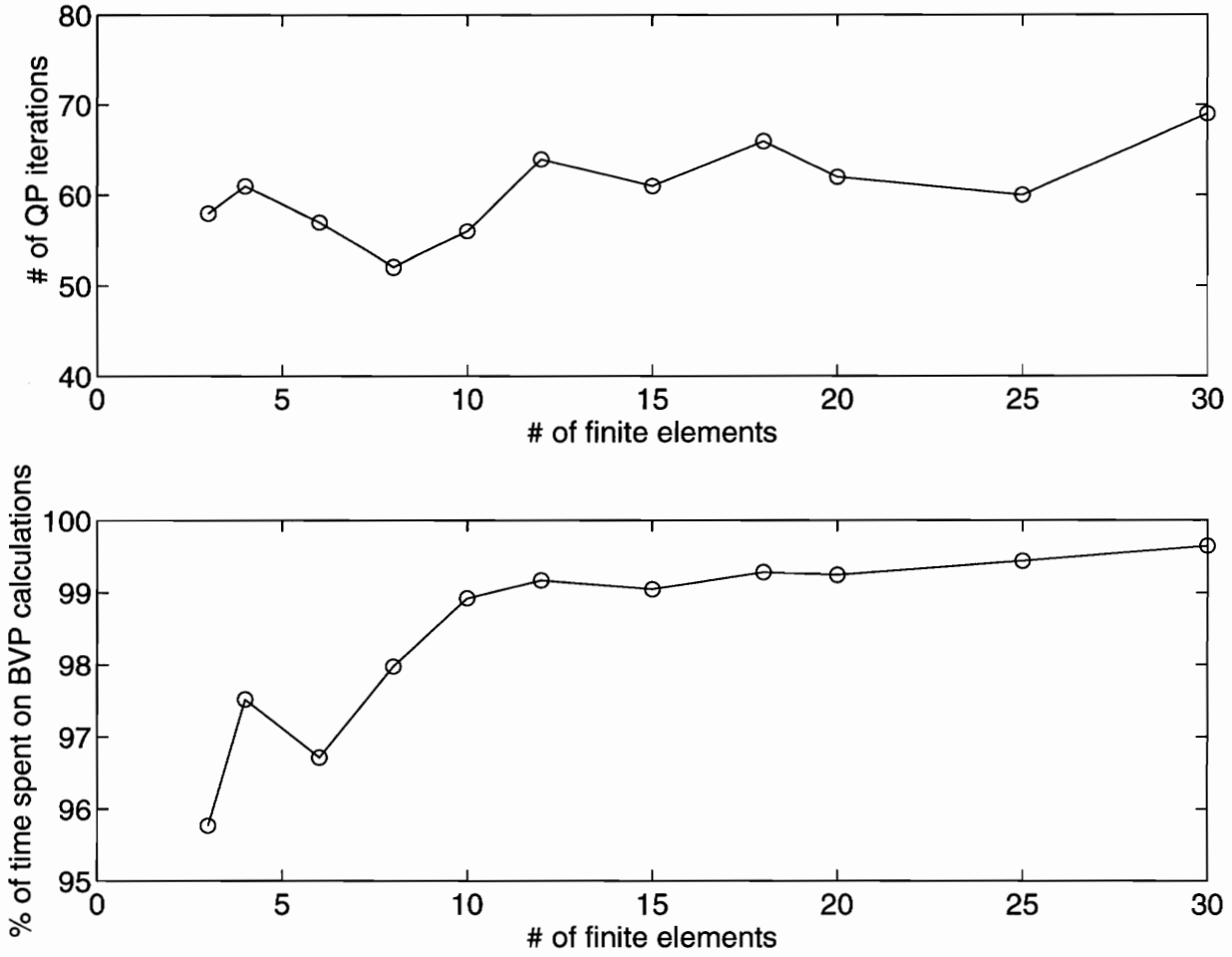


Figure 2.1: Results for the Black-Box method

2.3 Sensitivity Equation Method

The gradient of the cost function (2.4) with respect to control parameters $u_i, i = 1, \ldots, 3$ is given by:

$$\frac{\partial J}{\partial u_i} = 2 \int_0^1 \frac{\partial T}{\partial u_i} \left(T(u) - \hat{T} \right) dx, \qquad i = 1, \ldots, 3.$$

Here, $\frac{\partial T}{\partial u_i}, i = 1, \ldots, 3$ denotes the temperature sensitivity functions, which express the character of the dependence of temperature distribution T on the design parameters. The Sensitivity Equation Method [**4**] expresses the temperature sensitivity as a solution of the boundary-value problem:

$$\begin{aligned} &\frac{d}{dx}\left(A(x,u) \frac{dT_{u_i}(x)}{dx} \right) - \frac{h}{k} \frac{dS(x,u)}{dx} T_{u_i}(x) \quad = \\ &- \left[-\frac{d}{dx}\left(A(x,u) \frac{dT(x)}{dx} \right) - \frac{h}{k} \frac{dS(x,u)}{dx} T(x) \right] \end{aligned} \tag{2.6}$$

$$T_{u_i}(0) = 0 \qquad T_{u_i}(1) = 0 \qquad \text{for i=1,2,3,}$$

where $T_{u_i} = \frac{\partial T}{\partial u_i}$, $i = 1, 2, 3$.

Since the model (2.1) is linear, the work done in the computation of T_{u_i} by the Sensitivity Equation Method is the same as that required by a one-sided difference scheme. Figure (2.2) shows a comparison of the discrete finite-difference approximation of T_{u_i}, indicated by "o"s, and the finite-element solution of (2.6), represented by the solid line, when $u_1 = -2, u_2 = 4, u_3 = 1$ and $T(0) = 1, T(1) = 0$.

This method was used to compute the gradient of the cost function $\nabla_u J$ in the previous section.

2.4 Domain Decomposition Method

In this section we address certain features of the domain decomposition techniques, that can be implemented in optimization-based design in order to reduce the computational cost of the problem by parallelizing the procedure and spreading the analysis over a network of workstations.

In general, the solution of the analysis problem in optimization-based design, which is always based on expensive discretization schemes, is known to be computationally intensive. It was shown above that most of the computational time is spent on solving the analysis problem at each iteration step, and the finite-dimensional representation of the state vector y_M^N constitutes the main part of the memory requirements. Therefore, various forms of parallel numerical algorithms have been studied in order to speed-up these computations.

Domain decomposition techniques have two principal elements:

(1) decomposition of the computational domain into subdomains and

(2) communication between subdomains.

The number of subdomains, the size of subdomain grids and connectivity influence the performance of the global iteration and also the efficiency on a given parallel com-

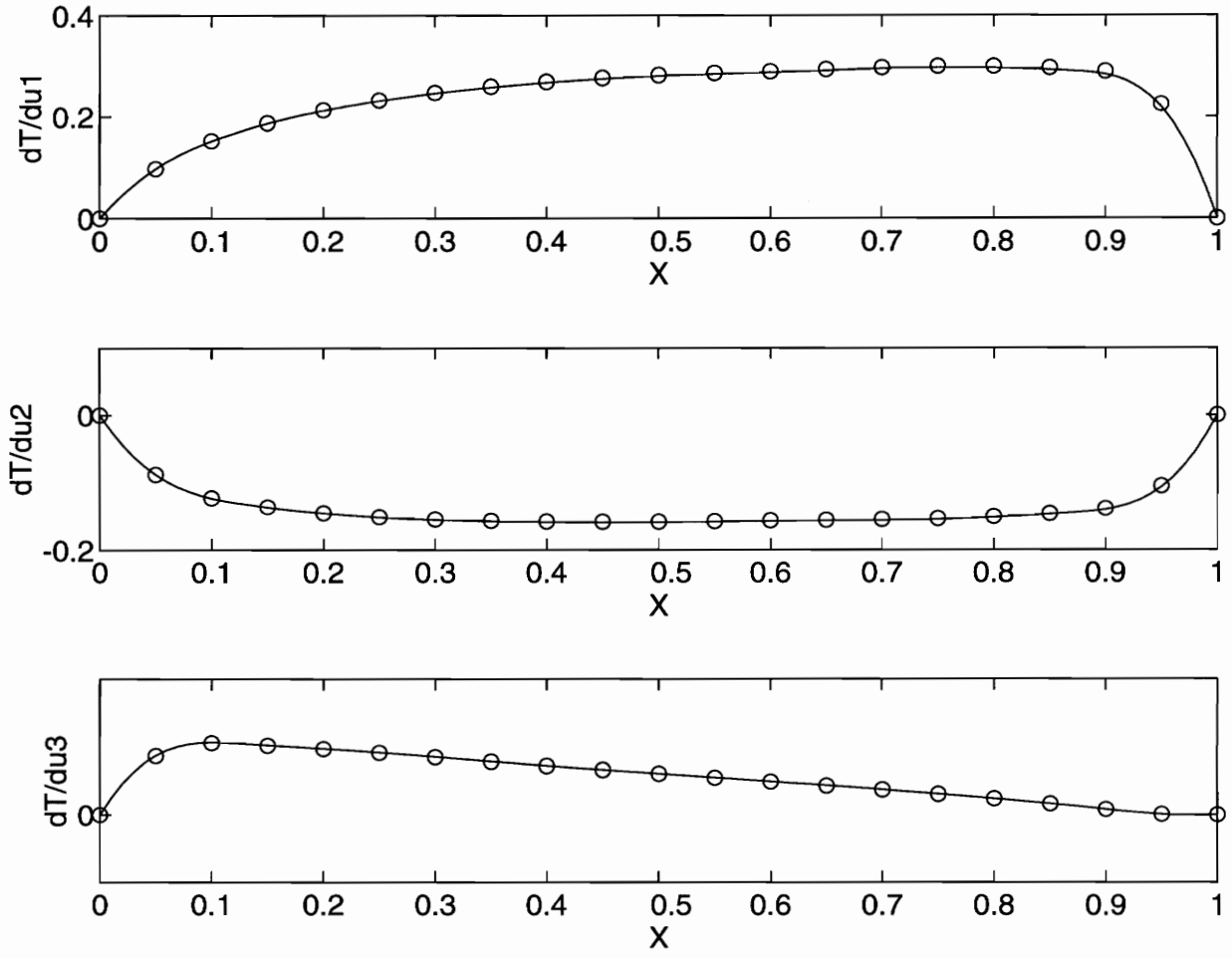


Figure 2.2: Sensitivity Functions

puter architecture. Several strategies to subdivide the computational region and establish communication between subregions have been explored in this work.

2.4.1 Non-Overlapping Domain Decomposition

The main idea of non-overlapping domain decomposition methods currently in use is that the computational region is partitioned into non-overlapping regions with common boundaries. One starts with initial guesses for boundary data and solves the boundary-value problems on subdomains. The continuity of the overall solution is achieved by imposing proper matching relations on the interfaces, which are imposed in such a way that only communication between adjacent subdomains is required. The interface values are then updated using new solution data, and the process is repeated until convergence.

2.4.2 Multidomain Formulation of the Mathematical Problem

We consider the decomposition of the initial computational domain by partitioning it into non-overlapping subdomains. We introduce the following notation to describe the procedure.

Let $\Omega_i, i = 1, \ldots, M$ be a partition of Ω such that:

$$\Omega = \bigcup_{i=1}^{M} \Omega_i, \qquad \Omega_j \cap \Omega_k = 0, \quad j \neq k$$

and let Γ_i be an interface, that is, the common boundary piece of two neighboring subdo-

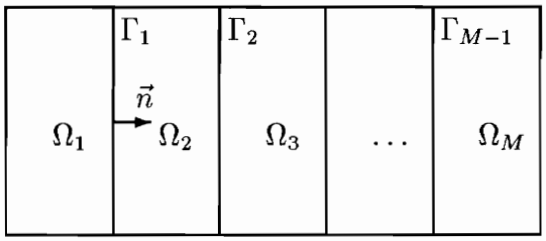


Figure 2.3: Non-overlapping domain decomposition

mains Ω_i and Ω_{i+1}.

$$\Gamma_i = \partial\Omega_i \cap \partial\Omega_{i+1}, \quad i = 1, \ldots, M-1 \qquad \Gamma = \bigcup_{i=1}^{M-1} \Gamma_i.$$

In practice, each Ω_i is usually a rectangular region (see Figure (2.3)) and for the one-dimensional problem it is just a subinterval of [0,1] with the coupling boundary Γ_i reduced to a point. For our purposes

$$\Omega_i = \begin{cases} \{x : 0 < x < \hat{x}_i\}, & \text{if } i = 1; \\ \{x : \hat{x}_i < x < \hat{x}_{i+1}\}, & \text{if } i = 2, \ldots, M-1; \\ \{x : \hat{x}_{i-1} < x < 1\}, & \text{if } i = M \end{cases}$$

where $\hat{x}_i, i = 1, \ldots, M-1$ denote the interfaces between the subdomains.

We consider decomposition of the problem (2.1)–(2.2) over $\{\Omega_i\}$. Let us denote $T_i = T|_{\Omega_i}, i = 1, \ldots, M$, then the original boundary-value problem (2.1)-(2.2) is equivalent to:

$$\frac{d}{dx}\left(A(x,u)\frac{dT_i(x)}{dx}\right) - \frac{h}{k}\frac{dS(x,u)}{dx}T_i(x) = 0 \qquad \text{in } \Omega_i, \quad i = 1, \ldots, M \tag{2.7}$$

$$T_i = u_i \qquad \text{on } \Gamma \cap \partial\Omega_i \tag{2.8}$$

with the following regularity conditions of the solution on the interface:

$$T_i = T_{i+1} \qquad \text{on } \Gamma_i, \quad i = 1, \ldots M-1 \tag{2.9}$$

(continuity of the solution)

$$\frac{\partial T_i}{\partial n} = \frac{\partial T_{i+1}}{\partial n} \qquad \text{on } \Gamma_i, \quad i = 1, \ldots M-1 \tag{2.10}$$

(continuity of the gradient),

where n is the outward unit normal to Γ_i and $\{u_i\}, i = 1, \ldots, M$ represents the set of boundary values of local solutions T_i on Γ_i.

In our approach we will solve subproblems which are Dirichlet on subregions Ω_i, with matching conditions at the interfaces of Neumann type. These subproblems are independent and can be solved in parallel. Many applications, especially two- and three-dimensional ones, use a scheme where the boundary conditions for these subproblems are of Neumann type and the compatibility conditions are of Dirichlet type.

2.4.3 Formulation of the Control Problem

The numerical approximation of the solution to problems (2.7)–(2.8) is carried out by finite-element scheme on each subdomain $\Omega_i, i = 1, \ldots, M$. Since this method assumes the continuously differentiable approximation to the solution, *i.e.*, $y_M^N \subset C^2$, the continuity of the solution, expressed by (2.9) is satisfied if the continuity of the gradient, expressed by (2.10) is satisfied. Let $\hat{T}(x)$ be the target temperature distribution on [0,1] and $T =$

T_i, if $x \in \Omega_i$, where T_i is the solution of (2.7)–(2.8) on the appropriate Ω_i. The optimal control problem (2.4) can be reformulated in the framework of domain decomposition in the following way: *Given* $\hat{T}(x)$, *minimize*

$$J(T,u) = \int_0^1 \left[T(x,u) - \hat{T}(x)\right]^2 dx \tag{2.11}$$

subject to

$$k\frac{dT(x)}{dx}\Big|_{x=\hat{x}_i^-} = k\frac{dT(x)}{dx}\Big|_{x=\hat{x}_i^+} \qquad \text{for } i = 1,\ldots,M-1. \tag{2.12}$$

Equations (2.12) are the constraints of the optimization problem, which express the feasibility requirement that the heat flux across the boundary of each subdomain is continuous.

The dimension of the control space is $M+2$, since in addition to the parameters u_1, u_2 and u_3, describing the geometrical configuration, we have $M-1$ control parameters u_{3+i}, denoting the boundary temperature at each interface point $\hat{x}_i$.

2.4.4 Parallel Implementation of the Domain Decomposition

Referring to the partition of Ω, described above, the solution of the optimal control problem (2.11) can be implemented step by step as follows:

Step 1: Construct the matrices on each subdomain Ω_i

Step 2: Given controls $u_m, m = 1, \ldots, M+2$, construct y_M^N on each Ω_i

Step 3: Calculate the integral (2.11)

Step 4: Evaluate the constraints (2.12) based on the solution computed in Step 2

Step 5: Redo from Step 2 until convergence

More precisely, in Step 1, M different processors can simultaneously compute the matrices associated with finite-element representation of y_M^N on M different subdomains. The degree of parallelism of this step is M. In Step 2, M processors can solve independently each Direchlet problem on Ω_i, so that the degree of parallelism of this step is also M. There is no exchange of information between the processors up to this point. The communication between the processors is restricted to passing the temperature gradients and information necessary to calculate the cost function to the "master" processor, which evaluates the cost function and constraints and updates the controls u_m.

