RICE UNIVERSITY

# The Use of Optimization Techniques in the Solution of Partial Differential Equations from Science and Engineering

by

**Anthony José Kearsley**

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

**Doctor of Philosophy**

APPROVED, THESIS COMMITTEE:

---

Roland Glowinski, Co-chairman
Adjunct Professor of Computational & Applied Mathematics

---

Richard A. Tapia, Co-chairman
Noah Harding Professor of Computational & Applied Mathematics

---

John E. Dennis, Jr.
Noah Harding Professor of Computational & Applied Mathematics

---

Mary F. Wheeler
Noah Harding Professor of Computational & Applied Mathematics

---

Balasubramaniam Ramaswamy
Assistant Professor of Mechanical Engineering and Material Sciences

Houston, Texas
August, 1996

## Abstract

# The Use of Optimization Techniques in the Solution of Partial Differential Equations from Science and Engineering

by

Anthony José Kearsley

Optimal Control of systems governed by Partial Differential Equations is an applications–driven area of mathematics involving the formulation and solution of minimization problems. Given a physical phenomenon described by a differential equation, the Optimal Control Problem (OCP) seeks to force *state variables* to behave in a particular, desired way. This manipulation of state variables is achieved through the *control variables*. Many problems arising in applications of science and engineering can fruitfully be viewed and formulated as OCP problems. From the OCP point of view, one sees the structure underlying the optimization problem. In this thesis we will propose and analyze algorithms for the solution of Nonlinear Programming Problems (NLP) designed to exploit the OCP structure.

## Acknowledgments

This thesis is a very important milestone in a journey I began more than ten years ago. People too numerous to mention have helped me along the way; a few are singled out here.

When I was an undergraduate at the University of Maryland, Baltimore County, the Mathematics faculty, in particular Professors James Greenberg, Søren Jensen, and Marc Teboulle, taught me to love applied mathematics; their patience with me was endless and I will always be grateful to them.

At the time, Professor Greenberg introduced me to Dr. Paul Boggs at NIST who, again, with infinite patience, spent countless hours teaching me the subtler aspects of optimization. Dr. Boggs, in turn, introduced me to Professor Jon Tolle of the University of North Carolina. Together, they metamorphosed me from student-aide to collaborating researcher. They have become close friends.

At Rice University, in addition to my thesis advisors, I have worked with and learned from Professors John Dennis, Mary Wheeler and William Symes who always made time in their busy schedules to talk to me and to help me. Without them, my education at Rice would have been incomplete. Friends from Rice, whom I will have forever, Martin Berggren, Lawrence Cowsar, Mark Gockenbach, Andrea Reiff and Ivan Yotov were always available to discuss ideas or to grind through details.

During a six-month visit to CERFACS in Toulouse, France I had the privilege of working with Drs. Ian Duff, Luc Giraud and the entire parallel algorithm team, fruitful collaborators and good friends.

My thesis advisors, Professor Richard Tapia and Professor Roland Glowinski, are due the greatest gratitude and thanks for their direction, encouragement and help. Their definitive influence on this work is obvious.

In remembrance of things past, I dedicate this thesis to the *Bistro des Vins* across from the Place Saint Etienne in Toulouse, France.

# Contents


Abstract ii
Acknowledgments iii
List of Illustrations vii
List of Tables x

**1 Introduction 1**

**2 A Sequential Linear/Quadratic Programming Algorithm that Uses Relaxed-Constraint Subproblems 4**

2.1 Preliminary considerations . . . . . 4
2.2 Introduction . . . . . 5
2.3 Basic algorithms . . . . . 7
2.4 Previously suggested relaxations . . . . . 10
2.5 The proposed relaxation . . . . . 12
2.6 Theory of convergence of the algorithm . . . . . 15
2.7 Interior point method for LP and RQP . . . . . 21
2.8 Implementation of the algorithm . . . . . 26
2.9 Numerical results and conclusions . . . . . 29

**3 A Derivative–Free Algorithm for Small Scale Nonlinear Programming Problems 33**

3.1 Introduction . . . . . 33
3.2 The direct-search method . . . . . 35
3.3 The algorithm . . . . . 37
3.4 Numerical results . . . . . 40
3.5 Conclusions . . . . . 43

**4 Exact and Approximate Neuman Boundary Control of the Heat Equation 49**

4.1 Introduction . . . . . 49
4.2 Neumann boundary-control of the heat equation . . . . . 50
4.3 Optimization formulations . . . . . 53
4.4 Discretization . . . . . 54
4.5 Test targets . . . . . 56
4.5.1 First target state . . . . . 56
4.5.2 Second target state . . . . . 56

4.5.3 Third target state . . . . . . . . . . . . . . . . . . . . . . . . 57
4.6 Numerical results . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.7 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
4.8 Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

**5 Hierarchical Control of Systems Governed by PDE's 70**
5.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
5.2 The control problem . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.3 Calculation of the adjoint equation . . . . . . . . . . . . . . . . . 74
5.4 Discretization of the state equation . . . . . . . . . . . . . . . . . 80
5.5 The test problem and NLP formulations . . . . . . . . . . . . . . . 81
5.6 Numerical results . . . . . . . . . . . . . . . . . . . . . . . . . . 82
5.7 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84

**6 Optimal Control of Melting; A Stefan Control Problem 86**
6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
6.2 The state equation . . . . . . . . . . . . . . . . . . . . . . . . . . 87
6.3 The control problem . . . . . . . . . . . . . . . . . . . . . . . . . 89
6.4 Discretizing the problem . . . . . . . . . . . . . . . . . . . . . . . 90
6.5 Numerical results and conclusions . . . . . . . . . . . . . . . . . . 92

**7 Simulation and Control of Dynamical Systems with Dry Friction 96**
7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
7.2 Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
7.3 Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
7.4 Optimization technique . . . . . . . . . . . . . . . . . . . . . . . 102
7.5 Numerical results . . . . . . . . . . . . . . . . . . . . . . . . . . 104
7.6 Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108

**8 An Algorithm for the Numerical Solution of Some Shape Optimization Problems 114**
8.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
8.2 The shape optimization problem . . . . . . . . . . . . . . . . . . . 115
8.3 Fictitious-domain decomposition method for elliptic problems . . . . 116
8.4 An elliptic, shape-optimization test problem; optimizing eigenvalues . 119
8.4.1 Test problem 1 . . . . . . . . . . . . . . . . . . . . . . . . . 120
8.4.2 Test problem 2 . . . . . . . . . . . . . . . . . . . . . . . . . 121
8.4.3 Test problem 3, an eigenvalue maximization . . . . . . . . . . 121
8.4.4 Summary of numerical results . . . . . . . . . . . . . . . . . . 123

8.5 Fictitious-domain method for Navier-Stokes . . . . . . . . . . . . . . . 123
8.6 Shape optimization for flow problems . . . . . . . . . . . . . . . . . . . . 127
8.7 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129

**9 Conclusions** **131**

**Bibliography** **132**

# Illustrations


2.1 Quadratic program with inconsistent constraints . . . . . . . . . . . . 14
2.2 Relaxed quadratic program with consistent constraints . . . . . . . . 14
2.3 Sketch of proposed trust-region algorithm . . . . . . . . . . . . . . . . 29

3.1 Initial Simplex Shapes when $n = 2$ . . . . . . . . . . . . . . . . . . . . 36
3.2 An illustration for $n = 2$ of all three modifications (reflection, contraction and expansion) to the simplex $\Omega$ defined by the points $x_0, x_1, x_2$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
3.3 Different “direct search methods” . . . . . . . . . . . . . . . . . . . . . 42

4.1 Triangulation of the domain $\Omega$ . . . . . . . . . . . . . . . . . . . . . . . 55
4.2 Analytic target shape . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
4.3 Pyramid target shape . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.4 Cylinder target shape . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
4.5 Profile of state and first target for 4.15 with $\epsilon = 1.E - 9$ . . . . . . . 61
4.6 Profile of state and first target for 4.16 with $\eta = .029$ . . . . . . . . . 61
4.7 Profile of state and first target for 4.17 with $\epsilon = 1.E - 9$ . . . . . . . 61
4.8 Profile of state and first target for 4.18 with $\eta = .029$ . . . . . . . . . 61
4.9 Profile of state and second target for 4.15 with $\epsilon = 1.E - 9$ . . . . . . 62
4.10 Profile of state and second target for 4.16 with $\eta = .0933$ . . . . . . . 62
4.11 Profile of state and second target for 4.17 with $\epsilon = 1.E - 9$ . . . . . . 62
4.12 Profile of state and second target for 4.18 with $\eta = .0933$ . . . . . . . 62
4.13 Profile of state and third target for 4.15 with $\epsilon = 1.E - 9$ . . . . . . . 63
4.14 Profile of state and third target for 4.16 with $\eta = .352$ . . . . . . . . . 63
4.15 Profile of state and third target for 4.17 with $\epsilon = 1.E - 9$ . . . . . . . 63
4.16 Profile of state and third target for 4.18 with $\eta = .352$ . . . . . . . . . 63
4.17 Computed boundary Neumann control for 4.15 with $\epsilon = 1.E - 9$ . . . 64
4.18 Computed boundary Neumann control for 4.16 with $\eta = .029$ . . . . . 64
4.19 Profile of state and first target for 4.17 with $\epsilon = 1.E - 9$ . . . . . . . 64
4.20 Profile of state and first target for 4.18 with $\eta = .029$ . . . . . . . . . 64
4.21 Computed boundary Neumann control for 4.15 with $\epsilon = 1.E - 9$ . . . 65
4.22 Computed boundary Neumann control for 4.16 with $\eta = .093$ . . . . . 65
4.23 Computed boundary Neumann control for 4.17 with $\epsilon = 1.E - 9$ . . . 65
4.24 Computed boundary Neumann control for 4.18 with $\eta = .093$ . . . . . 65
4.25 Computed boundary Neumann control for 4.15 with $\epsilon = 1.E - 9$ . . . 66

4.26 Computed boundary Neumann control for 4.16 with $\eta = .352$ . . . . . 66
4.27 Computed boundary Neumann control for 4.17 with $\epsilon = 1.E-9$ . . . 66
4.28 Computed boundary Neumann control for 4.18 with $\eta = .352$ . . . . . 66
4.29 Mesh of state variables solving 4.15 with $\epsilon = 1.E-9$ . . . . . . . . . 67
4.30 Mesh of state variables solving 4.16 with $\eta = .029$ . . . . . . . . . . . 67
4.31 Mesh of state variables solving 4.17 with $\epsilon = 1.E-9$ . . . . . . . . . 67
4.32 Mesh of state variables solving 4.18 with $\eta = .029$ . . . . . . . . . . . 67
4.33 Mesh of state variables solving 4.15 with $\epsilon = 1.E-9$ . . . . . . . . . 68
4.34 Mesh of state variables solving 4.16 with $\eta = .093$ . . . . . . . . . . . 68
4.35 Mesh of state variables solving 4.17 with $\epsilon = 1.E-9$ . . . . . . . . . 68
4.36 Mesh of state variables solving 4.18 with $\eta = .093$ . . . . . . . . . . . 68
4.37 Mesh of state variables solving 4.15 with $\epsilon = 1.E-9$ . . . . . . . . . 69
4.38 Mesh of state variables solving 4.16 with $\eta = .352$ . . . . . . . . . . . 69
4.39 Mesh of state variables solving 4.17 with $\epsilon = 1.E-9$ . . . . . . . . . 69
4.40 Mesh of state variables solving 4.18 with $\eta = .352$ . . . . . . . . . . . 69

5.1 - State profiles at $t = T$ . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.2 - Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . 83
5.3 - State profiles at $t = T$ . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.4 - Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . 83
5.5 - State profiles at $t = T$ . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.6 - Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . 84

6.1 State variables $U$ at time $t = T$ . . . . . . . . . . . . . . . . . . . . . 92
6.2 State variables $U$ at time $t = T$ . . . . . . . . . . . . . . . . . . . . . 94
6.3 Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . . 94
6.4 Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . . 94
6.5 Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . . 95
6.6 Profiles of the controls . . . . . . . . . . . . . . . . . . . . . . . . . 95

7.1 - (TP1) - Displacements of uncontrolled system . . . . . . . . . . . . 108
7.2 - (TP1) - Displacements of uncontrolled system . . . . . . . . . . . . 108
7.3 - (TP1) - Multipliers of uncontrolled system (NR) . . . . . . . . . . . 108
7.4 - (TP1) - Multipliers of uncontrolled system (R) . . . . . . . . . . . . 108
7.5 - (TP1) - Displacements of controlled system . . . . . . . . . . . . . . 110
7.6 - (TP1) - Displacements of controlled system . . . . . . . . . . . . . . 110
7.7 - (TP1) - Multipliers of controlled system (NR) . . . . . . . . . . . . 111
7.8 - (TP1) - Multipliers of controlled system (R) . . . . . . . . . . . . . 111
7.9 - (TP2) - Displacements of uncontrolled system . . . . . . . . . . . . 111
7.10 - (TP2) - Velocities of uncontrolled system . . . . . . . . . . . . . . . 111

7.11 - (TP2) - Multipliers of uncontrolled system (NR) . . . . . . . . . . . 112
7.12 - (TP2) - Multipliers of uncontrolled system (R) . . . . . . . . . . . . 112
7.13 - (TP2) - Displacements of controlled system . . . . . . . . . . . . . . 112
7.14 - (TP2) - Velocities of controlled system . . . . . . . . . . . . . . . . . 112
7.15 - (TP2) - Multipliers of controlled system (NR) . . . . . . . . . . . . 113
7.16 - (TP2) - Multipliers of controlled system (R) . . . . . . . . . . . . . 113

8.1 - Starting shape for test problem #1 . . . . . . . . . . . . . . . . . . . 121
8.2 Final shape and target shape for test problem #1 . . . . . . . . . . . 121
8.3 Starting shape for test problem #2 . . . . . . . . . . . . . . . . . . . . 122
8.4 - Final shape and target for test problem #2 . . . . . . . . . . . . . . 122
8.5 Starting shape for test problem #3 . . . . . . . . . . . . . . . . . . . . 123
8.6 Final shape for test problem #3 . . . . . . . . . . . . . . . . . . . . . . 123
8.7 Construction of the closed trailing edge . . . . . . . . . . . . . . . . . 128
8.8 Initial guess shape and solution . . . . . . . . . . . . . . . . . . . . . 128

# Tables


2.1 Minimization parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
2.2 Numerical results from perturbed SQP algorithm using $\phi$ merit function with LM-BFGS . . . . . . . . . . . . . . . . . . . . . . . . 32
2.3 Numerical results from perturbed SQP algorithm using $\psi$ merit function with LM-BFGS . . . . . . . . . . . . . . . . . . . . . . . . 32
2.4 Numerical results from perturbed SQP algorithm using $\phi$ merit function with LM-SR1 . . . . . . . . . . . . . . . . . . . . . . . . . 32
2.5 Numerical results from perturbed SQP algorithm using $\psi$ merit function with LM-SR1 . . . . . . . . . . . . . . . . . . . . . . . . . 32

3.1 Numerical Test Parameters . . . . . . . . . . . . . . . . . . . . . . . . . 41
3.2 Performance of constrained-PDS with $\ell_1$ penalty function . . . . . . . 45
3.3 Performance of constrained-PDS with $\ell_2$ penalty function . . . . . . . 46
3.4 Performance of constrained-PDS with $\ell_\infty$ penalty function . . . . . . 47
3.5 Performance of constrained-PDS with suggested $\Pi_L$ penalty function 48

4.1 Numerical Test Parameters . . . . . . . . . . . . . . . . . . . . . . . . . 59
4.2 Numerical results for formulation 4.15 . . . . . . . . . . . . . . . . . . . 59
4.3 Numerical results for formulation 4.16 . . . . . . . . . . . . . . . . . . . 59
4.4 Numerical results for formulation 4.17 . . . . . . . . . . . . . . . . . . . 60
4.5 Numerical results for formulation 4.18 . . . . . . . . . . . . . . . . . . . 60

5.1 Numerical relults for formulation 5.9 . . . . . . . . . . . . . . . . . . . 84
5.2 Numerical results for formulation 5.10 . . . . . . . . . . . . . . . . . . . 85

6.1 Numerical results for formulation 5.9 . . . . . . . . . . . . . . . . . . . 93
6.2 Numerical results for formulation 5.10 . . . . . . . . . . . . . . . . . . . 93

7.1 Problem I: Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
7.2 Problem I: Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
7.3 Problem 2: Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
7.4 Problem 2: Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
7.5 Difference between regularization and no reglarization . . . . . . . . . 110

8.1 Numerical results for test problem # 1 using the two loop formulation 123
8.2 Numerical results for test problem # 1 using the single loop formulation124

8.3 Numerical results for test problem # 1 using the two loop formulation 124
8.4 Numerical results for test problem # 2 using the single loop formulation 124
8.5 Numerical results for test problem # 3 using the two loop formulation 124
8.6 Numerical results for test problem # 3 using the single loop formulation 125
8.7 Coarse minimization: 10 points per processor . . . . . . . . . . . . . . 130
8.8 Fine minimization: 25 points per processor . . . . . . . . . . . . . . . . 130

# Chapter 1

# Introduction

Until quite recently, the study of Optimal Control Problems (OCP) and the study of Nonlinear Programming Problems (NLP) have been conducted independently. While these two areas of mathematics are strongly related, mathematical tools employed by researchers from these two fields have remained separate. We intend to address some algorithmic issues in an area somewhere between these two closely related fields.

Arguably, the most significant modern advance made both in theory and computation of Control Problems (CP) and Optimal Control Problems (OCP) is the Hilbert Uniqueness Method (HUM) of J. L. Lions. (For a comprehensive description see the papers by Glowinski and Lions [70] and [71]; for an illustrative review article also written by Lions, [98]). The theory surrounding the HUM has led to remarkable advances in the understanding of OCP, including questions of well-posedness, exact controllability and approximate controllability. Using HUM, one can formulate OCP in various forms of optimization problems. Then, satisfaction of the governing Partial Differential Equation (PDE) can be achieved through variational methods. (A comprehensive survey of numerical methods for nonlinear variational problems can be found in Glowinski [65] and the references therein). These two tools, numerical methods for nonlinear variational problems and HUM, create an arena in which the resulting discretized problems can be solved by optimization techniques.

There have been several significant recent advances in optimization. The solution of the general NLP (minimizing a function while fulfilling a collection of equality and inequality constraints) has become better understood. The rates of convergence of classes of algorithms have been established and, in some cases, theorems have been proved guaranteeing convergence from any starting point. Robust methods have been developed for approximating the quantities used in various algorithms. These advances have become standard tools of optimization. Coupled with the advent of new, larger and faster computers, they have spawned a family of very effective algorithms for various classes of optimization problems.

One special case of NLP that has seen extraordinary growth is the smooth NLP (when the function to be minimized and all the constraints are twice continuously differentiable). Recently, particular attention has been paid to the use of 'interior point' methods for solving smooth optimization problems with inequality constraints (see for example the classic book by Fiacco and McCormick [57] and modern review papers by Gonzaga [77] and [78], F. Jarre [90] and M. Wright [149]). Interior point methods have been the subject of over 1500 research papers over the past decade and this work has resulted in numerous rapidly convergent algorithms that solve NLP.

Recently, solving non-differentiable NLP (when derivatives are unavailable, either too costly to compute or simply too inaccurate) has also been the focus of much research. Certain classes of CP and OCP have a non-smooth nature that is simply unavoidable; as a result they give rise to non-differentiable NLP.

We propose the design and application of two specialized and hybrid algorithms. The first is intended for the solution of twice continuously differentiable NLP in problem formulations like CP and OCP. This algorithm uses interior-point methods plus perturbation ideas for handling both equality and inequality constraints. As the principal mechanism for solving these problems, we will use the Sequential Quadratic Programming (SQP) algorithm (for a modern review of SQP see [12]). An SQP algorithm approximates the NLP through a local model with a quadratic objective function and linear constraints. Solution of the problem associated with this model yields a direction or *step* along which one must locate a desirable point to stop and then construct a new local model. For use within the SQP arena, Lagrange multiplier estimates, Hessian approximation techniques, and numerous variants of globalization schemes have been suggested; most of them have proved quite successful on relatively broad classes of problems. This algorithm is one of the most effective mathematical and computational tools for the solution of problems such as NLP.

The second hybrid algorithm we propose is to be used for the solution of non-differentiable NLP. It is based on modifications to a class of algorithms called direct search methods. Originally, direct search methods were designed to solve unconstrained optimization problems.

The main thrust of this work is in the collection of CP and OCP that are representative of those arising in science and engineering. Each of these problems contains some special structure generally not shared by the others. Formulations tailored to the structures are discussed and tested numerically.

The thesis is organized as follows: the chapter which follows contains a description of our hybrid SQP algorithm. The 3rd chapter explains another hybrid algorithm intended to solve non-differentiable NLP. The remainder of the thesis deals with applications. We implement these two algorithms to solve problems which are CP or OCP. The 4rth chapter considers boundary control of the heat equation through Neumann boundary conditions. The 5fth chapter contains work on an interesting special class of OCP called hierarchical control problems. In the 6th chapter we consider a class of control problems with special governing equations, sometimes called *Stefan problems.* These problems model the propagation of zones of melting or freezing in materials. The subject of the 7th chapter is control of systems in the presence of Coulomb friction (sometimes called dry friction). In chapter 8, a control theory approach is applied to the solution of a class of shape optimization problems. Conclusions and comments on future research are in the 9th and last chapter of the thesis.

Each chapter of the thesis is written as an independent unit. Occasionally the notation and symbols used will differ between chapters. Notation is explained in each chapter so that no confusion should arise.

# Chapter 2

# A Sequential Linear/Quadratic Programming Algorithm that Uses Relaxed-Constraint Subproblems

## 2.1 Preliminary considerations

Modern computing environments and new algorithms for the solution of structured optimization problems have resulted in the birth of a new and vibrant sub-field of optimization, solution techniques for large-scale optimization. We are using the term 'large-scale' here to denote a problem so large that, to achieve its solution, it is of vital importance to exploit some special structure inherent in the problem. In this we are following the definitions presented in [30] and [33]. The term 'large-scale' seems to change almost daily to denote larger and harder problems. Indeed, the solution of a problem of a size considered 'large-scale' today would have been absolutely unthinkable only ten years ago. Research in this field has spawned a family of very effective algorithms for various classes of optimization problems. Two excellent surveys, Coleman [30] and Conn, Gould, and Toint [33] describe how these recent trends in algorithms have been affected by the new architectures for computation.

Among the techniques for the solution of small- to mid-sized nonlinear programming problems, the Sequential Quadratic Programming algorithm (SQP) has been one of the most useful and robust. An comprehensive survey of the SQP methods has been published recently by Boggs and Tolle [12]. The SQP algorithm, in spite of its popularity and success in solving small- to mid-sized problems, has never competed successfully in the large-scale arena.

In this chapter, we will be particularly concerned with sparsity as the special structure, though other structural features of a problem may often be of equal or greater importance. Many problems arising in science and engineering can be formulated as sparse and large-scale nonlinear programming problems. Consequently, a variety of SQP algorithms have been developed for the solution of such problems (see, for examples, Nickel and Tolle [111], Murray and Preito [107], and Eldersveld [54] among other excellent papers and algorithms). In most of these algorithms, the consistency of a linear system (the linearized constraints) is assumed. This consistency is necessary for the valid completion of each iteration of the SQP algorithm (the exact or approximate solution of the quadratic program subproblem).

The powerful engine that drives SQP (and many other optimization algorithms) is Newton's method. This method is extremely expensive but works very well when iterates are close to the solution. When iterates lie far from the solution, however, the use of the method becomes more complicated because the non-local iterations are most likely to be plagued by inconsistency. An approach that seeks to lessen the cost of these non-local iterations is the *inexact Newton's methods* (see for example papers by Eisenstadt and Walker [50], Dembo, Eisenstadt, and Steihaug [42], Nash and Nocedal [108] and for inexact SQP methods see the papers by Dembo and Tulowitzki [43] and Yabe, Yamaki, and Takahashi [151]). Roughly put, the idea behind inexact Newton's methods is that one aims to solve a sequence of approximate problems whose solutions converge to the solution of the original problem. When the approximations are poor, probably at early iterations, one need solve the approximate problem only inexactly but, as iterations approach the solution, the approximate problems should be solved with increasing accuracy. This method can drastically lessen the cost of far away iterates without precluding the rapid local convergence inherent in Newton's method. But these ideas do not yet address the problem of inconsistency.

Here, we propose a somewhat different approach. The idea is to solve a set of perturbed problems whose solutions form a sequence converging to the solution of the original problem. The perturbed problems are solved as accurately as possible, regardless of the proximity of iterates to the solution of the original nonlinear problem. While iterates are far from the solution, perturbations are large, but as iterates approach the solution they shrink to zero. The important effect of the perturbation is that it yields a problem much easier to solve than the unperturbed problem.

In the next section we introduce the nonlinear programming problem and the associated Lagrange functions. In section 2.3 we describe a basic SQP line-search algorithm. Inconsistent subproblems in SQP algorithms have been studied and some modifications suggested. A summary of some of these modifications is presented in section 2.4. We follow this with our suggestions for a relaxation scheme and with some theoretical convergence results in sections 2.5 and 2.6. In section 2.7 we briefly describe an interior-point method for the solution of the subproblems that arise in our method. A detailed description of a practical and implementable version of the algorithm that incorporates our relaxation ideas follows in section 2.8. Specifically, we discuss some heuristic modifications and alternative globalizations of the SQP algorithm. We conclude in section 2.9 with numerical results, some ideas regarding strengths and weaknesses plus some comments regarding future work. An appendix contains special test problems that give rise to inconsistency.

## 2.2 Introduction

In this paper we consider the problem of finding an $x^*$ to minimize a nonlinear objective function $f(x)$ subject to a set of constraints which include some mix of equality constraints $h(x) = 0$, and inequality constraints $g(x) \leq 0$. This general nonlinear programming problem (NLP) is written

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & h(x) = 0 \\ & g(x) \leq 0, \end{array} \tag{2.1}$$

where $f : \Re^n \rightarrow \Re$, $h : \Re^n \rightarrow \Re^{m_e}$, and $g : \Re^n \rightarrow \Re^{m_i}$ are assumed to be twice continuously differentiable nonlinear functions, $f, g, h \in C^2(\Re^n)$. We introduce the notation for the Lagrangian and augmented Lagrangian functions associated with this NLP problem. Let

$$\ell(x, \lambda, \mu) = f(x) + \lambda^{\mathsf{t}} g(x) + \mu^{\mathsf{t}} h(x) \tag{2.2}$$

and

$$\ell_{\rho}^{W}(x, \lambda, \mu) = f(x) + \lambda^{\mathsf{t}} g_{\oplus}(x) + \mu^{\mathsf{t}} h(x) + \rho \|(g_{\oplus}(x), h(x))^{\mathsf{t}}\|_{W(x)}^{2} \tag{2.3}$$

be the Lagrangian and augmented Lagrangian associated with (NLP). The matrix $W(x)$ is a symmetric, positive-definite matrix and the matrix norm $\|\cdot\|_{W(x)}$ is the Euclidean norm weighted by $W(x)$. For this exposition, one can assume that $W(x) = I$, the identity matrix. Here, $g_{\oplus}(x)$ is the standard notation for the function

$$g_{\oplus}(x) \equiv \begin{cases} g(x) & \text{if } \ g(x) > 0 \\ 0 & \text{if } \ g(x) \leq 0. \end{cases} \tag{2.4}$$

We call any 3-tuple, $(x_*, \lambda_*, \mu_*)$ a *Karush–Kuhn–Tucker* point (K. K. T. point) if it satisfies both first-order necessary conditions and second-order sufficient conditions for solutions of NLP. Specifically, $(x^*, \lambda^*, \mu^*)$ is a K. K. T. point if it satisfies

$$\begin{array}{l} \nabla_x \ell(x_*, \lambda_*, \mu_*) = 0 \\ (\lambda_*)_i g_i(x_*) = 0 \ \text{ for all } i = 1, \ldots m_i \\ h(x_*) = 0 \\ g(x_*) \leq 0 \\ \lambda_* \geq 0 \end{array} \tag{2.5}$$

For a detailed discussion of classical *Karush–Kuhn–Tucker conditions* see books by M. Avriel [4], J. Céa [24] and D. Luenberger [102].

The tool we use to solve this problem is the Sequential Quadratic Programming (SQP) algorithm. This algorithm is one of the many effective mathematical and computational tools in the solution of problems such as NLP. The algorithm approximates NLP through a model problem with a quadratic objective function and linear constraints. Numerous variants of globalization schemes have been suggested for SQP; most of them have proved to be quite successful on broad classes of problems. In later sections we will employ a globalization strategy that relies on an auxiliary, squared slack-variable version of the problem. We quickly state this auxiliary problem, the squared slack-variable, nonlinear programming problem (SNLP),

$$(SNLP)\begin{cases} \min\limits_{s,x} & f(x) \\ s.t. & c\left(\dfrac{1}{2}s^2, x\right) = g(x) + \dfrac{1}{2}s^2 = 0 \\ & h(x) = 0. \end{cases} \tag{2.6}$$

As an algebraic convenience we will define $z = \frac{1}{2}s^2$ where $z \geq 0$. This usage, while decreasing the algebraic clutter, may cloud the issue of smoothness . In particular, $c(z, x)$ is not differentiable with respect to the variable $s$. The function $c(z, x)$ has a discontinuity in its derivatives with respect to the variables $s$ at the point $s = 0$. But, provided $z > 0$, this lack of smoothness is not relevant. A more rigorous treatment of this inconveninece can be found in Boggs, Tolle, and Kearsley [16] and [17].

We can now define squared slack variable analogs to (2.2) and (2.3) for our auxiliary problem SNLP in a straightforward way.

$$L(x, z, \hat{\lambda}, \hat{\mu}) = f(x) + \hat{\lambda}^{\mathsf{t}}(g(x) + z) + \hat{\mu}^{\mathsf{t}}h(x) \tag{2.7}$$

and

$$L_\rho^W(x, z, \hat{\lambda}, \hat{\mu}) = f(x) + \hat{\lambda}^{\mathsf{t}}(g(x) + z) + \hat{\mu}^{\mathsf{t}}h(x) + \rho\|(g(x) + z), h(x)^{\mathsf{t}}\|^2_{W(x)}. \tag{2.8}$$

The Lagrange multipliers from SNLP, $(\hat{\lambda}, \hat{\mu})$, differ from those of NLP, $(\lambda, \mu)$, because of the dependence of the former on the non-negative slack variables $z$.

With the NLP and SNLP defined along with the associated Lagrangian functions, we can begin to describe an algorithm for the solution of NLP. We emphasize that the SNLP is *never* solved explicitly. The SNLP is used solely to define a globalization scheme, thus avoiding the classical criticisms of increased problem size, increased condition number, and asymptotic singularities. For a more complete justification and a discussion of the use of squared slack-variables in this context, see the paper by Tapia, [136].

## 2.3 Basic algorithms

The SQP algorithm for NLP in an abbreviated form can be simply stated as follows:

Basic SQP algorithm

1. *Given $x_0$, and $B_0$*

2. *Solve Quadratic Programming Problem (QP)*

$$(QP)\begin{cases} \min\limits_{\delta_k} & \nabla f(x_k)^{\mathsf{t}}\delta_k + \dfrac{1}{2}\delta_k{}^{\mathsf{t}} B_k \delta_k \\ s.t. & \nabla g(x_k)^{\mathsf{t}}\delta_k + g(x_k) \leq 0 \\ & \nabla h(x_k)^{\mathsf{t}}\delta_k + h(x_k) = 0 \end{cases} \tag{2.9}$$

$$\|D\delta_k\| \leq \Delta_k \tag{2.10}$$

3. *Update $\Delta_k$ as a function of $x_k$ and $\delta_k$*

4. *Choose step-length $\alpha_k$ using merit function and line-search algorithm*

5. *Update $B_{k+1}$ as a function of $(x_k + \alpha_k \delta_k)$*

6. *Check convergence*

7. *Go to 2*

The matrix $B_k$ should approximate the Hessian with respect to the current iterate $(x_k, \lambda_k, \mu_k)$ of either the Lagrangian function, (2.2), or the augmented Lagrangian function, (2.3), associated with NLP. We will consider,

$$B_k \approx \nabla_x^2 \ell_\rho^I(x_k, \lambda_k, \mu_k). \tag{2.11}$$

Subscripts are dropped from differentiation symbols here and for the remainder of this work. Differentiation can be assumed to be with respect to the $x$ variable (e.g., $\nabla^2 = \nabla_x^2$). Clearly, the initial guesses $x_0$ and $B_0$ from Step 1 should be as close as possible to the solution $x_*$ and to the Hessian of the augmented Lagrangian evaluated at the solution, respectively.

The trust-region constraint (2.10) is a safeguard added to avoid steps that are predictably too long. It is not intended to play a role in the globalization procedure. The diagonal scaling matrix $D$ is constructed from estimates of the orders of magnitude of $x_*$. For example, if [ diag $(\tilde{x}_*)$] denotes a diagonal matrix with elements of the magnitudes of the component of $x_*$, then a good choice for $D$ would be $D = \{$ diag $(\tilde{x}_*)\}^{-1}$. The symbol $\Delta$ denotes an estimate of the maximum allowable size of the scaled step. To maintain and monitor an estimate of the trust-region radius, a conservative version of a standard technique is used. This technique is discussed in section 2.7. For a complete discussion of updating strategies for trust regions, the reader is referred to the book by Dennis and Schnabel [44] and the paper by Moré and Sorensen [105].

We are interested primarily in the problems arising when the linearized constraints are inconsistent at step 2. In that event, no $\delta_k$ exists satisfying the linearized form of (2.9). We anticipate that this difficulty will occur when iterates lie far from the solution.

The calculation of the step-length parameter, $\alpha_k$, using a line search procedure, requires the employment of a merit function. There are a myriad of possible merit functions to be used for problem (2.1) that have been suggested in the literature. For the analysis, we will restrict ourselves to a very simple one but, for comparison of implementation details, we will employ a merit function more appropriate to numerics.

The merit function considered in the analysis is a simple $\ell^\infty$ penalty function. We denote the function giving maximum constraint violation as $V$,

$$V(x) = \max\left\{|h_1(x)|, |h_2(x)|, \ldots, |h_{m_i}(x)|, 0, g_1(x), g_2(x), \ldots, g_{m_i}(x)\right\}, \tag{2.12}$$

and our merit function as $\phi_\rho(x)$. Specifically, we choose the merit function,

$$\phi_\rho(x) = f(x) + \rho V(x). \tag{2.13}$$

The variable $\rho$ is a penalty parameter. We have used the same variable for the augmentation parameter in the earlier definitions (2.2), (2.7), (2.3) and (2.8). However, the role that it plays will be clear from the context and this slight abuse of notation should cause no problem.

While this merit function is easy to analyze, it has many drawbacks. It is not a Fréchet differentiable function with respect to $x$, for instance. Nevertheless, $\phi_\rho(x)$ is a Gâteaux or one-sided differentiable function and, for this reason, we introduce the directional derivative of $\phi$ in the direction $\delta$ denoted by,

$$\Pi(\phi_\rho(x); \delta) = \lim_{t \to +0} \frac{(\phi_\rho(x + t\delta) - \phi_\rho(x))}{t}. \tag{2.14}$$

The line search we will use is similar to the one introduced in the work of S. P. Han [81] in which, instead of requiring both an upper and a lower $Armijo-Goldstein-Wolfe$ condition, a single combined condition is enforced. The line search is given below.

### Line Search

1. *Given* $\delta_k$ , $x_k$ *and* $(\nu_1, \nu_2)$ ; *Set* $j = 0$
2. *Set* $\alpha_{k,1} = 1$
3. *If we satisfy*
$$\phi_\rho(x_k + \alpha_{k,j}\delta_k) \leq \phi_\rho(x_k) - \nu_1 \alpha_{k,j} \delta_k^{\mathrm{t}} B_k \delta_k \tag{2.15}$$
then *set* $\alpha_k = \alpha_{k,j}$ *and return*
4. *Set* $\alpha_{k,j+1} = \nu_2 \alpha_{k,j}$
5. *Set* $j = j + 1$ ; *Go to Step 2*

When a step reduces the merit function insufficiently, the length of the step is reduced to a fraction ($\nu_2$) of its original length. The parameter $\nu_1$ dictates how much reduction in the merit function is needed before a step is deemed acceptable. The reader is referred to a survey paper [12] for a more complete discussion of line search procedures in the context of merit functions for SQP.

Updating the approximation of the Hessian of the Lagrangian or augmented Lagrangian function is the subject of much research. We are primarily interested in methods that are least taxing of memory. Some popular and numerically effective updates that are not memory-intensive include limited memory BFGS (LM-BFGS) and the symmetric rank-one (SR1) updates (see for example papers by Liu and Nocedal [101], Nocedal[112], and Byrd, Nocedal and Schnabel [20] for papers describing limited memory updates). We will restrict our discussion to (LM-BFGS) because of its hereditary, symmetric, positive-definite properties, for reasons presented in section 2.7. Numerical performance of this algorithm with different choices of Hessian approximations and merit functions will be discussed in section 2.7.

Note, finally, that to initiate a successful termination of the algorithm at step 7, there will be two convergence criteria. A precise statement of these conditions is left for section 2.7. Here, we only mention briefly that one condition will require that the gradient of the Lagrangian function with respect to $x$ be desirably small; the second that the size of the solution to the quadratic program be sufficiently small. The second condition is the less desirable of the two stopping criteria.

## 2.4 Previously suggested relaxations

The subject of perturbations of nonlinear programs has been a fruitful area of optimization research. Indeed, one way of viewing relaxations of constraints is through examining nonlinear programs with data perturbations (see for example Fiacco [56]). Many algorithms for the solution of linear, nonlinear, integer, and even stochastic optimization problems have been improved by modifications based on data perturbations. Little attention has been given, however, to the systematic development of an algorithm tailored specifically to deal with inconsistency through perturbations of nonlinear programming.

Various suggestions have been made for coping with inconsistency in the subproblem arising when SQP methods are used to solve the general NLP. We mention only the few which motivated or are closely related to the work described in this paper. M. J. D. Powell [118], K. Tone [139] and Bartholomew–Biggs [6] suggest relaxations for problems with inconsistent linearized inequality-constraints. As early as 1977, M. J. D. Powell [118] suggested a relaxation based upon introducing an auxiliary variable, $p$, into the quadratic programming problem. The linearized inequality constraints in

(2.1) are then replaced by

$$\begin{aligned} &(\nabla h_i(x_k)^{\mathsf{t}}\delta_k) + p h_i(x_k) = 0 \\ &(\nabla g_i(x_k)^{\mathsf{t}}\delta_k) + p_i g_i(x_k) \leq 0 \end{aligned} \tag{2.16}$$

where $p_i$ is defined to be,

$$p_i \equiv \begin{cases} 1 & \text{if } \; g_i(x) > 0 \\ p & \text{if } \; g_i(x) \leq 0 \end{cases} \tag{2.17}$$

Here, $p$ is taken to be as large as possible, subject to the simple bounds $0 \leq p \leq 1$. The auxiliary variable $p$ must be included among the variables being minimized over, while the objective function in $QP$ is augmented with a penalty term $(-Mp)$. The parameter $M$ is taken to be a large constant.

Some serious drawbacks may arise with this early formulation. If the extended quadratic program admits only one feasible solution (i.e. $p = 0$ and $\delta = 0$) then, because it is assumed that the original nonlinear constraints in (2.1) are consistent, the associated recursive algorithm will halt. The assumption is certainly valid if all the constraints in (2.1) are linear. However, it is not difficult to construct a problem which is consistent without the auxiliary variable and yet is inconsistent when the auxiliary variable is included (see test problems in the paper by Tanabe [134]).

In a later work, K. Tone introduced two methods of constraint relaxation. The first relaxed only the equality constraints; the second relaxed equality and inequality constraints with equal bias. These relaxations result in larger feasible regions than the method suggested by Powell and always result in a consistent linear system. The second method replaces (2.16) by

$$\begin{aligned} &(\nabla h_i(x_k)^{\mathsf{t}}\delta_k + h_i(x_k) - \bar{t}_i + \hat{t}_i) = 0 \;\text{ for all } i \text{ such that } h_i(x_k) \neq 0 \\ &(\nabla h_i(x_k)^{\mathsf{t}}\delta_k = 0 \;\text{ for all } i \text{ such that } h_i(x_k) = 0 \\ &(\nabla g_i(x_k)^{\mathsf{t}}\delta_k) + g_i(x_k) \leq 0 \;\text{ for all } i \text{ such that } g_i(x_k) \leq 0 \\ &(\nabla g_i(x_k)^{\mathsf{t}}\delta_k) + p g_i(x_k) \leq 0 \;\text{ for all } i \text{ such that } g_i(x_k) > 0 \\ &\bar{t}_i, \hat{t}_i \geq 0 \\ &0 \leq p \leq 1. \end{aligned} \tag{2.18}$$

The quadratic program is modified by adding the penalty term

$$M_1 \sum_i (\bar{t}_i + \hat{t}_i) - M_2 p, \tag{2.19}$$

where the parameters $M_1$ and $M_2$ are sufficiently large constants. Both this modification and that of Powell result in a quadratic programming problem substantially larger than the original, with potentially 3 times as many variables to adjust in minimizing. It is exactly this growth in problem size that makes the suggestions of Powell and Tone difficult or impractical to use on some large-scale problems.

The work of M. C. Bartholomew–Biggs [6] suggests a different approach. Instead of relaxing the right-hand side of the constraints with only multiples of constraint violations, a nonlinear perturbation of the product of a penalty parameter with the square of a current estimate of constraint multipliers is used,

$$\begin{aligned} (\nabla h_i(x_k)^{\mathsf{t}}\delta_k) + p h_i(x_k) + \frac{1}{2}\rho\bar{\mu}_i &= 0 \\ (\nabla g_i(x_k)^{\mathsf{t}}\delta_k) + p_i g_i(x_k) + \frac{1}{2}\rho\bar{\lambda}_i &\leq 0. \end{aligned} \tag{2.20}$$

The $\bar{\mu}$ and $\bar{\lambda}$ are special estimates of the Lagrange multipliers associated with equality and inequality constraints, respectively. It can be shown that the system (2.20) with these special estimates of $\bar{\mu}$ and $\bar{\lambda}$ will always be consistent for positive $\rho$. This appears to be the earliest example of genuinely perturbed constraints for the purpose of eliminating inconsistency of linearized constraints. Significantly, this modification does not introduce new variables such as those of Powell and Tone into the quadratic subproblem.

It is worth mentioning that much of the work in trust region analysis of nonlinear programs, with only equality constraints, can be viewed as similar relaxations (see for example models by Vardi [145] and Celis, Dennis, and Tapia [25]). A more recent, beautiful work on shifted, modified, barrier-function algorithms by R. Polak has the effect of enlarging the feasible region (see [117]).

## 2.5 The proposed relaxation

The algorithm we propose adheres to certain principles. Only violated constraints will be relaxed. The strategy is to eliminate inconsistency in the linear system of constraints by relaxing constraints but never to relax any constraint that is satisfied at the current iterate. Moreover, a linear program yields the 'minimum' perturbation while still ensuring a constraint matrix consistent with the perturbed right-hand side.

As a first step, replace each linear equality constraint appearing in (2.9) with two linear inequality constraints. The Lagrange multiplier vector, $(\lambda, \mu)$, from (2.9) is replaced by the vector $(\lambda, \theta, \omega)$ where

$$\mu = \theta - \omega. \tag{2.21}$$

The second step uses (2.4) to add a perturbation to the right-hand side of the constraints in (2.9) for those linear equations associated with violated constraints. The size of the perturbation is determined by solving a linear program which yields an 'optimal' magnitude for the perturbation and a feasible point for the perturbed, quadratic programming subproblem.

We state the perturbed SQP algorithm in more detail.

Perturbed SQP algorithm

1. *Given $x_0$, $B_0$, and $\rho > 0$*

2. *Solve Linear Programming problem (LP) using feasible starting point*

$$(LP)\left\{\begin{array}{ll} \min\limits_{\tau,\delta_{0k}} & \nabla f(x_k)^{\mathsf{t}}\delta + M_\tau \tau \\ \text{s.t.} & \nabla g(x_k)^{\mathsf{t}}\delta_{0k} + g(x_k) \le \tau g_\oplus(x_k) \\ & \nabla h(x_k)^{\mathsf{t}}\delta_{0k} + h(x_k) \le \tau h_\oplus(x_k) \\ & -(\nabla h(x_k)^{\mathsf{t}}\delta_{0k} + h(x_k)) \le \tau h_\oplus(x_k) \\ & \tau \ge 0 \end{array}\right. \tag{2.22}$$

   yielding solution ($\tau_k$,$\delta_{Jk}$).

3. *Solve Quadratic Programming Problem (QP) using the solution above, $\delta_{Jk}$, as the feasible starting point*

$$(RQP)\left\{\begin{array}{ll} \min\limits_{\delta} & \nabla f(x_k)^{\mathsf{t}}\delta_k + \frac{1}{2}{\delta_k}^{\mathsf{t}}B_k\delta_k \\ \text{s.t.} & \nabla g(x_k)^{\mathsf{t}}\delta_k + g(x_k) \le (\tau + \epsilon_\tau)g_\oplus(x_k) \\ & \nabla h(x_k)^{\mathsf{t}}\delta_k + h(x_k) \le (\tau + \epsilon_\tau)h_\oplus(x_k) \\ & -(\nabla h(x_k)^{\mathsf{t}}\delta_k + h(x_k)) \le (\tau + \epsilon_\tau)h_\oplus(x_k) \end{array}\right. \tag{2.23}$$

$$\|D\delta\| \le \Delta \tag{2.24}$$

   yielding $(\delta_k, \lambda_k, \omega_k, \theta_k)$.

4. *Update $\Delta$ as a function of $x_k$, $\delta_k$ and the merit function $\phi_\rho$*

5. *Choose step-length $\alpha$ with the merit function $\phi_\rho$ and line search algorithm*

6. *Update $B_{k+1}$ as a function of $(x_k + \alpha\delta)$*

7. *Check convergence*

8. *Go to 2*

The parameter $M_\tau$ and $\epsilon_\tau$ are large and small positive constants, respectively. Making $M_\tau$ large ensures that the perturbation remains small; the positive $\epsilon_\tau$ guarantees that the feasible region in (2.23) will have a non-empty interior.

Other perturbations of the right-hand side are certainly possible. The perturbation, as stated, is restricted to lie in the subspace defined by the vector of constraint violations, $(g(x_k), h(x_k), -h(x_k))^{\mathsf{t}}_\oplus$. It is clear that this restriction could be removed by allowing $\tau$ to be a vector of $m_i + 2m_e$ variables in (2.22), but at the expense of greatly increasing the size of the problem. That choice of $\tau$ yields a smaller perturbation while still guaranteeing consistency. From the results of numerical experiments,

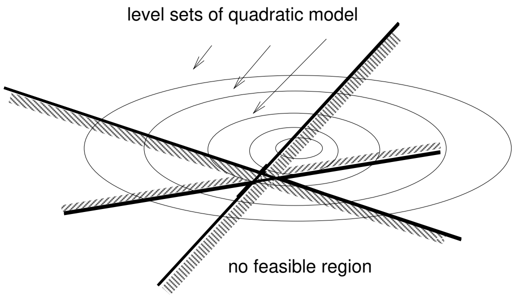


**Figure 2.1** Quadratic program with inconsistent constraints

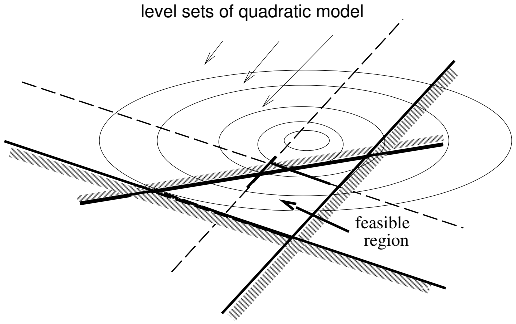


**Figure 2.2** Relaxed quadratic program with consistent constraints

however, it seems better to abandon this approach in favor of the less expensive approach above (fix the perturbation vector and solve only for the magnitude, $\tau \in \Re_+$).

It is clear, upon examining our system of linear inequalities as a homogeneous system, that we can find a solution to (2.22) that will yield a non-empty feasible interior and feasible starting point for the perturbed problem (2.9).

Whenever, at any point $\hat{x}$, we cannot find $(\bar{\tau}, \bar{\delta})$ to satisfy

$$\begin{bmatrix} \nabla g(\hat{x}) \\ \nabla h(\hat{x}) \\ -\nabla h(\hat{x}) \end{bmatrix}^{\mathsf{t}} \bar{\delta} + \begin{pmatrix} g(x_k) - \bar{\tau} g_{\oplus}(\hat{x}) \\ h(x) - \bar{\tau} h_{\oplus}(\hat{x}) \\ -h(x) - \bar{\tau} h_{\oplus}(\hat{x}) \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.25}$$

it must be the case that there exist positive $(\lambda, \omega, \theta)$ such that

$$\begin{pmatrix} \lambda \\ \omega \\ \theta \end{pmatrix} \begin{bmatrix} \nabla g(\hat{x}) \\ \nabla h(\hat{x}) \\ -\nabla h(\hat{x}) \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \lambda \\ \omega \\ \theta \end{pmatrix}^{\mathsf{t}} \begin{pmatrix} g(x_k) - \tau g_{\oplus}(\hat{x}) \\ h(x) - \tau h_{\oplus}(\hat{x}) \\ -h(x) - \tau h_{\oplus}(\hat{x}) \end{pmatrix} > \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.26}$$

and vice-versa. The existence of $(\lambda, \omega, \theta)$ satisfying (2.26) whenever the inequality (2.25) does not hold is guaranteed as an obvious consequence of a theorem of the alternative (see for example Theorem 22.1 on page 198 of the book by Rockefellar [124] and also Ekeland and Temam [51]). Because our relaxation is quite simple, the machinery of [124] and [51] is sufficient to establish the existence of $(\lambda, \omega, \theta)$.

We mention that a deeper study of the stability of inequalities in the context of constrained optimization exists (see Robinson [122] and [123]). When the relaxation is viewed as a perturbation of data, the work on differential stability in nonlinear programming by Gauvin and Tolle [62] is also pertinent.

## 2.6 Theory of convergence of the algorithm

In 1981, S. P. Han ([81]) proposed a variable metric algorithm for minimizing a class of non-differentiable functions, along with an associated global convergence proof. The convergence theorem is a strong one; it proves that all accumulation points of the iteration sequence for this algorithm are K.K.T. points. Here, the analysis of convergence is very close to Han's analysis, but some important and significant modifications are necessary in applying the methodology to our perturbation procedure.

Standard notation will be used for the matrix and vector norms. The symbols $\|\cdot\|_\infty, \|\cdot\|_1, \|\cdot\|_2$ will denote the infinity norm, 1-norm, and 2-norm, respectively. We first define some quantities convenient for the analysis.

$$R1_k(x_k, \lambda_k, \theta_k, \omega_k) = \tau_k \left(\lambda_k^{\mathsf{t}} g_\oplus(x_k) + (\theta_k - \omega_k)^{\mathsf{t}} h_\oplus(x_k)\right), \tag{2.27}$$

$$R2_k(\lambda_k, \mu_k) = \|\lambda_k\|_1 + \|\mu_k\|_1 = \|\lambda_k\|_1 + \|\theta_k - \omega_k\|_1 \tag{2.28}$$

and

$$R3_k(\lambda_k, \theta_k, \omega_k) = \tau_k \max\left\{\|g_\oplus(x_k)\|_\infty, \|h_\oplus(x_k)\|_\infty\right\}. \tag{2.29}$$

We also write down the K.K.T. conditions associated with our perturbed problem,

$$(B_k\delta_k + \nabla f(x_k) + \lambda_k \nabla g(x_k) + (\theta_k - \omega_k)\nabla h(x_k)) = 0 \tag{2.30}$$

$$\begin{aligned} \Lambda_k \left(\nabla g(x_k)^{\mathsf{t}} + g(x_k) - \tau_k g_\oplus(x_k)\right) &= 0 \\ \Theta_k \left(\nabla h(x_k)^{\mathsf{t}} + h(x_k) - \tau_k h_\oplus(x_k)\right) &= 0 \\ \Omega_k \left(-\nabla h(x_k)^{\mathsf{t}} - h(x_k) - \tau_k h_\oplus(x_k)\right) &= 0 \end{aligned} \tag{2.31}$$

$$\begin{aligned} \left(\nabla g(x_k)^{\mathsf{t}} + g(x_k) - \tau_k g_\oplus(x_k)\right) &= 0 \\ \left(\nabla h(x_k)^{\mathsf{t}} + h(x_k) - \tau_k h_\oplus(x_k)\right) &= 0 \\ \left(-\nabla h(x_k)^{\mathsf{t}} - h(x_k) - \tau_k h_\oplus(x_k)\right) &= 0 \end{aligned} \tag{2.32}$$

$$(\lambda_k, \theta_k, \omega_k) \geq 0. \tag{2.33}$$

Upper-case Greek letters denote diagonal matrices formed from the associated multiplier vectors designated by corresponding lower-case Greek letters. (e.g., $\Lambda_k =$ diag $\{\lambda_k\}$, etc.)

Assume the following:

Assumption 1: Objective and constraint functions are all sufficiently smooth ($f, h_i, g_j \in C^2(\Re^n)$ for $i = 1 \ldots m_e$ and for $j = 1 \ldots m_i$).

Assumption 2: For every $k$, the perturbed subproblem (RQP) can be solved.

Assumption 3: For the given $\rho$, the level set of the merit function at the initial guess, Lev $(\phi_\rho(x_0))$, is a compact set.

$$\text{Lev } (\phi_\rho(x_0)) = \{x \in \Re^n | \phi_\rho(x) \leq \phi_\rho(x_0)\}, \tag{2.34}$$

and

$$\rho \geq \|\lambda_k\|_1 + \|\mu_k\|_1 \quad \text{for all } k. \tag{2.35}$$

Assumption 4: The approximations to the Hessian are both positive–definite and bounded; i.e., there exist positive constants $\eta_{Max}, \eta_{Min}$such that

$$\eta_{Min}\|\delta\|^2 \leq \delta^{\mathsf{t}} B_k \delta \leq \eta_{Max}\|\delta\|^2$$

for any $\delta \in \Re^n$ and for all $k$.

Assumption 5: The perturbations remain sufficiently small with respect to the size of the step for all $k$. Specifically,

$$\tau_k \left(R1_k + \rho V(x_k)\right) < \|\delta_k\|_2^2 \eta_{Min} \left(\frac{1}{2} - \nu_1\right). \tag{2.36}$$

The first assumption is common to most global convergence theorems of this type. At the end of the last section, we demonstrated that the second assumption will always be satisfied if (2.22) is solved to completion. Indeed, this assumption is much less of a restriction than requiring the unperturbed quadratic program (2.9) to be soluble. The third assumption can be replaced by an assumption that the iterates $x_k$ plus the step $(x_k + \delta_k)$ remain in a compact set. The assumption that the Hessian approximations are uniformly symmetric and positive-definite is quite crucial to our convergence proof and to this algorithm. Allowing negative curvature in our quadratic model introduces local (non-global) minimizers into the subproblem, complicating the analysis significantly. This curvature can also have deleterious effects on the numerics for the case with large perturbations. Assumption 5 is the least straightforward and certainly the most restrictive of the assumptions. It is tantamount to assuming that the size of the perturbation remains small with respect to the size of the step, thus ensuring that the line-search procedure doesn't degenerate. A condition necessary (but not sufficient) for this assumption to hold is that at least one constraint is satisfied at every iteration, a condition that is ensured if the original nonlinear program, (2.1), has at least one equality constraint (i.e. $m_e \geq 1$).

We prove a set of lemmas that will yield our global convergence theorem. The first two lemmas supply an estimate of a directional derivative of the objective function and show that the solution to (2.23) is a descent direction for our merit function. The third lemma proves that the line-search procedure terminates after a finite number of iterations. These lemmas, together with our assumption 3, prove that the iterates, $x_k$, remain in a compact set defined by Lev $(\phi_\rho(x_0))$. Finally, a uniform bound on the step size, $\alpha_k$, is found and the size of the step is shown to converge to zero as $k \to \infty$. With the assumptions and the five lemmas in hand, we will state our convergence theorems.

**Lemma 2.1** *For any $k$,*

$$\nabla f(x_k)^{\mathsf{t}}\delta_k \leq -\delta_k^{\mathsf{t}} B_k \delta_k + V(x_k)R2_k + R1_k \tag{2.37}$$

**Proof** From (2.30) we know that

$$\delta_k^{\mathsf{t}}\left(B_k\delta_k + \nabla f(x_k) + \lambda_k \nabla g(x_k) + (\theta_k - \omega_k)\nabla h(x_k)\right) = 0.$$

Combining this with the complementarity equations, (2.31), yields

$$\begin{aligned}\delta_k^{\mathsf{t}}\nabla f(x_k) &\leq -\delta_k^{\mathsf{t}} B_k\delta_k + \lambda_k^{\mathsf{t}} g(x_k) + \mu_k h(x_k) + R1_k \\ &\leq -\delta_k^{\mathsf{t}} B_k\delta_k + R2_k V(x_k) + R1_k.\end{aligned}$$

The inequalities above are identical to (2.37) establishing the lemma.

□

**Lemma 2.2** *For any iteration $k$, the step $\delta_k$ is a descent direction for the merit function, and the following holds;*

$$\begin{aligned}\Pi(\phi_\rho(x_k);\delta_k) &\leq -\delta_k^{\mathsf{t}} B_k\delta_k + R1_k + R3_k + V(x_k)(R2_k - \rho) \\ &\leq -\eta_{Min} + R1_k + R3_k + V(x_k)(R2_k - \rho) \\ &< 0.\end{aligned} \tag{2.38}$$

**Proof** Let the largest constraint violation at iteration $k$, $V(x_k)$, be attained by constraint functions $I, J, K$. In other words, we partition $V(x_k)$ into three sets, $I = \{i : g_i(x_k) = V(x_k)\}$, $J = \{i : h_i(x_k) = V(x_k)\}$ and $K = \{i : -h_i(x_k) = V(x_k)\}$. Then, the feasibility conditions (2.32) allow us to write

$$\max_{i\in I}\left(\nabla g_i(x_k)^{\mathsf{t}}\delta_k\right) \leq \max_{i\in I}\left(g_i(x_k)(\tau_k - 1)\right)$$

$$\max_{i\in J}\left(\nabla h_i(x_k)^{\mathsf{t}}\delta_k\right) \leq \max_{i\in J}\left(h_i(x_k)(\tau_k - 1)\right)$$

$$\max_{i\in K}\left(-\nabla h_i(x_k)^{\mathsf{t}}\delta_k\right) \leq \max_{i\in K}\left(h_i(x_k)(1-\tau_k)\right)$$

Thus, on re-examining our directional derivative, we see that

$$\begin{aligned}\Pi(V(x_k);\delta_k) = \quad & \max\left\{\max_{i\in I}\left(\nabla g_i(x_k)^{\mathsf{t}}\delta_k\right), \max_{i\in J}\left(\nabla h_i(x_k)^{\mathsf{t}}\delta_k\right), \max_{i\in K}\left(-\nabla h_i(x_k)^{\mathsf{t}}\delta_k\right)\right\} \\ & \leq R3_k - V(x_k).\end{aligned}$$

The merit function separates into two components so that

$$\begin{aligned}\Pi(\phi_\rho(x_k);\delta_k) = \quad & \Pi(f(x_k);\delta_k) + \rho\Pi(V(x_k);\delta_k) \\ & \leq -\delta_k^{\mathsf{t}}B_k\delta_k + R1_k + R3_k + V(x_k)(R2_k - \rho) \\ & \leq -\eta_{Min} + +R1_k + R3_k + V(x_k)(R2_k - \rho).\end{aligned}$$

The second lemma has been established because the inequalities above are identical to those in (2.38).

□

> **Lemma 2.3** *For each iterations $k$, the step-length parameter $\alpha_k$ has a positive value and the suggested line-search given earlier will locate an acceptable positive value.*

**Proof** We prove this lemma by contradiction. Assume that the lemma is not true and, therefore, that at iteration $k$ the line-search condition (2.15) is not satisfied for any $j$. This implies that

$$\phi_\rho(x_k + \alpha_{k,j}\delta_k) \geq \phi_\rho(x_k) - \nu_1\alpha_{k,j}\delta_k^{\mathsf{t}}B_k\delta_k.$$

We form a difference quotient of $\phi_\rho(x_k)$ at $x_k$ in the direction of $\delta_k$ by an algebraic manipulation of this inequality. Then the limit of this difference quotient as $j \to +\infty$ results in $\alpha_{k,j} \to +0$ and yields a lower bound to the directional derivative. By applying Lemma 2, we attain the following:

$$\begin{aligned}-\nu_1\delta_k^{\mathsf{t}}B_k\delta_k &< \frac{1}{\alpha_{k,j}}\left(\phi_\rho(x_k + \alpha_{k,j}\delta_k) - \phi_\rho(x_k)\right) \\ -\nu_1\delta_k^{\mathsf{t}}B_k\delta_k &\leq \lim_{j\to+\infty}\frac{1}{\alpha_{k,j}}\left(\phi_\rho(x_k + \alpha_{k,j}\delta_k) - \phi_\rho(x_k)\right).\end{aligned}$$

The directional derivative can then be shown to satisfy

$$\Pi(\phi_\rho(x_k);\delta_k) \leq -\nu_1\delta_k^{\mathsf{t}}B_k\delta_k + R1_k + \rho R3_k.$$

Then, in view of assumptions 4 and 5, we arrive at the following:

$$\begin{aligned} & (\nu_1 - 1)\delta_k^{\mathsf{t}}B_k\delta_k + R1_k + \rho R3_k \\ \leq \quad & (\nu_1 - \frac{1}{2})\eta_{Min} - \delta_k^{\mathsf{t}}B_k\delta_k + R1_k + \rho R3_k \\ < 0. & \end{aligned}$$

There is clearly a contradiction here and the lemma is proved.

□

We pause, briefly, to use the previous three lemmas and our assumption 3 to define three useful constants.

$$\begin{aligned} F &\geq \sup_{\alpha\in(0,1)} \|\nabla^2 f(x_k+\alpha\delta_k)\|_2, \\ G &\geq \max_{i=1\dots m_i} \sup_{\alpha\in(0,1)} \|\nabla^2 g_i(x_k+\alpha\delta_k)\|_2, \\ H &\geq \max_{i=1\dots m_e} \sup_{\alpha\in(0,1)} \|\nabla^2 h_i(x_k+\alpha\delta_k)\|_2, \end{aligned} \tag{2.39}$$

for each iteration $k > 0$.

> **Lemma 2.4** *The sequence of step-length parameters $\{\alpha_k\}$ are a bounded sequence.*

**Proof** This lemma is also proved by contradiction. For some $k$, assume that $\alpha_k < \alpha_M$. It is obvious that $\alpha_M \leq \nu_2 \leq 1$ on recalling the role of $\nu_2$ in (2.15) of our line-search procedure. Therefore, there exists an integer $\hat{J} > 1$ such that $\alpha_k = \alpha_{k,\hat{J}}$. This, in turn, implies that

$$\phi_\rho(x_k + \alpha_{k,\hat{J}-1}\delta_k) > \phi_\rho(x_k) - \nu_1\alpha_{k,\hat{J}-1}\delta_k^{\mathsf{t}}B_k\delta_k,$$

and thus

$$-\nu_1\delta_k^{\mathsf{t}}B_k\delta_k < \frac{\phi_\rho(x_k+\alpha_{k,\hat{J}-1}\delta_k) - \phi_\rho(x_k)}{\alpha_{k,\hat{J}-1}}. \tag{2.40}$$

In view of the mean-value theorem, we can find $\hat{t} \in \Re^n$, $\hat{y} \in \Re^{m_i}$ and $\hat{z} \in \Re^{m_e}$ such that

$$f(x_k + \alpha_{k,\hat{J}-1}\delta_k) = f(x_k) + \alpha_{k,\hat{J}-1}\nabla f(x_k)^{\mathsf{t}}\delta_k + \frac{\alpha_{k,\hat{J}-1}^2}{2}\delta^{\mathsf{t}}\nabla^2 f(\hat{t})\delta_k.$$

The constraint violation function, $V$, satisfies

$$\begin{aligned} V(x_k + \alpha_{k,\hat{J}-1}\delta_k) &= \max\Big\{0, g_i(x_k+\alpha_{k,\hat{J}-1}\delta_k), h_j(x_k+\alpha_{k,\hat{J}-1}\delta_k)\Big\} \\ &= \max\Bigg\{0, g_i(x_k) + \alpha_{k,\hat{J}-1}\nabla g_i(x_k)^{\mathsf{t}}\delta_k + \frac{\alpha_{k,\hat{J}-1}^2}{2}\delta_k^{\mathsf{t}}\nabla^2 g_i(\hat{y})\delta_k, \\ &\qquad h_i(x_k) + \alpha_{k,\hat{J}-1}\nabla h_i(x_k)^{\mathsf{t}}\delta_k + \frac{\alpha_{k,\hat{J}-1}^2}{2}\delta_k^{\mathsf{t}}\nabla^2 h_i(\hat{z})\delta_k\Bigg\} \\ &\leq (1-\alpha_{k,\hat{J}-1}\delta_k)V(x_k) + \frac{1}{2}\alpha_{k,\hat{J}-1}^2(G+H)\|\delta_k\|_2^2 + \alpha_{k,\hat{J}-1}R1_k. \end{aligned}$$

The upper bound, $\alpha_M > \nu_2 \alpha_{k,\hat{\jmath}-1} = \alpha_{k,\hat{\jmath}} = \alpha_k$, combined with upper bounds given in (2.39) imply that

$$\frac{1}{\alpha_{k,\hat{\jmath}-1}} > \frac{1}{\eta_{Min}}(F + \rho(G + H)). \tag{2.41}$$

Applying Lemma 1, and recalling assumption 3 yields

$$\frac{1}{\alpha_{k,\hat{\jmath}-1}}\phi_\rho(x_k + \alpha_{k,\hat{\jmath}-1}\delta_k) - \phi_\rho(x_k)$$

$$\begin{aligned}
&= \frac{1}{\alpha_{k,\hat{\jmath}-1}}\Big((f(x_k + \alpha_{k,\hat{\jmath}-1}\delta_k) - f(x_k)) + \rho(V(x_k + \alpha_{k,\hat{\jmath}-1}\delta_k) - V(x_k))\Big) \\
&\le \left(\nabla f(x_k)^{\mathsf{t}}\delta_k + \frac{1}{2}\alpha_{k,\hat{\jmath}-1}\delta_k^{\mathsf{t}}\nabla^2 f(t)\delta_k\right) \\
&\quad +\rho\left(\frac{1}{2}\alpha_{k,\hat{\jmath}-1}(G + H)\|\delta_k\|_2^2 + (R3_k - V(x_k))\right) \\
&\le \nabla f(x_k)^{\mathsf{t}}\delta_k + \rho(R3_k - V(x_k))\frac{1}{2}\alpha_{k,\hat{\jmath}-1}(F + \rho(G + H))\|\delta_k\|_2^2 \\
&\le (R2_k - \rho)V(x_k) + \frac{\tau_k}{\|\delta_k\|_2^2}(R1_k + \rho V(x_k)) - \frac{1}{2}\delta_k^{\mathsf{t}}B_k\delta_k.
\end{aligned}$$

This inequality, when combined with (2.40), yields

$$\begin{aligned}
0 <\;& \frac{\tau_k}{\|\delta_k\|_2^2}(R1_k + \rho V(x_k)) + (R2_k - \rho)V(x_k) + (\nu_2 - \frac{1}{2})\delta_k^{\mathsf{t}}B_k\delta_k \\
\le\;& \|\delta_k\|_2^2\left(R1_k + \rho V(x_k) + (\nu_2 - \frac{1}{2})\delta_k^{\mathsf{t}}B_k\delta_k\right) + (R2_k - \rho)V(x_k) \\
\le\;& 0.
\end{aligned}$$

Therefore, choosing for $\alpha_M$,

$$\alpha_M = \nu_1 \min\left\{1, \frac{\eta_{Min}}{F + \rho(G + H)}\right\}, \tag{2.42}$$

for example, we have located a value for $\alpha_M$ producing the desired contradiction. The proof of the fourth lemma is complete.

□

**Lemma 2.5** *The limit of the descent directions converges to zero,*

$$\lim_{k\to\infty} \delta_k = 0. \tag{2.43}$$

**Proof** From the nature of the line-search procedure we know that

$$\phi_\rho(x_{k+1}) \geq \phi_\rho(x_k) - \nu_1 \alpha_k \delta_k^{\mathsf{t}} B_k \delta_k.$$

We get, upon summing both sides of the above equation, for any $N > 0$,

$$\begin{aligned} \phi_\rho(x_0) \; & \geq \phi_\rho(x_N) + \nu_1 \sum_k^N \alpha_k \delta_k^{\mathsf{t}} B_k \delta_k \\ & \geq f(x_{N+1}) + \nu_1 \alpha_M \eta_{Min} \sum_k^N \|\delta_k\|_2^2. \end{aligned}$$

Because of the compactness of the level sets of $\phi_\rho(x)$ and the smoothness of the objective function, we know that there exists $f_M$ such that $f(x_{N+1}) \geq f_M$; that is, $f(x)$ is bounded below by $f_M$. This fact, in turn, implies

$$\begin{aligned} 0 \; & < \nu_1 \alpha_M \eta_{Min} \sum_k^N \|\delta_k\|_2^2 \\ & \leq \phi_\rho(x_0) - f_M. \end{aligned}$$

We conclude that the series of norms of the steps generated by our algorithm is summable.

□

It is a trivial (if tedious) matter to substitute these lemmas and assumptions into Han's proofs [81] and thus construct proofs of the following convergence theorems.

> **Theorem 2.1** *If $\delta_k \neq 0$ for all $k \geq 0$, then any cluster point of the sequence $\{(x_k, \lambda_k, \mu_k)\}$ is a K.K.T. point of (NLP).*

> **Theorem 2.2** *If $\delta_{k_*} = 0$ for some positive $k_*$, then the three-tuple $(x_{k_*}, \lambda_{k_*}, \mu_{k_*})$ is a K.K.T. point of (NLP).*

We remark that our convergence proof (like that of Han [81]) guarantees more than $\liminf$ convergence.

## 2.7 Interior point method for LP and RQP

Interior-point methods offer elegant and rapid schemes for the computation of solutions of linear and quadratic programming problems. Developing interior-point method algorithms for the solution of quadratic programming problems is the subject of numerous recent research papers. There are various approaches. See Vanderbei and Carpenter [144], Monteiro [104] and El-Bakry, Tsuchyia, Tapia, and Zhang [53] for primal-dual interior point methods, Anstreicher, den–Hertog, and Ross [2] and

Anstreicher [1] for logarithmic barrier methods, Coleman and Liu [31] for interior-point Newton methods. Each of these approaches has its successes and invites further research.

Recently, Boggs, Domich, Rogers, and Witzgall ([9] and [10]) have developed, analyzed, and tested numerically an interior point method for the solution of large-scale linear and quadratic programming problems. The method is based on ideas presented originally by P. Huard [89] (see also the seminal paper by Renegar [121]) and has been shown to perform well numerically when applied to the solution of linear programming problems and quadratic programming problems. In particular, this method has been used successfully to solve the quadratic programming problems arising in SQP algorithms (see the papers by Boggs, Tolle, and Kearsley [15] and [14]). In this section we briefly describe this algorithm and the reason for an inequality constrained formulation of the quadratic programming problem (i.e., treating equalities as two inequalities).

To avoid cumbersome notation, we now let $\nabla f = \nabla f(x_k)$, $B = B_k$ and let

$$a = \begin{pmatrix} g(x_k) - \tau_k g_{\oplus}(x_k) \\ h(x_k) - \tau_k h_{\oplus}(x_k) \\ -h(x_k) - \tau_k h_{\oplus}(x_k) \end{pmatrix} \quad \text{and } A = \begin{bmatrix} \nabla g(x_k) \\ \nabla h(x_k) \\ -\nabla h(x_k) \end{bmatrix}.$$

For the sake of simplicity, in this section we will no longer indicate dependence on $x_k$ in our notation; that will always be obvious from context. In this simpler notation, our subproblem is written

$$(QP) \begin{cases} \min\limits_{\delta} & \nabla f^{\mathsf{t}} \delta + \dfrac{1}{2}\delta^{\mathsf{t}} B \delta \\ \quad \text{s.t.} & A\delta + a \le 0 \end{cases} \tag{2.44}$$

where $\delta \in \Re^n$ and the constraint matrix $A \in \Re^{m \times n}$ (thus, $m_i + 2m_e = m$).

We now state the algorithm, $O3D$, presented in [10]. We will employ $O3D$ for the solution of our quadratic subproblems;

### Interior Point Quadratic Program Solver (O3D)

1. *Given feasible* $\delta_0$
2. *Generate* $I$ *linearly independent search directions,* $\gamma_j$ *for* $j = 1 \ldots I$
3. *Solve the small quadratic problem letting* $\hat{\delta} = \delta + \sum_{i=1}^{I} \xi_i \gamma_i$

$$\begin{aligned} \min_{\xi} \quad & \nabla f^{\mathsf{t}} \hat{\delta} + \tfrac{1}{2}\hat{\delta}^{\mathsf{t}} B \hat{\delta} \\ & A\hat{\delta} + a \le 0 \end{aligned} \tag{2.45}$$

getting optimal values for $\{\xi_j\}_1^I$.

4. *Set* $\delta^* = \delta + \zeta \sum_{i=j}^{I} (\xi_j \gamma_j)$

5. Go to 2

We emphasize that, in step 1, there is absolutely no work required to generating feasible starting points for either (2.22) or (2.23). We have described the interior point algorithm only as used in the solution of a quadratic program; the algorithm for a linear program is obvious. (When the solution to a linear program (LP) is desired one simply replaces the small quadratic program in step 3 with an appropriate small linear program.) The parameter $\zeta$ in step 4 is taken to be slightly less than one (e.g., $\zeta = .99$). This parameter forces quadratic programming problem iterates to remain strictly feasible with respect to all of the linear constraints.

Various methods for the calculation of the descent directions, $\gamma_i$, have been proposed and tested. The currently preferred algorithm is chosen for its good numerical performance and uses $I = 3$ (hence the name of the algorithm, $O3D$). It depends on the following. At each early iteration, the three directions, $\gamma_i$ are (1) the *dual affine* direction, (2) a *3rd order correction to the dual affine direction*, and (3) a *Newton recentering* direction. A point of switch-over is calculated, based on monitoring the duality gap of the QP or LP. For all iterations after the switch-over, the algorithm uses (1) the *dual affine*, (2) a *3rd order correction*, and (3) a *rank-one correction* direction. Among these two sets of three directions, there are only four distinct directions. Their description is now necessary.

We discuss briefly the current choice implemented for the four distinct and linearly independent search directions. For a more detailed discussion and motivation of these specific search directions the interested reader is refered to the papers by Boggs, Domich, Rogers, and Witzgal ([8]).

Let the residual function associated with the objective function be denoted by

$$r_0(\delta, t) = t - (\nabla f^{\mathsf{t}} \delta + \frac{1}{2} \delta^{\mathsf{t}} B \delta) \tag{2.46}$$

and the residual function associated with the $j$th linear constraint (where $j = 1 \dots m$) be denoted by

$$r_j(\delta, t) = t - (\nabla f^{\mathsf{t}} \delta + \frac{1}{2} \delta^{\mathsf{t}} B \delta). \tag{2.47}$$

We wish to characterize feasible points for which the objective function is less than the quantity $\bar{t}$, where

$$\bar{t} \equiv \nabla f^{\mathsf{t}} \bar{\delta} + \frac{1}{2} \bar{\delta}^{\mathsf{t}} B \bar{\delta}.$$

From the solutions of the re-centering problems for these points, we can write the center of the corresponding feasible region

$$\min_{\delta}\left(-\prod_{j=0}^{m} r_j(\delta,\bar{t})\right) \tag{2.48}$$

or equivalently

$$\min_{\delta}\left(-\sum_{j=0}^{m} \log r_j(\delta,\bar{t})\right). \tag{2.49}$$

Let the diagonal matrix $D$ be defined by

$$D \equiv [\ \mathrm{diag}\ \{r_j(\delta,t)\}]^{-1}$$

and let $e$ be a vector with all its components unity (of a size appropriate to its use). With these quantities, we can define the four step directions above. The first direction, (dual affine), is given by the formula,

$$\gamma^1 \equiv -\left[A^{\mathsf{t}}D^2A + \frac{1}{r_0(\delta,t)}B\right]^{-1}(\nabla f + B\delta).$$

The second direction, (Newton re-centering), is

$$\gamma^2 \equiv -\left[A^{\mathsf{t}}D^2A + \frac{1}{r_0(\delta,t)}B\right]^{-1}(A^{\mathsf{t}}De).$$

The third direction, (rank-one correction), is

$$\gamma^3 \equiv -\left[A^{\mathsf{t}}D^2A + \frac{1}{r_0(\delta,t)}B\right]^{-1}(A^{\mathsf{t}})_{i^*}.$$

Finally, the fourth direction, (third-order correction), is

$$\gamma^4 \equiv -\left[A^{\mathsf{t}}D^2A + \frac{1}{r_0(\delta,t)}B\right]^{-1}\sum_{i=1}^{m}\left(\frac{\left((A)_i^{\mathsf{t}}\gamma^1\right)^2}{r_i(\delta,t)^3}\right).$$

The index $i^*$ that appears in the definition of $\gamma^3$ denotes the column of the matrix of linear constraints associated with the first constraint encountered.