2.4.5 Numerical Results

The parallel scheme, outlined above, was tested on a single-processor machine to analyze the convergence properties and stability of the algorithm. Some conclusions were made about the possible speed-up obtainable on a multi-processor machine. The target temperature distribution $\hat{T}$ for the experiments, described in this section, was taken to be the solution of (2.1)–(2.2) with $R(x,u) = -2x^2 + 4x + 1$ on [0,1] with $N = 8$. Figure (2.4) shows the

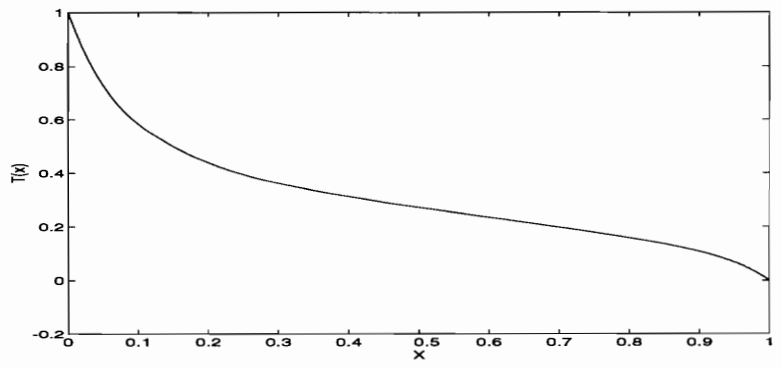


Figure 2.4: Target temperature distribution

temperature distribution for the target set of the shape parameters $u_{target} = (-2, 4, 1)$.

In the case of domain discretization, M has a profound effect on the optimization procedure. In order to solve for $T(u)$ on the whole interval, the optimization code has to solve M boundary-value problems. Of course, one expects a trade-off between the N and M parameters for a given level of accuracy. In Figure (2.5), two different decompositions (2 subdomains and 16 subdomains) are compared with the computation performed on a single grid covering the entire domain. It is seen that significant increase in the number of subproblems does not seriously penalize the number of iterations necessary to reach the optimal solution. In this case the cost-function convergence does not tell the whole story. With domain decomposition, the "unconverged" temperature profiles are not feasible (nor physical) since the gradient matching conditions are not satisfied. That is, the SQP procedure moves to optimality and feasibility at the same time.

Figure (2.6) shows the results of optimization for different number of partitioning points.

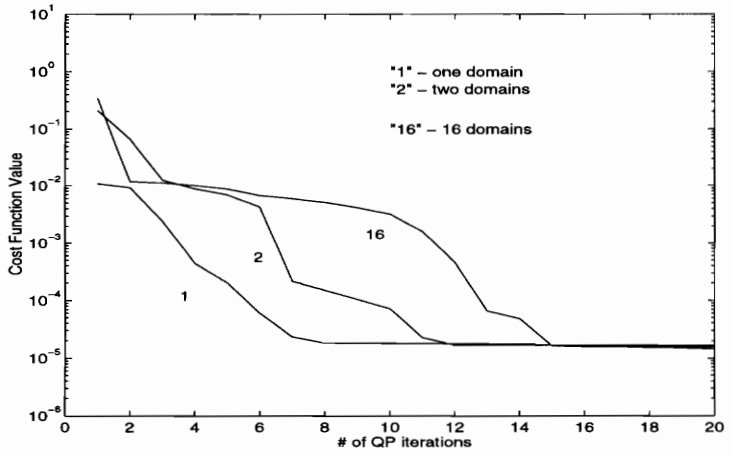


Figure 2.5: Convergence history of the cost function

The first graph depicts the number of iterations and the second graph shows the relative amount of total time spent on the solution of (2.7). The initial guess ($u_1^0 = -1, u_2^0 = 7, u_3^0 = 5$) for the geometrical parameters u_1, u_2, u_3 was kept the same for different number of subdomains. There is some degree of arbitrariness here, since as M increases initial estimates are needed for the boundary temperature parameters $u_{3+i}, i = 1, \ldots, M-1$. Figure (2.6) shows only 1 possible outcome of the optimization procedure.

Table (2.4.5) shows a comparison of computational effort required for the solution of the optimizaiton problem by domain decomposition method and by "Black-Box" approach. The size of the problem was the same for both methods in the sence that the total number of discretization nodes required for the representation of y_M^N was the same. In each case the number of finite-element nodes was 8 ($N = 8$) for each subdomain problem, and the

number of finite elements for the one-domain problem was proportional to the number of subdomains for that case.

Table 2.4.5 : Comparison of computational effort for one-domain and multi-domain optimization

# of domains/ finite elements	cost function value	number of iterations	elapsed time	% of total time spent on BVP solution
2	0.2595×10^{-5}	43	33.19 sec	99.83
16	0.2984×10^{-12}	47	32.23 sec	99.07
3	0.2577×10^{-5}	35	51.37 sec	99.87
24	0.5396×10^{-12}	33	43.74 sec	99.87
4	0.2564×10^{-5}	33	69.60 sec	99.91
32	0.3197×10^{-13}	30	58.42 sec	99.59

One of the major advantages of domain decomposition method, illustrated above, is represented by its potential for parallelization. Although the application of the domain decomposition method on a single-processor machine takes considerably longer than the solution of the problem of the same size on the entire domain, this scheme is very promising for implementation on a network of workstations. Each subdomain can be assigned to a separate processor, with one "master" processor performing the global iteration and synchronizing the process. A linear speed-up can be expected if the computational domain

is partitioned in a number of subdomains, proportional to the number of processors in the network. If the local solver allows the boundary data to be imposed explicitly, this method can be used to parallelize existing analysis codes without making significant changes in the code. If only coarse-grained parallelization is considered, application of domain decomposition is a strong alternative to creating new parallel algorithms.

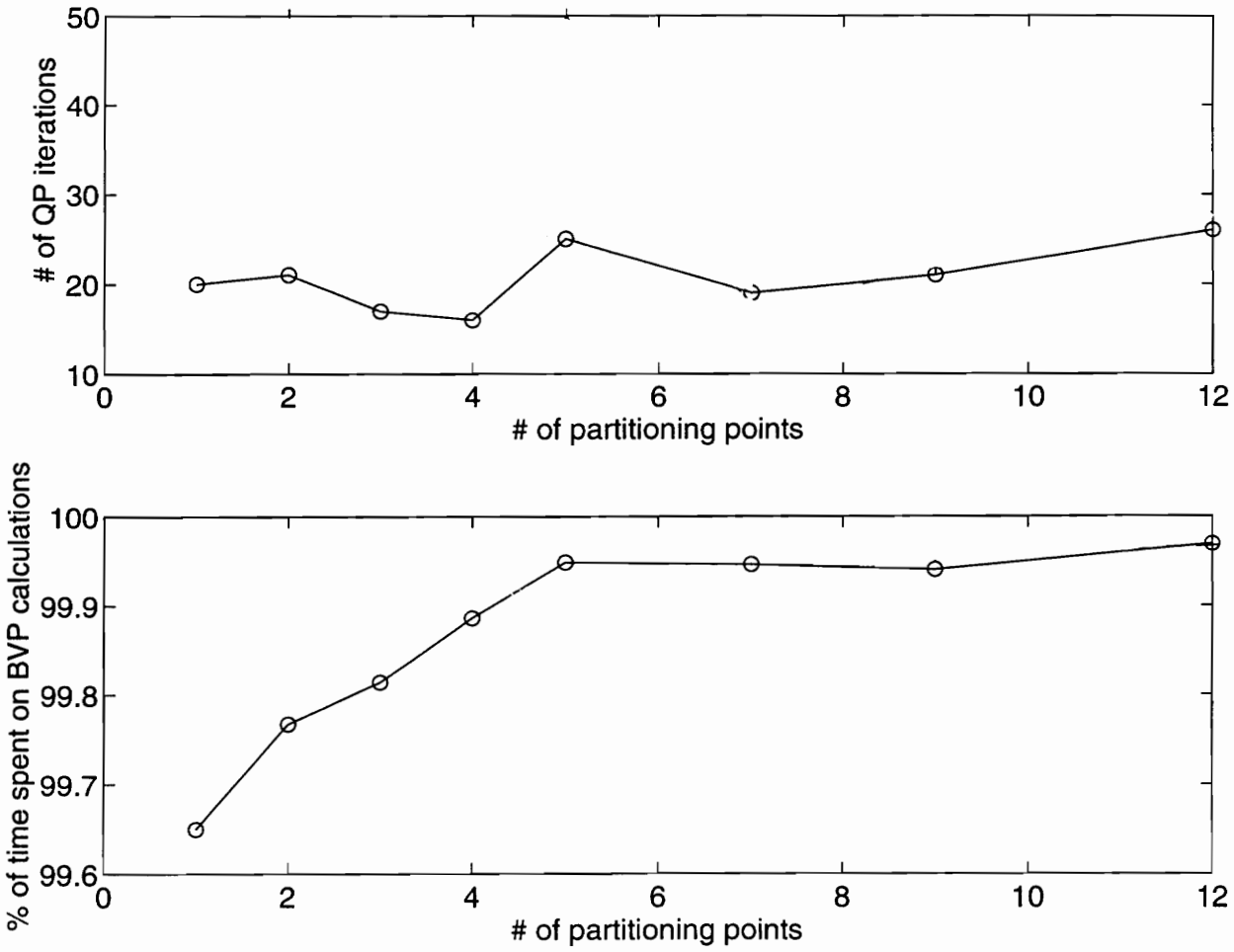


Figure 2.6: Results for domain decomposition method

Chapter 3

Heat Equation: Chimera Scheme

One of the attractive features of domain decomposition methods is that they allow the subdivision of the computational domain into subregions which admit a more easily constructed mesh. J. Benek *et al* [**2**], [**12**], [**13**] developed a grid-embedding approach for solution of complex flows which provides the flexibility to employ boundary-comforming grids on component parts of geometry, to refine the mesh selectively in regions of interest, and to permit the solution of different flow models on the component grids. The concept is based upon the subdivision of the computational domain into regions that encompass simplified geometry, simplified fluid dynamics models, or both. Because of its structural diversity, the authors have called it a chimera method after the creature from Greek mythology.

3.1 General Concept

The chimera scheme is a grid-embedding or oversetting technique, which, contrary to non-overlapping domain decomposition methods, does not require common boundaries between subdomains, but rather, a common or overlap region is required to provide the means of matching the solutions across boundary interfaces. Solution proceeds on each grid separately with boundary conditions for the embedded meshes supplied from the solution at interior points of the containing meshes. The chimera scheme is general in that it allows solid surfaces to be embedded within the computational domains of other meshes. In addition, overlapping mesh outer boundaries are allowed.

The concept behind the chimera scheme is illustrated in Figure (3.1), which depicts two independently generated meshes modeling a flapped airfoil. The flap mesh is embedded within the airfoil mesh. Clearly, the flap mesh outer boundary can receive flow field information interpolated from appropriate mesh elements of the airfoil grid. However, a reverse process must occur as well; the flap grid must pass flow information to the airfoil mesh. Since the airfoil mesh has no boundary through which flow field information can be obtained from the flap mesh, an artificial boundary is defined within the airfoil mesh. Grid points on the artificial boundary which are defined within the airfoil mesh and which are fully contained within the computational region of the flap mesh can be updated by interpolation from the appropriate mesh elements of the flap grid. Generally, any grid

can receive information from other grids through outer boundary and artificial boundary points.

The interpolation process is further illustrated in Figure (3.2), which depicts a portion of the overlap region between the airfoil and the flap. Airfoil mesh points which are within a certain region surrounding the flap are excluded from the computational domain of the airfoil mesh (they are called "hole" points). The points in the airfoil mesh surrounding the "hole" points are hole boundary points which receive flow-field information interpolated from mesh elements within the airfoil mesh [**11**].

The chimera procedure has two major parts, (1) generation of the composite mesh and associated interpolation data and (2) solution of the flow model or models on the composite mesh.

3.2 Composite Mesh Generation

Automatic generation of a composite mesh from the input component grids requires

(1) establishing the proper lines of communication among the grids

through appropriate data structures,

(2) construction of holes within grids,

(3) identification of points within holes,

(4) location of points from which the boundary values can be interpolated

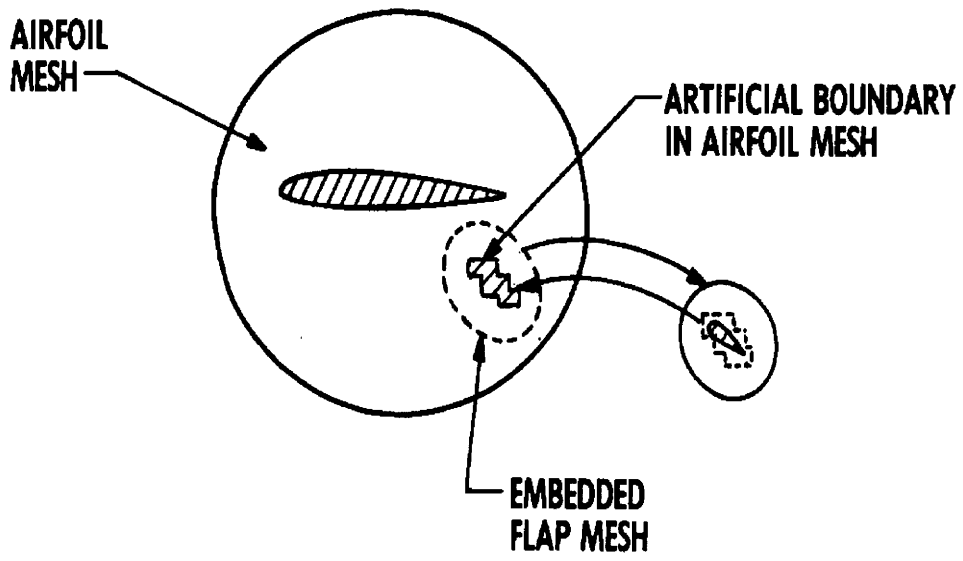


Figure 3.1: Illustration of the Chimera Scheme

(5) evaluation of interpolation parameters.

3.2.1 Embedding Hierarchy

In order to implement the chimera method, it is necessary to impose some structure on the collection of grids. A hierarchical form follows naturally from the embedding process – a given mesh appears to be at a more fundamental level than those grids embedded within it. Hierarchical forms also have convenient representation as graphs. To facilitate the discussion, we introduce the following nomenclature: grids which comprise level l of the hierarchy are denoted $G_{l,i}$, $i = 2, \ldots$. In general, grids on a given level l are embedded within grids on level $l-1$, overlap other grids on level l, and have one or more grids on level

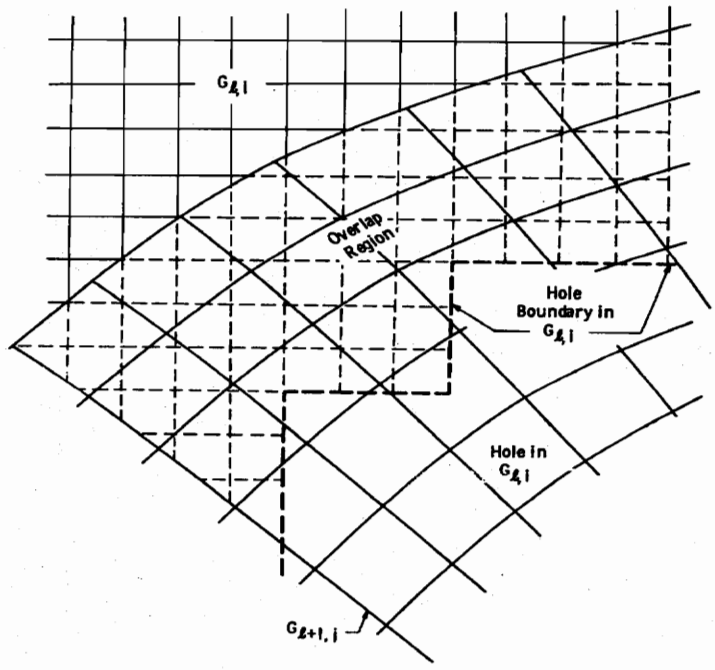


Figure 3.2: Hierarchical structure of embedded grids

$l+1$ embedded within them. Such an arrangement is shown in Figure (3.3). The Figure includes the corresponding graph, where lines connecting the grids indicate the intergrid communication links that must be supported by data structures.