We make the following interesting observation. Upon examining each of the steps of $O3D$, it becomes clear that the factorization needed for calculating will be sparse if the matrix $A$ is sparse. It is not at all clear that this is the case for the calculation of steps of other interior point methods mentioned earlier. We conclude (substantiated by numerical testing) that this algorithm does indeed harness the sparse linear algebra machinery that has been developed recently (see for example Duff, Erisman, and Reid [47] and the numerical software guide, Yale Sparse Matrix Package (SMPAK)). For a complete discussion of the computation of the steps and for a report on the numerical performance of *O3D* on large-scale problems, the interested reader is referred to the papers [9], [10], and [46].

We mention in passing that the calculation of the steps as described above requires an additional assumption not mentioned in (2.6), viz.,

> Assumption 6: The matrix of constraint gradients has full column rank.

However, alternative hybrid factorizations based on the modified Choleski factorizations are described in Gill, Murray, and Wright [64] which have been incorporated into *O3D*, releasing the need for assumption 6. Again, we refer the interested reader to the papers cited above.

For the conclusion of this section, we explain why we use a purely inequality-constrained formulation of the quadratic programming problem. Consider a quadratic programming problem formulated with both equality and inequality constraints. We partition our variables, vectors and matrices into parts according to their association with constraints and according to their boundedness. Let the subscript $(\cdot)_f$ and $(\cdot)_b$ denote free and bound quantities, respectively. Moreover, let $(\cdot)_g$ and $(\cdot)_h$ denote quantities associated with inequality constraints and equality constraints, respectively. Then we define the following partitions of constraint quantities,

$$a = \begin{pmatrix} a_b \\ a_f \end{pmatrix} \quad \text{and } A = \begin{bmatrix} A_{bh} & A_{fh} \\ A_{bg} & A_{fg} \end{bmatrix},$$

of objective function quantities,

$$\nabla f = \begin{pmatrix} \nabla f_b \\ \nabla f_f \end{pmatrix} \quad \text{and } B = \begin{bmatrix} B_{bb} & B_{bf} \\ B_{bf} & B_{ff} \end{bmatrix},$$

and of quantities associated with primal and dual variables,

$$\delta = \begin{pmatrix} \delta_b \\ \delta_f \end{pmatrix} \quad \text{and } z = \begin{bmatrix} z_h \\ z_g \end{bmatrix}.$$

The variable $z$ denotes the Lagrange multiplier. We state the partitioned primal quadratic programming problem

$$\text{(primal)} \left\{ \begin{array}{l} \min\limits_{\delta} \nabla f^{\mathrm{t}}\delta + \frac{1}{2}\delta^{\mathrm{t}} B\delta \\ \quad \text{subject to:} \\ A_{bh}\delta_b + A_{fh}\delta_f + a_h = 0 \\ A_{bg}\delta_b + A_{fg}\delta_f + a_g \le 0 \\ \delta_b \ge 0, \end{array} \right. \tag{2.50}$$

and the dual quadratic programming problem

$$\text{(dual)} \begin{cases} \min\limits_{z} a^{\mathsf{t}} z + \dfrac{1}{2}\delta^{\mathsf{t}} B\delta \\ \quad \text{subject to:} \\ \nabla f_f - A_{fh}^{\mathsf{t}} z_h - A_{fg}^{\mathsf{t}} z_g + [B_{ff} \quad B_{ff}]\,\delta = 0 \\ \nabla f_b - A_{bh}^{\mathsf{t}} z_h - A_{bg}^{\mathsf{t}} z_g + [B_{fb} \quad B_{fb}]\,\delta \leq 0 \\ z_g \geq 0 \end{cases} \tag{2.51}$$

We will now form a partitioned matrix which plays a crucial role in the quadratic programming problem (Axelsson [5]). The K.K.T conditions for problems (2.50) and (2.51) are extremely messy expressions which we will not bother to exhibit here. When these K.K.T. conditions are condensed by forming a Schur complement, the following partitioned matrix results:

$$\begin{bmatrix} L & A_h^{\mathsf{t}} \\ A_h & 0 \end{bmatrix}.$$

To solve the quadratic programming problem, this matrix must be factored or else the action of its inverse must be approximated (see for example the iterative methods discussed in the book by Axelsson [5]). Examination of the matrix reveals some interesting aspects of the computational cost of solving our equality constrained formulation. If there are any equality constraints at all, the solution of a potentially very dense matrix equation including $A_h L^{-1} A_h$ will be necessary. If there are many equality constraints, the cost of factoring this matrix will be prohibitive. Furthermore, if the number of equality constraints is larger than the number of variables, $A_h L^{-1} A_h$ may be a rank deficient system adding considerably to the difficulties of solution. Such considerations are compelling arguments for formulating the quadratic programming problem solely with inequality constraints.

## 2.8 Implementation of the algorithm

Empirical enhancements of an algorithm can effect remarkable improvement in numerical performance. It often happens that such improvements are even necessary in order to generate a practical, working code from a model theoretical algorithm. In this section, we describe some modifications to the algorithm proposed in section 5, modifications needed to make it practical for large-scale computations. Tests of the enhanced algorithm are reported in section 9 of this chapter.

Algorithms for updating the penalty parameter in SQP methods are the subject of numerous papers (see for instance the work by Tapia [135], El Alem [52], and Boggs, Tolle, and Kearsley [14]). The observation is made in [63] that, if the penalty parameter is increased too rapidly and subsequently never increased, the algorithm will tend to produce iterates that follow the feasible region too closely. This noxious behavior is particularly worrisome in the presence of highly nonlinear constraint manifolds when following the manifold could result in unreasonably small steps. Experience has demonstrated that, in this situation, a dynamic choice of the penalty parameter is essential for good performance.

A heuristic algorithm for varying the penalty parameter is used in obtaining numerical results, even though it is not considered in our convergence theorem. The updating scheme is not sophisticated. The angle between the gradient of the merit function $(\nabla\phi)^{\mathsf{t}}$ and the solution to the QP step direction $\delta^{\mathsf{t}}$ is monitored. When this angle becomes too large, we conclude that $\rho$ is not large enough to ensure sufficient decrease in our merit function. We calculate

$$w(\rho) = \frac{(\nabla\phi_\rho)^{\mathsf{t}}\delta}{\|\nabla\phi_\rho\| \cdot \|\delta\|}$$

and, using this information, we increase $\rho$ attempting to find a $\rho_{new}$ such that $w(\rho_{new}) \approx -.5$. If $\rho$ is *not* decreased by this procedure, we consider a different modification. To do so, we compute an interval of values for the penalty. For some fixed integer $I_c$, we desire a value of $\rho$ such that

$$\phi_\rho(x_k) < \phi_\rho(x_{k-i}) \quad \text{where} \quad i = 1, 2, \ldots I_c. \tag{2.52}$$

where the subscript, $k$, denotes iteration number. This additional mechanism is incorporated into the algorithm in order to extinguish any cycling behavior (see for instance [26]). In addition to enforcing this inequality, we insist on a heuristic that bounds the absolute and relative changes of the penalty parameter. Finally, we require that the predicted reduction in $\phi$ be greater than the predicted reduction in the linearized constraints. This prohibits a change in $\rho$ that is based solely on large decreases in linearized constraint infeasibilities.

We used a limited-memory BFGS (LM-BFGS) Hessian approximation to the matrix $B_k$. For an explanation and discussion of limited-memory updating schemes see Liu and Nocedal [101], Nocedal[112], and Byrd, Nocedal, and Schnabel [20]. Even though our global convergence theorem assumed a uniformly bounded family of symmetric, positive-definite, Hessian approximations, we employed a limited-memory version of the symmetric rank-one matrix (LM-SR1) for comparison purposes. In order to carry out our Hessian approximation, we need to store $m$ pairs of vectors for each iterate $x_k$ (say $\{s_i, y_i\}$, where $i = k-1, \ldots k-m$). Our choice for vectors $\{s_k, y_k\}$ is a very standard one;

$$s_k = x_{k+1} - x_k \quad \text{and} \quad y_k = \nabla\ell^I_\rho(x_{k+1}, \lambda_{k+1}, \mu_{k+1}) - \nabla\ell^I_\rho(x_k, \lambda_k, \mu_k). \tag{2.53}$$

The LM-BFGS update procedure guarantees that the Hessian approximations will remain positive-definite for all iterations $k$, provided the inner product $s_k^{\mathsf{t}} y_k$ remains positive. We enforced the condition that

$$s_k^{\mathsf{t}} y_k > \epsilon_{BFGS} \tag{2.54}$$

using the techniques described in the papers above. With the choices for the $s$ and $y$ vectors appearing in (2.53) all details of the construction of the Hessian approximation were taken directly from the papers mentioned above.

The trust-region radius $\Delta$ was never active for any of the test problems whenever a positive-definite Hessian approximation was used. We will describe the method used to monitor the trust-region radius $\Delta$, even though it was rarely active. For any merit function taking $\Re^n$ into $\Re$ we define a predicted reduction $\wp(\cdot)$ and actual reduction $\aleph(\cdot)$.

$$\wp(\phi_\rho(x_k)) = \nabla\phi_\rho(x_k)^{\mathsf{t}}\delta_k \tag{2.55}$$

and

$$\aleph(\phi_\rho(x_k)) = \phi_\rho(x_k) - \phi_\rho(x_k + \delta_k) \tag{2.56}$$

We will judge how accurately our model fits the nonlinear model by comparing these two reductions when applied to our merit function, $\phi_\rho(x)$. Actually, as shown in Figure 2.3, we evaluate (2.10) only heuristically in that, if the descent direction, $\delta_k$, violates our trust-region constraint, we simply scale it to bring it into compliance.

To conclude this section, we suggest an alternative merit function for comparison, a merit function for equality and inequality constrained nonlinear programming problems from Boggs, Tolle, and Kearsley [13] and from the associated paper on implementation [14]. The globalization procedure suggested in that work is quite complicated. We will not give complete details here, but will mention some interesting points. The merit function used in the squared slack-variable problem is based on the augmented Lagrangian function described in section 2.2. That merit function, $\psi$, is defined by

$$\psi_d(x,z) = f(x) + \bar{\lambda}(x,z)^{\mathsf{t}}c(x,z) + \frac{1}{d}\|c(x,z)\|^2_{W(x,z)}. \tag{2.57}$$

The weighting matrix $W$ was

$$W(x,z) = \left[\nabla g(x)^{\mathsf{t}}\nabla g(x) + \text{ diag } \{z\}\right]^{-1} \tag{2.58}$$

Least-squares estimates of the Lagrange multipliers are employed;

$$\bar{\lambda}(x,z) = -\left[W(x,z)\right]^{-1}\nabla g(x)^{\mathsf{t}}\nabla f. \tag{2.59}$$

A related approximate merit function was used in a *watch-dog* procedure so that derivatives were not evaluated at every trial point. This scheme is not unlike the first watch-dog procedure described in Chamberlain, Lemarechal, Pedersen, and Powell [27]). The globalization procedure described in [14] and analyzed in [16] and [17]

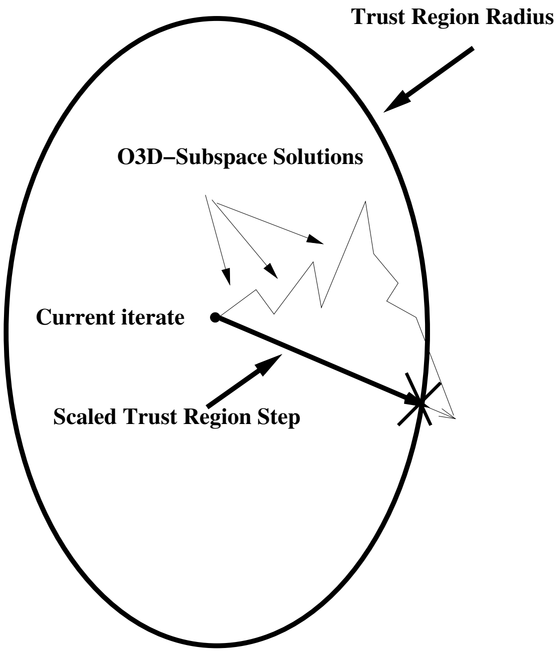


**Figure 2.3** Sketch of proposed trust-region algorithm

fits naturally into our algorithm (but the theoretical justification in section 2.6 does not apply). We include it only among the numerical results, in order to have an alternative globalization procedure for comparison to Han's merit function.

Attempts to analyze a neat algorithm after it is sullied by empirical enhancement usually fall short of satisfactory. As an alternative, we present some numerical results from the algorithm proposed in section 2.4 for comparison with those of the modified algorithm.

## 2.9 Numerical results and conclusions

The algorithm and its variants described in this work were implemented in Fortran 77 programs and tested on a selection of applied test problems. These problem are summarized in the appendix. All numerical results presented here were obtained on a SPARCstation20 using *IEEE floating point arithmetic (64 bit)*.

The suite of test problems is not large. The perturbation being tested is intended for solving problems with both inequality and equality–constraints that give rise to inconsistent linearized constraints. Public-domain test problems are of little interest here because they usually lead to consistent subproblems. Only a few of the problems from the Hock and Schitkowski test–set lead to inconsistency, but only very rarely, when extremely poor initial guesses were used [85]). Furthermore, we are more interested in problems that are mid- to large-scale.

The first test problem ([14]) arises in the post-processing of Nuclear Magnetic Resonance (NMR) data. The second, third, and fourth problems are from optimal control. Their numerical solution by the methodology presented here is the subject of chapters 4, 5 and 6. Related work on similiar problems can be found in Kelley and Sachs [93], Kelley and Wright [94], and Wright [150] among numerous other papers. The final test problem was fabricated to guarantee a test of inconsistent linearizations when iterates are far from the solution.

The stopping criteria used are standard and not overly stringent. The first stopping criterion requires that the gradient of the Lagrangian be sufficiently small; the second, that the size of the step be sufficiently small. Both of these criteria were augmented with constraints to preclude satisfying convergence tests at obviously infeasible points. Specifically, we required

$$\frac{\|\nabla f(x_k) + \nabla g(x_k)\lambda_k + \nabla h(x_k)(\sigma_k - \theta_k)\|}{|1 + f(x^k)|} + V(x_k) \leq \epsilon_1 \tag{2.60}$$

or

$$\|D\delta_k\| + V(x_k) \leq \epsilon_2. \tag{2.61}$$

Values of the important parameters of the algorithm are given in the table below. With reasonable choices for its parameters, the algorithm appears to behave extremely well.

| | |
|---|---|
| number of LM updates | 4 |
| gradient tolerance - $\epsilon_1$ | $\epsilon_M^{1/3}$ |
| step tolerance - $\epsilon_2$ | $\epsilon_M^{2/3}$ |
| value of $M_\tau$ | $1.0E6$ |
| value of $\epsilon_\tau$ | $1.0E-3$ |
| value of $\epsilon_{BFGS}$ | $1.0E-6$ |
| maximum number of iterations | 500 |

**Table 2.1** Minimization parameters

The names and the sizes of the test problems are recorded in the first two columns of each table. The third and fourth columns give the number of nonlinear and linear iterations required to solve the problem. The fifth column records the number of iterations where a non-zero relaxation parameter $\tau_k$ was encountered. The seventh and eighth columns give the values of the first and second convergence tests (i.e. the values given by the left-hand side of (2.60) and (2.61), respectively).

Some less obvious positive aspects of this algorithm are the following:

- The perturbation procedure is inactive for any iterations when the linearized constraints are consistent ($\tau_k = 0$).

- The perturbation procedure normally becomes inactive when iterates approach the solution. Inconsistency is usually encountered only far from the solution.

- The same subproblem solution technique can usually be used for solving (2.22) and (2.9). There is then only a one-time per iteration linear-algebra set up cost. The linear constraints in (2.22) and (2.9) are similar.

- There are no Phase-I procedures needed in any part of the algorithm.

As always, there are some undesirable aspects of the algorithm. The following points should be mentioned:

- Attempts to employ a symmetric, indefinite, limited-memory Hessian approximation (specifically the (LM-SR1)) led to a deterioration in numerical performance. It was also seen on a few problems that the trust region constraint was active very close to the solution.

- The theory of convergence is not quite satisfying (especially Assumption 5).

- Local convergence has not yet been examined. A more sophisticated merit fuction might well give better local convergence.

All the work suggested here was based on the availability of matrix factorizations. For solving the largest problems of interest (e.g., hierarchical control of Navier–Stokes in three space dimensions), iterative linear solvers must be employed for all matrix equations. The effect of these alternative iterative solvers on our perturbation technique is actively under investigation.

In summary, an SQP algorithm for the solution of a special class of nonlinear programming problems (NLP) has been proposed, analyzed, implemented, and demonstrated. Some examples of the special class (problems with many nonlinear constraints leading to inconsistent linearizations) have been considered. The formulations of the constrained distance problem and the hierarchical control problem examined here, frequently arise as members of this class. The numerical results presented support the utility of this algorithm for solving problems in the special class.

| Problem | NL-iters | QP-iters | $\tau_k \neq 0$ | $\epsilon_1(x_*)$ | $\epsilon_2(x_*)$ |
|---|---|---|---|---|---|
| MDP | 67 | 1267 | 12 | 9.E-6 | 2.E-4 |
| PCB | 25 | 478 | 2 | 7.E-7 | 8.E-5 |
| NCH | 81 | 1493 | 13 | 8.E-7 | 2.E-4 |
| OTD | 49 | 991 | 6 | 5.E-7 | 1.E-4 |
| RGP | 121 | 2121 | 35 | 7.E-7 | 1.E-4 |

**Table 2.2** Numerical results from perturbed SQP algorithm using $\phi$ merit function with LM-BFGS

| Problem | NL-iters | QP-iters | $\tau_k \neq 0$ | $\epsilon_1(x_*)$ | $\epsilon_2(x_*)$ |
|---|---|---|---|---|---|
| MDP | 55 | 1044 | 10 | 3.E-7 | 3.E-5 |
| PCB | 21 | 400 | 3 | 6.E-8 | 4.E-5 |
| NCH | 74 | 1339 | 12 | 9.E-9 | 9.E-7 |
| OTD | 47 | 889 | 7 | 1.E-8 | 8.E-6 |
| RGP | 96 | 1773 | 41 | 2.E-7 | 3.E-4 |

**Table 2.3** Numerical results from perturbed SQP algorithm using $\psi$ merit function with LM-BFGS

| Problem | NL-iters | QP-iters | $\tau_k \neq 0$ | $\epsilon_1(x_*)$ | $\epsilon_2(x_*)$ |
|---|---|---|---|---|---|
| MDP | 89 | 2423 | 18 | 5.E-7 | 5.E-4 |
| PCB | 178 | 5389 | 13 | 4.E-3 | 9.E-9 |
| NCH | 109 | 4218 | 21 | 3.E-5 | 1.E-8 |
| OTD | 51 | 1491 | 9 | 6.E-7 | 4.E-5 |
| RGP | 500 | 19184 | 500 | nc | nc |

**Table 2.4** Numerical results from perturbed SQP algorithm using $\phi$ merit function with LM-SR1

| Problem | NL-iters | QP-iters | $\tau_k \neq 0$ | $\epsilon_1(x_*)$ | $\epsilon_2(x_*)$ |
|---|---|---|---|---|---|
| MDP | 83 | 3108 | 17 | 2.E-7 | 9.E-4 |
| PCB | 142 | 4543 | 15 | 8.E-7 | 2.E-5 |
| NCH | 78 | 2872 | 14 | 7.E-7 | 6.E-5 |
| OTD | 50 | 1597 | 8 | 3.E-8 | 5.E-4 |
| RGP | 203 | 5949 | 40 | 1.E-6 | 8.E-9 |

**Table 2.5** Numerical results from perturbed SQP algorithm using $\psi$ merit function with LM-SR1

# Chapter 3

# A Derivative–Free Algorithm for Small Scale Nonlinear Programming Problems

## 3.1 Introduction

There is a great need to solve nonlinear programming problems in many, varied applications of computation. When all the functions involved are smooth to the degree that their derivatives can be computed or approximated, there are many excellent algorithms to employ (see Gill, Murray, and Wright [64]). These algorithms generally use the derivatives, sometimes second derivatives, in locating minima of the objective function. In many applications, such algorithms are useless because the needed derivatives are not accessible. It may be that their computation is simply not possible. Even when the derivatives are computable in principle, many times the noisy or ill-conditioned data make them useless (see, as examples, some statistical applications in the books by Tapia and Thompson [138] and [137]). In other applications, the cost of computing a derivative can be just too expensive (for an example from geophysics, see Symes [133]). There are numberless important problems for which derivatives simply do not exist (examples include control of dynamical systems with *Coulomb* friction [66], minimization of condition numbers in linear algebra [84], and others). Recently, there has been a great deal of interest in the non–differentiable optimization of shapes parameterized by a small number of control variables (see Glowinski, Kearsley, Pan, and Periaux [67] and [68]). With an eye towards important applications of these types, the aim in this chapter is to implement and test a robust parallel algorithm to be used when derivatives are unavailable, unreliable, or when they simply do not exist.

We propose a parallel algorithm that requires no derivative information to solve nonlinear programming problems . The goal of the algorithm is to find the vector $x^*$ minimizing an objective function $f(x)$ subject to a set of constraints which include some mix of equality constraints $h(x) = 0$ and inequality constraints $g(x) \leq 0$. No attempt is made to approximate derivatives; the algorithm uses only the values at trial points of the objective function and the constraint functions.

The nonlinear programming problem (NLP) we consider takes the form

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ s.t. & h(x) = 0 \\ & g(x) \leq 0 \end{array} \tag{3.1}$$

where $f : \Re^n \rightarrow \Re$, $h : \Re^n \rightarrow \Re^{m_e}$, and $g : \Re^n \rightarrow \Re^{m_i}$ are assumed to be continuous nonlinear functions. Here, we have $n$ variables, $m_e$ equality constraints, and $m_i$ inequality constraints.

Most algorithms designed for the solution of optimization problems, either unconstrained or constrained, depend heavily on the availability of derivatives. For example, the conjugate gradient algorithm and most algorithms based on Newton's method require the approximation of the first derivatives of a function, at the very least. A direct-search method is an algorithm for minimizing a function that relies solely on function values. Many modern and popular algorithms for global optimization fall into this category of algorithms with no dependence on derivatives. They include, for example, *simulated annealing* and *genetic algorithms*.

Recently, a parallel, direct-search method (PDS) for solving unconstrained problems was analyzed and implemented by Dennis and Torczon, [45]. Direct-search methods (sometimes called methods of moving polytopes, see for instance Hamma [80]) are far from new (for some early examples of pattern-search methods see Box [18] and Hooke and Jeeves [88]). Until recently, there was no good theory for their convergence. Lately, convergence theory for particular direct-search schemes has been published (see Torczon [143]) and numerical testing has suggested that direct-search methods, when implemented on parallel machines, can yield striking speed-up. There have been successful applications of the algorithm of Dennis and Torczon to the solution of many, varied, unconstrained problems; see N. Higham [84] for an example from linear algebra. Successes, such as the one mentioned, motivate us to investigate an implementation of parallel, direct-search methods tailored for the solution of constrained problems, specifically (NLP).

The method we employ for dealing with constraints is well known; constraints will be enforced via penalties. In this way, we transform the constrained problem into an unconstrained one. For many standard algorithms, these transformations have been thoroughly investigated by various researchers (see for example Fiacco and McCormick [57]). Our penalty function technique is nonstandard, however, and special provisions must be made to

- preserve the computational advantages of the parallel implementation,
- avoid premature numerical degeneracy of the simplex as infeasible iterates approach a feasible point, and
- minimize chattering or cycling of feasible/infeasible iterates when near the constraint set.

With an eye toward applications posing non-differentiable problems, (such as [66], [67] and [68]), our method makes no attempt to use any sophisticated technique requiring smoothness, to construct interpolations (see the work of Powell [119]), or even to approximate derivatives. The contributions in this chapter apply to constrained optimization. We note that neither solving without using derivatives (see Zangwill [152] and Fletcher [58]) nor adding constraints to direct-search methods (see Subrahmanyam [131] and Swann [132]) is a novel idea for unconstrained problems.

The organization of the remaining sections of this chapter is as follows. The next section is a brief discussion of the construction and evolution of the simplex used by the underlying direct-search method. The third section contains a formulation and explanation of the special penalty function employed by our algorithm. The fourth section reports the numerical results obtained on applying the algorithm to solve some test problems briefly described in the appendix. Our conclusions, observations and ideas for future research are presented in section 5. The short appendix describes the set of test problems which were assembled from various sources.

## 3.2 The direct-search method

Many different simplex shapes and associated algorithms have been proposed in the optimization literature. Those of Box [18], Hooke and Jeeves [88], and Nelder and Mead [109] are robust minimization algorithms. They are among the most frequently employed by practitioners of applied optimization. Recently, a particular family of direct search methods for unconstrained optimization was proposed by Dennis and Torczon [45] and a parallel implementation of these algorithms was developed and tested (see Torczon [142]). The performance of these algorithms on parallel machines was promising, prompting the development of a first-order stationary-point convergence theory (see Torczon [143]). We briefly describe this particular pattern-search method here; for an in depth discussion of the details the interested reader is referred to Torczon's thesis [140].

Given a 'seed' vector, say $x_0 \in \Re^n$, our simplex can be described with $x_0$ and $p$ additional vectors in $\Re^n$. The initial simplex, say $\Omega_O$, is chosen from one of three different shapes and evolves through a search along combinations of three distinct modifications made to this initial simplex. A good choice of initial simplex shape (right angle, regular or scaled) is usually made by correctly predicting the scale of the problem to be solved (see Figure 3.1). Experience suggests that the regular or right angle simplex is a best initial guess for problems with moderate nonlinearity and that the scaled simplex works best on problems where the nonlinearity in the objective function or constraints is poorly scaled. In doubtful cases, the right angle simplex appears to the best default choice.

When the 'starting simplex shape' is chosen, say $\{x_0, x_1, x_2, ..x_n\}$, subsequent nodes or vertices are added to the simplex by searching along expanding, contracting or reflecting search directions (see Figure 3.2). The nodes $x_1, x_2 \ldots x_n$ are reflected about $x_0$ along the lines that contain $(x_0, x_1), (x_0, x_2) \ldots (x_0, x_n)$, respectively. These points are indicated in Figure 3.2 as $x_1^R, x_2^R$. If a 'good' point is found among these

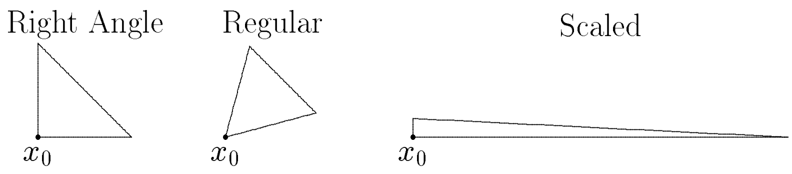


**Figure 3.1** Initial Simplex Shapes when $n = 2$

reflection-step points, then the expansion steps are constructed. This is done by increasing the distances between $x_0$ and $x_i^R$ for $i = 1, \dots n$ (usually the distance is doubled). This creates $n$ new points, indicated in Figure 3.2 as $x_1^E, x_2^E$. The simplex $\{x_0, x_1, x_2, ..x_n\}$ is then replaced by either $\{x_0, x_1^E, x_2^E, ..x_n^E\}$ or $\{x_0, x_1^R, x_2^R, ..x_n^R\}$ depending on whether the 'best' point was among the expansion points or the reflection points. In the event that neither the expansion nor reflection searches yield a point better than the points in the simplex $\{x_0, x_1, x_2, ..x_n\}$, then contraction steps are constructed. The distances between $x_0$ and $x_1, x_2, ..x_n$ are diminished (usually the distance is halved). If the best point is located among the points in this contraction simplex, say $\{x_0, x_1^C, x_2^C, ..x_n^C\}$, then this simplex replaces the original simplex (indicated by $\{x_0, x_1^C, x_2^C\}$ in Figure 3.2). The process described above constitutes one iteration (one outer-loop iteration) of our simplex generating algorithm.

The algorithm terminates when either of two stopping conditions is met: the maximum number of simplex iterations have been performed (maximum number of outer iterations) or a simplex is generated with the relative distance between any two distinct points less than a user-specified tolerance. Specifically, the second stopping criterion is

$$\text{tolerance} > \frac{\max\limits_{0 \le i < j \le n} \|x_i - x_j\|_2}{\max(1, \|x_0\|_2)}. \tag{3.2}$$

This pattern does not degenerate as iterates approach the solution (unlike the popular *Nelder–Mead* simplex [109]). For a discussion of issues involving the con-

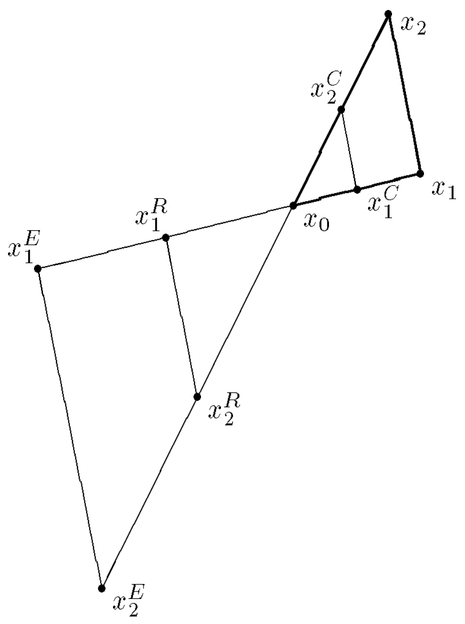


**Figure 3.2** An illustration for $n = 2$ of all three modifications (reflection, contraction and expansion) to the simplex $\Omega$ defined by the points $x_0, x_1, x_2$.

struction of this particular pattern and for more details, the reader is referred to [45] and [140].

## 3.3 The algorithm

The notion of transforming constrained problems into unconstrained problems has been employed successfully in a variety of ways and for many different applications. Usually, the success or failure of the transformation is dependent on compatibility of (1) the transformation and (2) the elimination of the constraints, with the method used to solve the resulting unconstrained problem. (See for example [57] or [64] for a survey of such issues.) In this section we propose a penalty function that currently appears 'most compatible' and robust when (PDS) is the underlying unconstrained solution method.

If the volume of the simplex becomes too small, one sees a large increase in the number of function evaluations needed to locate a new point that is closer to optimality. While a decrease in the volume of the simplex is needed when iterates are close to the solution, if this occurs far from the solution, the result can be a significantly longer run time or even a premature halting of the algorithm. In order to keep the average volume of the simplex as large as possible (while not making the penalty function too complicated), a relaxation term is added to a simple penalty function.

We now describe our suggested modification of the algorithm of Dennis and Torczon in order to incorporate the constraints in (3.1). Given a penalty constant $\rho$ plus a point (or a 'seed point') $x^\sharp \in \Re^n$ about which a pattern is constructed, we define the penalty function at a point $x \in \Re^n$:

$$\Pi_L(x; x^\sharp, \rho) = f(x) + \rho(G_L(x)) + Z_L(x, x^\sharp). \tag{3.3}$$

Here, the point $x_k^\sharp \in \Re^n$ is our previous estimate of the solution, the 'best' among points on the previous simplex. The function $G_L(x)$ is some norm of the absolute constraint violations. The norm varies dynamically, based on the distance of the current iterate from feasibility.

$$G_L(x) = \|G(x)\|_L \tag{3.4}$$

where

$$G(x) = (|h_1(x)|, ...|h_{n_e}(x)|, \max\{0, g_1(x)\} ... \max\{0, g_{n_i}(x)\})^{\mathsf{t}} \tag{3.5}$$

is the vector of constraint violations. The function $Z_L(x, x^\sharp)$ is identically zero if the objective function has increased (i.e. if $f(x) \leq f(x^\sharp)$). Otherwise (i.e. if $f(x) > f(x^\sharp)$), $Z_L$ increases the penalty function in proportion to any decrease in constraint violations. Specifically, if $\epsilon_M$ is the machine dependent zero, then

$$Z_L(x, x^\sharp) = \min\left\{\frac{G_L(x^\sharp)}{\max\{G_L(x), \sqrt{\epsilon_M}\}}, 1\right\} \max\left\{f(x^\sharp) - f(x), 0\right\}. \tag{3.6}$$

This function has the effect of forcing the pattern to 'tumble' towards the nearest region of the constraint manifold, regardless of whether the objective function increased or decreased. It has the effect of producing iterates that tend to follow the constraint manifold as they move towards a solution. Moreover, the penalty function does not decrease so quickly as to cause the vertices of the newly generated patterns to contract upon themselves. If such a contraction of the pattern occurs far from the solution, it can dramatically lengthen the time needed for the algorithm to converge. (This was the case when more standard penalty functions were employed.) This penalty function automatically coordinates progress towards the solution through two phases. The first phase gets the points of the pattern close to feasibility; the second drives the objective function to a smaller value. Adding the relaxation function, $Z_L$, to the penalty function has the effect of making the penalty function decrease less drastically from iteration to iteration, but more rapidly from starting point to solution (see

Figure 3.3). The relaxation tends to postpone the contraction steps until nodes of the simplex enter a neighborhood of the solution.

Though the idea of dynamically changing the norm in the penalty term might seem naive, it has proved effective. The active norm is determined by the distance of $x^\sharp$ from feasibility. Far from the solution, the infinity norm is used to drive down the most violated constraint, regardless of which way less-violated constraints are moved. In subsequent iterations, when the constraints become manageable, the 2-norm is employed. When constraint violations become less than 1, the 1-norm is employed because choosing a point based on decreasing the 2-norm can be too restrictive, causing the volume of the simplex to become excessively small. Numerical trials demonstrate that this is an acceptable strategy.

$$L = \begin{cases} \infty & \text{if} \quad G_2(x^\sharp) \geq 2 \\ 2 & \text{if} \quad 1 < G_2(x^\sharp) < 2 \\ 1 & \text{if} \quad G_2(x^\sharp) \leq 1 \end{cases} \tag{3.7}$$

The parameter $\rho$ can be a constant or it can be updated dynamically. Unlike derivative-based Newton methods or quasi–Newton methods for (NLP), the algorithm depends on $\rho$ only minimally. This is inherent in the fact that the generating of search directions and trial points is independent of any quantities which depend on the penalty parameters. Moreover, a new point is selected from the set of trial points based on the smallest values of the associated penalty function. There are no conditions enforced of minimum decrease or maximum allowable change. Thus, there is no ill-conditioning introduced into the problem by penalty parameters that are too large. There is a price paid for this robustness: there is no rapid local convergence. Numerical experience confirms that changes in $\rho$ have only a slight effect on the convergence behavior of the algorithm (shown in [132] and observed numerically in [66]). For our numerical tests, the penalty parameter was kept constant until the distance between nodes of a simplex dropped beneath the user-specified tolerance at some very infeasible point. Only then was the penalty parameter $\rho$ increased. The updating strategy employed in the numerical tests was

$$\rho^+ \leftarrow \rho^c(1 + \|G(x)\|_\infty) \tag{3.8}$$

This strategy results in slight improvements in numerical behavior when iterates tend to halt too far from feasibility.

A combining of the pattern construction and the specific penalty function from above can be summarized in the following elementary and brief statement of the proposed algorithm.

Algorithm: Constrained - PDS

1. Given $x_0^\sharp$, $\Omega_0$, $\rho_0$

2. Generate simplex : $\Omega_i$ about the point $x_i^\sharp$.
3. If $G_1(x) \approx 0$ and $(\max \|x_j - x_k\|_2) \leq \epsilon$ where $x_j, x_k \in \Omega_i$, then STOP
4. Solve $x_{i+1}^\sharp = \arg\min_{x_j} \{\Pi_L(x_j, \hat{x}_i; \rho) | x_j \in \Omega_i\}$
5. Update $L$ by (3.7) and if desired update $\rho \leftarrow \rho + \rho G(x_{i+1}^\sharp)$
6. $i \leftarrow i + 1$ ; go to step 2

A few comments are necessary. The first step requires an initial guess of the solution, an initial simplex shape, and a penalty parameter. The initial guess need not be close to the solution. Again, the behavior of our algorithm departs from that of the standard Newton or quasi-Newton type; the Euclidean proximity of a starting point to the actual solution in no way suggests that it is a local or non–local starting point, because there is no local rapid convergence. Provided the search scale is large enough, nothing prevents the algorithm from taking tremendously large steps, regardless of the local behavior of the objective and constraint functions in a neighborhood of the starting point. (There is no trust–region, for example, nor dependence on derivatives.) The unconstrained subproblem is solved using any implementation of PDS [141]. A heuristic concerning the penalty parameter which we have found to be numerically effective is the following: if the inter–node distance on a simplex drops below the user-specified tolerance but the constraint violations have not vanished (or decreased sufficiently), step 5 should include an increase of $\rho$. This has the effect of pushing iterates towards feasibility.

This algorithm is extremely easy to implement.The cost per iteration is $O(n)$ function evaluations with a *minimum cost* of $2n$ function evaluation. The value of the penalty function depends only on the function values at the current point, on the minimizer at the previously searched simplex and on the penalty constant. With little pain, each of these quantities can be made available to processors of a distributed-memory parallel machine. Therefore, the algorithm decomposes quite naturally for a parallel environment (with either shared or distributed memory); speed-up can be achieved as more processors become accessible.

## 3.4 Numerical results

It is important to bear in mind that this algorithm is not meant to terminate at an 'exact' solution of a minimization problem. Otherwise, results from the testing of this algorithm can be very misleading. The algorithm is designed to locate an approximate minimizer of a constrained optimization problem. Once this approximate solution is found, many efficient alternatives for improving the estimate of the solution are possible. One could, for example, run the same direct search algorithm repeatedly, each time using the approximate solution from the previous run as the starting value, increasing the number of processors used, decreasing the scale of the search strategy and tightening the convergence tolerance. This procedure will yield successively better solutions. For smooth problems, another possibility is to employ a higher-order local method to polish the solution found by PDS. In what follows, bear in mind that the purpose of the test results presented here is to illustrate how effectively the algorithm can locate an approximate solution. A computation should be considered a success when the value of the objective function agrees with an 'exact' solution with an accuracy of two or more digits while the constraint violations are small.