3.2.2 Hole Generation

If $G_{l+1,i}$ is embedded in $G_{l,i}$, it is likely that some points of $G_{l,i}$ will fall within a solid boundary of $G_{l+1,i}$ or within a mesh contained within $G_{l+1,i}$. In this case, such points of $G_{l,i}$ will be outside the computational domain or will violate the constraints on the grid hierarchy. In addition, a large number of points must be interpolated if every point

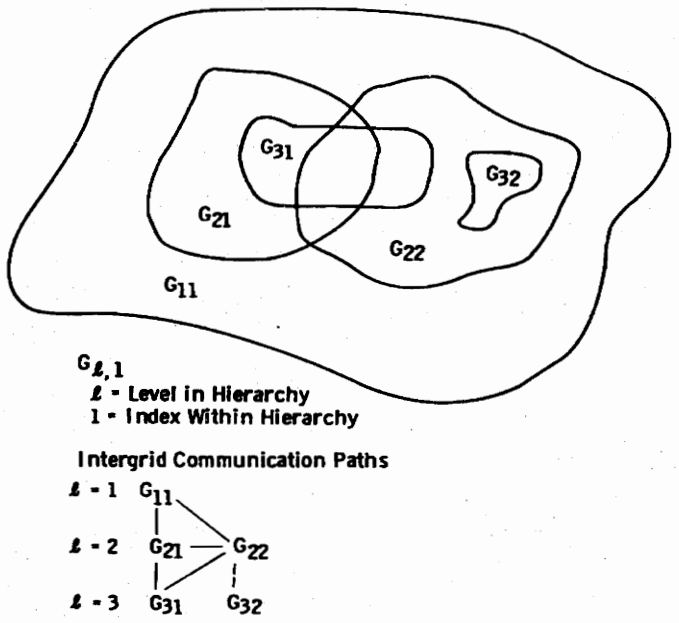


Figure 3.3: Overlap region between grids

common to $G_{l,i}$ and $G_{l+1,i}$ is to be updated. To avoid these difficulties, only the boundary of each embedded grid is updated; the points of $G_{l,i}$ contained within a subregion of $G_{l+1,i}$ are excluded from the solution on $G_{l,i}$. Thus, the embedded mesh, $G_{l+1,i}$ introduces an artificial "hole" into $G_{l,i}$. The only computational requirement is that there is sufficient overlap (*i.e.*, points in $G_{l,i} \bigcap G_{l+1,i}$ and exterior to the hole) to support an interpolation for the outer boundary of $G_{l+1,i}$ from points in $G_{l,i}$ (see figure (3.2)). Similarly, the overlap must be sufficient to allow the hole boundary in $G_{l,i}$ to be interpolated from points in $G_{l+1,i}$. A minimum overlap exists that is dependent on the type of interpolation used.

A hole is constructed as follows: a surface , C, is introduced into $G_{l,i}$, which can be

conveniently defined by level curves in $G_{l+1,i}$. This surface serves as an initial hole boundary. Once the boundary curve C has been constructed, a search in $G_{l,i}$ is made to determine those points that lie within C. These points are marked by changing the value of an integer array, IBLANK, corresponding to these points, from 1 to 0.

Figures (3.4)-(3.7) illustrates the details of the search procedure in two dimensions. The procedure is as follows:

(1) Define the initial hole boundary by a level curve in $G_{l+1,i}$ (Figure (3.4)).

(2) Construct outward normals, $\bar{N}$, at each point, P_c, defining C (Figure (3.5)).

(3) Determine a temporary origin ,say P_0, located within C by averaging the coordinates. We assume here that the origin P_0 lies inside the surface C.

(4) Define a "search" circle about P_0 with radius R_{max}, where R_{max} is the maximum distance from P_0 to the points on C (Figure (3.6)).

(5) Test the magnitude of $\bar{r}$, the position vector relative to P_0 for every point P of $G_{l,i}$. If $|\bar{r}| \geq R_{max}$, P lies outside the search circle and hence need not be considered further. Whenever $|\bar{r}| \leq R_{max}$, P falls within the search circle and additional testing is required.

(6) Compute $\bar{N} \cdot \bar{R}_p$ where $\bar{N}$ is the outward normal at the point P_c on C closest to P, and $\bar{R}_p$ is the position vector to P from P_c (Figure (3.7)). If $\bar{N} \cdot \bar{R}_p \geq 0$, P is outside C; if $\bar{N} \cdot \bar{R}_p < 0$, P is inside C and IBLANK corresponding to this point is set to 0.

The points of $G_{l,i}$ within the hole are excluded from the solution and are not usable as boundary points. Therefore, additional points of $G_{l,i}$ are identified as hole-boundary or

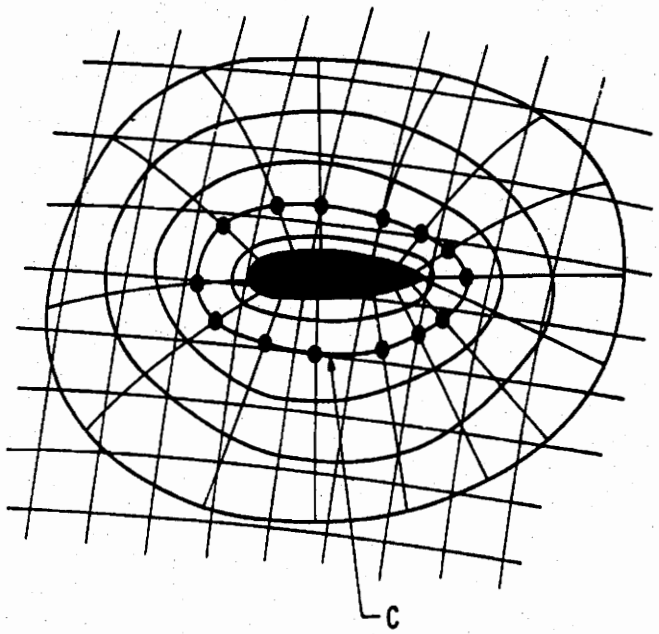


Figure 3.4: Initial hole boundary defined by level curve, C

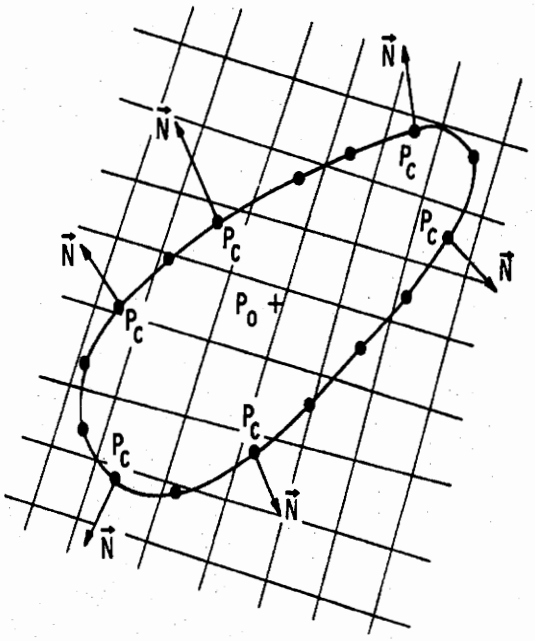


Figure 3.5: Construction of outward normal to C

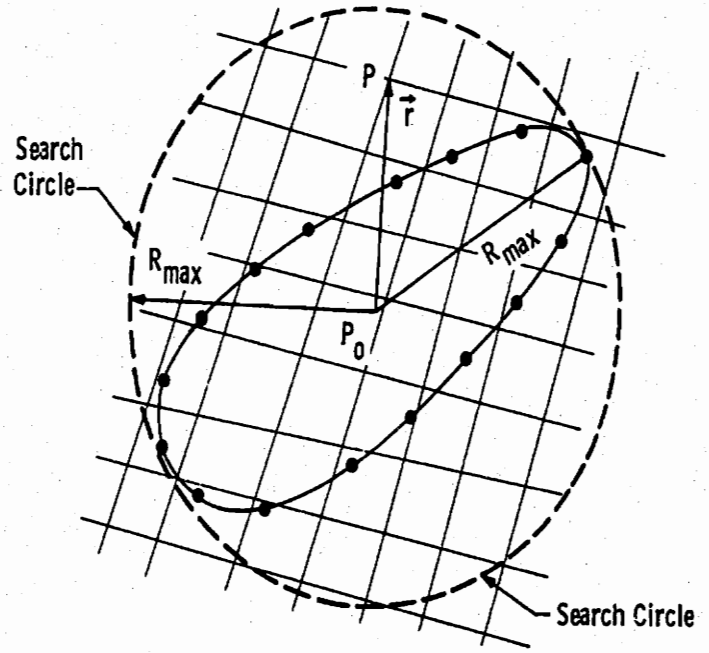


Figure 3.6: Temporary origin (P_0) and construction of search circle

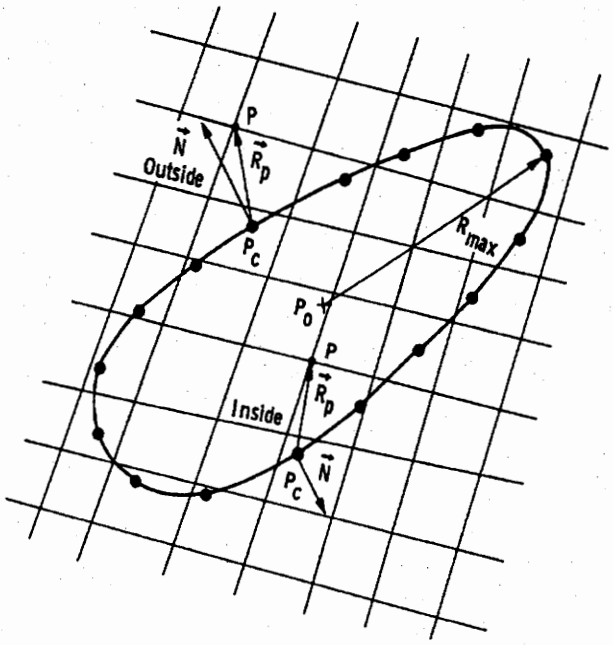


Figure 3.7: Construction of position vector $\vec{R}_p$ and dot product test

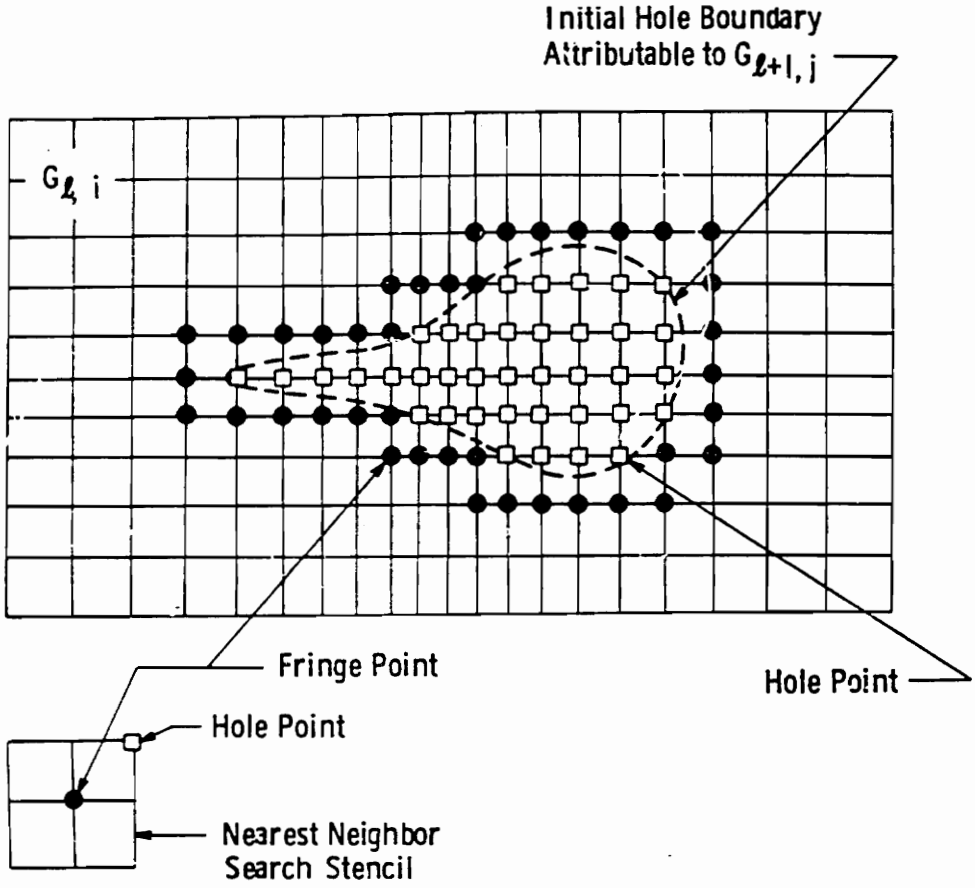


Figure 3.8: Final hole-boundary construction

fringe points. Values of the unknowns at these boundary points will be interpolated from the embedded mesh, $G_{l+1,i}$. The boundary points are constructed from points in $G_{l,i}$ which are not boundary points but which have nearest neighbors that are. Figure (3.8) illustrates the boundary construction. The procedure is to examine the nearest neighbors of each point, P, in $G_{l,i}$, at which IBLANK = 1. If a neighbor is a hole point, P is a boundary point. The indices of the fringe points are added to a list of boundary points which will require interpolated data.

3.3 Interpolation

Because the separate grids are treated as independent entities, boundary conditions must be supplied to each. The boundary conditions of the differential equations which model the flow provide data only at the boundaries of the computational domain. Thus, other data must be obtained for the subdomain boundaries which are not coincident with those of the computational domain. Because the subdomain boundaries typically lie in the interior of the computational domain where the differential equations are valid, it seems appropriate that the solution of these equations provide the necessary boundary data. These data can be obtained by interpolation of the solution in one mesh to provide boundary data for another, so that communication among grids is achieved by interpolation of boundary values from grids in which the boundaries are contained. The authors have employed a first-order accurate Taylor series approximation of the form

$$Q_B = Q_O + Q_X \delta x + Q_Y \delta y$$

for 2-D case where Q_B is the required boundary value. When higher order derivatives of the solution are not important, bilinear interpolation can be used instead of Taylor series expansion in two dimensions. In numerical examples for the one-dimensional problem, presented below, spline interpolation was used.

A four-to-five points overlap between grid boundaries was maintained in the original work [**2**] in order to accurately simulate complex two- and three-dimensional flows.

3.4 Applications

We attempted to incorporate the concept of the chimera grid-embedding scheme into the solution of the control problem, described by the boundary-value problem (2.1)-(2.2) and the cost function (2.4). Although the generation of the composite mesh, which is one of the key components of the chimera technique, is significantly simplified in one dimension, we preserved the original structure and the main aspects of the method.

The numerical example of the chimera approach was constructed in the following way. We consider a solid cylinder, whose radial distribution is given by $R(x) = -2x^2 + 4x + 1$ on [0,1] and whose thermal conductivity coefficient k is:

$$k = \begin{cases} 1, & \text{if } 0.29 < x < 0.79 \\ 0.5, & \text{if } 0 \leq x \leq 0.29 \text{ and } 0.79 \leq x \leq 1 \end{cases}$$

The temperature distribution in the cylinder is given by the solution of the boundary-value problem (2.1)–(2.2). The task is to find this solution.

We consider a two-level grid structure with the coarse grid being defined on $\Omega = \{x : 0 \leq x \leq 1\}$ and the embedded grid being defined on $\Omega^* = \{x : 0.29 \leq x \leq 0.79\}$. The partition of Ω into 10 elements and the partition of Ω^* into 20 elements is shown in Figure(3.9 a). These two meshes comprise the input to the composite grid generation code. The output is the grid, constructed according to the procedure, described in (3.2)-(3.3). It is shown in Figure (3.9 b). Here, e_1, e_2 denote the embedded grid boundaries and h_1, h_2 denote the

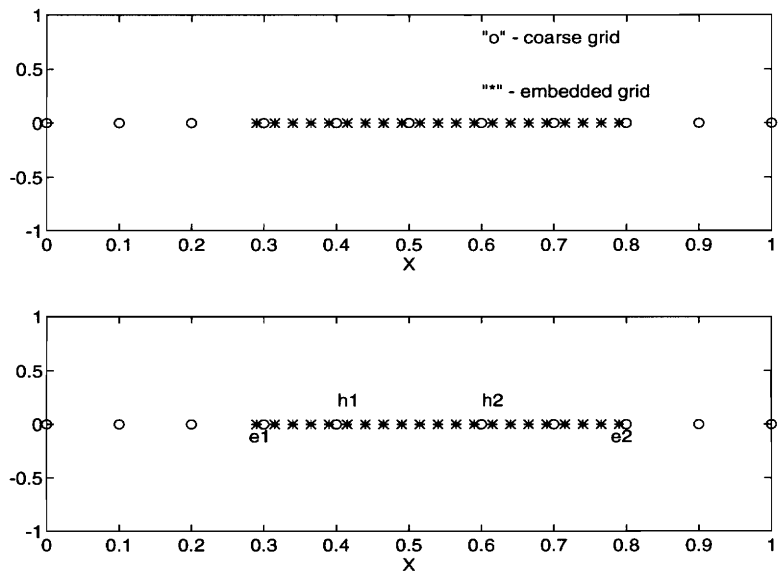


Figure 3.9: Illustration of the chimera approach

hole boundaries.

In order to employ domain decomposition principles to the composite grid, we divide it into three overlapping regions $\Omega_1, \Omega_2, \Omega_3$:

$$\begin{aligned} \Omega_1 &= \{x : 0 \leq x \leq h_1\}, \\ \Omega_2 &= \{x : e_1 \leq x \leq e_2\}, \\ \Omega_3 &= \{x : h_2 \leq x \leq 1\}, \end{aligned} \tag{3.1}$$

Here, Ω_2 coincides with the embedded grid Ω^*, Ω_1 is the part of the coarse grid Ω with its right boundary defined by the hole boundary h_1, and Ω_3 is the part of the coarse grid Ω with its left boundary defined by the hole boundary h_2.