The numerical tests reviewed here reflect the performance of a conservative version of the algorithm (see Table 3.1). There is no attempt to exploit the structure of any of these simple test problems. For example, no distinction was made between linear and non-linear constraints. Simple bounds were treated as general nonlinear inequalities. The scale of the search strategy was extremely conservative. (It was set at a large number.) It is worth remarking that problems with only simple bounds or linear constraints are a special class of problems which can be treated very efficiently. If an initial feasible point is found for a linearly-constrained problem, then a direct search method can be designed specifically for that problem. Simply stated, one can arrange to construct the simplex with nodes lying solely within the feasible set polytope. Bounded and linearly-constrained problems have been successfully solved using this feasible-point technique. In our tests, however, no effort was made to choose special starting points that were feasible. All our test problems have at least one nonlinear constraint. The convergence tolerance chosen was smaller than needed in general practice. Finally, no effort was made to adjust the algorithm dynamically while solving a problem. That is to say, size, type and scale of search strategy remained unchanged throughout each attempt to solve a problem. The following Table 3.1 gives the values for the stationary algorithm parameters.

| | |
|---|---|
| Search Scale | 4 |
| stopping tolerance | $\epsilon_M (\approx 2.e-16)$ |
| maximum iterations | 500 |
| Search thingy | $2n$ |
| penalty parameter $\rho$ | 1000 |
| $\max \lvert\tau\rvert$ | 10.0 |

**Table 3.1** Numerical Test Parameters

When $L = 1$ in (3.9), the resulting penalty function, $\ell_1$, is sometimes called Han's merit function (among other names). Work that describes using this penalty function as a merit function include [81], [60] and [32]. In Figure 3.3, we compare typical behavior of Han's merit function (applied to problem HOC78 from the appendix) to that of the proposed penalty function, (3.3). It must be emphasized that we are using this merit function in a way that it was not meant to be used. However, it is a natural way to incorporate the constraints appearing in (3.1) into the framework of the direct-search method. Figure 3.3 is representative of the numerical behavior of this penalty function which we observed for nearly all our test problems. The straightforward penalization of constraints led to an initial rapid decrease in the penalty function, but the volume of the simplex became quite small before iterates drew close to the solution. The resulting solution took 15 iterations of PDS. The suggested penalty function (3.3), on the other hand, decreased much more slowly during early iterations but managed to locate the solution after only 6 iterations.

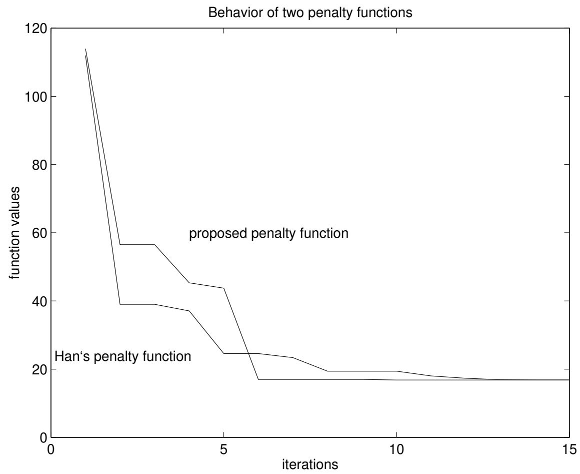


**Figure 3.3** Different "direct search methods"

The test problem suite was chosen by collecting problems from various sources. Some of the problems are from papers, others from books of test problems, and some are applied problems from work described later in this thesis. An appendix is at the end of this chapter, describing the problems and, when needed, giving an appropriate reference.

There are four tables of results, corresponding to four different penalty functions. The first three tables correspond to the penalty functions $\ell_1$, $\ell_2$ and $\ell_\infty$, respectively, where

$$\ell_L(x) = f(x) + \rho G_L(x) \quad \text{with} \quad L = 1, 2, \infty. \tag{3.9}$$

These penalty functions have been used extensively in both smooth and nonsmooth optimization algorithms (see Fletcher [59], Maratos [103], and Boggs and Tolle [11] among numerous other examples).

The second column, labelled 'start', indicates the starting points. Randomly generated starting points are denoted by 'rand'. When the starting points were those suggested in the associated reference, they are marked 'sug'. For a few problems, there was no suggested starting point, so a perturbed solution was employed for comparison. These are denoted by 'pert'. The perturbation was of the magnitude of the solution,

$$(x_0)_i = (x_*)_i + 2(\tau)|(x_*)_i| \tag{3.10}$$

where $\tau$ is a random number taken from a standard (zero mean) normal distribution. We report on numerical results from a $\tau$ chosen to have a variance of 10. We also report on numerical performance from random starting points, denoted by 'rand'.

The following quantities are results for the algorithm when it halted. The first column contains the four-letter label corresponding to the description of the problem from the appendix. The second column indicates whether the starting point was provided in the associated reference, randomly generated, or a perturbation of the solution. The third, fourth, and fifth columns give the number of variables and constraints (we record equality constraints/inequality constraints) respectively. The sixth column records the number of iterations needed to solve the problem. The seventh and eighth columns attempt to measure the quality of the approximate solution. A relative error in optimal function value and the absolute constraint violation are recorded, respectively. If $f^*$ is the optimal value of the objective function, then

$$\text{rel }(f(x^*)) = \frac{|f^* - f(x^*)|}{|f^* + 1|}. \tag{3.11}$$

The seventh column records rel $(f^*)$ and the eighth, $G_2(x^*)$. The ninth column gives the type of pattern used.

A short appendix contains very brief descriptions of the problems along with appropriate references. An effort was made to choose problems from a variety of sources. Two of the problems are from applications. All problems had some degree of nonlinearity.

## 3.5 Conclusions

Many important but non-standard small-scale constrained minimization problems call for algorithms that require no knowledge of derivatives, exact or approximate. The algorithm presented here is specifically designed to run on a particular class of such non-standard problems with a small number of variables. It is not designed to compare or to compete with any algorithm that incorporates proper use of derivatives or curvature information into the choice of a search direction.

This algorithm is a viable solution method for small-scale global optimization problems (see for example Floudas and Pardalos [61], Pardalos and Rosen [113]). The algorithm's lack of dependence on derivatives makes local minimizers less likely to appear as solutions. The method used to select trial points is such that the algorithm is guaranteed to sample 'up–hill' directions, or directions that appear to increase the penalty function. Evidence of this can be seen in the fact that, for all the test problems with multiple minimizers, the algorithm always found the global minimizer or halted in a neighborhood of the global minimizer.

An important observation is that the algorithm succeeded in solving most of the problems in a reasonable number of iterations. Moreover, the relaxation scheme of adding the $Z_L$ term clearly increases robustness and performance. On the other hand, a limitation of this algorithm is that the dimension of the problem must be reasonable (perhaps $n < 20$) for the algorithm to be a viable solution technique.

On certain problems, particularly those when both the suggested starting point and the solution are integers, the algorithms performed particularly well. These problems are not representative tests of the algorithm. Because of the way the simplex is constructed, the algorithm appears to solve these problems almost effortlessly. The true effectiveness of the algorithm is best illuminated by results from the random (non–integral) starting points.

When the algorithm is run on a parallel machine, one sees significant speed-up as the number of processors grows. To date, this algorithm has solved difficult problems with as many as 20 variables in a reasonable amount of time. The algorithm requires at least $O(n)$ function and constraint evaluations per iteration. (In fact, at least $2n$ each of function and constraint evaluations are required per iteration.) This makes it an expensive algorithm. On the other hand, it has been shown to be very robust on some highly nonlinear problems. In addition, given special exploitable information about a particular problem, one can customize the algorithm to adjust search direction, choose particular appropriate starting patterns, or even adjust dilation and contraction parameters in a way most efficient for the particular problem.

Another interesting application not discussed in this thesis is that of the semi–infinite programming problem. Our algorithm might very well prove effective on semi–infinite programming problems for which the number of variables remains small (but there are infinitely many constraints).

In brief, this is a numerically effective modification of the pattern-search method described in [140]. The penalty function modification accelerates convergence and increases the robustness of the algorithm in the presence of constraints. For problems of small to moderate size, the algorithm is easy to implement and effective.

| problem | start | var. | eq.'s | ineq.'s | iter | rel($f^*$) | $G_2(x^*)$ | Simplex |
|---|---|---|---|---|---|---|---|---|
| Cham | sug | 2 | 0 | 2 | 1 | 0.e0 | 0.e0 | Right |
| Cham | rand | 2 | 0 | 2 | 8 | 0.e0 | 0.e0 | Right |
| Tana | sug | 2 | 0 | 5 | 7 | 5.e-1 | 3.e-3 | Right |
| Tana | rand | 2 | 0 | 5 | 21 | 5.e-1 | 2.e-3 | Right |
| HS71 | sug | 4 | 1 | 9 | 18 | 3.e-3 | 2.e-4 | Right |
| HS71 | rand | 4 | 1 | 9 | 27 | 5.e-3 | 1.e-3 | Right |
| HS78 | sug | 5 | 3 | 0 | 19 | 1.e-2 | 9.e-3 | Right |
| HS78 | rand | 5 | 3 | 0 | 49 | 8.e-2 | 5.e-3 | Right |
| PCBurg | rand | 2 | 0 | 3 | 16 | 3.e-3 | 3.e-2 | Right |
| PCBurg | rand | 2 | 0 | 3 | 18 | 3.e-3 | 3.e-2 | Regular |
| CoulFr | rand | 2 | 0 | 3 | 29 | 7.e-3 | 4.e-3 | Right |
| CoulFr | rand | 2 | 0 | 3 | 35 | 7.e-3 | 4.e-3 | Regular |
| Shape | rand | 2 | 0 | 3 | 17 | 7.e-3 | 4.e-3 | Right |
| Shape | rand | 2 | 0 | 3 | 23 | 7.e-3 | 4.e-3 | Regular |

**Table 3.2** Performance of constrained-PDS with $\ell_1$ penalty function

| problem | start | var. | eq.'s | ineq.'s | iter | rel($f^*$) | $G_2(x^*)$ | Simplex |
|---|---|---|---|---|---|---|---|---|
| Cham | sug | 2 | 0 | 2 | 1 | 0.e0 | 0.e0 | Right |
| Cham | rand | 2 | 0 | 2 | 12 | 0.e0 | 0.e0 | Right |
| Tana | sug | 2 | 0 | 5 | 10 | 0.e0 | 3.e-3 | Right |
| Tana | rand | 2 | 0 | 5 | 27 | 0.e0 | 1.e-4 | Right |
| HS71 | sug | 4 | 1 | 9 | 20 | 2.e-3 | 1.e-3 | Right |
| HS71 | rand | 4 | 1 | 9 | 28 | 1.e-3 | 1.e-4 | Right |
| HS78 | sug | 5 | 3 | 0 | 20 | 3.e-2 | 9.e-4 | Right |
| HS78 | rand | 5 | 3 | 0 | 94 | 3.e-2 | 5.e-3 | Right |
| PCBurg | rand | 2 | 0 | 3 | 14 | 1.e-3 | 3.e-3 | Right |
| PCBurg | rand | 2 | 0 | 3 | 13 | 1.e-3 | 4.e-3 | Regular |
| CoulFr | rand | 2 | 0 | 3 | 10 | 2.e-3 | 9.e-4 | Right |
| CoulFr | rand | 2 | 0 | 3 | 12 | 2.e-3 | 2.e-3 | Regular |
| Shape | rand | 2 | 0 | 3 | 10 | 2.e-3 | 2.e-4 | Right |
| Shape | rand | 2 | 0 | 3 | 10 | 2.e-3 | 3.e-3 | Regular |

**Table 3.3** Performance of constrained-PDS with $\ell_2$ penalty function

| problem | start | var. | eq.'s | ineq.'s | iter | rel($f^*$) | $G_2(x^*)$ | Simplex |
|---|---|---|---|---|---|---|---|---|
| Cham | sug | 2 | 0 | 2 | 1 | 0.e0 | 0.e0 | Right |
| Cham | rand | 2 | 0 | 2 | 13 | 0.e0 | 0.e0 | Right |
| Tana | sug | 2 | 0 | 5 | 15 | 0.e0 | 3.e-3 | Right |
| Tana | rand | 2 | 0 | 5 | 20 | 0.e0 | 2.e-3 | Right |
| HS71 | sug | 4 | 1 | 9 | 38 | 2.e-3 | 2.e-4 | Right |
| HS71 | rand | 4 | 1 | 9 | 27 | 3.e-3 | 9.e-4 | Right |
| HS78 | sug | 5 | 3 | 0 | 31 | 9.e-4 | 2.e-3 | Right |
| HS78 | rand | 5 | 3 | 0 | 72 | 1.e-3 | 8.e-3 | Right |
| PCBurg | rand | 2 | 0 | 3 | 12 | 4.e-3 | 3.e-3 | Right |
| PCBurg | rand | 2 | 0 | 3 | 12 | 5.e-3 | 9.e-3 | Regular |
| CoulFr | rand | 2 | 0 | 3 | 12 | 5.e-3 | 3.e-3 | Right |
| CoulFr | rand | 2 | 0 | 3 | 12 | 3.e-3 | 3.e-3 | Regular |
| Shape | rand | 2 | 0 | 3 | 12 | 8.e-3 | 3.e-3 | Right |
| Shape | rand | 2 | 0 | 3 | 12 | 3.e-3 | 3.e-3 | Regular |

**Table 3.4** Performance of constrained-PDS with $\ell_\infty$ penalty function

| problem | start | var. | eq.'s | ineq.'s | iter | rel($f^*$) | $G_2(x^*)$ | Simplex |
|---|---|---|---|---|---|---|---|---|
| Cham | sug | 2 | 0 | 2 | 1 | 0.e0 | 0.e0 | Right |
| Cham | rand | 2 | 0 | 2 | 8 | 0.e0 | 0.e0 | Right |
| Tana | sug | 2 | 0 | 5 | 8 | 0.e0 | 4.e-4 | Right |
| Tana | rand | 2 | 0 | 5 | 18 | 0.e0 | 4.e-4 | Right |
| HS71 | sug | 4 | 1 | 9 | 13 | 2.e-3 | 3.e-4 | Right |
| HS71 | rand | 4 | 1 | 9 | 20 | 2.e-3 | 3.e-4 | Right |
| HS78 | sug | 5 | 3 | 0 | 16 | 2.e-4 | 8.e-4 | Right |
| HS78 | rand | 5 | 3 | 0 | 29 | 2.e-4 | 7.e-4 | Right |
| PCBurg | rand | 2 | 0 | 3 | 14 | 3.e-3 | 4.e-4 | Right |
| PCBurg | rand | 2 | 0 | 3 | 15 | 3.e-3 | 5.e-4 | Regular |
| CoulFr | rand | 2 | 0 | 3 | 7 | 1.e-3 | 2.e-3 | Right |
| CoulFr | rand | 2 | 0 | 3 | 7 | 1.e-3 | 2.e-3 | Regular |
| Shape | rand | 2 | 0 | 3 | 8 | 4.e-4 | 1.e-3 | Right |
| Shape | rand | 2 | 0 | 3 | 8 | 4.e-4 | 1.e-3 | Regular |

**Table 3.5** Performance of constrained-PDS
with suggested $\Pi_L$ penalty function

# Chapter 4

# Exact and Approximate Neuman Boundary Control of the Heat Equation

## 4.1 Introduction

It is not difficult to think of industrial processes which pose problems of controlling the thermal state of a body through influences on its boundaries. The design of a sterilization process for canned goods, the heat-treating of metal parts, the vulcanization of complex rubber moldings, the baking of ceramic wafers for electronic parts are examples that come easily to mind. Optimal control of such systems through the boundary conditions on the governing partial differential equation can be of great practical advantage.

Various thermal boundary conditions are commonly considered for the heat equation. The simplest of these are (1) *Dirichlet* conditions in which the temperature at a surface is determined (2) *Neumann* conditions in which the heat flux through a boundary is determined through setting the temperature gradient (3) *Robin* conditions in which the ratio of temperature and its gradient is determined. We shall be concerned only with the first two of these. The Neumann conditions will be used for controlling the development of a temperature distribution within an object.

Consider the optimal control of the thermal state of an object by means of heat fluxes through a part of the surface of the object. The aim of the control is to reach a target thermal state in the interior of the object at some prescribed time. We will consider several closely related ways of formulating the control problem for optimization and will compare the results of optimizing for three test targets.

The technique for optimization here uses a finite-element spatial discretization coupled with a finite-difference time discretization. The original Neumann boundary control problem is transformed to an *identification problem* in which we determine the initial data for an adjoint equation. Solution of the control problem is actually achieved through solution of this identification problem. (The sophisticated reader may detect the motivation for this approach in the Hilbert Uniqueness Method.) The solutions of four discrete control problems are found iteratively with a Sequential Quadratic Programming algorithm (SQP) in three cases and a conjugate gradient method for the fourth.

The chapter will be organized as follows. The next section presents the state equation, the mixed Neumann/Dirichlet boundary conditions and the optimality systems. Section 4.3 presents some optimization formulations. Formulations using penalty terms and using nonlinear constraints are given for the cases with and without state variables included among the control variables. Section 4.4 outlines the discretizations. A suite of three test targets is given in section 4.5. Section 4.6 and 4.7 cover the numerical results and some conclusions, respectively.

## 4.2 Neumann boundary-control of the heat equation

*Fourier's law* of heat conduction states that heat energy flows from high to low temperature at a rate proportional to the temperature gradient. This law, when combined with Green's formula, implies the *heat equation.* The pertinent physical material properties (assumed constant here) are the heat conductivity, $K$, the heat capacity, $C_p$, the density, $\rho$. We treat a finite, physical-space domain and we add to this list a length of that domain, call it $d$. (In our test problems, $d$ is the side of a square.) As we did in the previous chapter, we also add to the list the mechanical equivalent of heat, $J$, and then form normalizers from these quantities in order to nondimensionalize time, length, and temperature.

$$\begin{aligned} n_t &= \rho C_p d^2/K \\ n_l &= d \\ n_\theta &= K^2/(\rho^2 J C_p^3 d^2) \end{aligned} \tag{4.1}$$

where $n_t$, $n_l$, and $n_\theta$ are the normalizers for time, length, and temperature, respectively. Again, all quantities in the following discussion are normalized to be dimensionless.

Consider a domain $\Omega \in \Re^N$ where $N = 1, 2, 3$. We assume that $\Omega$ has a boundary $\Gamma$ sufficiently smooth for our purposes. (Exactly what constitutes a 'sufficiently smooth' boundary is a subtle question not discussed here. Roughly speaking, we expect it to be a continuous boundary without sharp cusps.) Because our control acts on only a part of the boundary of $\Omega$, we formally split the boundary into two parts; $\Gamma = \Gamma_N \cup \Gamma_D$. Control will act on the first part through the Neumann boundary conditions on $\Gamma_N$. According to Fourier's law, this will control the rate of heat flow into the domain through this part of the boundary. On the second part of the boundary, $\Gamma_D$, Dirichlet boundary conditions are prescribed to determine the temperature as some constant in time over the second part of the boundary. We define the space-time quantities

$$Q = \Omega \times (0, T), \quad \Sigma_D = \Gamma_D \times (0, T) \text{ and } \Sigma_N = \Gamma_N \times (0, T). \tag{4.2}$$

After these preliminaries, we write down the heat equation in a nondimensional form.

$$\frac{\partial y}{\partial t} - \triangle y = f, \tag{4.3}$$

boundary conditions

$$\begin{aligned} y &= a \quad \text{on} \quad \Sigma_D \\ \frac{\partial y}{\partial n} &= v(t) \quad \text{on} \quad \Sigma_N \end{aligned} \tag{4.4}$$

and initial conditions

$$y(0) = y_0 \quad \text{in} \quad \Omega. \tag{4.5}$$

We have introduced the function $y(t)$ denoting $x \to y(x,t)$. The quantity $f$ on the right of the heat equation represents a heat source at a point of the domain. We will consider the case without heat sources and with a constant temperature on $\Gamma_D$, where the Dirichlet boundary conditions apply. We will set the zero of the temperature scale so that this constant temperature is zero, thus: $a = 0$, $f = 0$.

We denote the state we aim to reach at time $t = T$, the target state, by $y_d$. In the case of exact controllability, we have

$$y(T) = y_d. \tag{4.6}$$

The initial value of the state variables, $y_0$, and the target state, $y_d$, are given in $L^2(\Omega)$,

$$L^2(\Omega) \equiv \left\{ \phi | \int_\Omega |\phi|^2 dx < \infty \right\}.$$

We choose a space, $V$ say, such that $V \subset H^1(\Omega)$, where

$$H^1(\Omega) \equiv \left\{ \phi | \phi, \frac{\partial \phi}{\partial x_i} \in L^2(\Omega) \qquad i = 1,2 \right\}$$

and

$$V(\Omega) \equiv \left\{ \phi | \phi \in H^1(\Omega) \quad \text{and} \quad \phi = 0 \quad \text{on} \quad \Gamma_D \right\}.$$

Clearly, the Laplace operator appearing above is such that, $-\triangle : V \to V^*$, where $V^*$ denotes the dual space of $V$. Given $\psi$ an element from $V^*$ we can define $N : L^2(\Gamma_N) \to V^*$ by

$$(Nv, \psi)_{V \times V^*} = \int_{\Gamma_N} v \psi d\Gamma_N. \tag{4.7}$$

We search for the Neumann control, $v$, (necessarily unique, if it exists) that is of minimum $L^2(\Sigma_N)$ norm. Specifically, it is the solution of the problem

$$\min_{v \in U_f} J(v) \tag{4.8}$$

where

$$J(v) = \frac{1}{2} \int_{\Sigma_N} |v|^2 d\Gamma_N dt, \tag{4.9}$$

$$U_f \equiv \left\{ v | v \in L^2(\Sigma_N) \ \text{ and the pair } \{v, y\} \quad \text{satisfy} \quad (4.3) - (4.5) \right\}. \tag{4.10}$$

If we assume that $U_f$ is not an empty set, then we know it is a closed and convex subset of $L^2(\Sigma_N)$. In general, one cannot guarantee the existence of a solution to (4.8); $y_d$ must be sufficiently smooth, $y_d \in C^\infty(\bar{\Omega})$.

Approximating (4.8) by the following problem facilitates the derivation of the optimality conditions that must be satisfied by our solution (assuming it exists). The problem for the approximation is

$$\min_{v \in L^2(\Sigma_N)} J_\epsilon(v) \tag{4.11}$$

where $J_\epsilon(\cdot)$ is defined for any positive $\epsilon$ by

$$J_\epsilon(v) = \frac{1}{2}\int_{\Sigma_N} |v|^2 d\Gamma_N dt + \frac{1}{2\epsilon}\|y(T) - y_d\|^2_{L^2(\Omega)}, \tag{4.12}$$

and $y$ satisfies (4.3) - (4.5).

A unique solution to our approximate problem (4.11) exists (see [98] for a proof of this). Let us denote this solution by $u_\epsilon$. This is the gradient of temperature on $\Gamma_N$, the control. The optimal pair $(u_\epsilon, y_\epsilon)$ is characterized by the existence of $p_\epsilon$ where

$$\begin{aligned}
&\frac{\partial y_\epsilon}{\partial t} - \triangle y_\epsilon = 0 \quad \text{in} \quad Q \\
&y_\epsilon = 0 \quad \text{on} \quad \Sigma_D \\
&\frac{\partial y_\epsilon}{\partial n} = u_\epsilon \quad \text{on} \quad \Sigma_N \\
&y_\epsilon(0) = y_0 \\
&-\frac{\partial p_\epsilon}{\partial t} - \triangle p_\epsilon = 0 \quad \text{in} \quad Q \\
&\frac{\partial p_\epsilon}{\partial n} = 0 \quad \text{on} \quad \Sigma_N \\
&p_\epsilon = 0 \quad \text{on} \quad \Sigma_D \\
&p_\epsilon(T) = \frac{1}{\epsilon}(y_\epsilon(T) - y_d) \\
&u_\epsilon = p_\epsilon \quad \text{on} \quad \Sigma_N
\end{aligned} \tag{4.13}$$

This system can then be used to derive optimality conditions for (4.8). If we assume that problem (4.11) has a solution, it can be shown that this solution must correspond to the solution of (4.13) in the limit as $\epsilon \to 0$. This limit is the optimality system

$$\begin{aligned}
&\frac{\partial y}{\partial t} - \triangle y = 0 \quad \text{in} \quad Q \\
&y = 0 \quad \text{on} \quad \Sigma_D \\
&\frac{\partial y}{\partial n} = u \quad \text{on} \quad \Sigma_N \\
&y(0) = y_0 \quad \text{in} \quad \Omega \\
&-\frac{\partial p}{\partial t} - \triangle p = 0 \quad \text{in} \quad Q \\
&\frac{\partial p}{\partial n} = 0 \quad \text{on} \quad \Sigma_N \\
&p = 0 \quad \text{on} \quad \Sigma_D \\
&y(T) = y_d \quad \text{in} \quad \Omega,
\end{aligned} \tag{4.14}$$

where $\dfrac{\partial p}{\partial n}$ denotes the normal derivative of the adjoint state variable, $p$.

## 4.3 Optimization formulations

The thermal problem can be formulated for optimization in a variety of ways. We shall consider four of them. We begin with two formulations with the same number of control variables; the first, an unconstrained minimization; the second, a minimization with constraints.

The first formulation follows directly from the starting point of the derivation of the optimality system. It is a re-statement of the regularized cost-function (4.11) with an appended penalty function.

$$\min_{v \in U_f} \int_{\Sigma_N} |v(t)|^2 d\Gamma_N dt + \frac{1}{2\epsilon} \|y_d - y_\epsilon(T)\|^2_{L^2(\Omega)}. \tag{4.15}$$

This formulation, for the special case of Dirichlet controls, is discussed extensively in [23] and also in [70] and [71].

The second formulation is one in which a nonlinear constraint is introduced in lieu of the penalty term. The constraint uses the same norm as the penalty function to restrict the optimal optimal controls so that

$$\|y_d - y(T)\|_{L^2(\Omega)} \leq \eta.$$

The parameter $\eta$ bounds (in the $L^2$ sense) the distance between the value of the state variables at time $T$ and the desired target. The formulation is

$$\begin{array}{l} \min\limits_{v \in U_f} \displaystyle\int_{\Sigma_N} |v(t)|^2 d\Gamma_N dt \\ \quad \text{subject to:} \\ \|y_d - y(T)\|^2_{L^2(\Omega)} \leq \eta^2, \end{array} \tag{4.16}$$

which is a *nonlinear programming problem*. Of course, the function $y(x,t)$ must satisfy (4.3)–(4.5).

The third and fourth formulations are arrived at by allowing the state variables to be control variables (subject to satisfying the state equation). Such formulations usually result in a significantly better-conditioned numerical approximation. They make intensive memory demands, however, and can easily prove cumbersome or impractical for large problems. The nonlinear programs associated with the unconstrained and constrained formulations are

$$
\begin{aligned}
&\min_{v,y} \int_{\Sigma_N} |v(t)|^2 d\Gamma_N dt + \frac{1}{2\epsilon}\|y_d - y(T)\|^2_{L^2(\Omega)} \\
&\quad \text{subject to:} \\
&\frac{\partial y}{\partial t} - \triangle y = 0 \quad \text{in} \quad Q \\
&y = 0 \quad \text{on} \quad \Sigma_D \\
&\frac{\partial y}{\partial n} = v(t) \quad \text{on} \quad \Sigma_N \\
&y(0) = y_0
\end{aligned}
\tag{4.17}
$$

and

$$
\begin{aligned}
&\min_{v,y} \int_{\Sigma_N} |v(t)|^2 d\Gamma_N dt \\
&\quad \text{subject to:} \\
&\|y_d - y(T)\|^2_{L^2(\Omega)} \leq \eta^2 \\
&\frac{\partial y}{\partial t} - \triangle y = 0 \quad \text{in} \quad Q \\
&y = 0 \quad \text{on} \quad \Sigma_D \\
&\frac{\partial y}{\partial n} = v(t) \quad \text{on} \quad \Sigma_N \\
&y(0) = y_0.
\end{aligned}
\tag{4.18}
$$

The discretizations described in the following section will result in linear constraints to replace the state equation. The adjoint portion of the optimality system (4.13) provides means to evaluate gradients of the cost function with respect to control variables. The gradients with respect to the state variables can be calculated analytically.

## 4.4 Discretization

In this section we discuss the discretization of the heat equation for the suite of test problems described in the next section. We use a combination of finite-difference discretization in time with finite-element discretization in space. For each test problem, the domain is simply the unit square in $\Re^2$, $\Omega = (0,1) \times (0,1)$. We will explain the discretization in this context, although it is clear that the explanation is valid for any piecewise polygonal domain.

In view of the observations of Carthel, Glowinski, and Lions [23], we employ a discretization of second-order accuracy for our state equation and the associated adjoint equation. We will give only an abbreviated description here. Details and accuracy estimates of this discretization are given in [23].

We begin by describing the time discretization. Let $N_T$ be the number of time steps desired, so that $\Delta t = \frac{T}{N_T}$. We will denote the estimate of $y$ at the $n$th time step by $y^n$ where $n = 1 \ldots N_T$. If $N_X$ denotes the number of spatial steps in the $x_1$

and in the $x_2$ directions, the spatial step is denoted by $h$ with $h = \frac{1}{N_X}$. The discrete approximation to $y$ is

$$y(n\Delta t, ih, jh) \approx y^n_{i,j}.$$

We follow the *two-step implicit scheme* for parabolic problems subject to Neumann boundary conditions, as outlined in Glowinski [65]. Accordingly, we define

$$\frac{\partial y}{\partial t}(n + 1\Delta t) \approx \frac{1}{2\Delta t}\left(3y^{n+1}_{i,j} - 4y^n_{i,j} + y^{n-1}_{i,j}\right).$$

Experience with this time discretization has led us to use it on stiff problems when we need to integrate to large values of $T$. In such cases, the fact that it assures unconditional stability and produces an accuracy to second order in time amply justifies the storage costs.

At each time step we must solve an elliptic problem to obtain $y^{n+1}_{i,j}$. The domain is so simple that we use the very common finite-element triangulation of $\Omega$ shown in Figure 4.1. The space of polynomials of degree $\leq 1$ is used to form a finite dimensional approximation to $L^2(\Omega)$, $H^1(\Omega)$ and to the space $V$ defined in section 4.2. (The reader

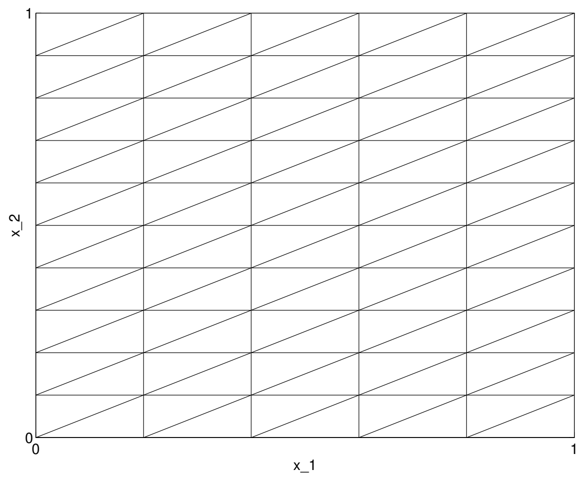


**Figure 4.1** Triangulation of the domain $\Omega$

is referred to Cialet [29] for a detailed description of this construction.) A standard 5 point stencil results, defined by

$$-\triangle y(n\Delta t, ih, jh) \approx \frac{1}{h^2}\left(4y^n_{i,j} - y^n_{i+1,j} - y^n_{i,j+1} - y^n_{i-1,j} - y^n_{i,j-1}\right).$$

More sophisticated schemes are certainly available for both linear and nonlinear parabolic equations. However, for comparing optimization formulations of the control problem here, this simple numerical scheme is both adequate and appropriate. For specific applications, more specialized or hybrid discretizations may be called for.

## 4.5 Test targets

Three target states, $y_T$, are used to compare the performance of the various formulations of the control problem from section 4.2. The targets are given in the order of increasing difficulty of solution.

### 4.5.1 First target state

The target shape $y_T$ (pictured in Figure 4.2) is the only differentiable target tried. Explicitly, it is

$$y_T(x_1, x_2) = 4xy^2. \tag{4.19}$$

This was the only target that did not satisfy homogeneous Dirichlet boundary conditions on all of $\Gamma_D$.

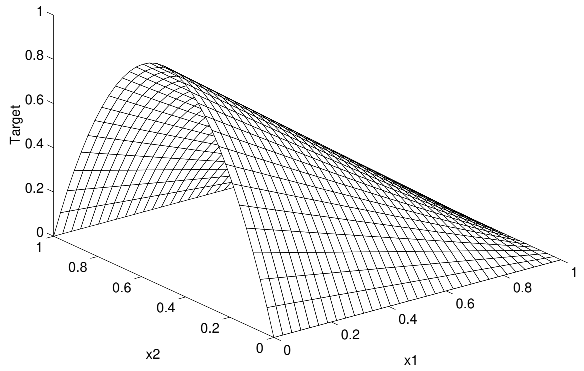


**Figure 4.2** Analytic target shape

### 4.5.2 Second target state

This target shape is a pyramid. (It is used for a test problem of Dirichlet control in [23].) It is a continuous but not differentiable function as shown in Figure 4.3. Specifically, it is

$$y_T(x_1, x_2) = 2\min\{x, y, 1-x, 1-y\}. \tag{4.20}$$

This target proved hard to reach with Neumann control because of its lack of smoothness. The control did achieve states closely resembling the pyramid away from the pinnacle (as did the Dirichlet control in [23]).

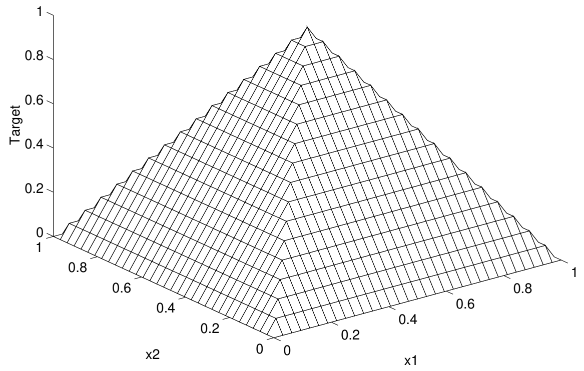


**Figure 4.3** Pyramid target shape

### 4.5.3 Third target state

The third target shape is a right circular cylinder, a target with very severe discontinuities. It is pictured in Figure 4.4. Explicitly, it is

$$y_T(x1,x2) \equiv \begin{cases} 0 & \text{if } (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 > \frac{1}{16} \\ +1 & \text{otherwise}. \end{cases} \tag{4.21}$$

The sharp discontinuities of the target played havoc with the diffusive nature of the heat equation. This target resulted in problems requiring the largest number of iterations and generated the most difficult optimization problems.

## 4.6 Numerical results

The performance of the optimizations on the four formulations of section 2 was examined for each of the three test targets. The combination of finite-difference/finite-element discretization and the SQP algorithm were used. In comparing among the formulations, in order to ensure that they are approximating the same problem, we proceed as follows: we first compute with the penalty function formulation for a given $\epsilon$; we then calculate $\eta$ from

$$\eta = \|y_d - y_\epsilon\|_{L^2(\Omega)};$$

we then compute with the corresponding nonlinear constraint formulation using this value of $\eta$ for the constraint. The various problems are solved with discretization parameters (number of time-steps and number of space-steps) as listed in the tables.

For the solution of (4.15) a conjugate gradient method is used. The method is equivalent to that of [23] except that Neumann conditions replace the Dirichlet

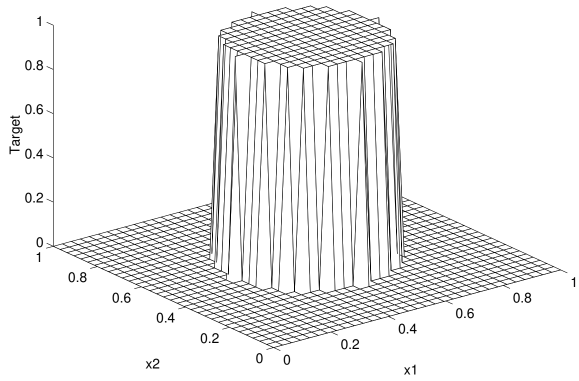


**Figure 4.4** Cylinder target shape

conditions of [23] at the control surfaces on $\Gamma_N$. The formulations (4.16)-(4.18) were solved using the SQP algorithm described in chapter 2. This algorithm did not perform as well with Neumann conditions as it did with Dirichlet conditions in [23].

A number of figures are presented to illustrate the behavior of the controls. By examining Figures 4.17-4.20, 4.21-4.24 and 4.25-4.28 one can see that oscillations of the optimal controls increased with decreasing smoothness of the targets. Upon examining the graphs of the state variables at time $T$, one sees that the control was successful in moving the states into a neighborhood of the desired target (see the profile graphs, 4.5-4.8, 4.9-4.12, and 4.14-4.16 and also the state variable graphs, 4.29-4.32, 4.33-4.36, and 4.37-4.40). On the other hand, the differences between results for penalty function formulations and nonlinear constraint formulations are essentially indistinguishable to the eye.

When the state variables are allowed to be control variables, however, some variance with respect to formulation is seen. Especially for the results with the first test target, it appears that this freedom allows the constrained optimization formulation, (4.18), to reach a smaller neighborhood of the target than when state variables are not control variables (4.16).

The tables summarize the numerical performance that resulted with each formulation.

## 4.7 Conclusions

While the formulation used here with nonlinear constraints does not improve convergence in every instance, it is a viable alternative when the penalization parameters

| | |
|---|---|
| $\Delta t$ | $1/32, 1/64$ |
| $h$ | $1/32$ |
| $\Omega$ | $(0,1) \times (0,1)$ |
| $T$ | 1 |
| $\epsilon$ | 1.E-9 |

**Table 4.1** Numerical Test Parameters

| Target | $\Delta t$ | CG-iters | $\|y_\epsilon - y_d\|$ |
|---|---|---|---|
| Analytic | 1/32 | 118 | 0.33 |
| Pyramid | 1/32 | 229 | 0.96 |
| Cylinder | 1/32 | 613 | 3.71 |
| Analytic | 1/64 | 190 | 0.29 |
| Pyramid | 1/64 | 319 | 0.93 |
| Cylinder | 1/64 | 877 | 3.52 |

**Table 4.2** Numerical results for formulation 4.15

become large. In such cases, the speed of convergence of the conjugate gradient algorithm deteriorates seriously because of the ill-conditioning caused by the penalization.

We observed that this control problem is sensitive to the accuracy of the time discretization. The minor extra expense of the computational book-keeping associated with time discretization of second-order accuracy is well worthwhile. It buys one good convergence and more accurate approximation to the target than schemes with first-order accuracy.

There are many aspects which affect the efficacy of the numerical methods. The formulations and algorithms used here vary so greatly that it is difficult to evaluate

| Target | $\Delta t$ | NL-iters | QP-iters | $\|y_d - y\|$ |
|---|---|---|---|---|
| Analytic | 1/32 | 35 | 612 | 0.32 |
| Pyramid | 1/32 | 37 | 633 | 0.96 |
| Cylinder | 1/32 | 51 | 945 | 3.71 |
| Analytic | 1/64 | 35 | 604 | 0.29 |
| Pyramid | 1/64 | 38 | 646 | 0.93 |
| Cylinder | 1/64 | 51 | 947 | 3.52 |

**Table 4.3** Numerical results for formulation 4.16

| Target | $\Delta t$ | CG-iters | $\|y_\epsilon - y_d\|$ |
|---|---|---|---|
| Analytic | 1/32 | 802 | 0.30 |
| Pyramid | 1/32 | 942 | 0.95 |
| Cylinder | 1/32 | 1009 | 3.61 |
| Analytic | 1/64 | 190 | 0.29 |
| Pyramid | 1/64 | 319 | 0.93 |
| Cylinder | 1/64 | 877 | 3.52 |

**Table 4.4** Numerical results for formulation 4.17

| Target | $\Delta t$ | NL-iters | QP-iters | $\|y_d - y\|$ |
|---|---|---|---|---|
| Analytic | 1/32 | 29 | 550 | 0.30 |
| Pyramid | 1/32 | 31 | 556 | 0.95 |
| Cylinder | 1/32 | 48 | 904 | 3.60 |
| Analytic | 1/64 | 32 | 561 | 0.29 |
| Pyramid | 1/64 | 33 | 623 | 0.93 |
| Cylinder | 1/64 | 46 | 872 | 3.51 |

**Table 4.5** Numerical results for formulation 4.18

the combination of computational costs, memory costs, and associated numerical linear algebra costs. It is apparent, in spite of the good numerical behavior observed in solving (4.16) and (4.18), that they are simply too memory-intensive for larger problems. Yet, the results do encourage the further exploration of optimization formulations that consider a subset of the state variables as control variables, even at the cost of introducing additional constraints.

A natural next step is to duplicate these experiments on a non-linear parabolic problem.

## 4.8 Figures

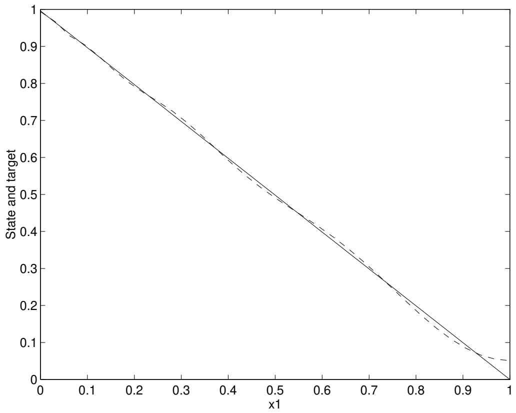


**Figure 4.5** Profile of state and first target for 4.15 with $\epsilon = 1.E - 9$

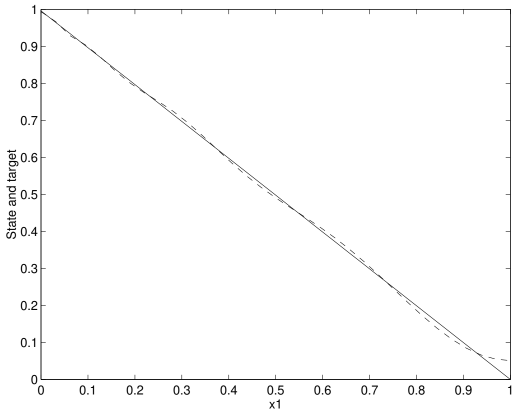


**Figure 4.6** Profile of state and first target for 4.16 with $\eta = .029$

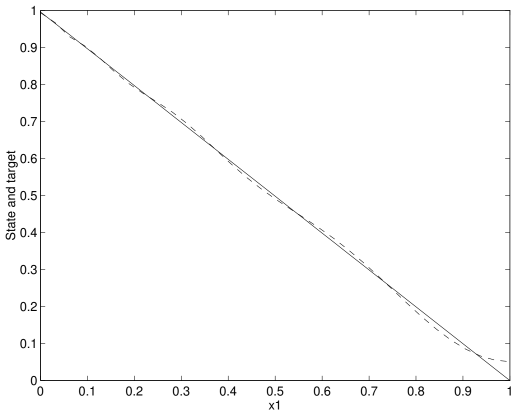


**Figure 4.7** Profile of state and first target for 4.17 with $\epsilon = 1.E - 9$

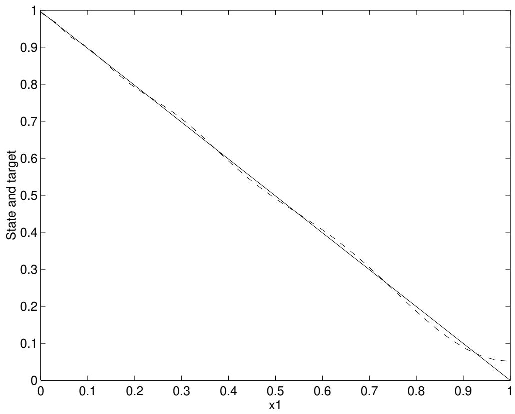


**Figure 4.8** Profile of state and first target for 4.18 with $\eta = .029$

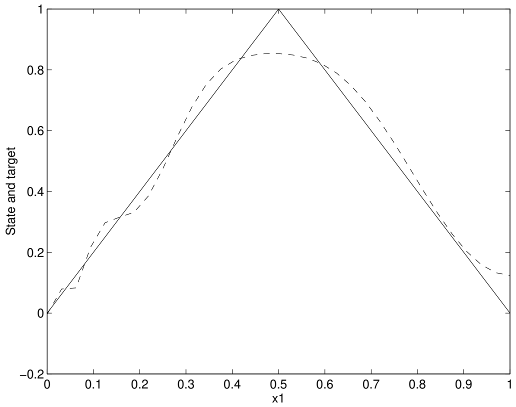


**Figure 4.9** Profile of state and second target for 4.15 with $\epsilon = 1.E - 9$

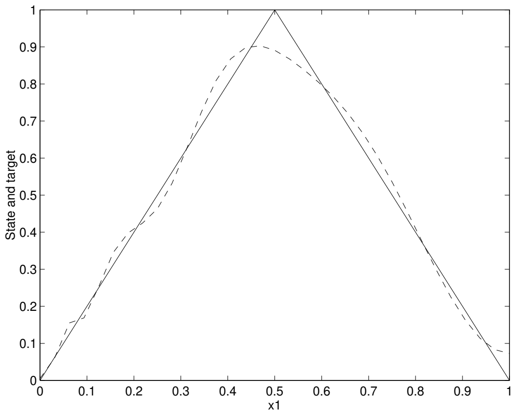


**Figure 4.10** Profile of state and second target for 4.16 with $\eta = .0933$

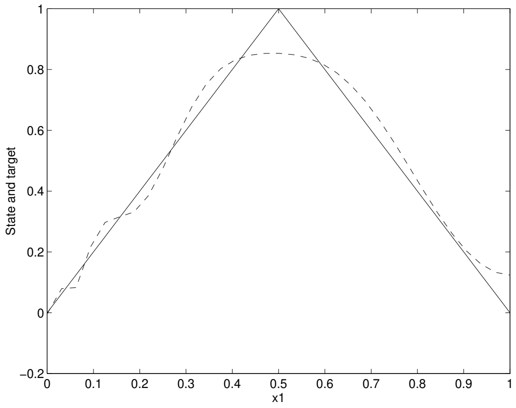


**Figure 4.11** Profile of state and second target for 4.17 with $\epsilon = 1.E - 9$

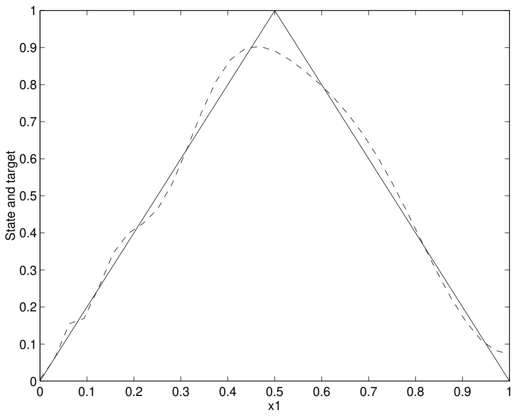


**Figure 4.12** Profile of state and second target for 4.18 with $\eta = .0933$

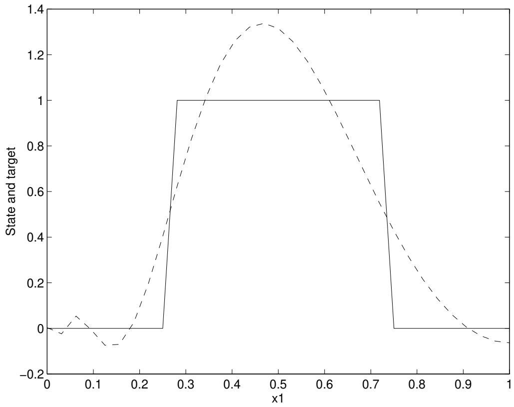


**Figure 4.13** Profile of state and third target for 4.15 with $\epsilon = 1.E - 9$

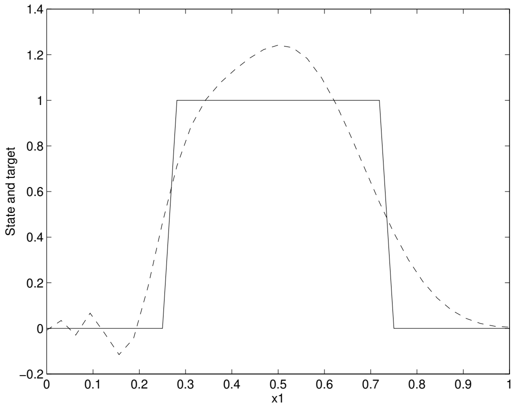


**Figure 4.14** Profile of state and third target for 4.16 with $\eta = .352$

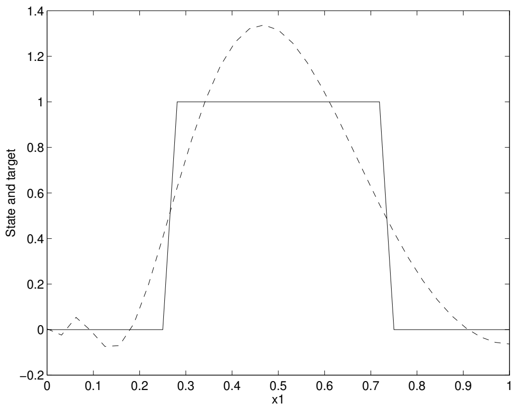


**Figure 4.15** Profile of state and third target for 4.17 with $\epsilon = 1.E - 9$

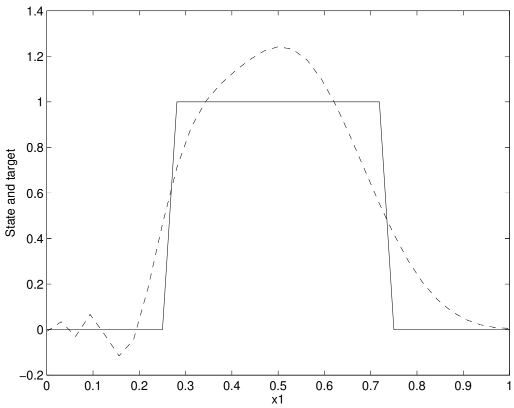


**Figure 4.16** Profile of state and third target for 4.18 with $\eta = .352$

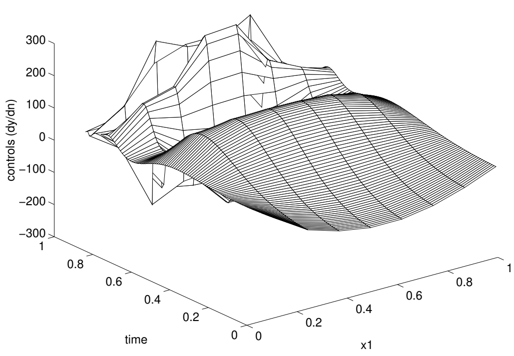


**Figure 4.17** Computed boundary Neumann control for 4.15 with $\epsilon = 1.E-9$

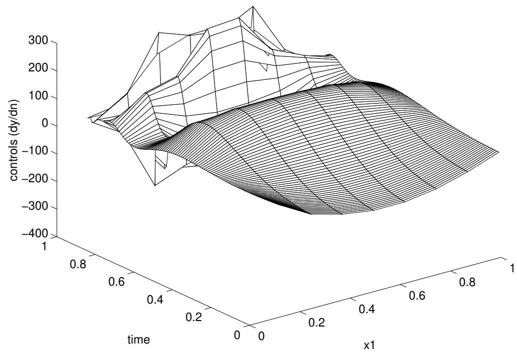


**Figure 4.18** Computed boundary Neumann control for 4.16 with $\eta = .029$

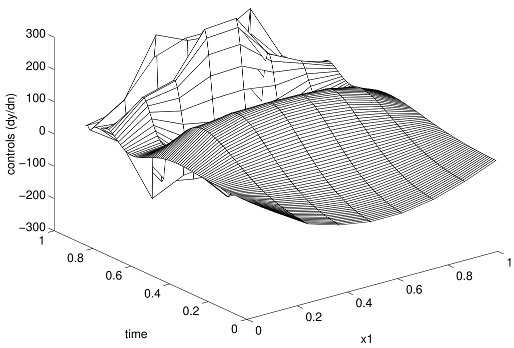


**Figure 4.19** Profile of state and first target for 4.17 with $\epsilon = 1.E-9$

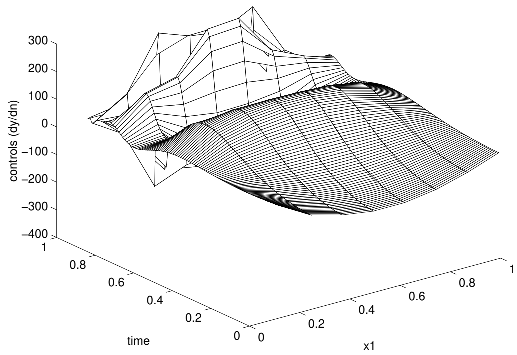


**Figure 4.20** Profile of state and first target for 4.18 with $\eta = .029$

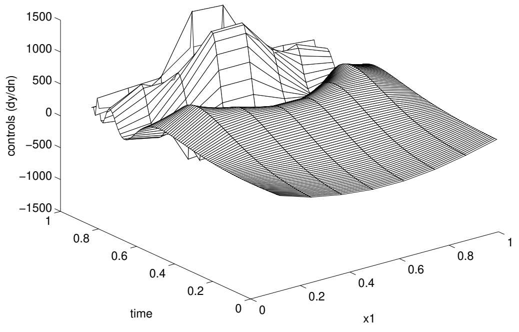


**Figure 4.21** Computed boundary Neumann control for 4.15 with $\epsilon = 1.E - 9$

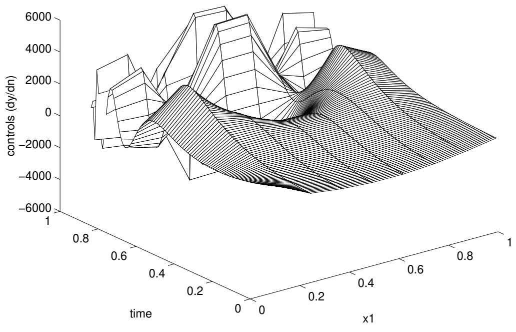


**Figure 4.22** Computed boundary Neumann control for 4.16 with $\eta = .093$

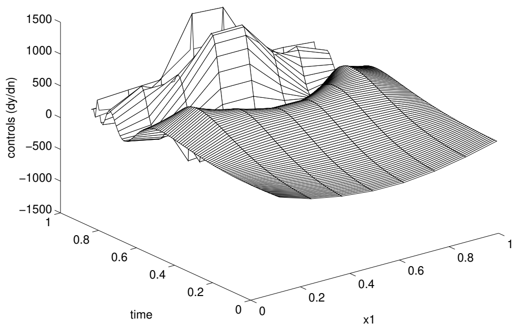


**Figure 4.23** Computed boundary Neumann control for 4.17 with $\epsilon = 1.E - 9$

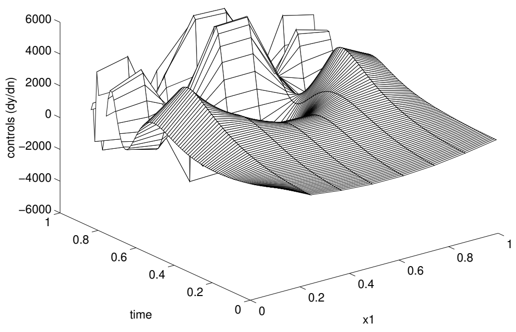


**Figure 4.24** Computed boundary Neumann control for 4.18 with $\eta = .093$

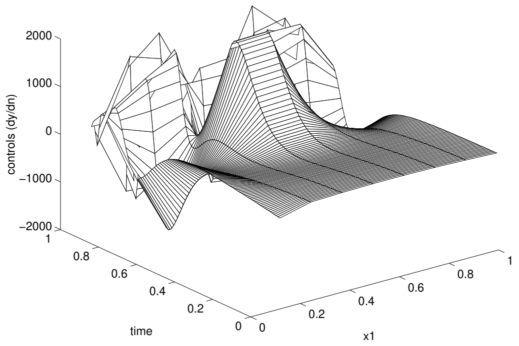


**Figure 4.25** Computed boundary Neumann control for 4.15 with $\epsilon = 1.E - 9$

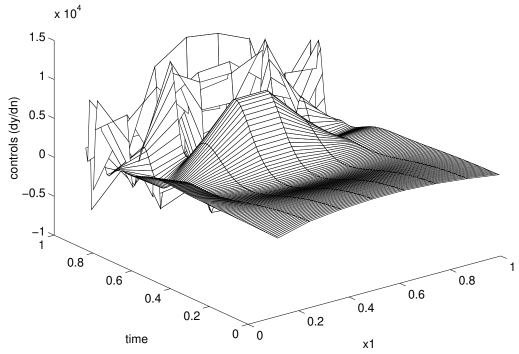


**Figure 4.26** Computed boundary Neumann control for 4.16 with $\eta = .352$

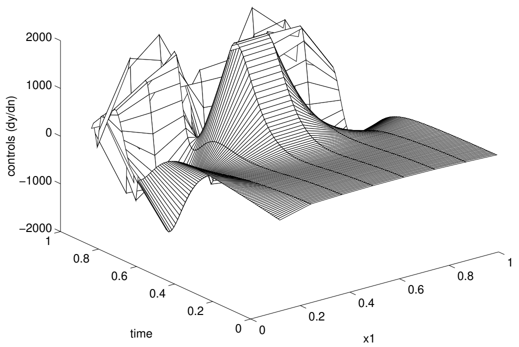


**Figure 4.27** Computed boundary Neumann control for 4.17 with $\epsilon = 1.E - 9$

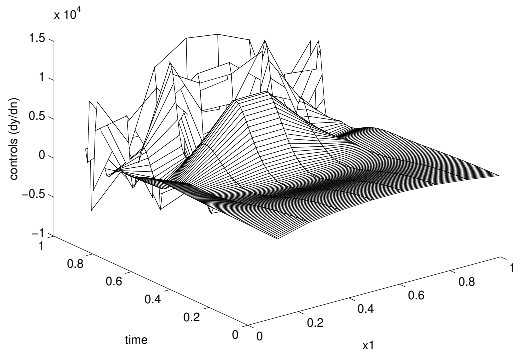


**Figure 4.28** Computed boundary Neumann control for 4.18 with $\eta = .352$

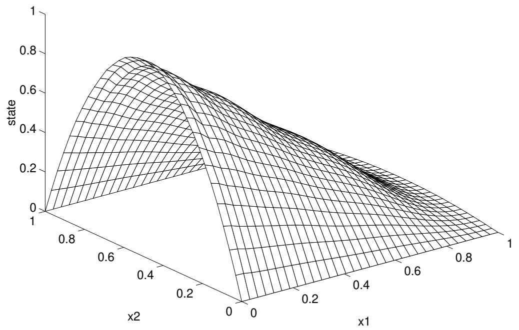


**Figure 4.29** Mesh of state variables solving 4.15 with $\epsilon = 1.E - 9$

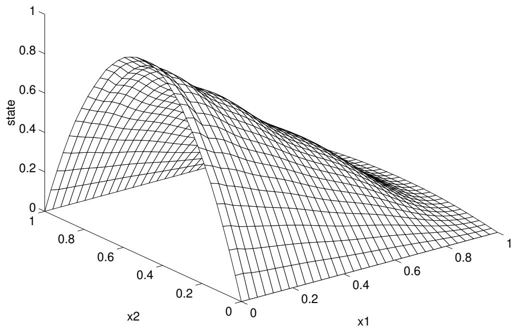


**Figure 4.30** Mesh of state variables solving 4.16 with $\eta = .029$

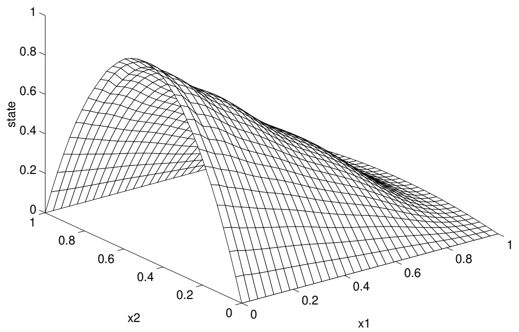


**Figure 4.31** Mesh of state variables solving 4.17 with $\epsilon = 1.E - 9$

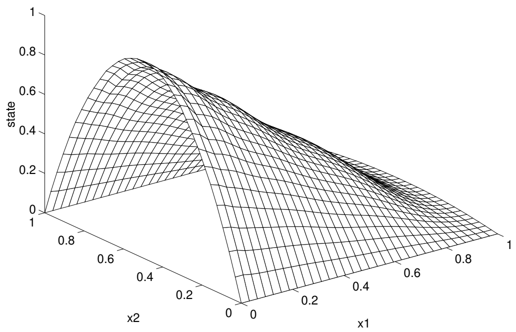


**Figure 4.32** Mesh of state variables solving 4.18 with $\eta = .029$

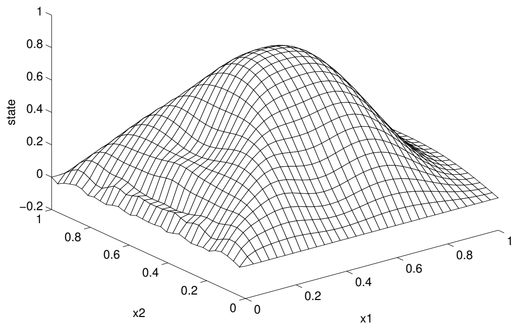


**Figure 4.33** Mesh of state variables solving 4.15 with $\epsilon = 1.E-9$

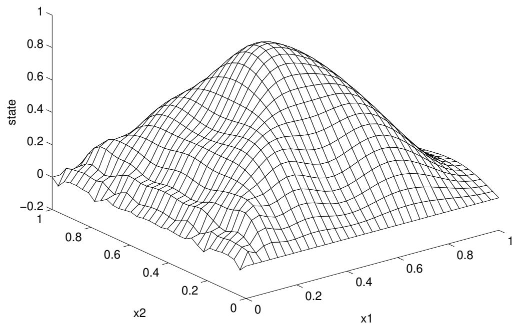


**Figure 4.34** Mesh of state variables solving 4.16 with $\eta = .093$

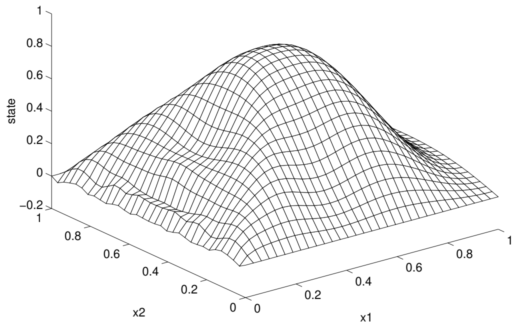


**Figure 4.35** Mesh of state variables solving 4.17 with $\epsilon = 1.E-9$

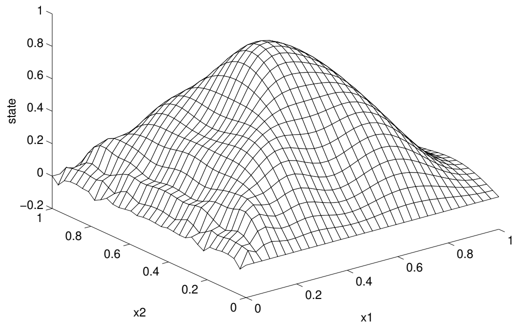


**Figure 4.36** Mesh of state variables solving 4.18 with $\eta = .093$

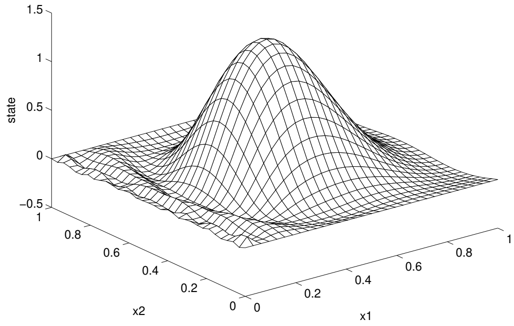


**Figure 4.37** Mesh of state variables solving 4.15 with $\epsilon = 1.E - 9$

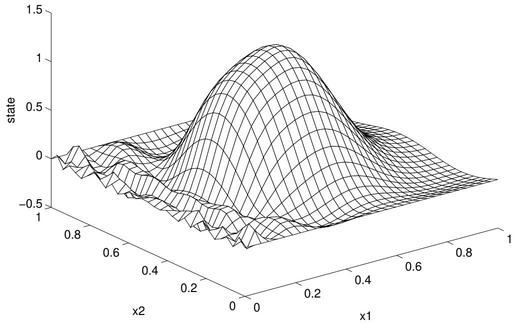


**Figure 4.38** Mesh of state variables solving 4.16 with $\eta = .352$

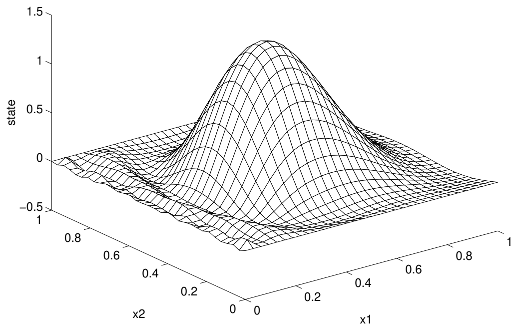


**Figure 4.39** Mesh of state variables solving 4.17 with $\epsilon = 1.E - 9$

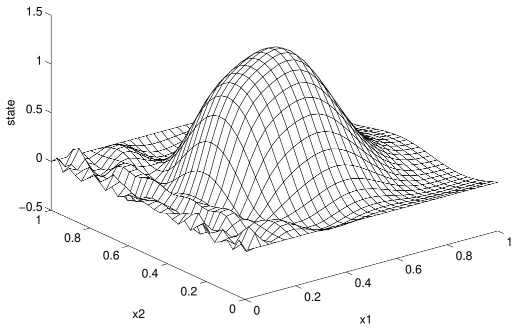


**Figure 4.40** Mesh of state variables solving 4.18 with $\eta = .352$

# Chapter 5

# Hierarchical Control of Systems Governed by PDE's

## 5.1 Introduction

A particularly fertile area of contemporary applied mathematics is the control and the optimal control of systems governed by differential equations. The ideas and methods developed in this field can have important applications throughout applied science and engineering. Sometimes a problem, though seemingly unrelated, can be posed in an unexpected way as an 'optimal control problem' . This often results in new insight and novel numerical solution techniques (see, for example, Dean and Gubernatis [41]).

Here, with an eye toward certain applications in petroleum and reservoir engineering, we shall introduce the idea of multiple controls, each with different objectives. We have in mind control of the Navier–Stokes equation which (among other advection-diffusion equations) has been a subject of much interest (see Burns and Kang [19] and Gunzburger, Hou and Svobodny [79]). In this chapter, we shall treat a simpler equation , however. The *Burgers' equation* is a nonlinear equation designed originally to give a preliminary insight into the nature of turbulence in fluid flows. It is a one-dimensional spatial model of a variety of three-dimensional physical phenomena, greatly simplifying the problems while retaining many of the complex behavior characteristics. It is also a useful tool for examining the robustness of numerical schemes (i.e., if a computational scheme does not work for Burgers' equation, it will certainly not work for the three-dimensional spatial problems posed by the Navier-Stokes equation.)

We consider a class of inequality-constrained, nonlinear programming problems which arise in a special class of control problems for systems governed by partial differential equations. We examine the situation where (1) there exists more than one control for a system and (2) a *hierarchy* is associated with these controls; thus, a multi-objective optimization problem arises. A brief overview of the theory for such problems follows. We then argue that this multi-objective optimization problem is well approximated by an inequality-constrained, nonlinear programming problem. We conclude with results from a nonlinear programming algorithm for the numerical solution of a set of these problems. It will be shown that the degree of difficulty in solving an approximate hierarchical-control problem is dependent on the *viscosity parameter* in the equation.

In the next section we describe the problem of hierarchical control for the one-dimensional Burgers' equation. Some comments regarding the choice of topology and function spaces are included. Section 5.3 contains a derivation of the optimality conditions for our model problem. Some of the finer details are covered because this derivation is non-standard. The discretization is explained in section 5.4, followed by some test problems in section 5.5. Comments on our numerical results are contained in section 5.6; our conclusions are in section 5.7.

## 5.2 The control problem

To demonstrate hierarchical control, we apply it to a model of a nonlinear, advection–diffusion phenomenon. We choose the Burgers' equation in one space-dimension and one time-dimension as our model problem in order to keep the numerical discretization simple. The governing equation is thus

$$y_t - \nu y_{xx} + y y_x = f \quad \text{in} \quad (0,1) \times (0,T). \tag{5.1}$$

We take $Q$ to be our 'space-time' domain, $Q = (0,1) \times (0,T)$.

This equation is a one-dimensional analog of the Navier-Stokes equation. In that interpretation, the variable $y(x,t)$ is the velocity of a fluid at the point $x$ at time $t$; $f(x,t)$ is the specific external force (force per unit mass) acting on a point of the fluid and $\nu$ is kinematic viscosity. In view of its role in this example, the parameter $\nu$ is generally called the *viscosity parameter*.

Various physical problems can suggest many interesting initial and boundary conditions for this problem. We elect to use a simple set.

$$\begin{array}{l} y(x,0) = y_0(x) \quad \text{for} \quad x \in (0,1) \\ y_x(0,t) = y(1,t) = 0 \quad \text{and} \quad y_x(0,t) = 0 \quad \text{for all} \quad t \in (0,T) \end{array} \tag{5.2}$$

for our initial and boundary conditions, respectively.

The problem of interest here is that of controlling the system above to produce a state variable $y(T)$ at some time $T$ that is as close as possible to a primary target, $y_{T_1}$, while remaining 'close to' additional targets, $y_{T_i}$, where $i = 2, \ldots N$. We aim to achieve this goal through multiple, pointwise controls in the spatial domain. The pointwise controls can be thought of as sources and/or sinks that enter the right-hand side of the equation. This statement of the problem is motivated by an environmental application. In reservoir engineering, the controls would correspond to well terms (Cowsar, Glowinksi, Kearsley Wheeler and Yotov [35]). Also of obvious interest is control through boundary conditions, but that will not be discussed here, since it is not appropriate to the motivating application.

Consider the following complete state equation,

$$
\begin{aligned}
&y_t - \nu y_{xx} + y y_x = f + \sum_1^N v_i(t)\delta(x - a_i(t)) \quad \text{in } \; Q \\
&y(x,0) = y_0(x) \quad \text{for } \; x \in (0,1) \\
&y_x(0,t) = y(1,t) = 0 \;\; \text{and} \;\; y_x(0,t) = 0 \quad \text{for all } \; t \in (0,T)
\end{aligned}
\tag{5.3}
$$

Here, $\delta(\cdot)$ denotes the Dirac measure at the argument $(\cdot)$. On occasion, we will also use $\delta$ in another sense, to indicate a perturbation. It will always be clear from context which use of $\delta$ is intended. We will denote by $v$ and $a$ the vectors with components $v_i$ and $a_i$, respectively. For clarity, we will consider the simple case where there are only two targets (only one secondary target, $N = 2$) and $a_1(t)$ and $a_2(t)$ are paths in $(0,1)$. This case is easier to work with than when $N > 2$, yet everything we do can be extended in an obvious way to more than two controls. The state equation appropriate when $N = 2$ is, then,

$$
\begin{aligned}
&y_t - \nu y_{xx} + y y_x = f + v_1(t)\delta(x - a_1) + v_0(t)\delta(x - a_0) \quad \text{in } \; Q \\
&y(x,0) = y_0(x) \quad \text{for } \; x \in (0,1) \\
&y_x(0,t) = y(1,t) = 0 \;\; \text{and} \;\; y_x(0,t) = 0 \quad \text{for all } \; t \in (0,T).
\end{aligned}
\tag{5.4}
$$

We do not consider here the details of necessary and sufficient conditions for a solution to (5.4); rather we concentrate on the notion of multiple targets. We assume, therefore, that our primary and secondary targets, $y_{T_1}$ and $y_{T_2}$, are compatible with the boundary conditions in (5.2). In view of the nonlinear nature of our model, this assumption will greatly simplify our task. Our objective is to find controls $v_1(t)$ and $v_2(t)$ (both sufficiently smooth; $(v_1(t), v_2(t) \in L^2(0,T))$, say) so that $y(x,t)$ is as close to $y_{T_1}$ as possible while $y(x,t)$ is also in a neighborhood of $y_{T_2}$. Given that $v_1(t), v_2(t) \in L^2(0,T)$ and that $f(x,t)$ is sufficiently well-behaved ($f \in L^2((0,T);H^1(0,1))$ ), we ask that our solution to the state equation be such that $y \in L^2((0,T);H^1(0,1))$. Choosing a topology will pose no difficulty in the case of this single space-dimension model; we can define 'close to' more precisely to mean 'close in the $L^2$ norm'. More elaborate machinery must be employed to define a topology in higher space-dimensions. The necessary tools have been developed by Lions and Magenes [100] and by Lions [97] and are the basis for the treatment of control problems in higher dimensions. Constructive algorithms based on these tools are thoroughly examined in Glowinski [65].

There are two explicit goals here: (1) minimizing

$$\|y(x,t) - y_{T_1}\|_{L^2(0,1)}$$

and (2) limiting

$$\|y(x,t) - y_{T_2}\|_{L^2(0,1)} \le \beta_2.$$

The constant $\beta_2$ sets a limit for deviations from the second target.

To further complicate matters, we would like to achieve these goals at minimum cost (i.e., we would like to control as gently as possible). This suggests the problem

$$
\begin{gathered}
\inf_{v_1, v_2 \in L^2(0,T)} \int_0^T (v_0(t)^2 + v_1(t)^2) dt \\
\text{subject to:} \\
\min_{v_1, v_2 \in L^2(0,T)} \|y(x,t) - y_{T_1}\|_{L^2(0,1)} \\
\|y(x,t) - y_{T_2}\|^2_{L^2(0,1)} \leq \beta_2^2.
\end{gathered}
\tag{5.5}
$$

This problem falls into the category of *multicriteria optimization*. In some sense, we can view this problem as having two objective or cost functions,

$$
J_1(v_1) = \frac{1}{2} \int_0^T v_1^2 dt + \frac{\epsilon_1}{2} \|y(T) - y_{T_1}\|^2_{L^2(0,1)} \tag{5.6}
$$

and

$$
J_0(v_0) = \frac{1}{2} \int_0^T v_0^2 dt + \frac{\epsilon_0}{2} \|y(T) - y_{T_0}\|^2_{L^2(0,1)} \tag{5.7}
$$

with two, well-chosen, positive constants, $\epsilon_1$ and $\epsilon_0$.

We follow the lead of Lions [99] who generalized and developed a suggestion of von Stackelburg [146]. The idea is to introduce a hierarchy among the controls. One such hierarchy is to think of our secondary goal as the necessary (hence, more important) goal because it requires only that the state remain in a neighborhood of a target. This aim can be achieved by introducing a constraint

$$
\phi(v_0(t)) - v_1(t) = 0. \tag{5.8}
$$

The specific form of this constraint relies on auxiliary variables introduced later into our optimality system. This matter will be clarified in the next section. It is not hard to anticipate, however, that the constraint will surface as a condition added to our optimality system.

It is interesting that the idea of hierarchical control of dynamical systems originated in attempts to model government and politics. This history led to some unusual nomenclature. We will follow the vocabulary of von Stackelburg and refer to the first control, $(v_1, a_1)$, as the *leader* and the second control, $(v_2, a_2)$, as the *follower*.

We can now write down nonlinear programming problems which approximate our original problem (5.5),

$$
\begin{aligned}
&\min_{v_1, v_2} \int_0^T (v_1^2 + v_2^2) dt + \beta \|y(T) - y_{T_1}\|^2 \\
&\text{subject to:} \\
&\|y(T) - y_{T_1}\| \leq \beta_2 \\
&\phi(v_0(t)) - v_1(t) = 0 \\
&v_1(t), v_2(t) \in L^2(0,T)
\end{aligned}
\tag{5.9}
$$

There is, of course, an implicit constraint which we do not write down, that $y(x,t)$ must satisfy (5.4). If we allow the state variables to be treated as control variables, we arrive at a nonlinear programming problem,

$$\begin{aligned}
&\min_{v_1,v_2,y} \int_0^T (v_1^2+v_2^2)dt + \beta\|y(T)-y_{T_1}\|^2 \\
&\text{subject to:} \\
&y_t - \nu y_{xx} + y y_x = f + v_1(t)\delta(x-a_1) + v_0(t)\delta(x-a_0) \quad \text{in } \ Q \\
&y(x,0) = y_0(x) \quad \text{for } \ x\in(0,1) \\
&y_x(0,t) = y(1,t) = 0 \ \text{ and } \ y_x(0,t) = 0 \quad \text{for all } \ t\in(0,T). \\
&\|y(T)-y_{T_1}\| \le \beta_2 \\
&\phi(v_0(t)) - v_1(t) = 0 \\
&v_1(t), v_2(t) \in L^2(0,T)
\end{aligned} \tag{5.10}$$

We have formulated here a nonlinear programming problem whose solution approximates the solution of our multicritera optimization problem. In a later section, we will comment on the numerical behavior of alternative formulations (for example, minimizing over state variables at the expense of adding the state equation as a constraint).

## 5.3 Calculation of the adjoint equation

To avoid algebraic clutter, we calculate the gradients of our cost functions (5.6) and (5.7) with respect only to the control variables $v_0$ and $v_1$. We will not calculate derivatives with respect to the moving location of the control (our test problems, however, will incorporate this trivial extension). We will assume for the remainder of this work that norms ( written $\|\cdot\|$ ) are $L^2$ norms. The notation $(a\cdot b)$ will be used to denote the inner product of $a$ and $b$.