We introduce the control parameters θ_1 and θ_2, denoting the temperature values on the

boundaries of the embedded grid Ω_2 at the points e_1 and e_2, respectively. Let $T_i = T|_{\Omega_i}, i = 1, \ldots, 3$ then the original boundary-value problem (2.1)–(2.2) is equivalent to the following problem:

Find θ_1 and θ_2 such that

$$\frac{d}{dx}\left(k(x)A(x,u)\frac{dT_2(x)}{dx}\right) - h\frac{dS(x,u)}{dx}T_2(x) = 0 \qquad \text{in } \Omega_2, \tag{3.2}$$

$$T_2(e_1) = \theta_1 \qquad T_2(e_2) = \theta_2 \tag{3.3}$$

$$\frac{d}{dx}\left(k(x)A(x,u)\frac{dT_1(x)}{dx}\right) - h\frac{dS(x,u)}{dx}T_1(x) = 0 \qquad \text{in } \Omega_1, \tag{3.4}$$

$$T_1(0) = 1 \qquad T_1(h_1) = T_2(h_1) \tag{3.5}$$

$$\frac{d}{dx}\left(k(x)A(x,u)\frac{dT_3(x)}{dx}\right) - h\frac{dS(x,u)}{dx}T_3(x) = 0 \qquad \text{in } \Omega_3, \tag{3.6}$$

$$T_3(h_2) = T_2(h_2) \qquad T_3(1) = 0 \tag{3.7}$$

and

$$k\frac{\partial T_1}{\partial x} = k\frac{\partial T_2}{\partial x} \qquad \text{at } x = e_1, \tag{3.8}$$

$$k\frac{\partial T_2}{\partial x} = k\frac{\partial T_3}{\partial x} \qquad \text{at } x = e_2. \tag{3.9}$$

The last two equations express the continuity of the heat flux across the boundaries of the embedded subdomain. The cost function definition of this requirement is given by:

$$J(\theta_1, \theta_2) = \left(k\frac{\partial T_1(e_1)}{\partial x} - k\frac{\partial T_2(e_1)}{\partial x}\right)^2 + \left(k\frac{\partial T_2(e_2)}{\partial x} - k\frac{\partial T_3(e_2)}{\partial x}\right)^2 = 0. \tag{3.10}$$

Note that the temperature conductivity k does not have to be continuous in the above expressions, and in our problem it assumes different values for the subdomains Ω_2 and Ω_1, Ω_3.

Suppose, we have an initial guess for control parameters (θ_1^0, θ_2^0). We solve the boundary-value problem (3.2)–(3.3) and calculate the heat fluxes through the boundaries of the subdomain Ω_2 using these values. With this newly computed local solution T_2 we evaluate boundary values $T_1(h_1)$ and $T_3(h_2)$, necessary to solve problems (3.4)– (3.5) and (3.6)–(3.7). We calculate the heat fluxes, entering the subdomain Ω_2, based on the temperature values at e_1 and e_2, obtained from the local solutions T_1 and T_3, respectively. The cost function (3.10) is evaluated by taking the squared differences of the fluxes computed at this step and the fluxes computed at the previous stage. Iterating this procedure until convergence, we obtain the global solution on [0,1]. The SQP code, described in the introduction, was used to calculate the converged values of θ_1, θ_2. Note, that the transfer of data between subdomains Ω_1, Ω_2 and Ω_2, Ω_3 is required at each iteration step. The solution of (3.4) and (3.6) must follow the solution of (3.2). Therefore, the degree of parallelism of this scheme is lower than that in the case of non-overlapping domains.

The temperature distribution, obtained with this procedure, is shown in figure (3.10). The initial parameters were $(\theta_1^0, \theta_2^0) = (0.5, 0.5)$ and the converged values were $(\theta_1^*, \theta_2^*) = (0.24363, 0.083214)$.

We applied the grid-embedding scheme, described above, to the shape optimization

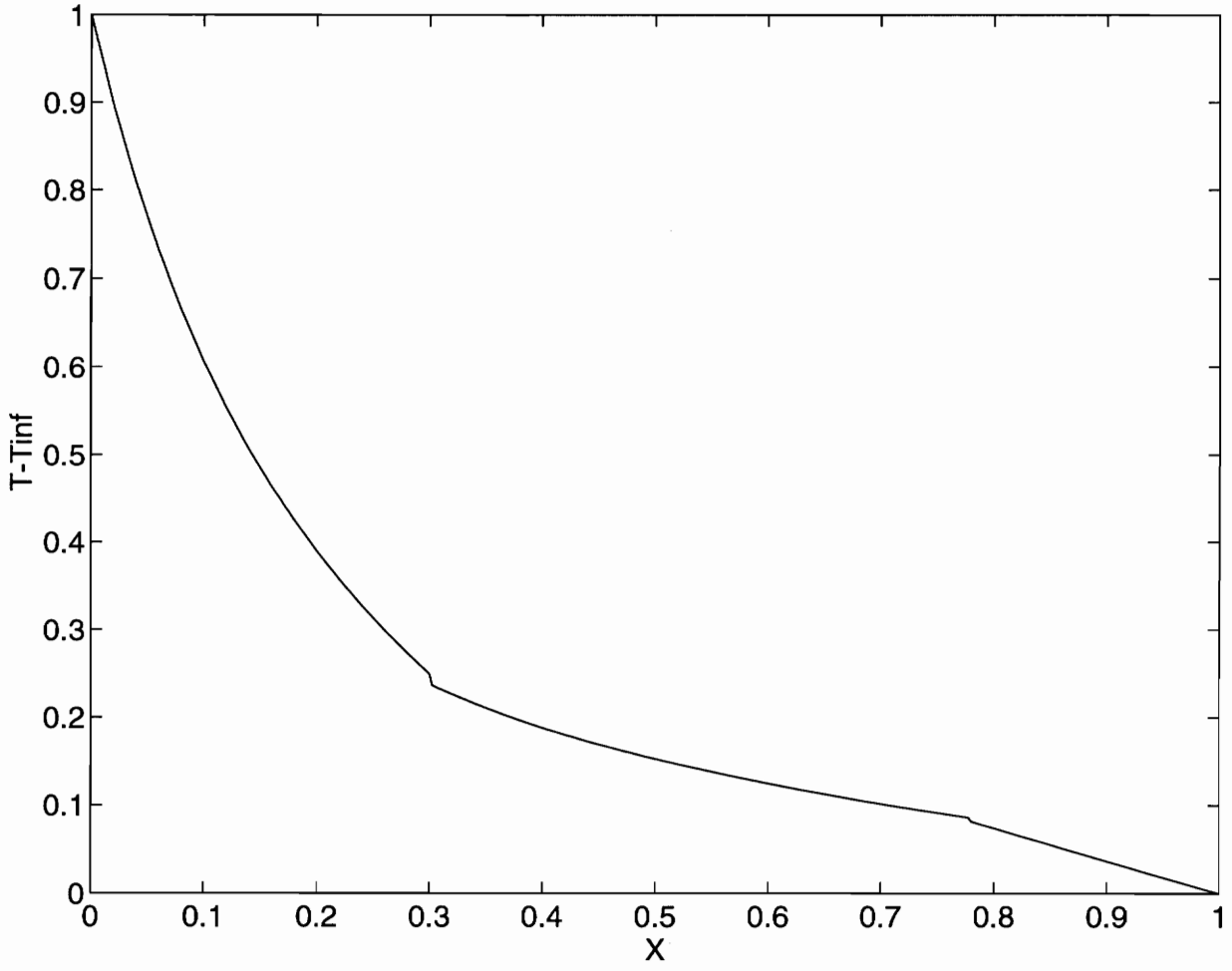


Figure 3.10: Temperature profile, computed with the chimera scheme

problem (2.4). The original computational region [0,1] was decomposed into 3 overlapping domains in the manner, described above. The control parameters for this case are $(u_1, u_2, u_3, \theta_1, \theta_2)$. The target distribution, $\hat{T}(x)$, was computed as an exact solution for (u_1, u_2, u_3) given by (-2,4,1). The constraints, expressing the continuity of the heat flux through the boundaries of the embedded subdomain are:

$$k\frac{dT(x)}{dx}\Big|_{x=e_i^-} = k\frac{dT(x)}{dx}\Big|_{x=e_i^+} \qquad \text{for } i = 1, 2. \tag{3.11}$$

Before the start of the optimization procedure, the chimera subroutine is called to create the composite mesh and identify the boundaries of the subdomains Ω_1 and Ω_3. Unlike the non-overlapping method, described in (2.4), in this scheme the subdomain problems are not entirely independent at each iteration step. The transfer of boundary data is required from the embedded subdomain Ω_2 to the outer domains Ω_1 and Ω_3.

The results of this procedure for several initial sets of control parameters are shown in Table (3.4). Although the converged values, found by the optimization code are not always very close to the target shape parameters $(\hat{u}_1, \hat{u}_2, \hat{u}_3) = (-2, 4, 1)$, the cost function achieved very small value and the constraints were satisfied with a good accuracy, which suggests that these target parameters $(\hat{u}_1, \hat{u}_2, \hat{u}_3)$ are not the only feasible set of the shape parameters, which yields the target temperature distribution $\hat{T}$. The gradients of the cost function and constraints in this example were computed by central-differences approximation.

On this simple example we showed that the chimera approach, which can significantly

simplify the grid generation procedure and reduce storage requirements for complex flows, can be successfully incorporated into the optimization-based design framework and that the original analysis problem can be split into the problems on subdomains, which can be solved in parallel.

Table 3.4 : Results of Optimization for the Chimera Scheme

$u_{target} = (-2, 4, 1)$

initial guess	approximation of solution	cost function value	constraint value	number of iterations	elapsed time
-1	-4.2712348				
7	8.6399413	0.74228769 $\times 10^{-9}$	-0.78757001$\times 10^{-11}$	11	26.43 sec
5	6.4391778		0.69743100 $\times 10^{-9}$		
0.8	0.54644045				
0.2	0.13604237				
-2	-2.2612347				
2	4.5360490	0.16381624 $\times 10^{-9}$	0.72600694 $\times 10^{-8}$	8	19.8 sec
2	3.3959106		0.30118956 $\times 10^{-7}$		
0.5	0.54643526				
0.3	0.13606525				

Chapter 4

Optimal Design of a Two-Dimensional Nozzle

4.1 Problem Statement

We consider an inviscid compressible steady flow governed by the Euler equations. We are specifically interested in the flow in a two-dimensional nozzle, described by the system:

$$\frac{\partial}{\partial t}\iiint Q dV + \iint F \cdot \hat{n} ds = 0, \tag{4.1}$$

where

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_0 \end{pmatrix},$$

with velocity components u, v, density ρ, total energy per unit mass $e_0 = e + (u^2 + v^2)/2$, where e is the internal energy per unit mass.

With the appropriate choice of inflow and outflow boundary conditions and the shape of the nozzle, the flow inside the nozzle can be supersonic, subsonic throughout the computational domain, or it can contain a shock. The goal of the optimal design problem is to find the shape of the cross-sectional area of the duct that minimizes the discrepancy between the flow, calculated according to (4.1) and a desired flow.

We consider the inverse design problem. Let $\hat{N}$ denote the target nozzle and let $\hat{Q} = (\hat{\rho}, \hat{u}, \hat{v}, \hat{p})$ be the given target flow in $\hat{N}$. Let N denote a nozzle having its end dimensions specified and let $y = A(x)$ denote the lower boundary of N. The corresponding flow in N is denoted by $Q = (\rho, u, v, p)$. We consider the following optimization problem: *given* $\hat{Q}$ *in* $\hat{N}$ *find* $A(x)$ *such that*

$$(4.2)\quad J(A) = w_1 \int_C [\rho - \hat{\rho}]^2 ds + w_2 \int_C [u - \hat{u}]^2 ds + w_3 \int_C [v - \hat{v}]^2 ds + w_4 \int_C [p - \hat{p}]^2 ds$$

is minimized.

The nozzle is assumed to be symmetric with respect to its centerline, so that the inte-

gration is performed along the centerline C and the flow only in the lower half of the nozzle is considered. The $w_i, i = 1, \ldots, 4$ are weights that can be chosen to better match all the components of $\hat{Q}$ and to scale the numerical value of the cost function J.

4.2 Geometry of The Nozzle

We consider a family of symmetric nozzles, with their lower boundary $y = A(x)$ being described by the following function [**3**] :

$$A(x) = A_l + (A_r - A_l)(1 + \tanh(\tan(x - 0.5)\pi) - \alpha_1))/2, \tag{4.3}$$

where A_l is the width of the inlet side and A_r is the width of the outlet side of the nozzle and α_1 is the design parameter. Thus the cost function J in (4.2) may be viewed as a function of α_1, i.e. $J = J(\alpha_1)$. The target domain $\hat{N}$ and the candidate optimal domains are defined by 4.3 with $A_l = 0.125, A_r = 0.1875$. Figure (4.1) shows the shape of the nozzle for various parameters α_1 in the range $(-1, 1)$.

4.3 The Flow Solver

The following section describes the computational procedures implemented in the flow solver ErICA (Euler Implicit Code for Aerodynamics) [**19**] to obtain analysis solutions for the flow in the nozzle.

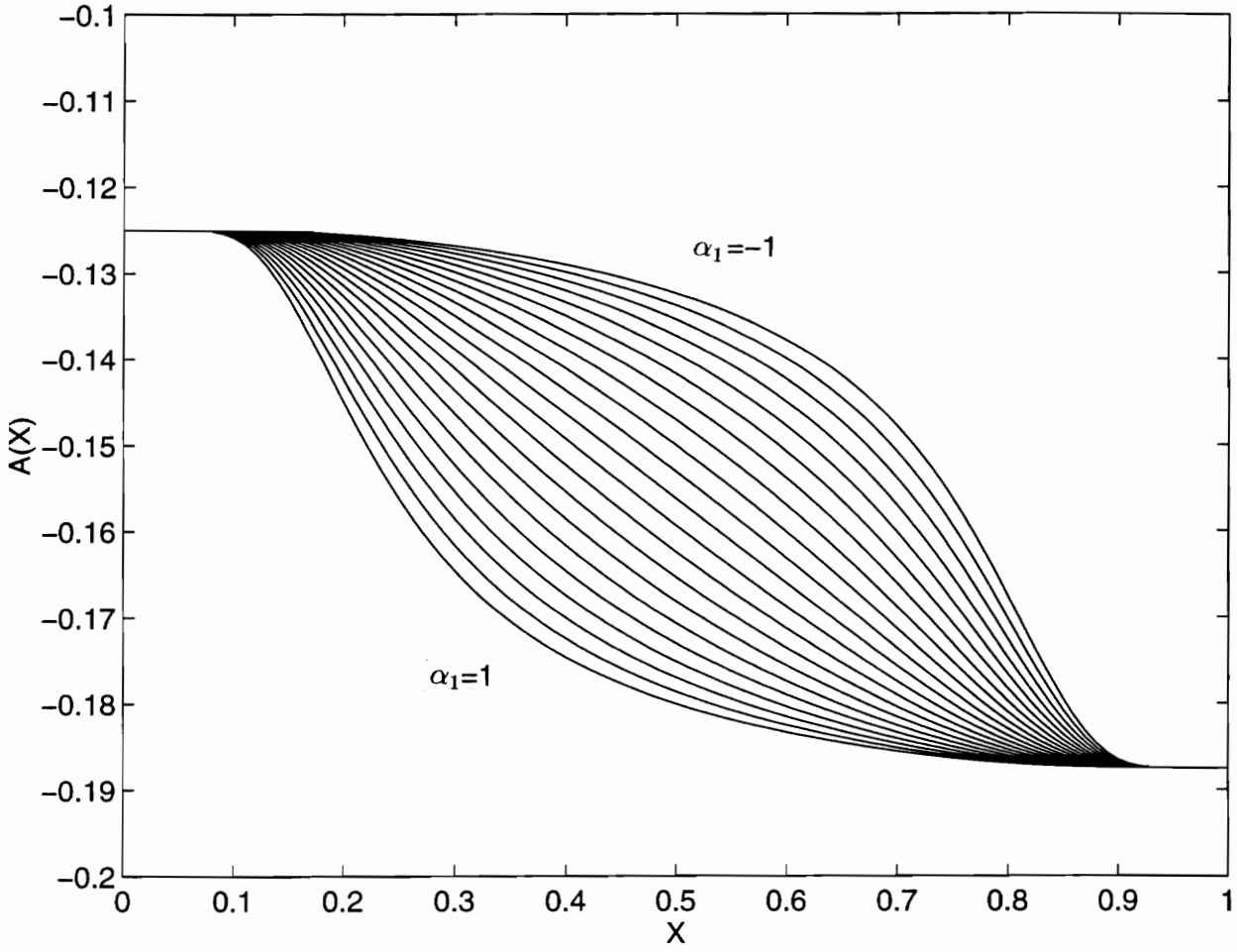


Figure 4.1: Nozzle shape for different α_1.