We begin by recalling (5.4) and calculating the Jacobian of the first cost function with respect to the first control,

$$J_1(v_1) = \frac{1}{2}\int_0^T v_1^2 dt + \frac{\epsilon_1}{2}\|y(T)-y_{T_1}\|^2.$$

To do this, we apply a Green's formula to (5.4) in order to write the *variational form*,

$$\begin{aligned}
\int_0^T (y_t\cdot p)dt \quad &+\nu \int_0^T (y_x\cdot p_x)dt + \int_0^T (y_x y\cdot p)dt \\
&= \int_0^T (f\cdot p)dt + \int_0^T v_0(t)p(a_0,t)dt + \int_0^T v_1(t)p(a_1,t)dt
\end{aligned}$$

in which $p(x,t)$ is yet to be specified. The following variational form, then, applies to a perturbation $\delta y$.

$$\int_0^T (\delta y_t \cdot p)dt \quad +\nu \textstyle\int_0^T (\delta y_x \cdot p_x)dt + \int_0^T (\delta y_x y + \delta y \cdot p)dt$$

$$= \int_0^T v_1(t)p(a_1,t)dt.$$

where we have used the facts that $\delta y(0) = 0$ and $\delta f = 0$. This, in turn, leads us to

$$(\delta y(T) \cdot p(T)) + \nu \int_0^T \Big[ p_x \delta y \mid_0^1 - \nu(p_{xx} \cdot \delta y) + yp\delta y \mid_0^1$$

$$-(y_x p + y p_x . \delta y) + (y_x p \cdot \delta y) ]$$

$$= \int_0^T \delta v_1(t) p(a_1,t)$$

and thus to

$$(p(T) \cdot \delta y(T)) + \int_0^T \left[ -\dot{p} - \nu p_{xx} - y p_x \right] dt + \int_0^T \left[ -\nu p_x(0,t) - y(0,t)p(0,t) \right] \delta y dt$$

$$= \int_0^T p(a_1,t)\delta v_1(t).$$

This leads us to the first component of our optimality conditions,

$$\begin{array}{l} -\dot{p} - \nu p_{xx} + y p_x = 0 \\ p(1,t) = 0 \\ \nu p_x(0,t) + y(0,t)p(0,t) = 0 \\ p(T) = \epsilon_1(y_{T_1} - y(T)) \end{array} \tag{5.11}$$

This implies that

$$(p(T) \cdot \delta y(T)) = \int_0^T p(a_1,t)\delta v_1(t)$$

and

$$(\epsilon_1(y_{T_1} - y(T)) \cdot \delta y(T)) = \int_0^T p(a_1,t)\delta v_1(t)$$

and

$$\delta J_1 = \int_0^T v_1 \delta v_1 dt - \int_0^T p(a_1,t)\delta v_1(t).$$

So that we have

$$v_1 = p(a_1,t) \quad \text{and} \quad \delta v_1 = \delta p(a_1,t). \tag{5.12}$$

We have arrived at the following:

Optimality System for the Follower

$$\begin{array}{l} y_t - \nu y_{xx} + y y_x = v_0 \delta(x - a_0) + p(a_1, t)\delta(x - a_1) \\ y(x,0) = y_0 \\ y_x(0,t) = 0 \\ y(1,t) = 0 \end{array} \tag{5.13}$$

and

$$\begin{array}{l} -p_t - \nu p_{xx} + y p_x = 0 \\ \nu p_x(0,t) + y(0,t)p(0,t) = 0 \\ p(1,t) = 0 \\ p(T) = \epsilon_1(y_{T_1} - y(T)). \end{array} \tag{5.14}$$

We proceed to take account of the second cost function,

$$J_0(v_0) = \int_0^T v_0^2 dt + \frac{\epsilon_0}{2}\|y(T) - y_{T_0}\|^2,$$

with

$$\delta J_0(v_0) = \int_0^T v_0 \delta v_0 dt + \epsilon_0(y(T) - y_{T_0} \cdot \delta y(T))$$

in a manner analogous to that for the first cost function. The variational form in this case is

$$\int_0^T (y_t \cdot P) + \nu \int_0^T (y_x \cdot P_x)dt + \int_0^T (y y_x \cdot P)dt = \int_0^T v_0(t)P(a_0,t)dt + \int_0^T p(a_1,t)P(a_1,t)dt$$

where $P(x,t)$ is a function yet to be determined. In this equation, we have taken into account that $y(0) = 0$ and we have set a boundary condition on $P$, $P(1,t) = 0$.

Differentiating this expression yields

$$\int_0^T (\delta y_t \cdot P)dt + \nu \int_0^T (\delta y_x \cdot P_x)dt + \int_0^T (y_x, \delta y + y\delta y_x \cdot P)dt$$

$$= \int_0^T \delta v_0 P(a_0,t)dt + \int_0^T P(a_1,t)\delta p(a_1,t)dt.$$

This in turn yields a variational equation on $P$.

$$
\begin{aligned}
&(\delta y(T)\cdot P(T)) + \int_0^T \Big[-(P_t\cdot\delta y) + \nu P_x\delta y\ |_0^1 - \nu(P_{xx}\cdot\delta y) + (y_x P\cdot\delta y) \\
&\qquad + yP\delta y\ |_0^1 - (y_x P + yP_x\cdot\delta y)\Big]\,dt \\
&= (P(T)\cdot\delta y(T)) + \int_0^T [-(P_t\cdot\delta y) - \nu P_x(0,t)\delta y(0,t) - \nu(P_{xx}\cdot\delta y) \\
&\qquad - y(0,t)P(o,t)\delta y(0,t) - (yP_x\cdot\delta y)]\,dt] \\
&= (P(T)\cdot\delta y(T)) + \int_0^T (-P_t - \nu P_{xx} - yP_x\cdot\delta y)\,dt \\
&\qquad + \int_0^T [-\nu P_x(0,t) - y(0,t)P(0,t)]\,\delta y(0,t)dt \\
&= \int_0^T P(a_0,t)\delta v_0(t)dt + \int_0^T P(a_t,t)\delta p(a_1,t)dt.
\end{aligned}
$$

Because this equation contains interactions between $p$ and $P$, we will need to introduce yet another function in a variational form. We introduce a function yet to be determined, $Y(x,t)$. We set $Y(1,t)=0$ and we write the variational form

$$
\begin{aligned}
&-\left(\int_0^T (p_t\cdot Y)dt + \nu\int_0^T (p_{xx}\cdot Y)dt + \int_0^T (yp_x\cdot Y)dt\right) \\
&= -\int_0^T (p_t\cdot Y)dt + \int_0^T \Big[-\nu p_x Y\ |_0^1 + \nu(p_x\cdot Y_x) - ypY\ |_0^1 + (p\cdot yY_x + y_x Y)\Big]\,dt \\
&= -\int_0^T (p_t\cdot Y)dt + \nu\int_0^T (p_x\cdot Y_x)dt + \int_0^T (y_x Y\cdot P)dt + \int_0^T (yY_x\cdot P)dt \\
&= 0.
\end{aligned}
$$

where we have used integration by parts. When we differentiate this equation, we get a variational form on the perturbation.

$$-\int_0^T (Y\cdot\delta p_t)dt+\nu\int_0^T (Y_x\cdot\delta p_x)dt+\int_0^T (Y\cdot p\delta y_x+y_x\delta p)dt+\int_0^T (Y_x\cdot p\delta y+y\delta p)dt$$

$$=-(Y(T)\cdot\delta p(T))+(Y(0)\cdot\delta p(0))+\int_0^T (Y_t\cdot\delta p)dt$$

$$+\int_0^T \Big[\nu Y_x\delta p\mid_0^1-\nu(Y_{xx}\cdot\delta p)+(Yy_x\cdot\delta p)+Yp\delta y\mid_0^1-(Y_xp\cdot\delta y)$$

$$-(p_xY\cdot\delta y)+(Y_xp\cdot\delta y)+(Y_xy\cdot\delta p)]\,dx$$

$$=-(Y(T),\delta p(T))+\int_0^T [(Y_t-\nu Y_{xx}+y_xY+yY_x\cdot\delta p)-(p_xY\cdot\delta y)$$

$$-\nu Y_x(0,t)\delta p(0,t)-Y(0,t)p(0,t)\delta y(0,t)]\,dt$$

$$=0$$

We are now in a position to sum this equation with the variational equation on P above and, upon combining terms, we find the following:

$$\int_0^T [(-P_t-\nu P_{xx}-yP_x-p_xY\cdot\delta y)(-Y(0,t)p(0,t)-\nu P_x(0,t)-y(0,t)P(0,t))\delta y(0,t)$$

$$(P(T)\cdot\delta y(T))-(Y(T)\cdot\delta p(T))(Y_t-\nu Y_{xx}+y_xY+y_xY,\delta p)-\nu Y_x(0,t)\delta p(0,t)]$$

$$=\int_0^T P(a_0,t)\delta v_0(t)+\int_0^T P(a_1,t)\delta p(a_1,t).$$

We now specify $Y$ and $P$. For $Y(x,t)$ we require

$$\begin{array}{l}
Y_t-\nu Y_{xx}+(yY)_x=P(a_1,t)\delta(x-a_1)\ \text{ on }\ Q\\
Y(x,0)=0\\
Y_x(0,t)=0\\
Y(1,t)=0
\end{array}$$

and, similarly, for $P(x,t)$ we require

$$\begin{array}{l}
-P_t-\nu P_{xx}-yP_x-p_xY=0\ \text{ on }\ Q\\
\nu P_x(0,t)+y(0,t)P(0,t)+Y(0,t)p(0,t)=0\\
P(1,t)=0\\
P(T)+\epsilon_1Y(T)=\epsilon_0(y_{T_0}-y(T)).
\end{array}$$

We note that
$$((P(T)+\epsilon_1 Y(T))\cdot \delta y(T)) = \int_0^T P(a_0,t)\delta v_0(t)$$
if and only if
$$-(\epsilon_0(y(T)-y_{T_0})\cdot \delta y(T)) = \int_0^T P(a_0,t)\delta v_0(t).$$
This leads to
$$\delta J_0 =< J_0', \delta v_0 >= \int_0^T (v_0 - P(a_0,t))\delta v_0.$$
We can now write the complete optimality system we have derived. It consists of a system of four coupled problems determining the four functions, $y(x,t)$, $p(x,t)$, $Y(x,t)$ and $P(x,t)$.

$$\begin{array}{l}
y_t - \nu y_{xx} + y y_x = P(a_0,t)\delta(x-a_0) + p(a_1,t)\delta(x-a_1)\\
y_x(0,t) = 0\\
y(1,t) = 0\\
y(x,0) = y_0\\
\hline
-p_t - \nu p_{xx} + y p_x = 0\\
\nu p_x(0,t) + y(0,t)p(0,t) = 0\\
p(1,t) = 0\\
p(T) = \epsilon_1(y_{T_1} - y(T))\\
\hline
Y_t - \nu Y_{xx} + (yY)_x = P(a_1,t)\delta(x-a_1)\\
Y_x(0,t) = 0\\
Y(1,t) = 0\\
Y(x,0) = 0\\
\hline
-P_t - \nu P_{xx} - y P_x - p_x Y = 0\\
\nu P_x(0,t) + y(0,t)P(0,t) + Y(0,t)p(0,t) = 0\\
P(1,t) = 0\\
P(T) = \epsilon_1 \left[\dfrac{\epsilon_0}{\epsilon_1}(y_{T_0} - y(T)) - Y(T)\right].
\end{array} \tag{5.15}$$

This coupled system is truly different from any standard 'adjoint' optimality systems. There are really two sets of strongly coupled optimality conditions here.

We remark that the explicit form of our hierarchical constraint promised at (5.8) of the last section can now be given. We have
$$v_1 - \frac{1}{\beta_2} p(v_0) = 0. \tag{5.16}$$

so that

$$\phi(v_0(t)) = \frac{1}{\beta_2} p(v_0(t)) \tag{5.17}$$

where $\beta_2$ is the limit on the displacement of states from the secondary target as described in (5.5).

## 5.4 Discretization of the state equation

We begin by dividing the time interval $(0, T)$ into $N_t$ uniform steps of size $\Delta t = \frac{T}{N_t}$. The various continuous quantities discussed in the previous two sections now have straightforward discrete analogs (e.g. $y(T) \approx y^{N_t}$, $\int_0^T (v_1(t))^2 dt \approx \sum_{i=1}^{N_t} |v_1^i|^2$, and so forth). The integration scheme we employ is nothing more than the *backward Euler* approximation.

Because of the boundary conditions and the variational formulations of our optimality system, we are motivated to introduce the space of functions $M$, such that

$$M = \left\{ w | w, \frac{\partial w}{\partial x} \in L^2(0,1) \text{ and } w = 0 \quad \text{when} \quad x = 1 \right\}. \tag{5.18}$$

Our integration scheme can then be written in the compact form

$$y^0 = y_0$$

$$\text{For} \quad i = 1, \ldots N_t :$$

$$y^{i+1} \in M,$$

$$\int_0^T \frac{1}{\Delta t}(y^{i+1} - y^i) w dt + \nu \int_0^T y_x^{i+1} w_x dt + \int_0^T y^{i+1} y_x^{i+1} dt$$

$$= \int_0^T f^{i+1} w dt + v_1^{i+1} w(a_1) + v_0^{i+1} w(a_0)$$

$$\forall w \in M$$

and

$$p^{N_t+1} = -\epsilon_1(y_{T_1} - y^{i+1})$$

$$\text{For} \quad i = N_t, \dots 1$$

$$p^i \in M,$$

$$\int_0^T \frac{1}{\Delta t}(p^i - p^{i+1})wdt + \nu \int_0^T p_x^i w_x dt + \int_0^T y^i p_x^i w_x dt$$

$$\int_0^T y_x^i p^i w dt = 0 \quad \forall w \in M$$

Obtaining a discrete version of the derivatives of our cost functions is also straightforward.

The space discretization is equally simple. We divide $(0,1)$ into $N_x$ equally spaced subintervals so that $\Delta x = N_x^{-1}$. We can now use the space of functions that are both continuous and linear on each subinterval, call it $M_\Pi$, to be our space of trial functions. These functions vanish (by construction) on the right spatial boundary. The variables $y$ and $w$ are also replaced by discrete analogs. We use perhaps the most common and straightforward of discretizations, polynomials of degree one.

## 5.5 The test problem and NLP formulations

We select a test problem from among those in [41] and modify it appropriately by adding a second target and requiring the solution-state to stay within a specified deviation from it. The targets satisfy the boundary conditions. The performance of a numerical algorithm is examined for different settings of the parameters.

The number of time steps and the number of spatial steps $N_t$ and $N_x$ were chosen to be of the same order of magnitude. Variable but moderate values for the viscosity parameter are taken. In addition, a constraint is applied on the location and the magnitude of the pointwise controls.

Test Problems

- $N_t = 64$, $N_x = 64$ and $T = 1$.
- $\nu = 10^{-2}, 10^{-3}, 10^{-4}$.
- $\epsilon_1 =$ and $\beta_2 =$.

$$f(x,t) = \begin{cases} 1 & x \in [0, \frac{1}{2}) \\ 2(1-x) & x \in [\frac{1}{2}, 1] \end{cases}$$

- $y_0(x) = 0$ for $x \in (0,1)$.

- $y_{T_1}(x) = 1 - x^3$ for $x \in (0, 1)$.
- $y_{T_2}(x) = \cos(\frac{\pi}{2}x)$ for $x \in (0, 1)$.
- $a(t) \in [.2, .6]$
- $v(t) \in [-10, 10]$

Even if a target satisfies the boundary conditions, that fact in no way guarantees that the target can actually be reached through the controls. The nature of Burgers' equation causes the influence of the control to die out downstream, eventually becoming negligible far downstream. The constraint on the location of the controls must be loose enough that their influence can extend to the downstream boundary. This point becomes especially important with small values of $\nu$ when the hyperbolic nature of the model becomes more pronounced.

## 5.6 Numerical results

The details of the algorithm used to solve the test problem of the previous section are found in chapter 2. The results are representative of a number of similar problems solved in the course of this work. Figures 5.1 and 5.3 show clearly that the state variable at time $t = T$ remains in a neighborhood of the second target. While the solutions look quite similar for different values of the viscosity parameter $\nu$, the results summarized in the numerical tables 6.1 and 6.2 show that the amount of work needed was substantially greater for the smaller values of $\nu$ .

The algorithm used (see chapter 2) is an SQP algorithm with an interior point method to solve the quadratic programming problem. A limited-memory BFGS approximation of the Hessian of the Lagrangian function is used at every iteration.

In Table 6.1 and 6.2, we report the numerical behavior of this algorithm. The first two columns define the size of the problem. The second column gives the viscosity parameter $\nu$, while the third and fourth columns give the relative distance of the state from the targets,

$$D(y_{T_1}) = \|y(T) - y_{T_1}\| / \|y(T)\| \tag{5.19}$$

and

$$D(y_{T_2}) = \|y(T) - y_{T_2}\| / \|y(T)\|, \tag{5.20}$$

The parameters $\epsilon_1$ and $\beta_2$ are given in the fifth and sixth columns. The last two columns give the number of nonlinear iterations and the number of quadratic programming iterations needed in solving the problem.

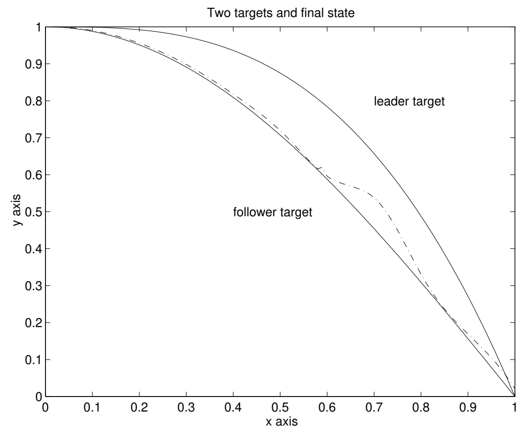


**Figure 5.1** - State profiles at $t = T$

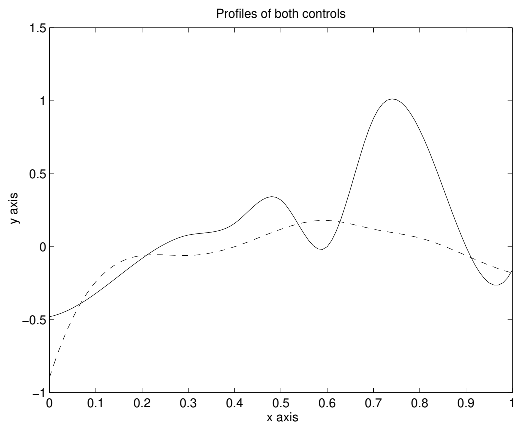


**Figure 5.2** - Profiles of the controls

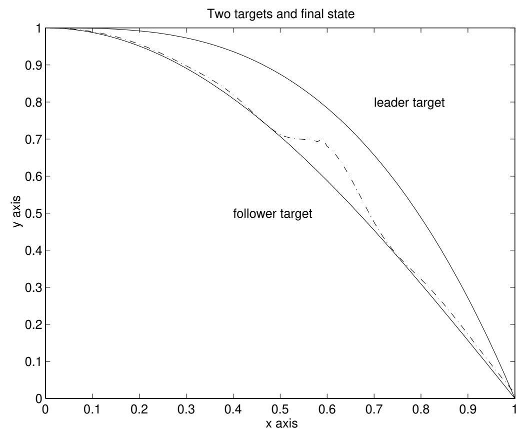


**Figure 5.3** - State profiles at $t = T$

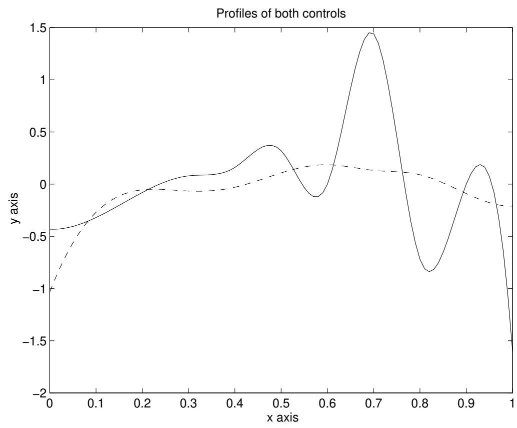


**Figure 5.4** - Profiles of the controls

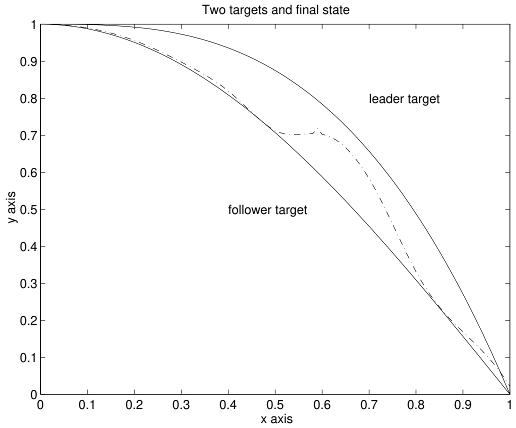


**Figure 5.5** - State profiles at $t = T$

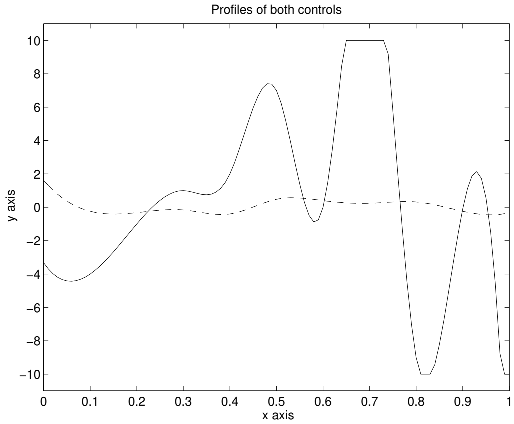


**Figure 5.6** - Profiles of the controls

| $D(y_{T_1})$ | $D(y_{T_2})$ | $\nu$ | NL-iters | QP-iters |
|---|---|---|---|---|
| $9.97e-2$ | $4.04e-2$ | $10^{-2}$ | 61 | 1090 |
| $1.15e-2$ | $2.88e-2$ | $10^{-3}$ | 69 | 1319 |
| $1.58e-2$ | $4.10e-2$ | $10^{-4}$ | 84 | 1675 |

**Table 5.1** Numerical relults for formulation 5.9

## 5.7 Conclusions

The difficulty of solving this hierarchical control problem increases with decreasing values of the viscosity parameter $\nu$. This effect is clear in the performance of the SQP method. It is not surprising in light of the fact that similar behavior has been observed for the simpler case of Burgers' equation with pointwise control from a fixed controller location ([41]).

The freedom to specify multiple targets is extremely important for many practical control problems. Because of the high cost of solving multicriteria optimization problems, however, it is not always clear that this advantage can be realized in an affordable way. The hierarchy employed here yields a useful and quite affordable way.

For the future, more delicate and complicated physical phenomenon should be examined in the context of hierarchical control. We anticipate that these ideas will be fruitful in attacking problems of optimal well placement, contaminant transport, and bioremediation.

| $D(y_{T1})$ | $D(y_{T2})$ | $\nu$ | NL-iters | QP-iters |
|---|---|---|---|---|
| $9.95e-2$ | $4.00e-2$ | $10^{-2}$ | 48 | 871 |
| $1.17e-2$ | $2.91e-2$ | $10^{-3}$ | 51 | 876 |
| $1.51e-2$ | $4.28e-2$ | $10^{-4}$ | 58 | 890 |

**Table 5.2** Numerical results for formulation 5.10

# Chapter 6

# Optimal Control of Melting; A Stefan Control Problem

## 6.1 Introduction

In 1889, J. Stefan proposed and solved the first of a large class of what are now called *Stefan problems*. These problems model the thermal history of a body of material containing a moving boundary between two phases of the material. Stefan's original problem modeled the freezing of soil permeated with water. The water could exist in two phases, liquid or solid, depending upon the effects of conductive heat transfer and a temperature of freezing. There is a *latent heat of fusion* absorbed by the material when it melts, or emitted by the material when it freezes. In physical terminology, this simplified freezing is an isothermal crystallization process without supercooling or volume change. Thus, cooling below freezing temperature at a surface of the liquid material produces an interface or *free boundary* between ice and liquid water which propagates through the material. In Stefan's original problem, the temperature of a half-space of the material is uniform and above freezing at the start. A constant temperature below freezing is imposed at the plane surface of the halfspace. Determining the subsequent propagation of an interface between the frozen and liquid soil is the problem. More elaborate versions of such problems have an obvious practical interest for the design of molds for metals and plastics.

Much mathematical analysis, numerical analysis, and development of algorithms for the solution of various Stephan problems has appeared in the literature since 1889. We cannot attempt even an abbreviated listing. We mention a few distinctive works of immediate pertinence. For a comprehensive review of mathematical analysis applicable to Stefan problems, see J. Crank [38] and Rubinstein [127]. For work on the calculation of dendrites in crystal growth, see the work contained in [147], [148] and [106]. Interesting numerical algorithms have been developed (see, for example, Heinkenschloss and Sachs [83]). Stefan problems remain a very important and fruitful area of research because of the practical applications.

The problem considered here is an optimal control problem for which the state equation is a Stefan problem. Specifically, we optimize the melting of a material by controlling the heat flux along part of the boundary of a two-dimensional domain over which the Stefan problem is defined. In the next section we discuss some specifics of the state equation we use and a transformation we apply which makes the state equation more tractable numerically. Section 6.3 contains an explanation of the control problem we set out to solve and a few comments on related control problems which appear in the literature. The simple discretization scheme we use is explained briefly in section 6.4. Numerical performance on some simple test problems is discussed in section 6.5, along with some conclusions.

## 6.2 The state equation

We will consider a Stefan problem involving the propagation of a surface of melting (rather than freezing) in two space-dimensions . In order to reveal the mathematics, we will nondimensionalize the equations of the Stefan problem. There are four physical properties which, for simplicity, we will take as constants: heat capacity, $C_p$; heat of fusion, $L$; conductivity, $K$; density, $\rho$. To this list we add the mechanical equivalent of heat, $J$, and then construct normalizers from these quantities to non-dimensionalize time, length, and temperature. These normalizers are as follows:

$$\begin{aligned} n_t &= K/(JC_pL\rho) \\ n_l &= K/(L^{1/2}J^{1/2}C_p\rho) \\ n_\theta &= L/C_p \end{aligned} \tag{6.1}$$

where $n_t$, $n_l$ and $n_\theta$ are the normalizers for time, length, and temperature respectively. The nondimensional temperature scale is set to zero at the freezing temperature. All quantities in the discussion that follows are normalized to be dimensionless.

Consider a two-dimensional, rectangular domain $\Omega$ and a temperature $u = u(x, y, t)$ defined on $\Omega$ where $\Omega = (0, l_x) \times (0, l_y)$ and $0 \leq t \leq T$. The heat fluxes along the northern and eastern boundaries of this domain are identically zero; the heat fluxes along southern and western boundaries are variable. The variable heat fluxes along two quarters of the boundary of $\Omega$ will be the control quantities, $f_s(y, t)$ and $f_w(x, t)$. We are concerned with the evolution of the temperature on $\Omega$ from time, $t = 0$ to $t = T$; we define the space-time domain $Q = (0, T) \times \Omega$. We consider only a *single-phase Stefan problem*, that is, transition between phases occurs at a definite temperature and, in determining the freezing interface, it is sufficient to consider only one phase. (In problems with supercooling, a *two-phase Stefan problem* results.) The problem we shall consider here, when expressed using these definitions, is the following single-phase Stefan problem:

$$u_t(x, y, t) - (u_{xx}(x, y, t) + u_{yy}(x, y, t)) = 0 \quad \text{on} \quad Q, \tag{6.2}$$

with boundary conditions

$$\begin{aligned} &u_x(0, y, t) = f_s(y, t) \quad \text{for all} \quad t \in [0, T] \\ &u_x(l_x, y, t) = 0 \quad \text{for all} \quad t \in [0, T] \\ &u_y(x, 0, t) = f_w(y, t) \quad \text{for all} \quad t \in [0, T] \\ &u_y(x, l_y, t) = 0 \quad \text{for all} \quad t \in [0, T] \end{aligned} \tag{6.3}$$

and initial conditions

$$u(x, y, 0) = u_0 = 0 \quad \text{on} \quad \Omega. \tag{6.4}$$

Points on the interface between the two phases (liquid and solid) are roots of an interface function

$$I(x, y, t) = 0.$$

A dimensionless parameter appears in the problem which we call the *Stefan number* and denote by $S$, where $S = \frac{C_p}{n_\theta L}$. The resulting system of heat balance equations can now be written

$$\begin{aligned} &u_x(x, y, t) I_x(x, y, t) + u_y(x, y, t) I_y(x, y, t) - \frac{1}{S} I_t = 0 \\ &u(x, y, t) = 0 \\ &I(x, y, t) = 0. \end{aligned} \tag{6.5}$$

When viewed as a necessary equality constraint, (6.5) proved prohibitively expensive to solve numerically. An alternative formulation was sought and that of Duvaut [48] was chosen.

Since the temperature at any point is temporally non-decreasing, we can define a temperature characteristic function $U(x, y, t)$ by

$$U(x, y, t) = \begin{cases} \int_\tau^t u(x, y, t) dt & \text{if } t > \tau \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

where $\tau = \tau(x, y)$ is the time at which the material at location $x, y$ changes phase from solid to liquid.With this function, the Stefan problem is transformed to read

$$U_t(x, y, t) - (U_{xx}(x, y, t) + U_{yy}(x, y, t)) + \frac{1}{S} \chi_{t-\tau}(x, y) = 0 \tag{6.7}$$

with boundary conditions

$$\begin{aligned} &U_x(0, y, t) = \int_\tau^t f_s(y, t) dt \\ &U_x(l_x, y, t) = 0 \\ &U_y(x, 0, t) = \int_\tau^t f_w(y, t) dt \\ &U_y(x, l_y, t) = 0 \end{aligned} \tag{6.8}$$

and initial conditions

$$U(x, y, 0) = 0. \tag{6.9}$$

The special characteristic function $\chi_{t-\tau}$, appearing on the right hand side of (6.7), is defined by

$$\chi_{t-\tau}(x, y) = \begin{cases} 1 & \text{if } t \leq \tau \quad \text{at} \quad (x, y) \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

This transformation is sometimes called the *Baiocchi-Duvaut transformation.* A complete and detailed account can be found in the paper by Duvaut [48]. The interested reader is also referred to Elliot and Ockendon [55] where the system (6.7)–(6.9) is analyzed and conditions guaranteeing existence and uniqueness of a solution are established.

The transformed formulation has many interesting features not shared by (6.2-(6.4). The following two advantages of the Duvaut transformation invite its use:

1. the possibility of solving for the interface $U(x, y, t)$ implicitly
2. the availability of numerical methods for the solution of parabolic variational inequalities. (A comprehensive description of numerical methods for variational inequalities is found in Glowinski [65].)

## 6.3 The control problem

We begin this section by reviewing pertinent studies of Stefan problems of the type we are concerned with here. The control of a two-phase Stefan problem in one space-dimension with 'on - off' control is analyzed by Hoffman and Sprekels [87]. Their objective is control of the solid-liquid interface, keeping it within some neighborhood of a specified target. They present the results of numerical experiments and a strong analysis of the mathematics of their problem.

Pawlow [114] considers the control of a two-phase Stefan problem in two space-dimensions. An $L_2$ cost function is constructed from the target solid-liquid interface. By coupling the steepest descent methods for minimization with the use of an adjoint parabolic variation inequality to calculate gradients, Pawlow achieves very promising numerical results.

A problem close to the problem we consider is found in Silva–Neto and White [110]. A single-phase Stefan problem in two space-dimensions is controlled to maximize the amount of melting of the material through heat flux applied to the boundary. They use a sophisticated Levenberg - Marquardt algorithm to approximate the solution of this problem. The scheme incorporates vectorization and pre-processing to speed up all calculation. It is a small problem in the sense that few controls are allowed.

We now describe the objective function and the constraints on the controls for the two space-dimensional Stefan problem we shall consider. The aim is to control the amount of material melted at time $t = T$ to be as close as possible to a prescribed quantity. Consequently, the objective function will be very close to those in [114] and [86]. The fluxes, considered as functions of time, are constrained by bounds from above and below (as they are in [110]). The maximum and minimum of the boundary integral of the fluxes are also limited. Most significantly, the Levenberg-Marquardt algorithm of [110] is replaced by a *Sequential Quadratic Programming* (SQP) algorithm in solving the nonlinear programming problem (NLP).

Let $A(f_n, f_w, U(x,y,T))$ denote the area in the liquid state at time $T$ and let $A_T$ denote the target area of liquid at time $T$. Our objective function is

$$\min_{f_n, f_w} |A_T - A(f_n, f_w, U(x,y,T))|^2. \tag{6.11}$$

The following constraints are imposed on the control variables (the flux functions):

$$\begin{aligned} &0 \le -f_n(x,t), -f_w(y,t) \le f_B \\ &\int_0^T \int_0^{l_x} -f_n(x,t)dxdt \le F_X \\ &\int_0^T \int_0^{l_y} -f_w(y,t)dydt \le F_Y. \end{aligned} \tag{6.12}$$

Here, $U(x,y,t)$ is implicitly assumed to satisfy (6.7)–(6.9) on the time-space domain $Q$.

Another formulation is attained by taking the state variables to be control variables,

$$\min_{f_n, f_w, U} |A_T - A(f_n, f_w, U(x,y,T))|^2. \tag{6.13}$$

subject to the constraints,

$$\begin{aligned} &U_t(x,y,t) - (U_{xx}(x,y,t) + U_{yy}(x,y,t)) + \frac{1}{S}\chi_{t-\tau}(x,y) = 0 \\ &U_x(0,y,t) = \int_\tau^t f_s(y,t)dt \\ &U_x(l_x,y,t) = 0 \\ &U_y(x,0,t) = \int_\tau^t f_w(y,t)dt \\ &U_y(x,l_y,t) = 0 \\ &U(x,y,0) = 0 \\ &0 \le -f_n(x,t), -f_w(y,t) \le f_B \\ &\int_0^T \int_0^{l_x} -f_n(x,t)dxdt \le F_X \\ &\int_0^T \int_0^{l_y} -f_w(y,t)dydt \le F_Y. \end{aligned} \tag{6.14}$$

The integral constraints as well as the simple bounds can be handled in a straightforward manner after discretizing.

## 6.4 Discretizing the problem

There are two issues that must be addressed here: (1) discretizing the state equation (2) discretizing the control problem.

Finite differences are used to discretize in a standard way. Let $N_x$ and $N_y$ be the number of spatial steps in the $x$ and $y$ directions, respectively. Denote the number of time steps by $N_t$. Uniform steps are taken for both the spatial and the time discretizations

$$\Delta x = \frac{l_x}{N_x}, \quad \Delta y = \frac{l_y}{N_x}, \quad \text{and } \Delta t = \frac{T}{N_t}. \tag{6.15}$$

The resulting sequence of elliptic variational inequalities (see [55]) are to be solved at each time step. A five-point finite-difference stencil is used for the spatial variables. A fully implicit time-discretization is employed for the solution of (6.7) - (6.9).

At each time step, the resulting discrete problem

$$\begin{aligned}
&\frac{1}{\Delta t}\left(U(x,y,t) - U(x,y,t-\Delta t)\right) - \left(U_{xx}(x,y,t) + U_{yy}(x,y,t)\right) + \frac{1}{S}\chi_{t-\tau}(x,y) = 0 \\
&U_x(0,y,t) = \int_\tau^t f_s(y,t)dt \\
&U_x(l_x,y,t) = 0 \\
&U_y(x,0,t) = \int_\tau^t f_w(y,t)dt \\
&U_y(x,l_y,t) = 0
\end{aligned} \tag{6.16}$$

can be posed as a quadratic programming problem with simple constraints, non-negativity bounds on the variables. The matrix in the quadratic objective function is the standard block-pentadiagonal matrix associated with (6.16). An interior-point method developed by Boggs, Domich, Rogers and Witzgall [10] then solves this quadratic programming problem.

The discretization of the control problem boils down to the discretization of the constraint functions. We approximate (6.12) by

$$\begin{aligned}
&0 \leq -f_n(x^i,t), -f_w(y^j,t) \leq f_B \quad \text{for} \quad i = 1, \ldots N_x, j = 1, \ldots N_y \\
&\sum_{k=1}^{N_t}\sum_{i=1}^{N_x} -f_n(x^i,t^k)\Delta x \Delta t \leq F_X \\
&\sum_{k=1}^{N_t}\sum_{j=1}^{N_y} -f_w(y^j,t^k)\Delta y \Delta t \leq F_Y.
\end{aligned} \tag{6.17}$$

The minimization problem arising in the course of approximating the solution of the control problem is solved using the Sequential Quadratic Programming algorithm of the second chapter. Derivatives of the objective function are calculated using a combination of finite- differences and adjoint-gradients. The derivative of the solution with respect to the controls can be calculated by solving an adjoint equation, but the derivative of the liquid area function $A$ with respect to the temperature function $U$ is calculated using a finite-difference approximation.

We emphasize that quadratic programs are solved at two junctures:

1. as part of the calculation of the search direction for the constrained minimization of (6.11) subject to (6.17) and

2. in the course of the solution of the state equation, (6.16).

## 6.5 Numerical results and conclusions

The (SQP) algorithm from chapter 2 is applied in the solution of our control problem. The performance of this numerical technique is evaluated on a set of problems of various sizes.

Test Problem

- $l_x = l_y = 1$ and $T = .5$
- $N_x = N_y = 16, 32$ and $N_t = 8, 16$
- $C_p =$ and $L =$
- $A_T = 1\frac{1}{3}$ $(= 33\%)$ and $A_T = 2$ $(= 50\%)$

It appears from the graphs of our state equations at time $T$, that the control was successful (see the figure 6.1 and 6.5). Because of the geometry of the problem, the flux control along each of the two axes is very similar. In figures 6.5 and 6.4, the scaled heat fluxes at $x_1 = .5$ and $x_2 = .5$ are plotted as functions of time. The scaling factor is the inverse of the flux element of largest magnitude.

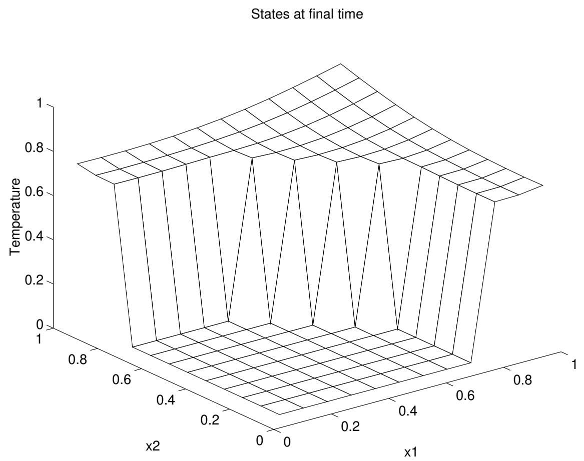


**Figure 6.1** State variables $U$ at time $t = T$

The summary table specifies the size of the problem in the first column. The simple bounds and integral bounds are shown in the second and third columns. The fourth and fifth columns report the number of nonlinear and quadratic iterations needed for the minimization process. (The number of quadratic iterations does not

| Target | $\nu$ | NL-iters | QP-iters | $A$ |
|---|---|---|---|---|
| 33 % | $10^{-2}$ | 61 | 1090 | 33.3 % |
| 33 % | $10^{-3}$ | 69 | 1319 | 33.3 % |
| 33 % | $10^{-4}$ | 84 | 1675 | 33.7 % |
| 50 % | $10^{-2}$ | 59 | 1117 | 51.0 % |
| 50 % | $10^{-3}$ | 69 | 1294 | 51.3 % |
| 50 % | $10^{-4}$ | 81 | 1496 | 51.7 % |

**Table 6.1** Numerical results for formulation 5.9

| Target | $\nu$ | NL-iters | QP-iters | $A$ |
|---|---|---|---|---|
| 33 % | $10^{-2}$ | 59 | 1058 | 33.0 % |
| 33 % | $10^{-3}$ | 59 | 1093 | 33.3 % |
| 33 % | $10^{-4}$ | 67 | 1201 | 33.3 % |
| 50 % | $10^{-2}$ | 51 | 971 | 51.0 % |
| 50 % | $10^{-3}$ | 58 | 1197 | 51.0 % |
| 50 % | $10^{-4}$ | 63 | 1200 | 51.3 % |

**Table 6.2** Numerical results for formulation 5.10

include the quadratic program solutions needed to satisfy the state equation.) In the next-to-last column we record the value of

$$B = \int_0^{l_x} \int_0^{l_y} \left( U(x,y,T) - U(x,y,T-\Delta t) + \frac{\Delta t}{C_S} \right) dxdy - \Delta t \int_{\partial\Omega} \frac{dU}{dn} d\partial\Omega. \qquad (6.18)$$

The quantity $\frac{dU}{dn}$ denotes the component of the gradient of the state variable $U$ in the direction of the outward-pointing normal. When the heat balance equation is satisfied, this quantity $B$ should be close to zero at every time step. (One can see this by applying Green's theorem to (6.16).) An earlier version of our control problem included this requirement as a constraint. However, the solutions with and without this constraint were indistinguishable from each other. We give the value of $B$ at the last time step. This quantity was monitored during all simulations; it was always smaller than $10^{-3}$. In the last column we give the value of the liquid-filled area, the quantity controlled to be near the target $A_T$.

The minimization procedure here is a viable technique for solving Stefan problems of the type considered. It allows a larger number of control variables than, for instance, [110]. Parallel variants of this algorithm with more sophisticated discretization schemes have yet to be explored.

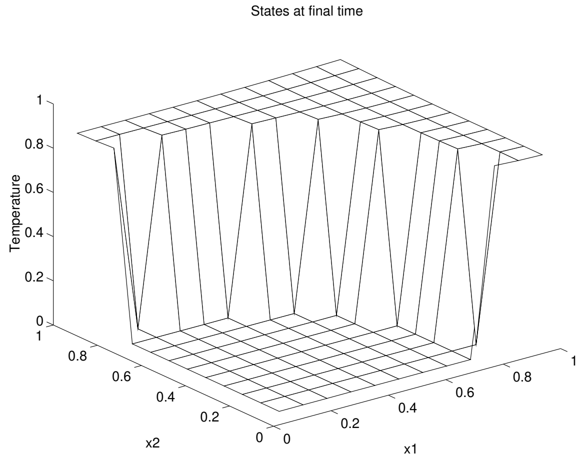


**Figure 6.2** State variables $U$ at time $t = T$

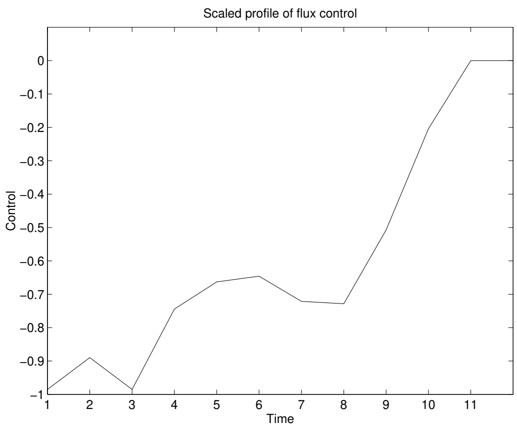


**Figure 6.3**
Profiles of the controls

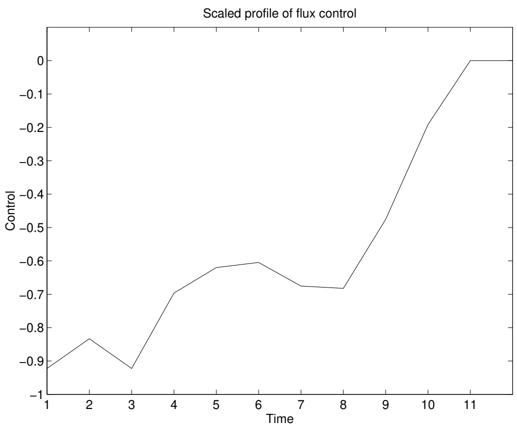


**Figure 6.4**
Profiles of the controls

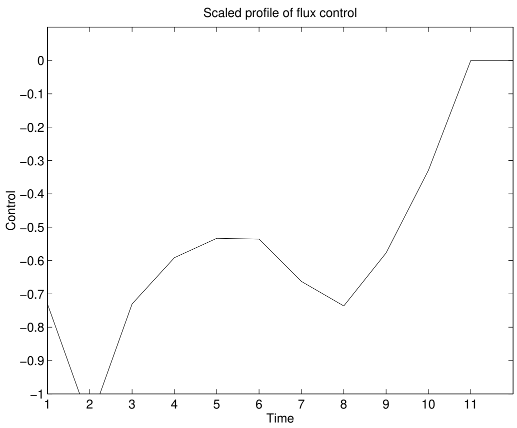


**Figure 6.5**
Profiles of the
controls

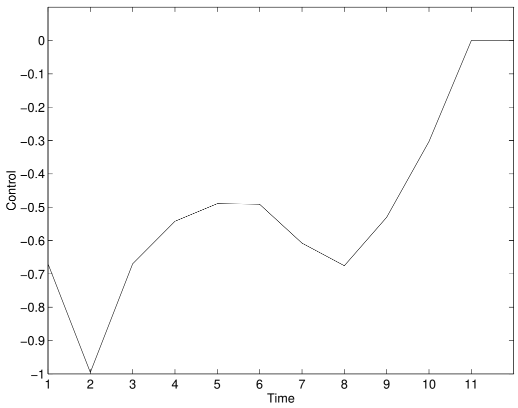


**Figure 6.6**
Profiles of the
controls

# Chapter 7

# Simulation and Control of Dynamical Systems with Dry Friction

## 7.1 Introduction

In this paper we discuss the simulation and control of some elasto-dynamic systems with dry friction. The phenomenon of dry or *Coulomb* friction has been described and analyzed in the comprehensive book by Kikuchi and Oden [95] and in the paper by Campos, Oden, and Kikuchi [22] (see also the interesting paper by Renardy [120]). In Cabannes [21], Coulomb friction is analyzed in the motion of a string. These works all model physical situations through problems of time-dependent variational inequalities (see the book by Duvaut and Lions [49]). The spatial semi-discretization of these problems gives rise to systems like the one examined here.

We consider the simple time-dependent problem

$$M\ddot{x} + Ax + C\lambda = f, \quad t \in (0, T], \tag{7.1}$$

$$x(0) = x_0, \quad \dot{x}(0) = x_1, \tag{7.2}$$

$$\lambda_i(t) = 0 \text{ if } \quad c_{ii} = 0, |\lambda_i(t)| \leq 1 \text{ if } \quad c_{ii} > 0, \text{ and } \quad C\lambda(t) \cdot \dot{x}(t) = \sum_{i=1}^{d} c_{ii} |\dot{x}_i(t)|. \tag{7.3}$$

We are using the standard inner product of $\Re^d$ (i.e., for $y, z \in \Re^d$, $y \cdot z = \sum_{i=1}^{d} y_i z_i$). In this system, $x_i(t)$ denotes the displacement of the $i$th component at time $t$. The vectors $x, \dot{x}, \ddot{x}, \in \Re^d$ represent displacement, velocity, and acceleration, respectively. Here, $\dot{x} = \frac{dx}{dt}$ and $\ddot{x} = \frac{d^2x}{dt^2}$. Also, $x_0, x_1 \in \Re^d$ are the initial displacement and velocity, respectively. The mass matrix $M \in \Re^{d \times d}$ is a symmetric and positive definite matrix. The stiffness matrix, $A \in \Re^{d \times d}$ is symmetric and positive semi-definite. The friction matrix $C \in \Re^{d \times d}$ is diagonal with only non-negative diagonal entries. Matrices $M$, $A$ and $C$ are all assumed to be constant with respect to time $t$. The function $f$ models the external forces acting on the system. We will assume that $f \in L^2(0, T; \Re^d)$. Finally, $\lambda \in L^\infty(0, T; \Re^d)$ is the normalized coefficient of Coulomb friction.

The system described in (7.1)–(7.3) is equivalent to a second-order time-dependent variational inequality whose solution requires finding a function $x \in H^2(0, T; \Re^d)$ such that

$$\int_0^T (M\ddot{x} + Ax - f) \cdot (y - \dot{x}) dt + \int_0^T (j(y) - j(\dot{x})) dt \geq 0, \qquad \forall y \in L^2(0, T; \Re^d), \tag{7.4}$$

$$x(0) = x_0, \qquad \dot{x}(0) = x_1. \tag{7.5}$$

Here, we have

$$j(y) = \sum_{i=1}^{d} c_{ii}|y_i|, \qquad \forall y \in \Re^d. \tag{7.6}$$

The space $H^2(0, T; \Re^d)$ is a Hilbert space with associated inner product

$$< y, z >= \sum_{j=0}^{2} \int_0^T \frac{d^j y}{dt^j} \cdot \frac{d^j z}{dt^j} dt. \tag{7.7}$$

The functional $j(\cdot)$ is convex and continuous. It is also non-smooth, unless $c_{ii} = 0$ for all $i = 1, 2, ...d$.

We do not prove existence and uniqueness of the solution of (7.1)–(7.3) here. That can be done using using the methods from [49].

## 7.2 Simulation

Let $N, \Delta t, T$ be positive scalars denoting the number of time steps in the discretized problem, the length of the time step, and the final time, respectively. Thus, we have $\Delta t = \frac{T}{N}$. We seek an approximate solution, $x^n$, to problem (7.1) –(7.3) for every $n\Delta t$. Similarly, we approximate $f(n\Delta t)$ by $f^n$. We will assume that $f$ is smooth ($f \in C^0([0, T]; \Re^d)$) so that we can take $f^n = f(n\Delta t)$. We then approximate system (7.1)– (7.3) by

$$M\frac{1}{\Delta t^2}(x^{n+1} + x^{n-1} - 2x^n) + A(\alpha x^{n+1} + \alpha x^{n-1} + (1-2\alpha)x^n) + C\lambda^n = f^n, \quad \forall n \geq 0; \tag{7.8}$$

$$\lambda_i^n = 0 \ \text{ if } \ c_{ii} = 0, \tag{7.9}$$

$$|\lambda_i^n| \leq 1 \ \text{ and } \ \lambda_i^n(x_i^{n+1} - x_i^{n-1}) = |x_i^{n+1} - x_i^{n-1}| \ \text{ if } \ c_{ii} > 0, \tag{7.10}$$

$$x^0 = x_0 \ \text{ and } \ x^1 - x^{-1} = 2x_1\Delta t, \tag{7.11}$$

$$\lambda_i^0 = 0 \ \text{ if either } \ c_{ii} = 0 \ \text{ or } \ \dot{x}_i^0(0) = 0 \ \text{ and } \ c_{ii} > 0, \tag{7.12}$$

$$\lambda_i^0 = sign(\dot{x}_i(0)) \ \text{ if } \ \dot{x}_i(0) \neq 0 \ \text{ and } \ c_{ii} > 0. \tag{7.13}$$

This discretization, (7.8)–(7.13), has been investigated in [40] for $\alpha \in [0, \frac{1}{2}]$. Stability properties of the scheme were established. Some convergence properties such as the following theorem have also been established.

**Theorem 7.1** *[Dean, Glowinski, Kuo, Nasser]*

*Suppose $(x, \lambda)$ is the solution of (7.8)–(7.13). Let $\lambda_{\Delta t}$ be the function of $t$ defined by*

$$\lambda_{\Delta t}(t) = \lambda^0 \quad when \quad t \in [0, \frac{\Delta t}{2}) \tag{7.14}$$

$$\lambda_{\Delta t}(t) = 0 \quad when \quad t \in (T - \frac{\Delta t}{2}, T] \tag{7.15}$$

$$\lambda_{\Delta t}(t) = \lambda^n \quad when \quad t \in (n\Delta t - \frac{\Delta t}{2}, n\Delta t + \frac{\Delta t}{2}). \tag{7.16}$$

*Then (7.8)–(7.13) is an unconditionally stable scheme if* $\alpha \in [\frac{1}{4}, \frac{1}{2}]$ *and the following convergence results hold:*

1. $\lim_{\Delta t \to 0} \max_{0 \le n \le N} \|x^n - x(n\Delta t)\| = 0,$

2. $\lim_{\Delta t \to 0} \max_{0 \le n \le N-1} \|\frac{1}{\Delta t}(x^{n+1} - x^n) - \dot{x}((n + \frac{1}{2})\Delta t)\| = 0,$

3. $\lim_{\Delta t \to 0} \lambda_{\Delta t} = \lambda$, *weakly * in* $L^\infty(0, T; \Re^d)$.

**Remark 2.1:** If $\alpha \in [0, \frac{1}{4})$, the above convergence results still hold, the stability condition in that case being

$$\Delta t < [(\frac{1}{4} - \alpha)\delta_M]^{-\frac{1}{2}}$$

with $\delta_M$ the largest eigenvalue of the matrix $M^{-1}A$.

Following [39], we can solve the nonlinear system (7.8)– (7.13) by observing that the conditions on $\lambda$ in (7.8)–(7.13) can be rewritten as

$$\lambda^n = P_\Lambda\left(\lambda^n + \frac{r}{2\Delta t}C(x^{n+1} - x^{n-1})\right), \quad \forall r \ge 0, \quad \forall n \ge 0. \tag{7.17}$$

The projector $P_\Lambda : \Re^d \to \Lambda$, is the orthogonal projection operator from $\Re^d$ onto the set $\Lambda$, where $\Lambda$ is a closed convex set defined by

$$\Lambda = \{\lambda : \lambda \in \Re^d, |\lambda_i| \le 1 \quad \text{and} \quad \lambda_i = 0 \quad \text{if} \quad c_{ii} = 0\}.[ \tag{7.18}$$

We have for all $\mu \in \Re^d$, $P_\Lambda(\mu) = z$ with $z_i = 0$ if $c_{ii} = 0$ and $z_i = \min\{1, \max(\mu_i, -1)\}$ if $c_{ii} > 0$.

Numerical solutions of the class of problems defined by (7.8)– (7.13) can be found by using techniques developed and described in [Glo84].

In the particular case of problem (7.8)–(7.13) and for $\alpha = \frac{1}{4}$, these techniques yield the following algorithm for computing the pair $(x^{n+1}, \lambda^n)$:

$$\lambda^{n,0} \in \Lambda; \tag{7.19}$$

For $k \geq 0$, $\lambda^{n,k}$ being known in $\Lambda$, solve the linear system

$$(M + \frac{\Delta t^2}{4} A) x^{n+1,k} = \Delta t^2 (f^n - C\lambda^{n,k}) + M(2x^n - x^{n-1}) - \frac{\Delta t^2}{4} A(2x^n + x^{n-1}) \quad (7.20)$$

Then, update $\lambda^{n,k}$ to

$$\lambda^{n,k+1} = P_\Lambda \left( \lambda^{n,k} + \frac{r}{2\Delta t} C(x^{n+1,k} - x^{n-1}) \right). \quad (7.21)$$

Stop iterating when

$$\frac{\|x^{n+1,k} - x^{n+1,k-1}\|}{\|x^{n+1,k-1}\| + 1} \leq \eta \quad (7.22)$$

for some well chosen norm $\|\cdot\|$ and parameter $\eta$.

The algorithm (7.19)–(7.22) applied to the solution of system (7.8)–(7.13), for test problem (7.1)-(7.3), the numerical shows very good convergence properties for the approximate displacement and velocity in numerical experiments (see Figures 7.1,7.2,7.5,7.6). However, the approximate multiplier $\lambda_{\Delta t}$ displays violent oscillations (shown in Figures 7.5 and 7.8). This behavior is compatible with the convergence Theorem 7.1 which only guarantees weak convergence for $\lambda_{\Delta t}$. Some applications are more sensitive to the friction forces and demand more accurate discrete multipliers. It is to meet this need that a regularization procedure was developed in [39]. The procedure is based on the observation that equation (7.21) is a discrete form of

$$\lambda = P_\Lambda (\lambda + rC\dot{x}), \quad (7.23)$$

It regularizes (7.23) via

$$\epsilon\dot{\lambda} + \lambda = P_\Lambda(\lambda + rC\dot{x}) \quad (7.24)$$

with $\epsilon > 0$, and discretizes (7.24) to get

$$\epsilon \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta t} \right) + \lambda^n = P_\Lambda \left( \lambda^n + rC \frac{x^{n+1} - x^{n-1}}{2\Delta t} \right). \quad (7.25)$$

The procedure can be viewed as a dynamical Tychonoff regularization procedure. The relation (7.24) clearly implies that the regularized multiplier is a Lipschitz-continuous function of $t$. From this it can be shown that when

$$\lim_{\Delta t \to 0} \frac{\epsilon}{\Delta t} = 0,$$

the convergence results of Theorem 2.1 still hold.

In light of (7.25), we shall use the regularized variant of algorithm (7.19)-(7.22) obtained by replacing (7.21) with

$$\lambda^{n,k+1} = \frac{\Delta t}{\epsilon + \Delta t} P_\Lambda \left( \lambda^{n,k} + \frac{r}{2\Delta t} C(x^{n+1,k} - x^{n-1,k}) \right) + \frac{\epsilon}{\epsilon + \Delta t} \lambda^{n-1}. \quad (7.26)$$

Of course, the parameter $\epsilon$ must be chosen so that $\lim_{\Delta t\to 0}\frac{\epsilon}{\Delta t}=0$.

The above regularization procedure has a double effect

- For a fixed value of $r$, it substantially improves the convergence of the algorithm (7.19)-(7.22).
- It improves the convergence of $\lambda_{\Delta t}$ to $\lambda$ without affecting the convergence of the approximate displacement and velocity to their respective limits. This is seen in Table 5. Indeed, from the figures and Table 5 we conjecture that the weak-* convergence of the discrete multipliers has been replaced by a strong convergence of the discrete regularized multipliers $L^s(0,T;\Re^d)$ for all $s\in[1,+\infty)$.

Let $\delta$ be the largest eigenvalue of the matrix $([M+\frac{\Delta t^2}{4}A]^{-1}C^2)$. Numerical observations suggest that a value of $r\approx\frac{2}{\delta\Delta t}$ yields the best results. While this value of $r$ appears optimal, $r\in(0,\frac{4}{\delta\Delta t})$ is necessary to guarantee the important fixed-point property that proves convergence of the pair $(x^{n+1,k},\lambda^{n,k})$.

## 7.3 Control

There are many practical applications which require controlling the evolution in time of the system modeled by (7.1)– (7.3). In particular, we may require $x$ and/or $\dot{x}$ to arrive at assigned values at some final time $t=T$. This target is especially applicable to the Coulomb friction problem because equilibrium can be reached for finite $T$ provided the external forces, $f$, are sufficiently smaller than the friction forces, $C\lambda$. The following theorem expresses this fact.

**Theorem 7.2** *Given the system (7.1)–(7.3), assume*

- **A1** $c_{ii}>0$, $\forall i=1,2.,d$,
- **A2** $\|f_i\|<c_{ii}$ $\forall i=1,2.,d$,
- **A3** *$A$ is identically* $0$.

*Then the system described by (7.1)–(7.3) reaches steady state in finite time.*

**Proof** Suppose that $\dot{x}(t)\neq 0$, for every $t\in(0,T]$. Multiply (7.1) by $\dot{x}(t)$ and use **A3** to obtain

$$(M\ddot{x}(t)+C\lambda)\cdot\dot{x}(t)=f\cdot\dot{x}(t).$$

Now, using **A2** and $\lambda\in\Lambda$, we have

$$\frac{1}{2}\frac{d}{dt}(M\dot{x}(t)\cdot\dot{x}(t))+\sum_{i=1}^{d}(c_{ii}-\|f_i\|_\infty)\dot{x}(t)\leq 0. \tag{7.27}$$

Since $M$ is a positive definite matrix,

$$M\dot{x}(t) \cdot \dot{x}(t) > 0. \tag{7.28}$$

Pick a positive scalar, $\beta$, such that for any $z \in \Re^d$, we have

$$\beta(Mz \cdot z)^{1/2} \leq \sum_{i=1}^{d}(c_{ii} - \|f_i\|_\infty)|z_i| \tag{7.29}$$

This implies that

$$\frac{d}{dt}(M\dot{x}(t) \cdot \dot{x}(t))^{1/2} + \beta \leq 0. \tag{7.30}$$

We can now integrate and, on recalling the initial conditions from (7.2) and (7.3)

$$(M\dot{x}(t) \cdot \dot{x}(t))^{1/2} + \beta t \leq (M\dot{x}(0) \cdot \dot{x}(0))^{1/2} = (Mx_1 \cdot x_1)^{1/2}. \tag{7.31}$$

This is impossible, so there must be an equilibrium. In fact, if we define $t^* = \frac{1}{\beta}(Mx_1 \cdot x_1)^{1/2}$ we notice that $\dot{x}(t^*) = 0$. So for any $t \geq t^*$, the solution of (7.1)–(7.3) is

$$x(t) = x_0 + \int_0^{t^*} \dot{x}(\tau)d\tau, \tag{7.32}$$

where

$$\lambda(t) = C^{-1}f(t) \tag{7.33}$$

and $\dot{x}(t) = 0$; thus the system is in a position of equilibrium $\forall t \in [t^*, T]$.

□

These ideas can certainly be applied to the control of remote manipulator systems where final velocities must be kept small. We choose target velocity $\dot{x}_T$, and target position $x_T$. We employ a cost-function using penalty terms to attain target states. The resulting problem will look as follows:

$$M\ddot{x} + Ax + C\lambda = f + Bv, \qquad t \in (0, T], \tag{7.34}$$

$$x(0) = x_0, \qquad \dot{x}(0) = x_1, \tag{7.35}$$

$$\lambda_i(t) = 0 \ \text{ if } \ c_{ii} = 0, \qquad |\lambda_i(t)| \leq 1 \ \text{ if } \ c_{ii} > 0, \ \text{ and } \ C\lambda(t) \cdot \dot{x}(t) = \sum_{i=1}^{d} c_{ii}|\dot{x}_i(t)| \tag{7.36}$$

where we want to solve

$$\min_{v \in \Re^d} J(v) = \int_0^T Nv \cdot v dt \tag{7.37}$$

*subject to:*

$$x(T) = x_T \tag{7.38}$$

$$\dot{x}(T) = \dot{x}_T. \tag{7.39}$$

The vector, $v \in \Re^p$, is the vector of control variables. The matrix $B \in \Re^{d\times p}$ will dictate how the control is administered to the system. Likewise, the matrix $N \in \Re^{p\times p}$, may be different from the identity if some parts of the system are more difficult to control than others. For simplicity, we will assume that $p = d$ and that both $B$ and $N$ are multiples of the identity matrix. The algorithm used to solve this *nonsmooth* nonlinear programming problem (7.37)–(7.39) will be discussed in section 4.

## 7.4 Optimization technique

Problem (7.37)–(7.39) is solved by equality-constrained nonlinear programming, but a serious difficulty arises from the fact that the constraints *are not* differentiable with respect to the control variables, $v$. This fact prohibits the use of very fast algorithms (e.g. see [64]). We must expect that at the solution of (7.37)–(7.39), say $v^*$, the constraints will not be differentiable with respect to $v$. In fact, whenever the velocity of any of the components of our dynamical system vanishes, the derivative with respect to $v$ will not exist. For this reason we employ an algorithm that uses no information from the derivative.

In [45] Dennis and Torczon suggested an algorithm was suggested for unconstrained optimization that requires no information from derivatives. While such algorithms for optimization (usually referred to as *direct search* methods) are not new (see, for example, [24] and the references therein), a global first-order stationary point convergence theory for this particular method has only recently been developed by Torczon in [143]. A parallel implementation of the algorithm has also been developed and tested by Torczon (see [142]). Recently, this algorithm and its implementation have been modified to handle constraints (see [92] or chapter 3 of this thesis). The method samples points on an evolving simplex, moving to that vertex on the simplex with a value closest to optimality. The simplex then expands or contracts depending on the location of the point closest to optimality. The contractions reduce the size of the simplex; the procedure is halted when the lengths of the edges in the simplex fall below a tolerance set by the user. Direct search methods of this sort typically do not demonstrate rapid local convergence but, when functions are not smooth or the data is noisy, they are extremely robust and far less susceptible to the difficulties associated with faster, higher-order methods.

Because we are dealing with equality constraints only, minimization problem (7.37)–(7.39) can be handled in a straightforward manner. After each simplex is constructed, a penalty function is evaluated. Let $\rho_1$ and $\rho_2$ be two large constants. Then we employ a penalty function, such as

$$P(v;\rho_1,\rho_2) = \int_0^T Nv \cdot v dt + \rho_1\|x(T) - x_T\|_2 + \rho_2\|\dot{x}(T) - \dot{x}_T\|_2. \tag{7.40}$$

We can now minimize this penalty function. The new problem is then a continuous, nonsmooth, nonconvex, unconstrained optimization problem. Various strategies for updating the penalty parameters $\rho_1$ and $\rho_2$ were tried in early numerical testing. Most of our attempts at dynamically adjusting these parameters for the penalty function (7.40) failed significantly to improve the performance of the (PDS) algorithm. For instance, the dynamic penalty parameter updating schemes

$$\rho_1^+ \leftarrow \min\{10^6, (\rho_1^c + \|x(T) - x_T\|_\infty)\} \tag{7.41}$$

$$\rho_2^+ \leftarrow \min\{10^6, (\rho_2^c + \|\dot{x}(T) - \dot{x}_T\|_\infty)\} \tag{7.42}$$

for $\rho_1$ and $\rho_2$ were employed for comparison purposes only. (The superscript $c$ denotes a quantity associated with the minimizer on the previous simplex.) They proved fairly ineffective when compared to the performance with constant $\rho_1 = \rho_2 = 10^6$ (see Table 2 & 4).

More complicated penalty functions for (PDS) involving penalty parameter schemes, dynamical choices of norms for penalization, and the exploitation of feasibility are discussed in [92] and chapter 3 of this thesis. One such example is the use of the penalty function

$$\Pi_K(v; \rho_3) = \int_0^T Nv \cdot v dt + \rho_3(G_K(v)) + Z_K(v, v^c) \tag{7.43}$$

where $Z_K(v, v^c)$ is

$$Z_K(v, v^c) = \min\left\{\frac{G_K(v^c)}{\max\{G_K(v), 10^{-8}\}}, 1\right\} \max\left\{\int_0^T Nv^c \cdot v^c dt - \int_0^T Nv \cdot v dt, 0\right\} \tag{7.44}$$

and $G_K(v)$ is

$$G_K(v) = \|(x(T) - x_T, \dot{x}(T) - \dot{x}_T)^T\|_K. \tag{7.45}$$

Here $v^c$ denotes the value of control variables yielding the smallest value of the penalty function on the preceding simplex. The augmentation function $Z_K(v, v^c)$ incorporates a balance between a decrease in feasibility and motion towards optimality. The choice of norm, $K$

$$K = \begin{cases} \infty & \text{if} \quad G_2(v^c) \geq 2 \\ 2 & \text{if} \quad 1 < G_2(v^c) < 2 \\ 1 & \text{if} \quad G_2(v^c) \leq 1 \end{cases} \tag{7.46}$$

depends on the distance from feasibility. The parameter $\rho_3$ can be chosen initially to be a large constant or it can be updated dynamically. The updating strategy employed here averages that of (7.41) and (7.42)

$$\rho_3^+ \leftarrow \frac{1}{2}(\rho_1^+ + \rho_2^+). \tag{7.47}$$

It is worth commenting that consistent performance of (PDS) with respect to the penalty parameters $\rho_1$, $\rho_2$ and $\rho_3$, when minimizing penalty functions (7.40) and

(7.43), is characteristic of direct search methods. This is inherent in the way search directions and trial points are generated independently of any quantity dependent on the penalty parameters. Moreover, a new point is selected from the set of trial points by having the smallest associated objective function value. There is no condition enforced of minimum decrease or maximum allowable change. Therefore, no ill-conditioning is introduced into the problem by penalty parameters that are too large. However, as mentioned above, there is a price paid for this robustness, a loss of rapid local convergence.

## 7.5 Numerical results

The test problems presented here are small ones intended only to demonstrate the effectiveness of the simulation method and regularization. Much larger problems have actually been solved using similar techniques. An example is [40] in which an algorithm like (7.19)- (7.22) has been used to model the motion of an elastic string in the presence of dry friction; it is straightforward to employ the additional regularization procedure. The corresponding results are in [66].

Our first test problem is significant because we have chosen for $A$ the $3 \times 3$ Hilbert matrix, a small matrix with a large condition number ($\approx$ 524). This $A$ strongly couples the components of the state variable $x(t)$ which implies a fairly complicated dynamics, particularly for the friction multipliers. For this first test, we have chosen a final time, $T$, which is relatively small and we have prescribed only a desired final position, not a target velocity.

The second test problem is suggested by the physics of a spring-mass system consisting of $(d+1)$ springs with known spring constants and $d$ known point masses. The springs are in a chain between two fixed endpoints with exactly one of the masses attached between each pair of adjacent springs. The spring constants determine the stiffness matrix $A$ and the masses determine the matrix $M$.

We calculate values of the objective function (7.37) and the constraints (7.38) and (7.39) from solutions of the initial value problem (7.34)–(7.36). The following numerical experiments demonstrate the behavior of the optimization algorithm and, specifically, the multipliers with and without the regularization. For simplicity, we set $d = 3$ with the following values and functions:

Test problem (TP1)

- $x_0 = (1, 2, 3)^T$ (initial positions),
- $x_1 = (1, 1, 1)^T$ (initial velocities),
- $m_{11} = 4$, $m_{22} = 5$, $m_{33} = 6$ (diagonal matrix),

- $a_{ij} = \frac{1}{(i+j)-1}$ (Hilbert matrix),
- $c_{ii} = 10$ (diagonal matrix),
- $f_i(t) = -100e^{-10t}$ (external forces)
- $\Delta t = .0125$ (time step)
- $T = 5$ (final time)
- $\epsilon = (\Delta t)^{6/5}$,
- $r = \frac{2}{\delta \Delta t}$,
- $x_T = (2, 0, -2)^T$ (desired final position)
- $n_{ii} = \frac{1}{3}$ (diagonal matrix).
- $b_{ii} = 1$ (identity matrix)

Test problem (TP2)

- $x_0 = (1, 3, 5)^T$ (initial positions)
- $x_1 = (2, 2, 2)^T$ (initial velocities)
- $m_{11} = 1$ $m_{22} = 2$ $m_{33} = 3$ (diagonal matrix)
- $\kappa_1 = 1, \kappa_2 = 2, \kappa_3 = 3, \kappa_4 = 4$ (spring constants)
- $a_{11} = 3$ $a_{22} = 4$ $a_{33} = 5$ $a_{23} = a_{32} = -3$ $a_{12} = a_{21} = -2$ (Stiffness matrix)
- $c_{11} = 6$ $c_{22} = 5$ $c_{33} = 4$ (diagonal matrix)
- $f_i(t) = -10 \sin(\frac{3\pi t}{5}) e^{-t^2}$ (external forces)
- $\Delta t = .01$ (time step)
- $T = 10$ (final time)
- $\epsilon = (\Delta t)^{6/5}$
- $r = \frac{2}{\delta \Delta t}$
- $x_T = (5, 10, 15)^T$ (desired final position)

- $\dot{x}_T = (0,0,0)^T$ (desired final velocity)
- $n_{ii} = \frac{1}{3}$ (diagonal matrix)
- $b_{ii} = 1$ (identity matrix).

The control problems (7.40) and (7.43) were solved on an Intel iPSC/860 using 8 nodes simultaneously to calculate 216 function evaluations per processor for each simplex generated. The minimization procedure was halted in the event that the largest edge of a generated simplex was less than $10^{-10}$.

The simulation was performed on a Sun Sparcstation (1+) in double precision arithmetic (IEEE 64-bit floating point arithmetic). The fixed point iterations were applied until (using stopping criteria (7.22)) convergence was detected (The norm used was $\|\cdot\| = \|\cdot\|_2$ with a value of $\eta = 1.5 \times 10^{-8} \approx \sqrt{\epsilon_{machine}}$).

The tables illustrate how effective the regularization procedure was. In the simulation, the number of fixed point iterations decreases drastically with the use of regularization (see Table 2 & 4). While the regularization could not decrease the number of simplexes generated by (PDS) and, hence, did not decrease the number of solutions of the initial value problem (7.34)–(7.36), it did result in a decrease in the elapsed time. While the dynamical updating of the penalty parameter had little influence on the performance of (PDS), employing the sophisticated penalty function (7.43) did consistently result in fewer (PDS) simplexes than did the simpler function (7.40). The relative error in the final position, $E^1_\infty(v)$, is listed in the numerical results.

$$E^1_\infty(v) = \frac{\|x(T) - x_T\|_\infty}{\max\{1, \|x_T\|_\infty\}} \tag{7.48}$$

The relative error associated with the velocities, $E^2_\infty(v)$, is not reported in the tables.

$$E^2_\infty(v) = \frac{\|\dot{x}(T) - \dot{x}_T\|_\infty}{\max\{1, \|\dot{x}_T\|_\infty\}} \tag{7.49}$$

For the first test problem (TP1) no final velocities were prescribed; for the second problem the desired velocities ($\dot{x}_T = (0,0,0)^T$) were satisfied exactly.

The effect of regularizing on the positions and velocities of our test problems is summarized in Table 5. The maximum relative difference between regularized and unregularized position and velocity is recorded. More precisely, the discrepancies

$$D(x_R, x_{NR}) = \frac{\|\max_n |x_{NR}(n\Delta t) - x_R(n\Delta t)|\|}{1 + \max_n |x_R(n\Delta t)|}$$

and

$$D(\dot{x}_R, \dot{x}_{NR}) = \frac{\|\max_n |\dot{x}_{NR}(n\Delta t) - \dot{x}_R(n\Delta t)|\|}{1 + \max_n |\dot{x}_R(n\Delta t)|}$$

are in the third and fourth columns of Table 5, respectively. The subscripts $R$ and $NR$ denote regularized and nonregularized values, respectively.

Position, velocity, and the Coulomb multipliers are plotted versus time. The first set of figures (Figures7.1 and Figures7.1) represents the system coming to rest without any control present, hence the velocities of all three components vanish quite rapidly. In the second set of figures (Figures 7.5 and Figures 7.6) the control is activated and the final positions of the components are the desired targets.

There was a significant decrease in computation with the use of regularization (using (7.26) instead of (7.21)). An oscillating behavior of the multipliers was virtually extinguished by the regularization procedure both in the controlled and the uncontrolled systems. An interesting observation is that the values of the regularized multipliers were always close to the time-averaged values of the unregularized multipliers (as observed in [39].) The external forces were chosen to change the sign of the velocity in the interval $t \in (0, T]$. The only nonsmooth behavior in the multipliers occurred at these jumps, where the velocity of the system changed sign. Since we are most interested in behavior of these systems when velocities become small or vanish, dealing with the nonsmooth behavior of the multipliers was the primary difficulty in these simulations.

The graphs associated with test problem one and test problem two are labeled (TP1) and (TP2), respectively. Similarly, the graphs of the Coulomb multipliers calculated with and without the regularization procedure are labeled with (R) and (NR), respectively. In the uncontrolled system, the friction damped the displacements to zero very rapidly. Here, the weak convergence of the multipliers is evident and the benefit of the regularization is clear. In the controlled systems, the displacements did reach the correct targets, both with and without regularization as seen in Figures 7.12, 7.16, 7.4, and 7.8. We see weak convergence, again, of the multiplier associated with the particular component whose velocity vanishes for large $t$.

The effectiveness of the regularization and the smooth behavior of the multipliers when velocities are far from zero suggest that a combination of smooth and nonsmooth minimizations may prove useful in the study of these motions. In particular, using smooth or higher order minimization techniques (like those in [44] and [64] ) for (7.40) or (7.43) seems appropriate when velocities are far from zero. When velocities are small, however, more robust methods such as [45] may yield minima of a non-smooth function describing the system more accurately than do methods based on a smoothed analog. Perhaps it is possible to develop an algorithm for the minimization of (7.40) or (7.43) which harnesses the strengths of both types of minimization techniques.