4.3.1 Governing Equations

The flow inside the two-dimensional duct is governed by the Euler equations for a perfect gas, which may be written in integral conservation law form, valid across the discontinuities, in Cartesian coordinates as:

$$\frac{\partial}{\partial t}\iiint Q dV + \iint F \cdot \hat{n} ds = 0, \tag{4.4}$$

where

$$Q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_0 \end{pmatrix},$$

$$\hat{F} = F\hat{j} + G\hat{k},$$

$$F = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e_0 + p)u) \end{pmatrix} \qquad G = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e_0 + p)v \end{pmatrix}$$

where ρ is density, u is the x-component of the velocity, v is the y-component of the velocity, e_0 is the total energy per unit mass, and p is pressure. The ideal gas equation of state closes

the system. It has the form:

$$p = \frac{\rho a^2}{\gamma} = (\gamma - 1)\rho e = \rho RT$$

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.

4.3.2 Problem Discretization

The code implements a cell-centered finite volume method, which is based upon an integral form of equations to be solved. The computational domain is divided into elementary quadrilateral volumes within which the integration is carried out. Figure (4.2) shows a typical 30×10 grid, which is used in computations below.

Equation (4.4) is applied to a discretized flow field. The state variables are assumed to be constant within each cell. In a finite-volume formulation, (4.4) is applied to each cell volume and in two-dimensions the semi-discrete form is:

$$\frac{\partial}{\partial t}(QV)_{j,k} + \sum_{sides} (\hat{F} \cdot \hat{n}) ds = 0. \tag{4.5}$$

A typical structured grid has cell center (j, k) and boundary faces labeled $j - 1/2, j + 1/2, k - 1/2, k + 1/2$, as shown in Figure (4.3).

Using this cell structure, equation (4.5) becomes:

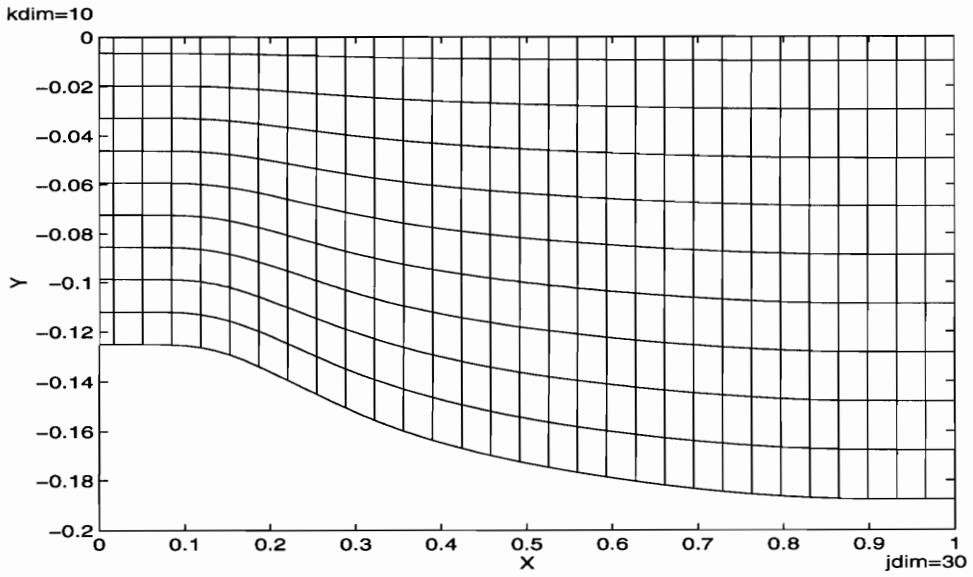


Figure 4.2: Computational Grid

$$\frac{\partial}{\partial t}(QV)_{j,k} + (\hat{F}\cdot\hat{n})ds_{j-1/2} + (\hat{F}\cdot\hat{n})ds_{j+1/2} +$$

$$(\hat{F}\cdot\hat{n})ds_{k-1/2} + (\hat{F}\cdot\hat{n})ds_{k+1/2} = 0, \tag{4.6}$$

where the unit normal $\hat{n}$ is positive outward.

4.3.3 Time Integration

The analysis flow problem is solved by time marching to a steady state. The governing equations of fluid flow can be rewritten for the (j,k)th cell in terms of the semi-discretized residual R as:

$$Q_t + R(Q) = 0,$$

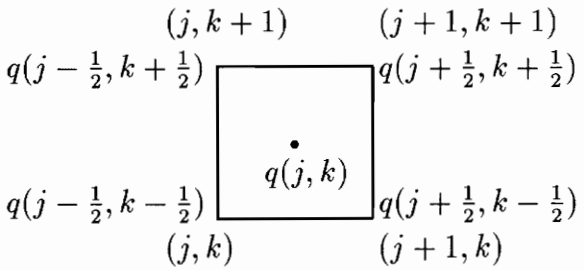


Figure 4.3: Finite-volume scheme: $(j,k)th$ cell

where R is:

$$
\begin{aligned}
R_{j,k}(Q) \;=\; & \frac{1}{V_{j,k}}\left(\left[\hat{F}(\hat{n}\cdot ds)\right]_{j-1/2} + \left[\hat{F}(\hat{n}\cdot ds)\right]_{j+1/2}\right) + \\
& \frac{1}{V_{j,k}}\left(\left[\hat{F}(\hat{n}\cdot ds)\right]_{k-1/2} + \left[\hat{F}(\hat{n}\cdot ds)\right]_{k+1/2}\right),
\end{aligned}
\tag{4.7}
$$

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.

The equation integrated in ErICA makes use of the set of primitive variables $[\rho, u, v, p]^T$. We write the time derivative using the chain rule as

$$Q_t = \frac{\partial Q}{\partial q} q_t = M q_t,$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & 0 & \rho & 0 \\ q^2/2 & \rho u & \rho V & 1/(\gamma - 1). \end{bmatrix}$$

In terms of primitive variables, the governing equations can be written

$$q_t + M^{-1} R(q) = 0,$$

where

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ \frac{(\gamma-1)(u^2+v^2)}{2} & -(\gamma-1)u & -(\gamma-1)v & (\gamma-1) \end{bmatrix}.$$

This equation is integrated explicitly in ErICA by a Jameson-style, multi-stage Runge-Kutta routine [**18**].

4.3.4 Upwind Differencing

Information propagates 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. In Godunov-type

schemes, known as flux difference splitting methods, 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 or 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 flux difference vector [**18**]. Roe proposed

$$\hat{F}^{+}_{j+1/2} = 1/2\left(\hat{F}^{+} + \hat{F}^{-} - |\tilde{A}|(Q^{+} - Q^{-})\right), \tag{4.8}$$

where

$$|\tilde{A}| = \tilde{T}|\Lambda|\tilde{T}^{-1},$$

and the subscripts + and - denote the right and left states at the cell interfaces. The matrix $\tilde{T}$ corresponds to the matrix of right eigenvectors of Λ written in columns

$$\tilde{T} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ \tilde{u} & \hat{n}_y & \tilde{u} + \tilde{a}\hat{n}_x & \tilde{u} - \tilde{a}\hat{n}_x \\ \tilde{v} & -\hat{n}_x & \tilde{v} + \tilde{a}\hat{n}_y & \tilde{v} - \tilde{a}\hat{n}_y \\ \frac{\tilde{q}^2}{2} & \tilde{\bar{v}} & \tilde{h}_0 + \tilde{a}\tilde{\bar{u}} & \tilde{h}_0 - \tilde{a}\tilde{\bar{u}} \end{bmatrix}, \tag{4.9}$$

where h_0 denotes the total enthalpy, and u and v are contravariant velocities defined as

$$\begin{aligned} \bar{u} &= \hat{n}_x u + \hat{n}_y v, \\ \bar{v} &= \hat{n}_y u - \hat{n}_x v. \end{aligned}$$

The matrix of eigenvalues along the diagonal, $\tilde{\Lambda}$, is

$$\tilde{\Lambda} = \begin{bmatrix} \tilde{u} & 0 & 0 & 0 \\ 0 & \tilde{u} & 0 & 0 \\ 0 & 0 & \tilde{u}+\tilde{a} & 0 \\ 0 & 0 & 0 & \tilde{u}-\tilde{a} \end{bmatrix}. \tag{4.10}$$

The $\tilde{}$ notation refers to Roe averaged variables defined according to

$$\begin{aligned} \tilde{\rho} &= \sqrt{\rho^+\rho^-}, \\ \tilde{u} &= \psi^- u^- + \psi^+ u^+, \\ \tilde{v} &= \psi^- v^- + \psi^+ v^+, \\ \tilde{h}_0 &= \psi^- h_0^- + \psi^+ h_0^+, \\ \tilde{a}^2 &= (\gamma-1)\left[\tilde{h}_0 - \frac{1}{2}(\tilde{u}^2+\tilde{v}^2)\right], \end{aligned} \tag{4.11}$$

where

$$\psi^{+,-} = \frac{\sqrt{\rho^{+,-}}}{\sqrt{\rho^+}+\sqrt{\rho^-}}.$$

4.3.5 Spatial Accuracy

The solution of the discretized governing equations 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 a left-running characteristic

and once for a right-running characteristic. Information from a right running characteristic is stored as left-sided quantities as the information comes from the left. Likewise, right-sided quantities store information from left running characteristics. The flux quantities can be interpolated from computed values at cell centers, or the state variables may be interpolated from cell centers to cell faces, and fluxes are computed from these values. The latter is called MUSCL (Monotone Upstream-centered Scheme for Conservation Laws) differencing and is the method implemented in ErICA. Interpolation accurate up to third order is available. The state variables at the cell centers are interpolated to the cell faces using

$$q^{-}_{j+1/2,k} = q_{j,k} + \frac{\phi}{4}\left[(1-\kappa)\nabla + (1+\kappa)\Delta\right] q_{j,k}, \tag{4.12}$$

and

$$q^{+}_{j+1/2,k} = q_{j+1,k} - \frac{\phi}{4}\left[(1-\kappa)\Delta + (1+\kappa)\nabla\right] q_{j+1,k}, \tag{4.13}$$

where Δ and ∇ are the forward and backward difference operators, so that

$$\begin{aligned} \Delta q_{j,k} &= q_{j+1,k} - q_{j,k} \\ \nabla q_{j,k} &= q_{j,k} - q_{j,k-1}. \end{aligned}$$

Similar formulae are used to extrapolate to the $k + 1/2$ and $k - 1/2$ faces. Third-order interpolation is possible with $\phi = 1$ and $\kappa = 1/3$. Second-order, one-sided interpolation is obtained with $\phi = 1$ and $\kappa = -1$. First-order interpolation results from $\phi = 0$.

Limiters, such as Van Albeda's limiter and minmod limiter, are used to avoid large

oscillations in the solution near regions of steep gradients by locally reducing the order of the interpolation. The minmod limiter modifies the gradient appearing in (4.12) and (4.13) according to 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:

$$\text{minmod}\,(x,y) = \begin{cases} 0, & \text{if } xy < 0; \\ \text{sign}(x)\,\min(|x|,|y|), & \text{if } xy > 0. \end{cases}$$

The variale β is related to the order of the interpolation and is given by:

$$\beta = \frac{3-\kappa}{1-\kappa}.$$

4.4 Boundary Conditions

Four sets of boundary conditions are specified on a set of "ghost" cells constructed around the computational domain.

At the inflow boundary $j = 0$, a supersonic boundary condition is given:

$$\begin{aligned} p(0,y) &= 66754.11Pa \\ u(0,y) &= 1.5M \\ v(0,y) &= 0 \end{aligned} \tag{4.14}$$

$$T(0,y) \;=\; 3573.0K^0,$$

where M is Mach number.

At the outflow boundary $j = jdim$, a subsonic boundary condition is imposed:

$$p(1,y) = p(0,y) * B = 66754.11 * B \tag{4.15}$$

By varying the parameter B we can move the position of the shock along the computational domain, and for a certain range of B we can obtain the flow without shock inside the computational domain. Numerical experiments together with a one-dimensional analysis lead to the conclusion that $B \leq 2.4$ yields supersonic flow throughout the nozzle, $B \geq 3.5$ yields subsonic flow throughout the nozzle and the flow contains a shock for the range $2.4 \leq B \leq 3.5$.

On the lower wall $k = 0$ and along the centerline $k = kdim$ a no penetration boundary condition is given:

$$u\cos\theta + v\sin\theta = 0, \tag{4.16}$$

where $(\cos\theta, \sin\theta)$ is the outward normal to the wall.

4.5 Numerical Examples

The optimization problem (4.2) is solved for 2 distinct cases: supersonic target flow and target flow with a shock. The 60×20 grid was used throughout this work to obtain the

target flow. Candidate optimal domains are given by (4.3) with $A_l = 0.125$, $A_r = 0.1875$ with α_1 as a free design parameter. The flow problem is solved for this parameter, then the objective function is computed. The integral in (4.2) is evaluated by exact integration of the spline function, fitted through the points, whose independent variables are specified by the discretization of the nozzle in the $x-$direction and whose dependent variables are the squared differencies between the target data and the computed flow. The output is sent to an optimizer, which updates the parameter and sends it back to the flow solver. This process is repeated until the optimal criteria is met. The optimization algorithm is the SQP scheme. The gradient information, required for the functional, is obtained by forward finite difference approximation.

4.5.1 Supersonic Target Flow

The target flow for the supersonic case was obtained with $\alpha_1 = 0.6$ and the outflow boundary condition

$$p(1, y) = 0.6 * p(0, y) \tag{4.17}$$

Figures (4.4) and (4.5) show the target data along the centerline, which was used to compute the cost function in this problem. Table (4.5.1) contains the results of optimization for several initial values of the design parameter α_1. Since the flow is symmetric with respect to the centerline of the nozzle, the $y-$component of the velocity is very small compared to the $x-$component. This can be seen in the Figures (4.4) and (4.5).

The weights in the cost function were chosen in such a way, that contributions of all the components of the integral in (4.2) are approximately the same. It can be seen that the difference between the converged optimal parameters and the target value of α_1 is negligible. The value of the cost function achieves a very small number at the end of iteration process, which indicates good accuracy in the analysis flow computations. As one would expect, the number of iterations grows significantly when the initial value of the design parameter is far from the target solution. It should be pointed out that the only obstacle for the optimal design in this case is the amount of time required to solve the underlying flow problem at each step. The main reason for this is the explicit time integration scheme used to obtain the flow in the nozzle.

Table 4.5.1 : Results of Optimization for Supersonic Flow

$q_{target} = 0.6$

initial guess	approximation of solution	cost function value	number of iterations	elapsed time
0.4	0.59999998	0.10949×10^{-11}	7	6863.61 sec
0.2	0.59999980	0.16468×10^{-9}	8	8709.97 sec
0.1	0.59999877	0.64160×10^{-8}	8	8820.03 sec
-0.1	0.59999906	0.37354×10^{-8}	11	14104.85 sec
-0.2	0.59999906	0.37354×10^{-8}	12	16431.16 sec

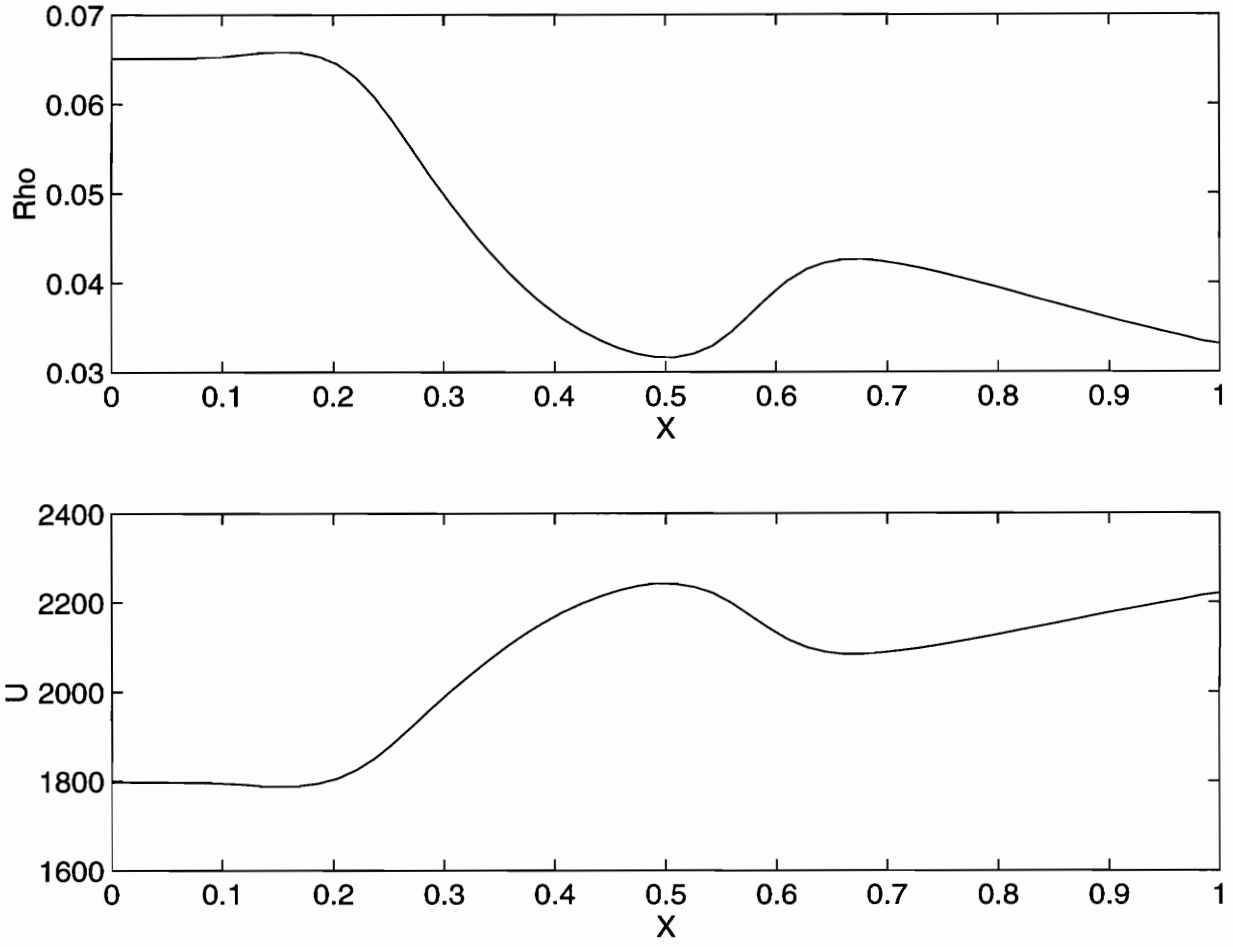


Figure 4.4: Target flow along the centerline for supersonic case

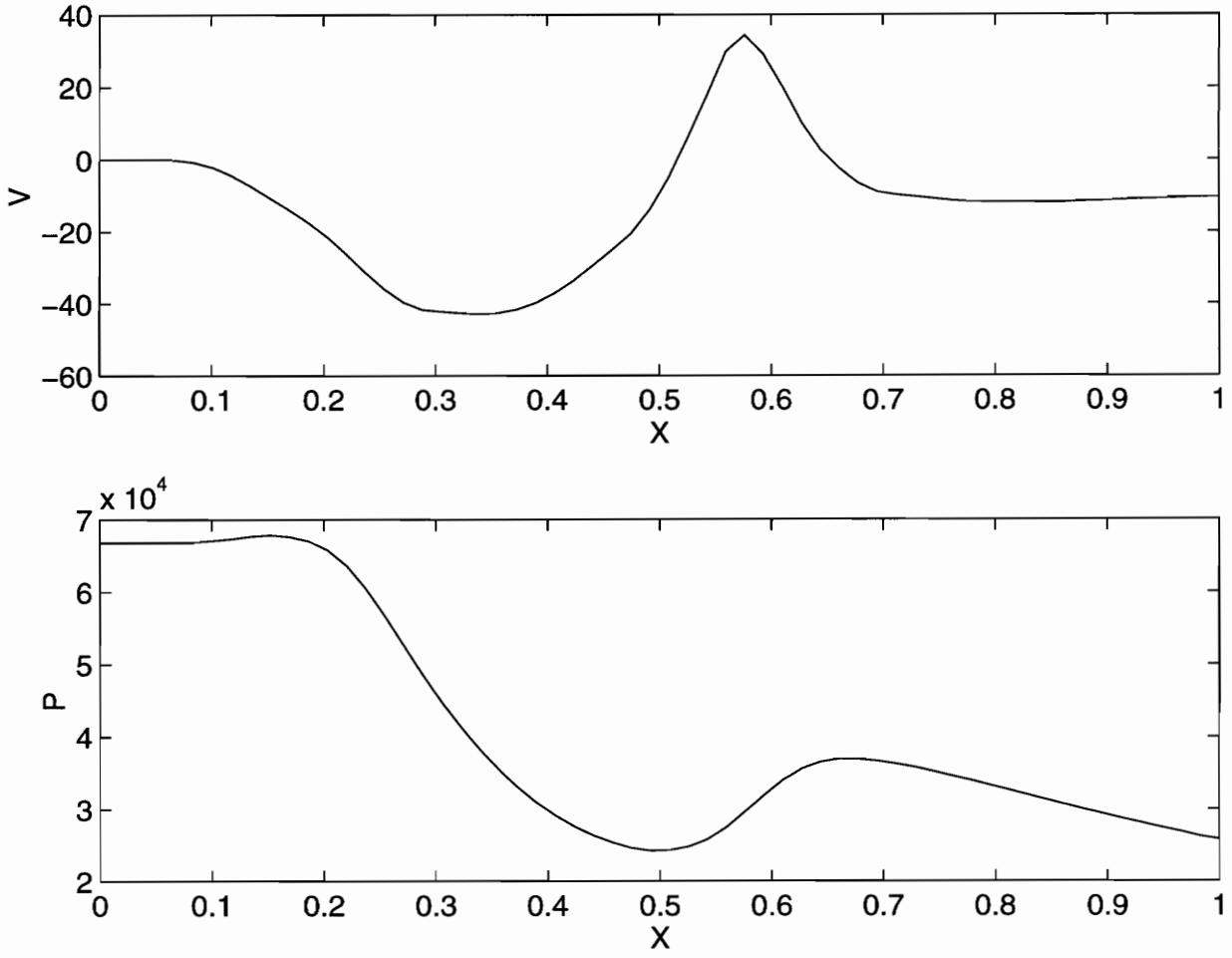


Figure 4.5: Target flow along the centerline for supersonic case (cont.)

4.5.2 Target Flow With a Discontinuity

The target flow with a shock was obtained with $\alpha_1 = -0.5$ and the outflow boundary condition

$$p(1, y) = 2.9 * p(0, y)$$

Figures (4.6) and (4.7) show the target data along the centerline. Table (4.5.2) contains the results of optimization for several initial values of the design parameter α_1. The number of SQP iterations necessary to obtain a solution in this case is considerably larger than that in the supersonic case. It should also be noted that the amount of time, required to obtain the flow data at each iteration step is approximately 3-4 times longer than that in the supersonic case. It was observed that only initial values of α_1, which were close to the target parameter, produced an accurate optimal solution. And conversely, when the initial guess was far from the desired α_1, the results were unsatisfactory. This results mainly from two factors. The flow solver iteration scheme showed very slow convergence rate for this particular case, resulting in inaccuracy in flow calculations. Secondly, the integral in the cost function was taken not over the whole computational domain, but only along the centerline, which may have prevented a good flow fit inside the nozzle away from the centerline.

Table 4.5.2 : Results of Optimization for the Flow with Discontinuity

$q_{target} = -0.5$

initial guess	approximation of solution	cost function value	number of iterations	elapsed time
-0.1	-0.50000021	0.28741×10^{-18}	12	34836.07 sec
0.0	-0.50000069	0.29641×10^{-7}	10	30870.37 sec
0.2	-0.50000011	0.77093×10^{-9}	11	33094.95 sec
0.15	-0.6759	0.874848	14	55008.76 sec
0.35	-0.40381	0.392928	14	54585.76 sec

4.6 Domain Decomposition

Accurate simulation of fluid flow, which is a necessary requirement for the solution of the analysis problem in optimization-based design, requires a great number of discretization points which can lead to saturation of a computer's memory. One approach to overcome this limitation is to split the geometrical domain into sub-regions which can be dealt with separately. Domain decomposition appears to us as a natural means to exploit coarse-grained parallelism. We attempt to adopt domain decomposition ideas to the solution of the design problem (4.2). This section begins by a description of the domain decomposition

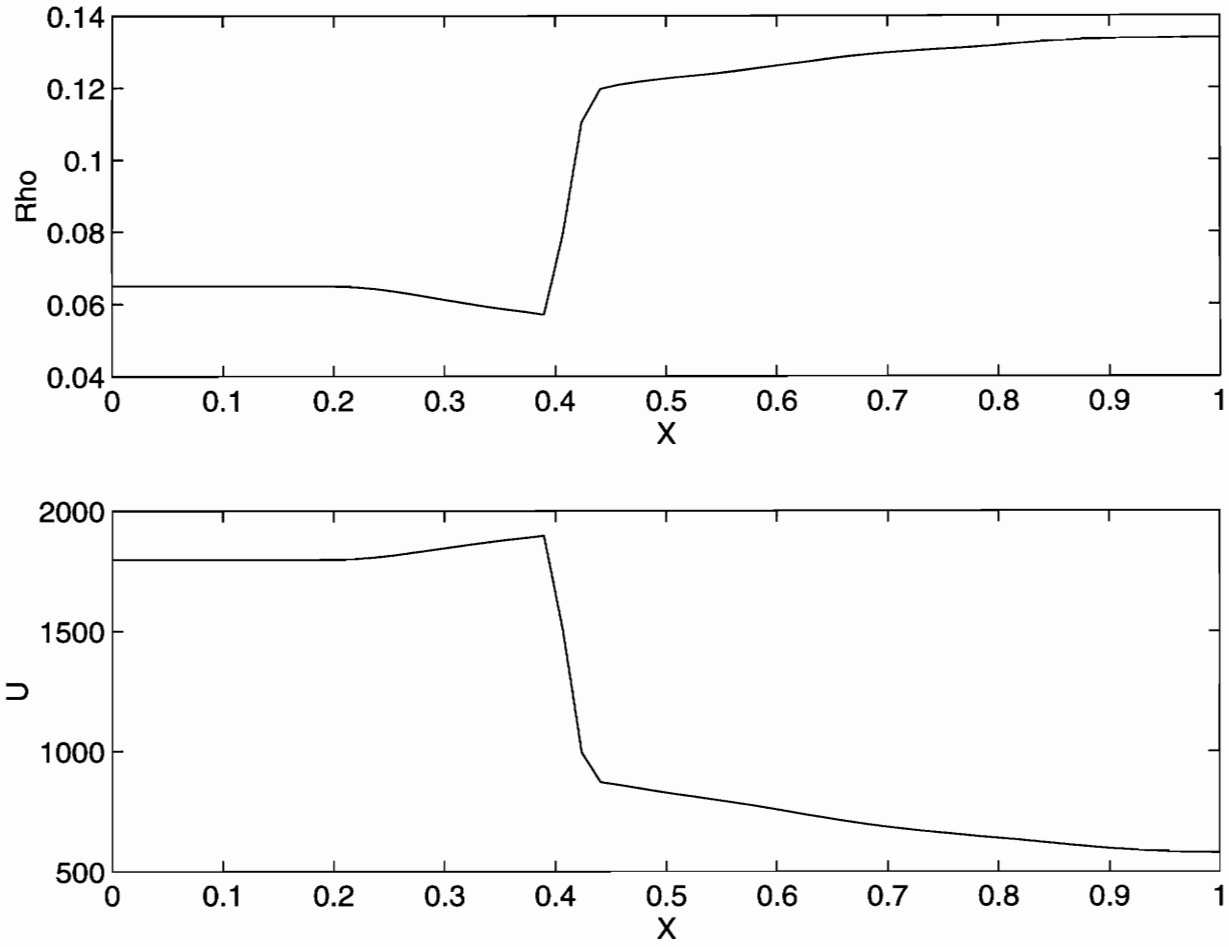


Figure 4.6: Target flow along the centerline for shock case

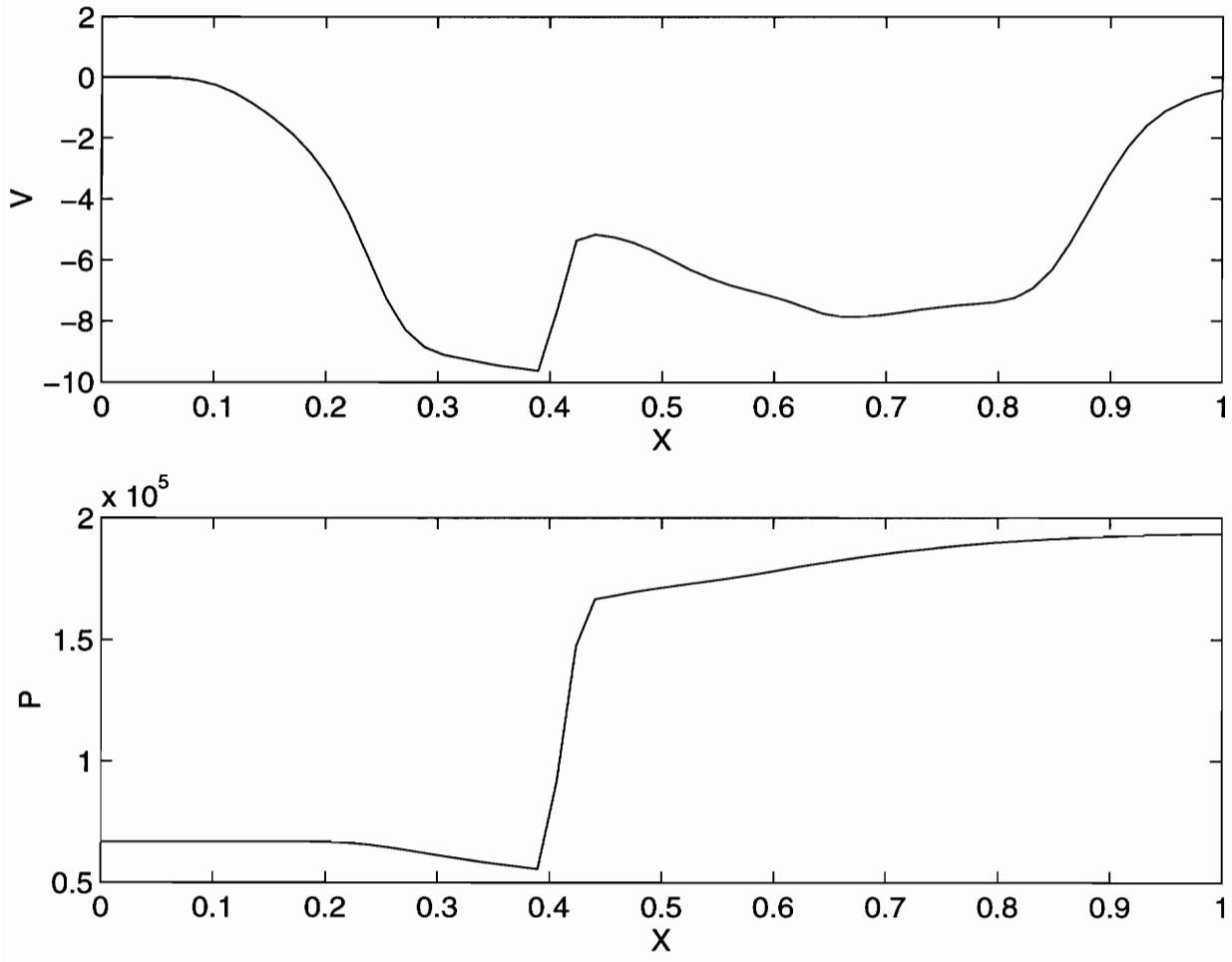


Figure 4.7: Target flow along the centerline for shock case (cont.)

algorithm. The resulting numerical method is validated on several examples. The section concludes with a brief analysis of numerical results.

4.6.1 Theoretical Formulation

We restrict ourselves to the case of overlapping subdomains, which is suggested by the numerical scheme (finite-volume), used to find the solution of the analysis flow problem. For the sake of simplicity, we consider here the partition of computational domain Ω into two overlapping subdomains Ω_1 and Ω_2, with the boundaries Γ_1 and Γ_2. The overlap region is denoted $\Omega_{1,2}$. We have that $\Omega_1 \cup \Omega_2 = \Omega$ and $\Omega_1 \cap \Omega_2 = \Omega_{1,2}$. Γ_1 represents a common boundary between the subdomains Ω_2 and $\Omega_{1,2}$ and Γ_2 represents a common boundary between Ω_1 and $\Omega_{1,2}$. This is illustrated in Figure (4.8). Let Σ represent a line, common to both subdomains Ω_1 and Ω_2, such that it divides $\Omega_{1,2}$ into $\Omega^1_{1,2}$ and $\Omega^2_{1,2}$. (See Figure (4.9)). In the present approach, the subdomains Ω_1 and Ω_2 and the interface Σ are kept fixed arbitrarily at the beginning of the calculations, but in general the definition of Ω_i can be automatically adapted to the physical characteristics of the solution. One assumption that we make concerns the grids associated with each subdomain. We require that they match at the internal interface Σ.

The partition of the spatial domain into adjoining, intersecting subdomains quite naturally implies the definition of the compatibility criteria at the interfaces and the way to impose boundary conditions. If one wishes to deal with the solution of the Euler equations

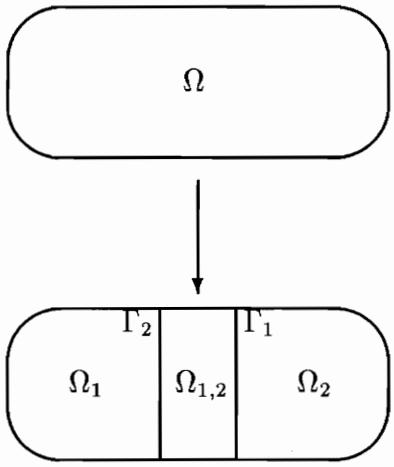


Figure 4.8: Decomposition of a domain into 2 overlapping domains

(4.4) separately on each Ω_i, one requires the boundary values for flow variables on each Γ_1 and Γ_2. These boundary values, imposed on Γ_1 and Γ_2 must involve the continuity of inviscid flux vector $\hat{F}$ across Σ, to be the restriction of the global solution. It is thus natural, in terms of the control problem, to view the values of flow variables at the boundaries as control parameters, and to look for such a set of these variables, that yields the zero value of the jump in $\hat{F}$ across Σ. This intuitive approach leads to the following formulation of the optimal control problem, described in (4.2).

Let Q_1 be the solution of (4.4) on the domain Ω_1 with boundary conditions $\vec{q}^1$, imposed on Γ_1 and Q_2 be the solution of (4.4) on the domain Ω_2 with boundary conditions $\vec{q}^2$, imposed on Γ_2. Let $\hat{F}_1$ and $\hat{F}_2$ denote the corresponding inviscid fluxes in the direction,

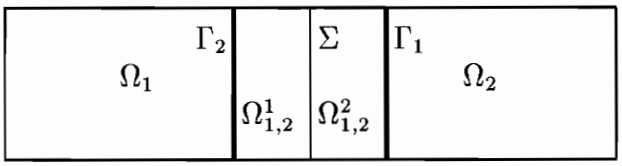


Figure 4.9: Domain Decomposition with an overlap

normal to Σ. The global solution Q on the entire computational domain Ω is defined as follows:

$$Q = \begin{cases} Q_1, & \text{if } (x,y) \subset \Omega_1 \setminus \Omega^2_{1,2}; \\ Q_2, & \text{if } (x,y) \subset \Omega_2 \setminus \Omega^1_{1,2} \end{cases} \tag{4.18}$$

Let $\hat{N}$ notify the target nozzle and $\hat{Q}$ notify the target flow in $\hat{N}$. *Given* $\hat{Q}$ *in* $\hat{N}$ *find* $A(x)$, $\vec{q}^1$ *and* $\vec{q}^2$ *such that*

$$J(A) = w_1 \int_C [\rho - \hat{\rho}]^2 ds + w_2 \int_C [u - \hat{u}]^2 ds + w_3 \int_C [v - \hat{v}]^2 ds + w_4 \int_C [p - \hat{p}]^2 ds \tag{4.19}$$

subject to

$$\hat{F}_1^+ - \hat{F}_2^- = 0 \qquad \text{on } \Sigma \tag{4.20}$$

is minimized.

The cost function (4.19) depends on the choice of boundary conditions $\vec{q}^1$ and $\vec{q}^2$ and therefore $J = J(\alpha_1, \vec{q}^1, \vec{q}^2)$. Equation (4.20) denotes the set of active constraints, expressing the compatibility conditions.

During each iteration step the partial solutions are brought together to form the global solution, which is used to compute the cost function of the control problem. From the structure of the time marching strategy, it follows that computations relative to individual subdomains can be performed in parallel, with data exchange being required only at the end of each updating step.

4.6.2 Discretization of Constraints

The choice of space-discretization induces quite naturally the discretization of the constraint equations and therefore defines the dimension of the control parameter space. In the case where the finite-volume formulation is used and fluxes are presumed constant across the cell-face, the dimension of the control space is $ND+1$, where N is the number of cell nodes on Σ, D is the number of independent flow variables (4 in our case) and 1 is the number of shape parameters.

After the discretization is performed, the boundary conditions are imposed at each cell center, which means that the grid dimension defines the dimension of control space. This can lead to a very large number of control variables and therefore to an unrealistic number of flow calculations necessary to evaluate the gradient of the cost and constraint functions at each step. This problem can be solved by evaluating the flux across Σ and imposing boundary conditions only at a limited number of points along Γ_1, Γ_2, and Σ and interpolating the rest of the data from these values. In the numerical computations,

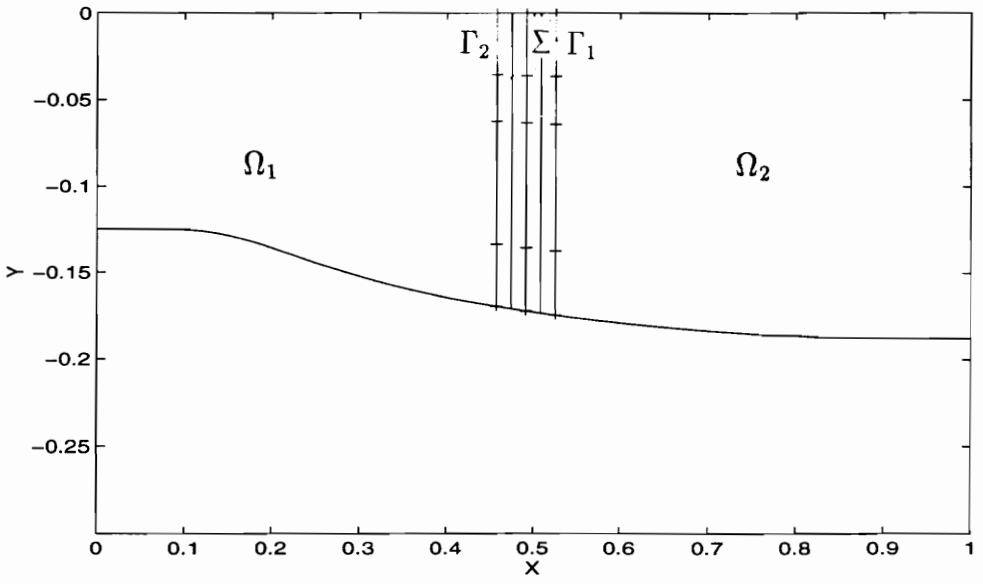


Figure 4.10: Decomposition of a nozzle into 2 domains with a 2-cell overlap

performed in this work, we chose 5 points along the width of the nozzle, denoting the relative distance from the centerline. The boundary conditions were imposed and the fluxes were evaluated at $(0, 4/19l, 7/19l, 15/19l, l)$, where l is the length of the appropriate grid line in the y-direction. By this we ensure that the data is properly matched for different shapes of the nozzle. Therefore, we have $2*5*4+1 = 41$ control variables and $5*4 = 20$ constraints imposed. Figure (4.10) shows the decomposition of a nozzle into 2 domains with a 2-cell overlap. Here, "+"'s denote the position of the points, at which the boundary conditions and compatibility conditions are imposed.

At each iteration step, the boundary data to be imposed at the "ghost" cell centers in domains Ω_1 and Ω_2 is interpolated from the corresponding set of control variables. The

flow solution is obtained in Ω_1 and Ω_2 and the global solution on Ω is defined according to (4.18). The integral (4.19) is evaluated in the manner, similar to that without domain decomposition. Constraints are evaluated by interpolating the flow data to the points where the control variables are defined. This information is passed to the optimizer, which produces the next set of boundary data and the shape parameter. This process is repeated until convergence.

It is also possible to match the fluxes at two different interfaces Σ_1 and Σ_2, so that the number of constraints is 40. By this we eliminate 20 degrees of freedom which remain for the problem with fixed shape parameter α_1 when the only control parameters are the flow components, associated with the boundaries of subdomains. Even though the procedure described above seems to converge to the solution, which is close to the target solution, the optimal solution, produced by the SQP code in this setting may not be the flow obtained with the target parameters, because 20 degrees of freedom remain in this formulation of the control problem.

4.6.3 Numerical Examples

The domain decomposition algorithm, described above was applied to the solution of shape optimization problem for supersonic target flow. The target flow was obtained with $\alpha_1 = 0.6$ and the set of boundary conditions, specified in section (4.4). Decomposition of the domain into two subdomains was performed for two different sizes of the overlap. In the case 1 the

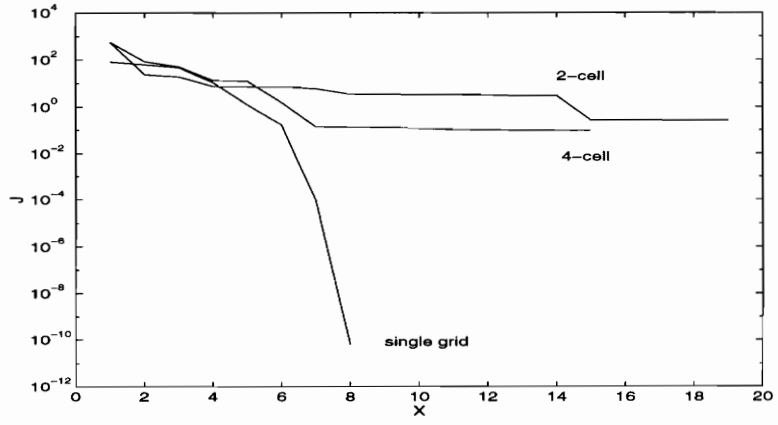


Figure 4.11: Convergence history

overlap was taken to be 2 cells and the computational grid for each subdomain was 32×20, in the case 2 the overlap was 4 cells and the computational grid for each subdomain was 34×20. The numerical results, illustrating the convergence of the algorithm are shown in Figure (4.11). Here, domain decomposition with two different overlaps is compared to the computation performed on a single grid, covering the entire domain. It can be seen that by increasing the size of the overlap between the adjacent domains, the convergence is achieved faster even though the computational cost per domain is higher.

Figures (4.12) and (4.13) show the optimal solution along the centerline, obtained with a 4-cell overlap, compared with the target solution for this case. Solid lines denote the target data, dashed lines denote the flow, obtained with the initial parameters and the "dash-dot" lines denote the flow, obtained with the converged parameters. The initial value of the cost function was 560.131 and the final value after the optimization was completed was

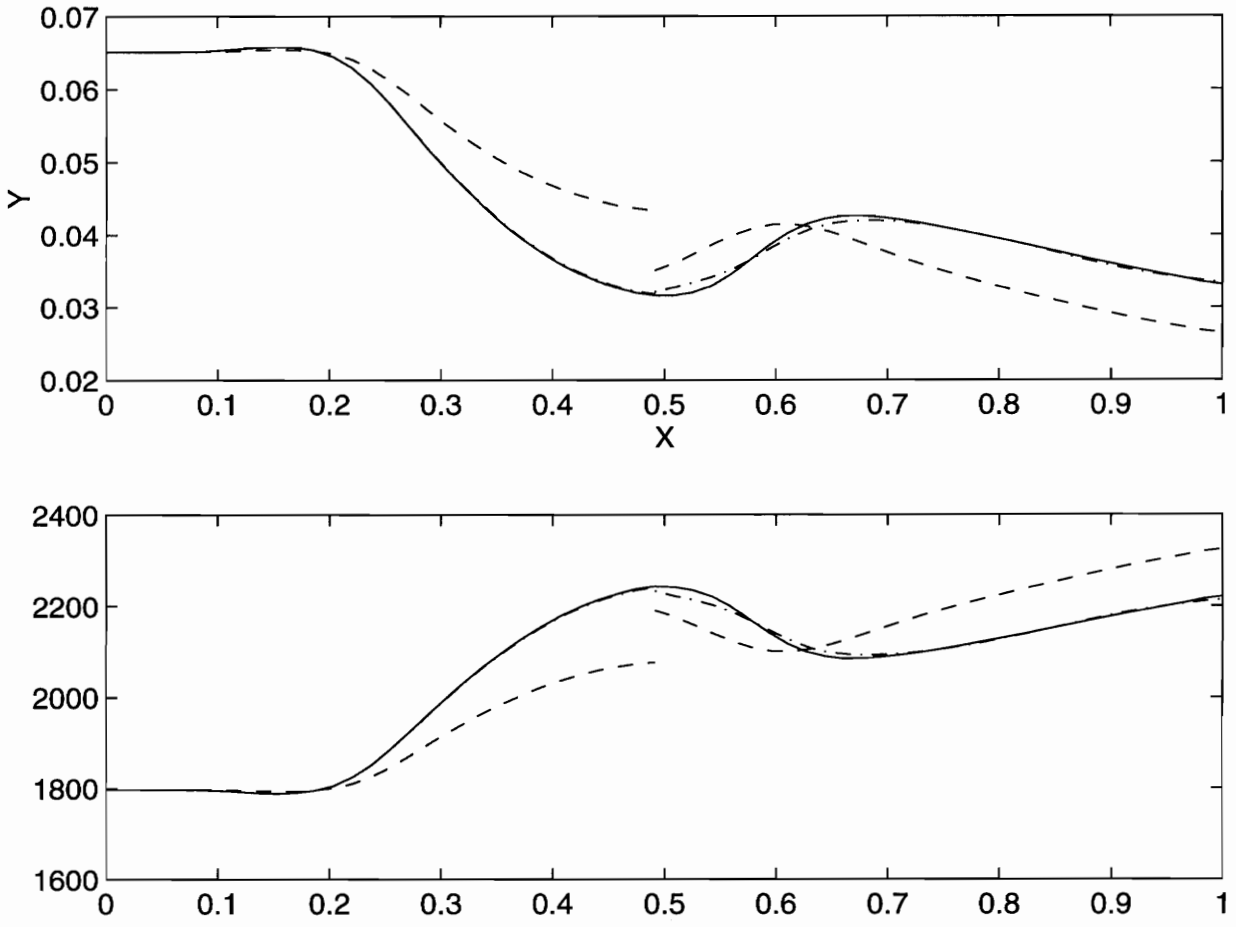


Figure 4.12: Results of Domain Decomposition method with 4-cell Overlap

0.094935. The initial value of the shape parameter α_1 was 0.1 and the converged value was 0.59267 (compare to the target value of 0.6).

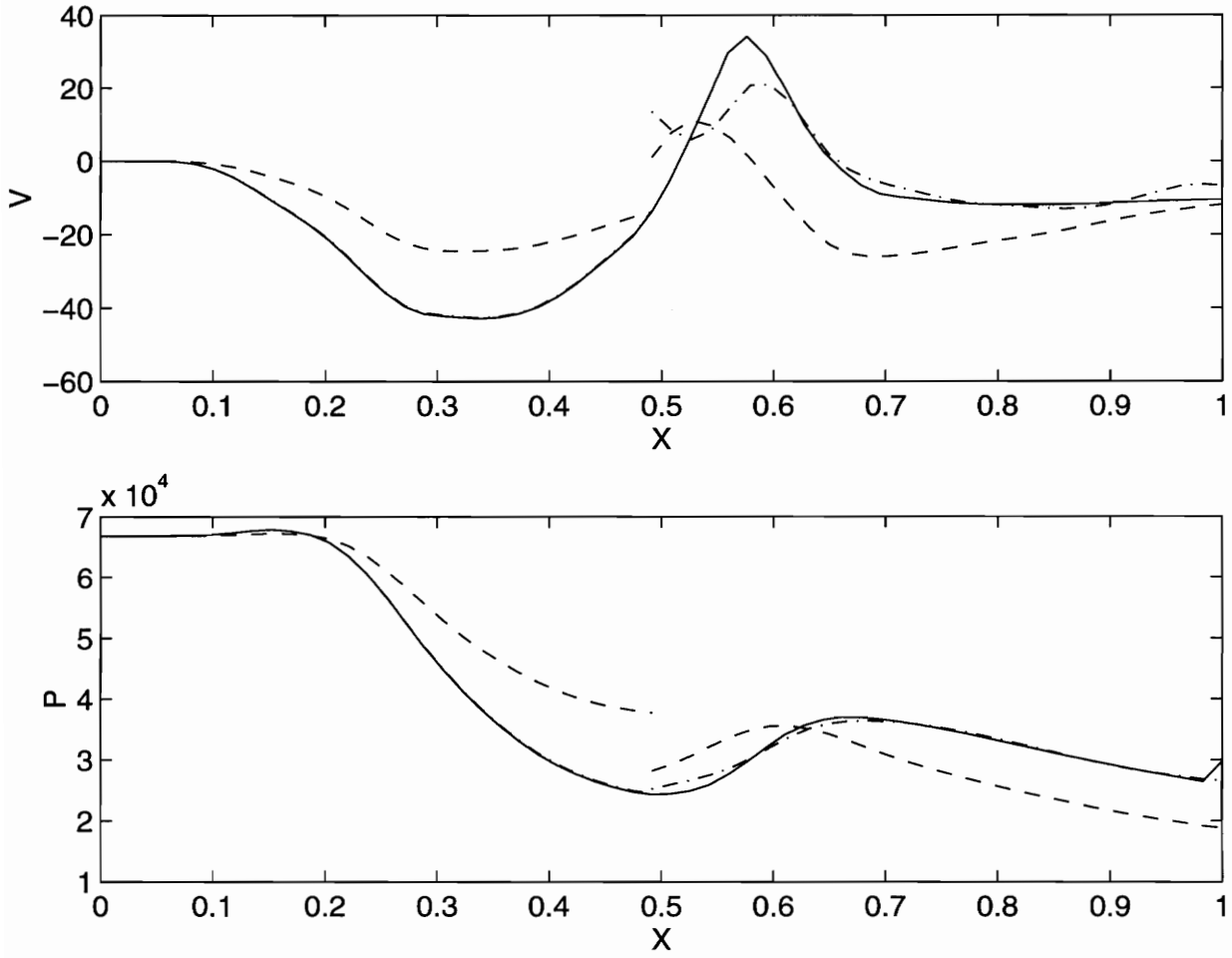


Figure 4.13: Results of Domain Decomposition method with 4-cell Overlap (cont.)

Chapter 5

Conclusions

In this work, we considered an "inverse-design" problem, where we specified the flow distribution in the computational domain (or on a subset) and sought the geometrical configuration that produced this flow. Based on this idea, we formulated a control problem, in which the optimization procedure minimizes the error between the target flow and the actual flow through successive adjustment of the design parameters. We were interested in exploring the computational efficiency of the numerical solution of this problem, particularly the implementation in a workstation cluster environment. In this setting we studied numerical algorithms that are suited to coarse-grained parallelization. These aspects were examined with an example of one-dimensional heat transfer in a cylinder of non-uniform cross-sectional area and the optimal design of a two-dimensional nozzle.

We compared the computational cost and convergence properties of the optimiza-

tion procedure for two approaches: the "Black-Box" method and domain decomposition method. When employing the "Black-Box" technique, the unconstrained control problem was solved in terms of the design variables and in the case of domain decomposition implementation, the constrained control problem was solved in terms of design variables and boundary data on interfaces. In the latter approach, the analysis problem was divided into a number of subproblems, which can be solved independently. The size of the discretized state variable in each subproblem is smaller to decrease the computational cost on each respective subdomain. The trade-off lies in global cost function's dependence on many more control variables that are introduced by this division process.

The results obtained for the one-dimensional heat-transfer case strongly suggest the use of domain decomposition methods in the context of an optimization procedure. The application of the domain decomposition method took considerably longer than the solution over the entire domain on a single-processor machine, yet shows promise if implemented on a network of workstations. Each subdomain can be assigned to a separate processor, with one "master" processor performing the global iteration and synchronizing the process. Subdomain problems are virtually independent from each other, and the data transfer is performed only between "master" processor and subdomain processors. A linear speed-up can be expected if the computational domain is partitioned in a number of subdomains proportional to the number of processors in the network. If the local solver allows the boundary data to be imposed explicitly, this method can be used to parallelize existing

analysis codes without making significant changes in the code. If only coarse-grained parallelization is considered, this a strong alternative to creating new parallel algorithms.

Also, we implemented the grid-embedding chimera technique to the solution of a one-dimensional heat transfer problem. The formulation of this scheme in terms of domain decomposition leads to an overlapping method. Subdomain problems are coupled for each pair of neighboring domains, therefore the degree of parallelism is lower in this case. We showed that the chimera approach, which can significantly simplify the grid generation procedure and reduce the storage requirements for complex flows, can be successfully incorporated into the optimization-based design framework. The original analysis problem can be split into problems on subdomains, which can be solved in parallel with data transfer at each step limited to exchanging boundary information between the neighboring subdomains and between the "master" processor.

In the case of a two-dimensional flow, the optimization was applied to supersonic flow and the discontinuous flow. The optimization results strongly depend on the accuracy of the flow calculations, therefore one should be careful with the choice of flow solver. Implicit algorithms are preferred to explicit methods since the flow calculations are performed several times in each iteration. The domain decomposition for the case of a two-dimensional flow leads to a very large number of control variables, associated with the discretization on the interfaces. If one wishes to implement this algorithm in a parallel environment, the computations should be spread between the processors in such a way, that the number of

processors is proportional to the number of control variables. This class of algorithms is known to have a slow convergence rate, therefore the use of a preconditioner is suggested.

Appendix A: Numerical Solution of the Heat Equation

We wish to construct a numerical solution of the boundary-value problem

$$\frac{d}{dx}\left(k(x)A(x,u)\frac{dT(x)}{dx}\right) - h\frac{dS(x,u)}{dx}T(x) = 0, \quad 0 < x < 1 \tag{5.1}$$

$$T(0) = T_0 \qquad T(1) = T_1, \tag{5.2}$$

This problem is equivalent to the problem of finding the solution of the second-order differential operator

$$L(w) \equiv [-pw']' + qw = f \tag{5.3}$$

with boundary data

$$w(a) = w_0 \qquad w(b) = w_1. \tag{5.4}$$

In the case of the heat equation $p(x,\alpha) = -k(x)A(x,\alpha)$, $q(x,\alpha) = h\frac{dS(x,\alpha)}{dx}$, and $f = 0$.

Due to linearity of the problem we can decompose the solution of (5.3) w as

$$w = w_h + w_i,$$

where w_i is a sufficiently smooth function, satisfying

$$w_i(a) = w_0 \qquad w_i(b) = w_1$$

and w_h is a solution of (5.3) with homogeneous boundary data. We can rewrite (5.3) as

$$L(w_h) = f - L(w_i) = f_n, \tag{5.5}$$

where $w_h(a) = 0 \qquad w_h(b) = 0.$

Let w_i be a linear function

$$w_i = \frac{w_1 - w_0}{b-a}(x-a) + w_0, \tag{5.6}$$

then

$$L(w_i) = -p'(x,\alpha)\frac{w_1 - w_0}{b-a} + q(x,\alpha)\left[\frac{w_1 - w_0}{b-a}(x-a) + w_0\right]. \tag{5.7}$$

Now the problem is reduced to the construction of a numerical solution for the second-order operator

$$L(w_h) = f - \left(-p'(x,\alpha)\frac{w_1 - w_0}{b-a} + q(x,\alpha)\left[\frac{w_1 - w_0}{b-a}(x-a) + w_0\right]\right) \tag{5.8}$$

with homogeneous boundary conditions.

The solution to this boundary-value problem satisfies a variational property. We assume that $p \subset C^1[0,1]$ and $q, f_n \subset C[0,1]$. Further, we require that

$$p(x) \geq \delta > 0, \qquad q(x) \geq 0, \quad \text{for } a \leq x \leq b,$$

and

$$q(x) \geq 0, \qquad \text{for } a \leq x \leq b.$$

These assumptions are sufficient to guarantee that the boundary-value problem (5.3) has a unique solution and that the numerical approximation, constructed with the finite-element scheme, described below satisfies the following property [**27**]:

$$\left(\int_a^b |w(x) - \phi(x)|^2 dx \right)^{1/2} = O(h^4), \qquad 0 \leq x \leq 1.$$

The variational property for the second-order operator (5.3) states that the function $w \subset C_0^2[a,b]$ is the unique solution of (5.3) under the above assumptions if and only if it minimizes the integral

$$I(u) = \int_a^b \{p(x)[u'(x)]^2 + q(x)[u(x)]^2 - 2f(x)u(x)\} dx. \tag{5.9}$$

The finite-element approach minimizes the integral(5.9) over a set of functions consisting of linear combinations of certain basis functions $\phi_i, i = 0, \ldots, N$. The basis functions are linearly independent and satisfy

$$\phi_i(a) = \phi_i(b) = 0, \qquad \text{for } i = 0, 1, \ldots, N.$$

An approximation $y^N(x) = \sum_{i=0}^{N} c_i\phi_i(x)$ to the solution of equation(5.3) is then obtained by finding constants $c_i, i = 0, \ldots, N$ to minimize $I\Big[\sum_{i=0}^{N} c_i\phi_i\Big]$.

From the equation (5.9)

$$I[y^N] = \int_a^b \{p(x)\Big[\sum_{i=0}^{N} c_i\phi_i\Big]^2 + q(x)\Big[\sum_{i=0}^{N} c_i\phi_i\Big]^2 - 2f(x)\sum_{i=0}^{N} c_i\phi_i\}dx \tag{5.10}$$

and, the necessary condition for the minimum is given by

$$\frac{\partial I}{\partial c_j} = 0, \qquad \text{for } i = 0, 1 \ldots, N. \tag{5.11}$$

Differentiating (5.10) and substituting into (5.11) yields:

$$\sum_{i=0}^{N}\Big[\int_a^b \{p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x)\}dx\Big]c_i - \int_a^b f(x)\phi_j dx \tag{5.12}$$

$$\text{for } j = 0, 1, \ldots, N.$$

These equations produce an $n \times n$ linear system $Ac = b$ in the variables $c_0, c_1, \ldots, c_N$, where the symmetric matrix A is given by

$$a_{ij} = \int_a^b [p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x)]dx,$$

and b is defined by

$$b_i = \int_a^b f(x)\phi_i(x)dx.$$

The basis functions, that allow to obtain the approximation $y^N \subset C^2[a, b]$, are bell-shaped splines or B-splines. The basis B-spline $S(x)$ satisfies the boundary conditions

$$S''(x_0) = S''(x_4) = 0 \qquad \text{and} \qquad S'(x_0) = S'(x_4) = 0$$

$$S(x) = \begin{cases} 0 & x \le -2 \\ (2-x)^3 - 4(1-x)^3 - 6x^3 + 4(1+x)^3 & -2 < x \le -1 \\ (2-x)^3 - 4(1-x)^3 - 6x^3 & -1 < x \le 0 \\ (2-x)^3 - 4(1-x)^3 & 0 < x \le 1 \\ (2-x)^3 & 1 < x \le 2 \\ 0 & 2 < x. \end{cases} \tag{5.13}$$

We define

$$S_i(x) = S\Big(\frac{x - x_i}{h}\Big), \quad \text{for } i = 0, 1, \ldots, N,$$

where x_i's are equally spaced nodes on [a,b]. For the set ϕ_i to satisfy the boundary conditions $\phi_i(0) = \phi_i(1) = 0$, we must modify S_0, S_1, S_{N-1}, and S_N. The basis with this modification is defined by

$$\phi_i(x) = \begin{cases} S_0(x) - 4S_{-1}(x) & i = 0 \\ S_0(x) - 4S_1(x) & i = 1 \\ S_i(x) & -2 \le i \le N-2 \\ S_N(x) - S_{N-1}(x) & i = N-1 \\ S_N(x) - 4S_{N+1}(x) & i = n. \end{cases} \tag{5.14}$$

Since $\phi_i(x)$ and $\phi_i'(x)$ are nonzero only for $x_{i-2} \le x \le x_{i+2}$, the matrix in the finite-

element approximation is a banded matrix with bandwidth at most seven:

$$
(5.15)\quad A = \begin{bmatrix}
a_{00} & a_{01} & a_{02} & a_{03} & \cdots & \cdots & \cdots & \cdots & 0 \\
a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & \ddots & & & \vdots \\
a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & \ddots & & \vdots \\
a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & a_{N-2,N+1} \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & a_{N-1,N+1} \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & a_{N,N+1} \\
0 & \cdots & \cdots & 0 & \cdots & a_{N+1,N-2} & a_{N+1,N-1} & a_{N+1,N} & a_{N+1,N+1}
\end{bmatrix}
$$

where

$$
a_{ij} = \int_a^b [p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x)]dx \quad \text{for } i,j = 0,1,\ldots,N+1
$$

The matrix A is positive definite, so the linear system(5.15) can be solved by Cholesky factorization or some other algorithm. In our application, we used a LINPACK subroutine BANDM.F to find the inverse of this matrix.

Bibliography

[**1**] Arina, R., Canuto, C., "A χ-Formulation of the Viscous-inviscid Domain Decomposition for the Euler/Navier Stokes Equations," *Proc. of the 7th International Conference on Domain Decompostion*, Pennsylvania, PA, October, 1993.

[**2**] Benek, J. Steger, J., Buning, P., "A 3-D Chimera Embedding Technique," NASA TM85-1523, December, 1985.

[**3**] Borggaard, J., Burkardt, J., Burns, J., Cliff, E., Gunzburger, M., Kim, H., Lee, H., Peterson, J., Shenoy, A., Wu, X., "Algorithms for Flow Control and Optimization," Proc. of the Workshop on Optimal Design Control, Blacksburg, VA, April, 1994.

[**4**] Borggaard, J., Burns, J., Cliff, E., Gunzburger, M., "Sensitivity Calculations for a 2-D, Inviscid, Supersonic Forebody Problem," in *Identification and Control of Systems Governed by Partial Differential Equations,* H.T. Banks, R. Fabiano and K. ITO, eds, SIAM Publications, Philadelphia, 1993.

[**5**] Bourget, J. F., LeTallec, P., Perthame, B., Qin, Y., "Coupling Boltzmann and Euler Equations Without Overlapping," *Proc. of the 6th International Conference on Domain*

Decomposition, Como, Italy, June, 1992.

[**6**] Brambilla A., Callenzoli, C., Gazzaniga, G., Gervasio, P., Sacchi, G., "Implementation of Domain Decomposition Techniques on nCUBE2 Parallel Machine," *Proc. of the 6th International Conference on Domain Decomposition*, Como, Italy, June, 1992.

[**7**] Burden, R., L., *Numerical Analysis*, PWS-Kent Publishing Compony, Boston, 1993

[**8**] Cahovet, J., "On Some Difficulties Occuring in the Simulation of Incompressible Fluid Flows by Domain Decomposition Methods," *Proc. of the First International Symposium on Domain Decomposition Methods for PDE's*, Paris, France, January, 1987.

[**9**] Cai, X.-C., Dryja, M., "Domain Decomposition Mehtods for Monotone Nonlinear Elliptic Problems," *Proc. of the 7th International Conference on Domain Decompostion*, Pennsylvania, PA, October, 1993.

[**10**] Chan, T. F., Resasco, P. C., "A Framework for the Analysis and Construction of Domain Decomposition Preconditioners," *Proc. of the First International Symposium on Domain Decomposition Methods for PDE's*, Paris, France, January, 1987.

[**11**] Dietz, W. E., Jacocks, J., L., Fox, J., H. "Application of Domain Decomposition to the Analysis of Complex Aerodynamic Configurations," *Proc. of the 3rd International Symposium on Domain Decomposition Methods for PDE's*, Houston, TX, March, 1989.

[**12**] Dougherty, F., Benek, J., Steger, J., "Chimera: a Grid-Embedding Technique," AEDC—TR—85–64, December, 1985.

[**13**] Dougherty, F., Benek, J., Steger, J., "A Flexible Grid Embedding Technique With

Application to the Euler Equations," NASA TM83-1944, June 1983.

[**14**] Dryja, M., Widland, O. B., "An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions," Technical Report 339, Department of Computer Science, Courant Institute, 1987.

[**15**] Dryja, M., Widland, O. B., "Some Domain Decomposition Algorithms for Elliptical Problems," *Proc. of the Conference on Iterative Methods for Large Linear Systems*, Austin, TX, October, 1988.

[**16**] Dryja, M., Wilwund, O. B., "Variant of the Schwarz Alternating Method for the Case of Many Subregions," Technical Report 339, Dept. of Computer Science, Courant Institute, 1987.

[**17**] Dinh, Q. V., Glowinsky, R., Reriaux, J., "Solving Elliptic Problems by Decomposition Methods with Applications." *Elliptic Problem Solvers 2*, Academic Press, New York, 1982

[**18**] Hirsch, C., *Numerical Computation of Internal and External Flows,* John Wiley & Sons, Ltd. West Sussex, 1991.

[**19**] Marchuck, G. I., Kuznetsov, Yu. A., Matsokin, A. M., "Fictitions Domain and Domain Decomposition Methods". *Sov. J. Numer. Anal. Math. Modelling*, Vol. 1, 1986, pp. 3—35

[**20**] Mu, M., Rice, John R., "Modelling with Collaborating PDE Solvers: Theory and Practice," *Proc. of the 6th International Conference on Domain Decomposition*, Como,

Italy, June, 1992.

[**21**] Narducci, R. P., "Selected Optimization Procedures for CFD-Based Shape Design Involving Shock Waves or Computational Noise," Ph.D. Dissertation, Virginia Tech, Blacksburg, VA, May 1995.

[**22**] Pernice, H., "Domain Decomposed Preconditioners with Krylov Subspace Methods as Subdomain Solvers", *Proc. of the 7th International Conference on Domain Decompostion*, Pennsylvania, PA, October, 1993.

[**23**] Quarteroni, A., "Domain Decomposition and Parallel Processing for the Numerical Solution of Partial Differential Equations," *Surv. MAth. Industry*, Vol 1, 1991, pp. 75—118.

[**24**] Sadchikova, E., Shenoy, A., Cliff, E. M., "Computational Issues in Optimization-Based Design," Proc. of the 34th IEEE Conference on Decision and Control, New Orleans, LA, December 1995.

[**25**] Schittkowski, K., "NLPQL: A FORTRAN Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, vol. 5, 1985/6, pp. 485—500.

[**26**] Shubin, G., Frank, P., "A Comparison of Optimization-Based Approaches for a Model Computational Aerodynamics Design Problem", *J. Computational Physics*, Vol. 98, 1992, pp. 74—89

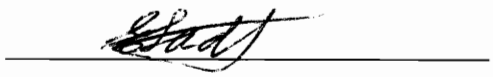
[**27**] Schwarz, H. A.: *Über einige Abbildungsaufgaben*, Ges. Math. Abh., 1869, pp. 65—83.

[**28**] Widlund, O., "Domain Decomposition Algorithms and the Bicentennial of the French Revolution," *Proc. of the 3rd International Symposium on Domain Decomposition Methods for PDE's*, Houston, TX, March, 1989.

Vita

After obtaining her Bachelor of Science degree in Aerospace Engineering from the Moscow Institute of Physics and Technology, Moscow, Russia, Lena spent a year at the Aerospace and Ocean Engineering Department at Virginia Tech, Blacksburg, Virginia, USA, as an exchange student under the Presidential Exchange Program.

Lena enrolled in the Master of Science program of the Aerospace and Ocean Engineering Department of Virginia Tech in the fall of 1994.



Lena V. Sadtchikova