## 7.6 Figures

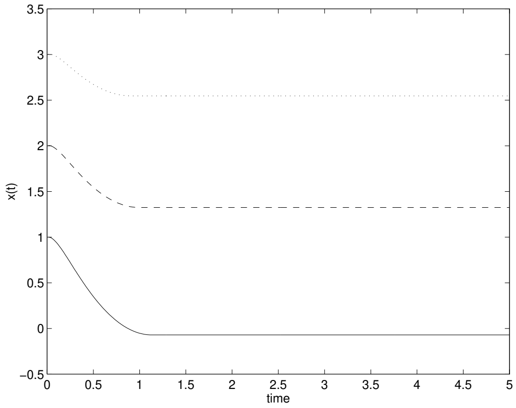


**Figure 7.1** - (TP1) - Displacements of uncontrolled system

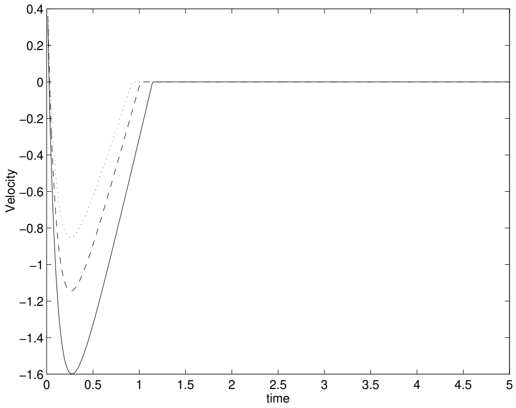


**Figure 7.2** - (TP1) - Displacements of uncontrolled system

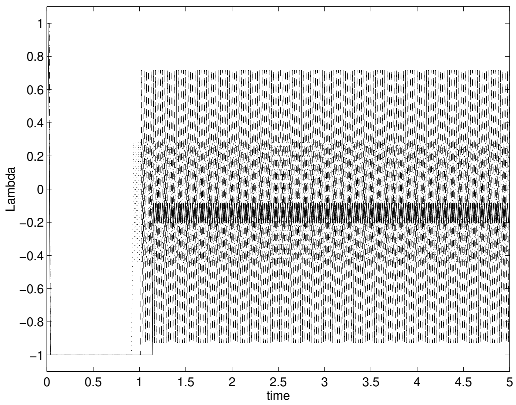


**Figure 7.3** - (TP1) - Multipliers of uncontrolled system (NR)

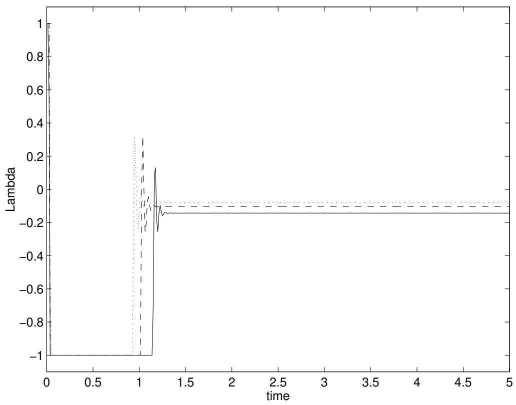


**Figure 7.4** - (TP1) - Multipliers of uncontrolled system (R)

| Multiplier Scheme | Fixed Point Iterations (FIP) | Average: (FIP) per time step | Control Present on RHS? |
|---|---|---|---|
| Not Regularized | 35633 | 89.0825 | no |
| Not Regularized | 16676 | 41.6900 | yes |
| Regularized | 5135 | 12.8375 | no |
| Regularized | 4184 | 10.4600 | yes |

**Table 7.1** Problem I: Simulation

| Penalty Scheme | Penalty constant | Multipliers Scheme | PDS Simplexes | # of IVPs solved | error $E^1_\infty(v^*)$ | elapsed time (msec) |
|---|---|---|---|---|---|---|
| $P(v;\rho)$ | Constant | Not Regularized | 21 | 36624 | 7.82E-8 | 1878366 |
| $P(v;\rho)$ | Constant | Regularized | 21 | 36624 | 7.82E-8 | 1141314 |
| $P(v;\rho)$ | Dynamic | Not Regularized | 21 | 36624 | 7.79E-8 | 1878338 |
| $P(v;\rho)$ | Dynamic | Regularized | 21 | 36624 | 7.80E-8 | 1141328 |
| $\Pi(v;\rho)$ | Constant | Not Regularized | 19 | 33136 | 2.01E-8 | 1707533 |
| $\Pi(v;\rho)$ | Constant | Regularized | 19 | 33136 | 1.99E-8 | 1040543 |
| $\Pi(v;\rho)$ | Dynamic | Not Regularized | 19 | 33136 | 2.08E-8 | 1707433 |
| $\Pi(v;\rho)$ | Dynamic | Regularized | 19 | 33136 | 1.94E-8 | 1040461 |

**Table 7.2** Problem I: Control

| Multiplier Scheme | Fixed Point Iterations (FIP) | Average: (FIP) per time step | Control Present on RHS? |
|---|---|---|---|
| Not Regularized | 20876 | 20.876 | no |
| Not Regularized | 11517 | 11.517 | yes |
| Regularized | 3238 | 3.238 | no |
| Regularized | 3142 | 3.142 | yes |

**Table 7.3** Problem 2: Simulation

| Penalty Scheme | Penalty constant | Multipliers Scheme | PDS Simplexes | # of IVPs solved | error $E^1_\infty(v^*)$ | elapsed time (msec) |
|---|---|---|---|---|---|---|
| $P(v,\rho)$ | Constant | Not Regularized | 23 | 40112 | 2.54E-9 | 2309235 |
| $P(v,\rho)$ | Constant | Regularized | 23 | 40112 | 2.54E-9 | 1424193 |
| $P(v,\rho)$ | Dynamic | Not Regularized | 23 | 40112 | 2.56E-9 | 2309188 |
| $P(v,\rho)$ | Dynamic | Regularized | 23 | 40112 | 2.56E-9 | 1424085 |
| $\Pi(v,\zeta)$ | Constant | Not Regularized | 20 | 34880 | 6.03E-8 | 2010565 |
| $\Pi(v,\zeta)$ | Constant | Regularized | 20 | 34880 | 6.02E-8 | 1241696 |
| $\Pi(v,\zeta)$ | Dynamic | Not Regularized | 20 | 34880 | 6.11E-8 | 2010213 |
| $\Pi(v,\zeta)$ | Dynamic | Regularized | 20 | 34880 | 6.11E-8 | 1241428 |

**Table 7.4** Problem 2: Control

| Problem | Control present on RHS ? | position discrepancy $D(x_R, x_{NR})$ | velocity discrepancy $D(x_R, x_{NR})$ |
|---|---|---|---|
| Test Problem 1 | yes | 4.4E-8 | 2.8E-5 |
| Test Problem 1 | no | 1.4E-7 | 3.8E-4 |
| Test Problem 2 | yes | 6.1E-7 | 5.7E-2 |
| Test Problem 2 | no | 8.9E-8 | 3.2E-3 |

**Table 7.5** Difference between regularization and no reglarization

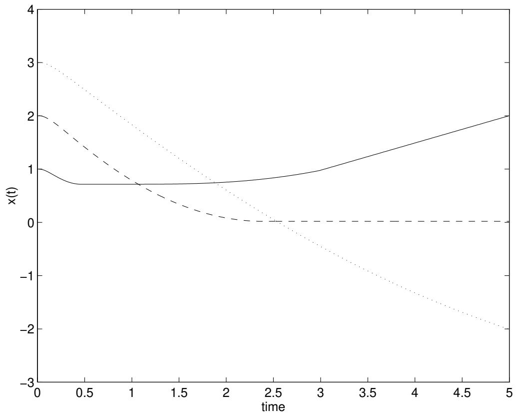


**Figure 7.5** - (TP1) - Displacements of controlled system

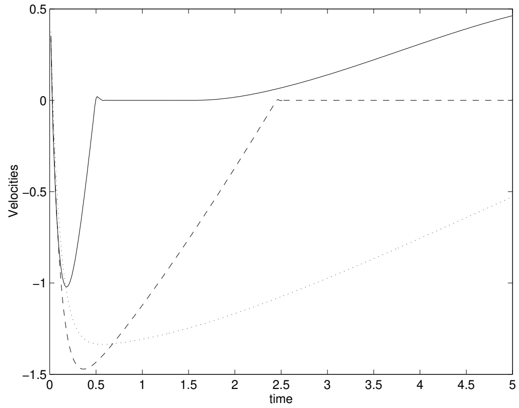


**Figure 7.6** - (TP1) - Displacements of controlled system

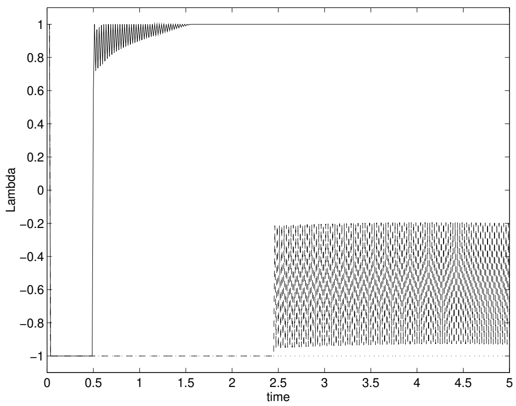


**Figure 7.7** - (TP1) -
Multipliers of controlled system
(NR)

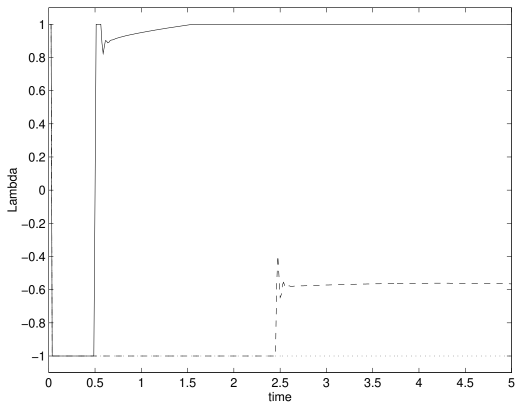


**Figure 7.8** - (TP1) -
Multipliers of controlled system
(R)

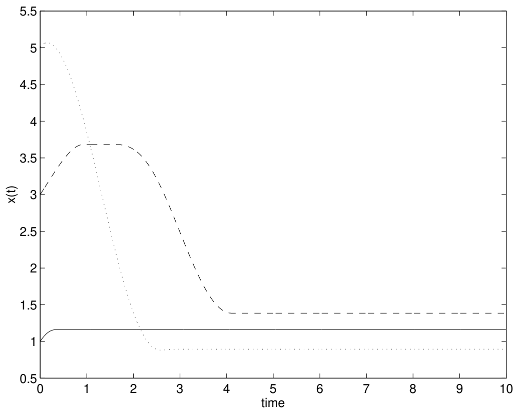


**Figure 7.9** - (TP2) -
Displacements of uncontrolled
system

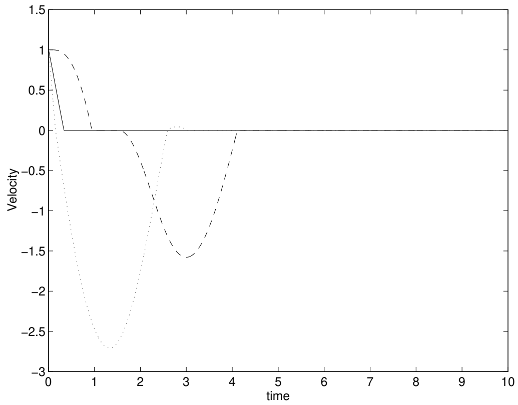


**Figure 7.10** - (TP2) -
Velocities of uncontrolled system

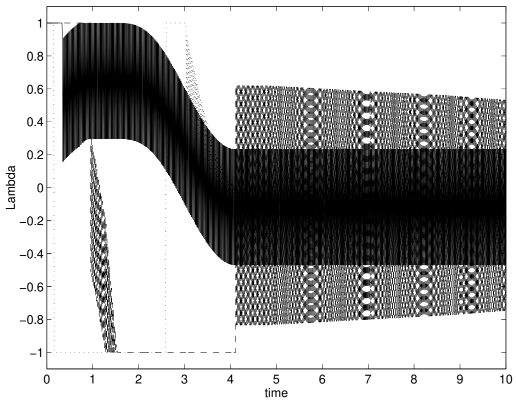


**Figure 7.11** - (TP2) - Multipliers of uncontrolled system (NR)

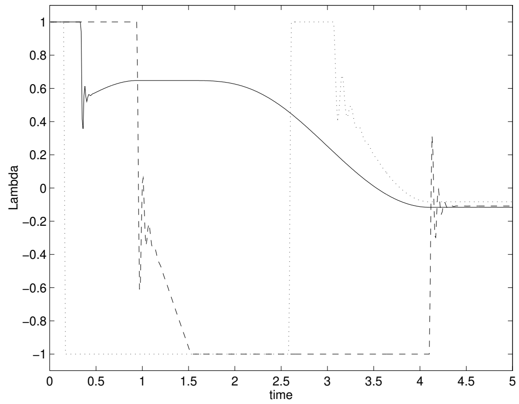


**Figure 7.12** - (TP2) - Multipliers of uncontrolled system (R)

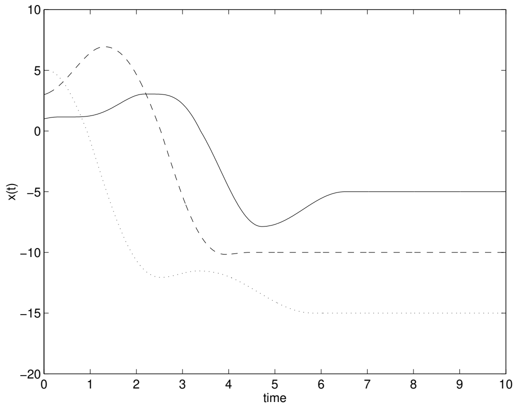


**Figure 7.13** - (TP2) - Displacements of controlled system

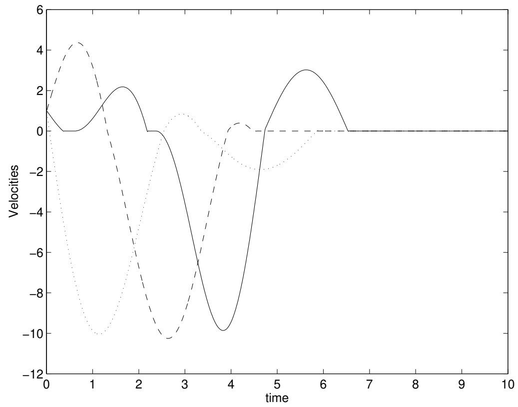


**Figure 7.14** - (TP2) - Velocities of controlled system

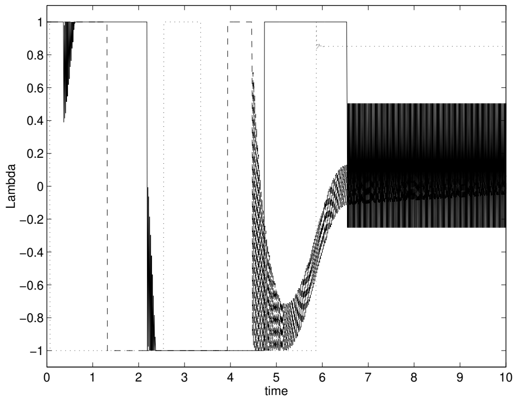


**Figure 7.15** - (TP2) -
Multipliers of controlled system
(NR)

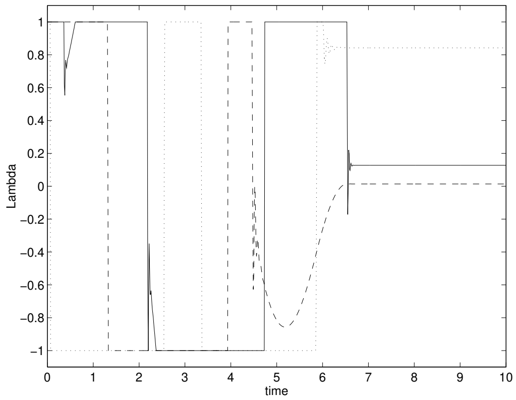


**Figure 7.16** - (TP2) -
Multipliers of controlled system
(R)

# Chapter 8

# An Algorithm for the Numerical Solution of Some Shape Optimization Problems

## 8.1 Introduction

The problem of finding an optimal shape subject to simulation constraints, *shape optimization*, is a problem fraught with difficulties. Such problems can be viewed as *free-boundary problems*, long considered among the hardest problems of applied mathematics. Yet, shape optimization problems are abundant in industrial applied mathematics, arising in structural mechanics, environmental engineering, basin modeling and, of course, in optimal control of distributed-parameter systems. Algorithms for shape optimization are an indispensable tool for the design and analysis of mechanical structures, accurate forecasting of the time evolution of such free boundaries as river basins and shallow bays, and of dendrites in crystal growth. They have proved useful even for the design of electro-magnetic structures such as magnets and antennas.

The problem we shall consider here is that of finding an optimal shape when the measure of quality of the shape is based upon the solution of some differential equation in a region bounded by the shape to be optimized. The algorithm used here presupposes the solution of the simulation problem associated with a 'trial shape'. It then alters that shape based on the simulation. For this reason, we choose a method for solving the partial differential equations (state equations) that does not require generation of a new grid for each trial-shape simulation. The method of choice is the fictitious domain method (FDM). This method is a recently popular tool for the solution of partial differential equations on irregularly shaped domains. Some recent examples of its use in electromagnetics are contained in the paper by Atamian, Dinh, Glowinski, He, and Pironneau [3] and the papers by Glowinski, T-W. Pan, and Periaux [73], [72] and [74] explains its use in problems arising from fluid flow.

One of the earliest solutions to this type of shape optimization problem is that of Begis and Glowinski (see [7]). They find the shape of an optimal 'cap' in an elegant way using the theory of elliptic partial differential equations. One cannot expect to solve all shape optimization problems as deftly as they are solved in that paper, but the important ideas can be abstracted and adapted to a more computational level. That course will be pursued in this chapter.

The chapter will be organized as follows. In the second section we will describe a general setting for our shape optimization problems. In the third section we will describe the fictitious-domain decomposition. This technique couples the fictitious-domain method with domain decomposition to generate numerical schemes for the solution of some linear, elliptic, partial differential equations that are efficient on either sequential or parallel computers. Fictitious-domain decomposition also has the important feature that it does not require re-gridding when functions are re-evaluated based on new trial shapes. In the fourth section we discuss a family of test problems calling for eigenvalues of linear elliptic differential equations on irregularly-shaped domains. We solve these test problems numerically and comment on the results. The fifth section describes a fictitious domain method for the solution of Naiver-Stokes flow of an incompressible fluid over an obstacle. In section six, the method of the fifth section is used to solve some problems of optimal shape design. Section seven contains the concluding remarks for the chapter.

## 8.2 The shape optimization problem

In this section, the formulation of a shape-optimization problem is described in general terms. Some specific examples will be discussed in the following sections.

Let $\omega$ be an irregularly-shaped domain upon which some partial differential equation with prescribed boundary conditions is given. Assume, further, that the boundary of this irregularly shaped domain, $\partial\omega$, can be parameterized. Let the set of parameters be $\kappa$. The vector $\kappa$ with elements $\kappa_1, \kappa_2, \ldots, \kappa_K$ serves as a shape control variable, so that $\partial\omega = \partial\omega(\kappa)$. The $\kappa$ might, for example, be the nodes and coefficients of a cubic spline or they might be the vertices of a piecewise polygonal domain. Usually, in a specific application of the shape optimization problem, the inherent structure will suggest a good parameterization of the boundary.

Let $u, p$ denote state variables that satisfy a partial differential equation, say $L_\omega(u,p) = 0$. The subscript $\omega$ on $L$ is natural because any initial or boundary conditions are incorporated in the definition of $L_\omega$. At least a portion of the boundary conditions or initial conditions will be prescribed on the irregularly shaped boundary. For example, $u$ and $p$ might be velocities and pressures in a fluid flow problem. Clearly, these quantities depend strongly on the irregular shape of the domain $\omega$ and, therefore, they depend on the shape control variable, $\kappa$. Let $S$ denote a functional of the state variables and hence of the irregularly-shaped domain. The objective is to minimize the quantity $S$ with respect to the shape control variables $\kappa$ subject to a set of constraints that include, at least, the constraint $L_\omega(u,p) = 0$. That is to say, the objective is to minimize the shape cost functional, $S$, over admissible values of $\kappa$, subject to the state variables satisfying $L_\omega(u,p) = 0$. Admissible values of $\kappa$ will be expressed by a constraint written $G(\kappa) \leq 0$. Such constraints will be on the shape itself. The following are examples of conditions which might arise in applications restricting admissible values of $\kappa$: enforcing a region to be star-shaped, bounding the volume of $\omega$, or bounding the length of the boundary $\partial\omega$. These examples necessarily constitute a constraint on the control variables.

The discussion in chapter 4 suggests two mathematically equivalent formulations of this shape optimization problem. The first possibility ensures that $L_\omega(u,p)=0$ is satisfied for all values of $(u,p)$. In other words, the equation $L_\omega(u,p)=0$ is satisfied implicitly in the following shape optimization problem

$$\begin{aligned}&\min_{\kappa} S\left(u(\omega(\kappa)), p(\omega(\kappa)\right)\\ &\quad\text{subject to}\\ &G(\kappa)\leq 0.\end{aligned} \tag{8.1}$$

This is sometimes referred to as a *Black Box* formulation because it treats the solution to the equation $L_\omega(u,p)=0$ implicitly.

The second formulation results from minimization over the shape control variables and the state variables. The state equation is treated as a constraint. This formulation is written

$$\begin{aligned}&\min_{\kappa,u,p} S\left(u, p, \omega(\kappa)\right)\\ &\quad\text{subject to}\\ &L_\omega(u,p)=0\\ &G(\kappa)\leq 0.\end{aligned} \tag{8.2}$$

Other formulations are certainly possible. One could minimize over only a subset of the state variables, for example, to arrive at an *in-between* formulation. Here, we will be concerned only with the two formulations (8.1) and (8.2).

A few comments are in order. Because there is a strong coupling between the constraints and the objective function through the shape of the boundary of the domain, differentiability of $S$ with respect to perturbations of the boundary has been and is an active area of research. Among the excellent references on this question are Simon [128] and Sokolowski and Zolesio [129]. For a discussion of shape optimization arising from elliptic partial differential equations, the book by Pironneau provides a thorough survey [116]. Another important reference is the book by Haslinger and Neittaanmaki [82] which is rich in applications of shape optimization problems.

## 8.3 Fictitious-domain decomposition method for elliptic problems

In this section we briefly describe a *fictitious-domain decomposition method* (FDDM) for the approximate solution of elliptic partial differential equations. Using the notation of the previous section, we define $L_\omega(u,p)=L_\omega(u)$ with the following equation

$$\begin{aligned}&\alpha u-\nabla\cdot\mathbf{K}\nabla u-f=0\quad\text{in}\quad\omega\\ &u-u_B=0\quad\text{on}\quad\partial\omega.\end{aligned} \tag{8.3}$$

Multiply equation (8.3) by a function belonging to some well-chosen function space, then integrate by parts to generate a quadratic form, $a(u,v)$.

$$a(u,v) = \alpha \int_\omega uv dx + \frac{1}{2}\int_\omega \nabla v \cdot (\mathbf{K}\nabla u) dx - \int_\omega fv dx. \tag{8.4}$$

If $\mathbf{K}$ is a positive-definite, symmetric matrix, the system is one which can be treated by an obvious generalization of the procedure that follows. However, in order to illustrate shape optimization without the confusion of a complicated state equation, we continue with Poisson's equation which follows upon taking $\mathbf{K} = I$ and $\alpha = 0$. We will consider more complicated state equations in later chapters.

The problem of concern with Poisson's equation as state equation is, then

$$\begin{aligned} \triangle u + f = 0 \quad &\text{in} \quad \omega \\ u - u_B = 0 \quad &\text{on} \quad \partial\omega \end{aligned} \tag{8.5}$$

and the quadratic form

$$a(u,v) = \frac{1}{2}\int_\omega \nabla u \cdot \nabla v dx - \int_\omega fv dx. \tag{8.6}$$

The major obstacle to the solution of (8.5) is the irregularly-shaped domain $\omega$. Indeed, if $\omega$ were a computationally friendly shape (such as a square) there would be an abundance of fast solvers at our disposal. For this reason we introduce a fictitious domain $\Omega$. Let $\Omega$ denote a domain completely containing $\omega$. We will always take $\Omega$ to be a square to take advantage of existing fast solvers designed to solve (8.3) in regular domains. Let $\bar{f}$ be a smooth extension of $f$ into the larger domain $\Omega$ (for example $\bar{f} \in L^2(\Omega)$). Then, define the function $F$ over the whole domain $\Omega$.

$$F_\Omega(v) = \int_\Omega \nabla v \cdot \nabla v - \int_\Omega \bar{f} v. \tag{8.7}$$

This function is just the quadratic form associated with our fictitious domain; call this quadratic form $a_\Omega(u,u)$.

Minimizing $F_\Omega(u)$ yields a weak solution to the differential equation on the fictitious domain. Of course, in order to satisfy the boundary conditions, constraints are added to the problem.

The nonlinear programming problem to be considered is

$$\begin{gathered} \min_{v \in H_0^1(\Omega)} F_\Omega(v) \\ \text{subject to;} \\ v - u_B = 0 \quad \text{on} \quad \partial\omega \end{gathered} \tag{8.8}$$

Solving (8.8) will yield a weak solution to the Poisson's equation problem (8.5).

It is worth writing down the augmented Lagrangian function associated with (8.8) in order to understand the functioning of the resulting numerical procedure. Let $\ell_\rho(v,\mu)$ denote the augmented Lagrangian associated with (8.8).

$$\ell_\rho(v,\mu) = F_\Omega(v) + \int_{\partial\omega} \mu(v-u_B) + \frac{\rho}{2}\int_{\partial\omega}(v-u_B)(v-u_B). \tag{8.9}$$

The variables $\mu$ are the Lagrange multipliers associated with the boundary conditions in (8.5); $\rho$ is a positive scalar, the convexifing penalty parameter. It is not hard to see that the resulting problem, after discretization, will yield a quadratic programming problem (minimize a quadratic objective function subject to linear constraints) to which the algorithm discussed in section 2.9 applies quite naturally. To find $\tilde{u}$ with this algorithm, one seeks the solution of the linear system

$$\int_\Omega \nabla\tilde{u}\nabla\phi + \rho\int_{\partial\omega}\tilde{u}\phi = \int_\Omega \bar{f}\tilde{u}\int_{\partial\omega}\lambda\phi + \rho\int_{\partial\omega}u_B\phi \tag{8.10}$$

at each iteration.

Domain decomposition offers a method for solving equation (8.10) over the large domain $\Omega$ (see, for instance, Letallec [96], Chan and Mathew [28], Cowsar and Wheeler [36], and Glowinski and Wheeler [75]). Experience with various applications of constrained optimization suggests the formulation to be used here. The formulation is explicitly designed to be computationally convenient. The large domain $\Omega$ is formally divided into sub–domains $\Omega = \cup\Omega_i$ with boundaries $\gamma = \cup\partial\Omega_i/\partial\Omega$. This leads to a whole family of optimization formulations which, when flux matching is enforced in patching together the solutions on subdomains, combines into the quadratic programming problem

$$\begin{aligned}
&\min_{v\in H_0^1(\Omega)} \; F_\Omega(v)\\
&\text{subject to;}\\
&\left[\frac{\partial v}{\partial \mathbf{n}}\right] = 0 \quad \text{on} \quad \gamma\\
&v - u_B = 0 \quad \text{on} \quad \partial\omega.
\end{aligned} \tag{8.11}$$

where the symbol $[\frac{\partial u}{\partial \mathbf{n}}]$ denotes the discontinuity in the normal component of the gradient of $v$ at an interface. An equivalent Lagrangian and augmented Lagrangian can be constructed for this nonlinear program exactly as was done for (8.8) with (8.9), except that additional Lagrange multipliers associated with the flux matching constraints must be included. Both formulations, (8.8) and (8.11), result in quadratic programming problems.

The formulation (8.11) is well suited for *mixed finite-element methods*. By following Glowinski and Wheeler [75], we can formulate an optimization problem that uses both domain decomposition and the fictitious-domain method with mixed finite-elements. The vector $v$ is chosen from a divergence-free space so that the variational principle (8.10) becomes

$$\inf_{v \in H(div,\Omega)} \sup_{u \in L^2(\Omega)} \frac{1}{2} \int_\Omega v \cdot v + \int (\nabla \cdot v + f) u. \tag{8.12}$$

To obtain numerical results, we make use of this observation. We use mixed finite-elements of Ravier-Thomas type (lowest order). The discretization and numerical tests will be discussed in the following section.

## 8.4 An elliptic, shape-optimization test problem; optimizing eigenvalues

In this section we formulate some problems designed to test the viability of the approach formulated in (8.11). The algorithm to solve the shape optimization is found in section 3. This algorithm is designed to solve the general nonlinear programming problem with both equality constraints and/or inequality constraints. The fact that this algorithm does not use computed derivatives is advantageous because, for our test problems, the map from the control variables to the state variables ($\kappa \to u(\kappa)$) will be only differentiable, at best. The evaluation of the function and of the constraints entails a forward simulation, the integration of 8.8 for some trial domain $\omega$. This is carried out using methods from the previous section.

All the test examples are restricted to domains $(\Omega, \omega)$ in $\Re^2$. We will also assume that all domains and all shapes are symmetric about the $x_1$ axis. Thus we can parameterize the upper portion of the boundary shape and complete the shape by reflection in the $x_1$ axis. We parameterize $\partial\omega$ using polar coordinates ( $x_1 = r\cos(\theta)$ and $x_2 = r\sin(\theta)$ ). Consider a trigonometric Fourier series with coefficients $\kappa$ parameterizing the curve, $\omega(\kappa)$. The series is written

$$\partial\omega(\kappa) = \kappa_0 + \sum_{i=1}^{K} \kappa_{2i} \sin(i\theta) + \kappa_{2i+1} \cos(i\theta) \tag{8.13}$$

where $0 \leq \theta \leq \pi$ (rather than $0 \leq \theta \leq 2\pi$) because the curve is defined by reflection across the $x_1$ axis. The boundary must be approximated by no more than 50 parameters if the minimization method (section 3) is to remain feasible. Experiments show that $7 < K < 21$ gives a healthy balance between the accuracy of the shape approximation and the size of the problem.

The first two test problems are inverse problems. We first solve 8.8 for a given target shape $\omega_d$. The aim of the test problems is to recover this target shape from information about the state variables $u_d$ corresponding to this solution. These first two test problems are convenient because, by examining their solutions and comparing them to the original target shapes, one can easily see how well the method succeeds. For a third test problem, we aim to find the convex shape that maximizes the minimum eigenvalue of the Laplacian when the length of the shape's boundary is bounded from above.

We now introduce three possible choices for our constraint $G(\kappa)$ appearing in (8.8) and (8.11). The admissible shapes are defined by the following sets:

$$\Pi_A = \{ \text{ set of subdomains in } \Omega \text{ with no area greater than } A \ \}, \tag{8.14}$$

$$\Pi_L = \{ \text{ set of subdomains in } \Omega \text{ with total boundary length, } \partial\omega \text{ no greater than } L \ \}, \tag{8.15}$$

and

$$\Pi_C = \{ \text{ set of convex subdomains in } \Omega \ \}. \tag{8.16}$$

The first two constraints can be handled straightforwardly. One simply evaluates the area or length using a numerical approximation of the appropriate integral. The second constraint must be implemented in a nonstandard way. Because we are using polar coordinates, we write our curve as the solution to $r = \eta(\theta)$, where $\eta$ if defined through (8.13). The curvature, $K$, for this curve is

$$K = \frac{\eta^2 + 2\eta_r^2 - \eta\eta_{rr}}{(\eta^2 + \eta_r^2)^{\frac{3}{2}}}. \tag{8.17}$$

Constraining $K$ to be positive ensures that the shape is locally convex. The denominator is always non-negative and it can be removed from this constraint. Thus, convexity can be assured by insisting that

$$\eta^2 + 2\eta_r^2 - \eta\eta_{rr} \leq 0. \tag{8.18}$$

Ideally, we would like this constraint to be satisfied at every point on the curve, something that cannot be enforced by pointwise computation. Instead, we approach this by requiring that the constraint be satisfied at $2N+1$ points uniformly distributed on the curve. Here, $N$ is the number of sines and cosines appearing in (8.13).

### 8.4.1 Test problem 1

The first set of cost-functional and constraints is

$$\begin{array}{l} \min\limits_{\kappa} \frac{1}{2}\int_{\omega\kappa} (u - u_d)^2 dx \\ \text{subject to:} \\ \omega \in \Pi_A(\omega_d). \end{array} \tag{8.19}$$

The starting shape is shown in Figure 8.1). It is not close to the target shape. Figure 8.2 shows both the target shape (dotted line) and the approximate solution located by our algorithm (solid line).

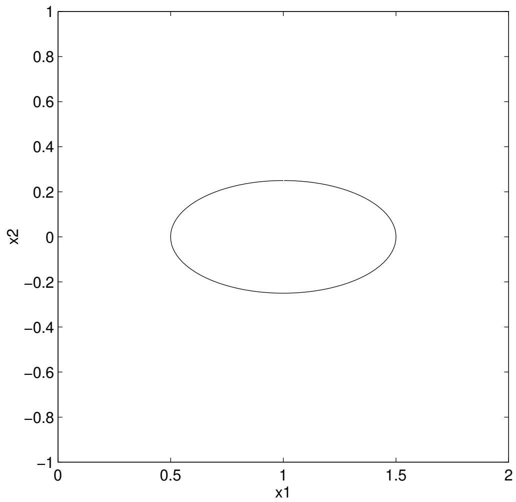


**Figure 8.1** - Starting shape for test problem #1

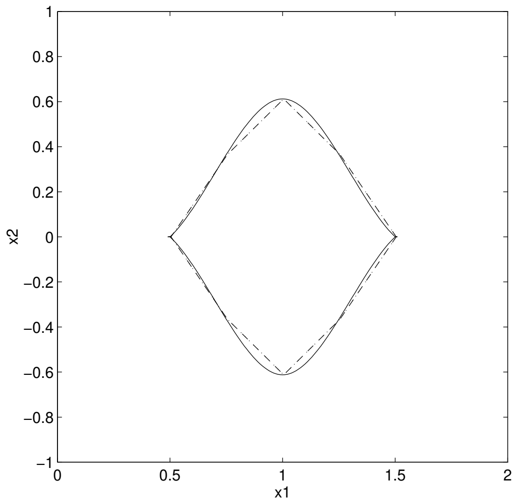


**Figure 8.2** Final shape and target shape for test problem #1

### 8.4.2 Test problem 2

The second set of cost functional and constraints is

$$\begin{array}{l} \min\limits_{\kappa} \frac{1}{2} \int_{\partial\omega\kappa} (u - u_d)^2 ds \\ \text{subject to:} \\ \omega \in \Pi_L(\omega_d). \end{array} \tag{8.20}$$

### 8.4.3 Test problem 3, an eigenvalue maximization

Another family of problems fitting naturally into the framework of equation (8.1) comes from classical eigenvalue problems of mathematical physics. An example is an *isoperimetric problem*, the basis of the third test problem. In simple terms, one asks, "What is the shape of a convex drum head that yields the maximum smallest eigenvalue for a perimeter of fixed length?" To answer this question, we will use the *shifted power method* to approximate the smallest eigenvalue of the Laplacian. An explanation of the shifted power method is found in Golub and Van Loan [76] or Stewart [130]. Consider the eigenvalue equation

$$\begin{array}{l} -\Delta u = \sigma u \quad \text{on} \quad \omega(\kappa) \\ u = 0 \quad \text{on} \quad \partial\omega(\kappa). \end{array} \tag{8.21}$$

where $\sigma$ is an eigenvalue of the Laplacian. The spectrum of the Laplacian is obviously unbounded above, but the smallest eigenvalue is bounded below and can be computed

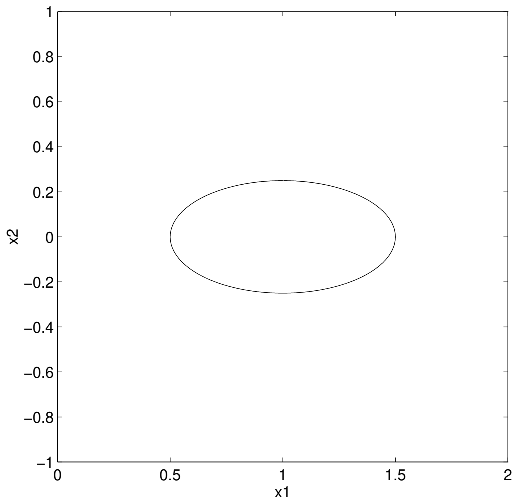


**Figure 8.3** Starting shape for test problem #2

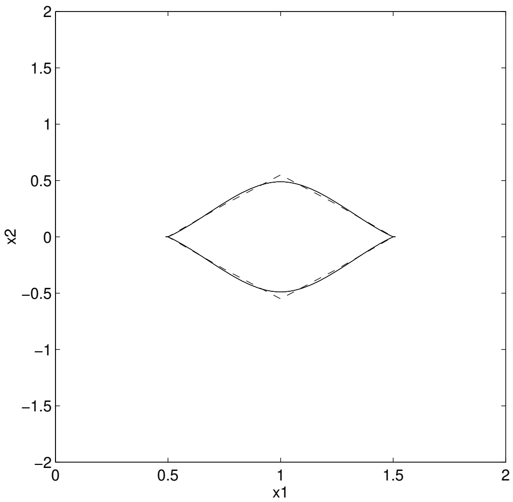


**Figure 8.4** - Final shape and target for test problem #2

using ideas from the previous section coupled with the shifted power method. Many fascinating optimization problems are posed by these facts. Solving such problems with functional analysis has been an active area of applied mathematics for many years. Interesting results are to be found in the classic book by Courant and Hilbert [34] and in papers by Cox and Ross [37], Rousselet [125] and [126], and Joseph [91]. We restrict ourselves to the following problem.

$$\begin{array}{l} \max\limits_{\kappa} \min\limits_{\sigma} \sigma \\ \quad \text{subject to:} \\ \omega \in \Pi_L \cup Pi_C \cup Pi_A \\ L = 4 \\ A = .75 \end{array} \tag{8.22}$$

where the pair $(\sigma, u)$ satisfies (8.21). Based on numerical results, this problem appears to admit local minimizers. The algorithm, from different starting shapes, did not always halt at similar end shapes. If a circle with perimeter $L$ was used as a starting shape, the algorithm failed to move from the starting point. With an elliptical starting shape, a more complex end shape resulted (see Figures 8.2 and 8.4). It should be pointed out that the convexity was not exact. The violations are only of the order square root of machine precision, yet the solution-shape in figure 8.6 has a slight but just visible concavity.

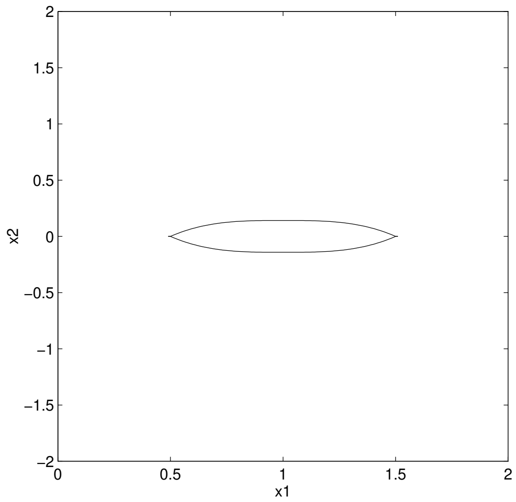


**Figure 8.5** Starting shape for test problem #3

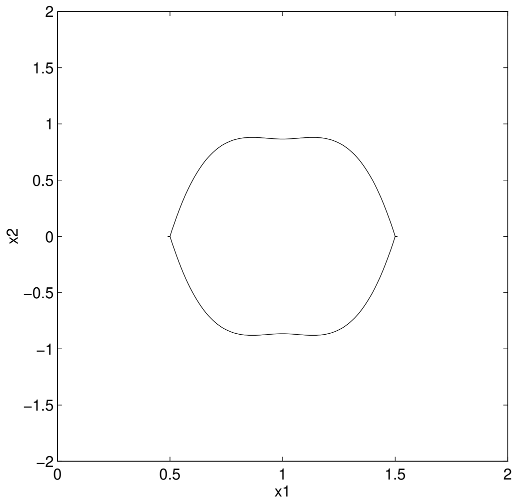


**Figure 8.6** Final shape for test problem #3

### 8.4.4 Summary of numerical results

For all of the numerical results presented here, mixed finite-elements (spaces of Ravier-Thomas mixed finite-elements of lowest order) were used. The discretization intervals of the $x_1$ and $x_2$ directions were of the same size, denoted by $h$. The number of iterations for all our test problems remained reasonable.

## 8.5 Fictitious-domain method for Navier-Stokes

In this section we discuss a fictitious-domain method for the solution of the Navier Stokes equations. We model flow around an obstacle. There is a small but important

| Grid | Shape Iter. | Iterations Inner | Iterations Outer | Exec. (sec) |
|---|---|---|---|---|
| 8x8 | 30 | 121 | 62 | 3.1 |
| 16x16 | 60 | 358 | 544 | 7.3 |
| 32x32 | 225 | 3332 | 3620 | 3395 |
| 64x64 | 440 | 9674 | 12191 | 67666 |

**Table 8.1** Numerical results for test problem # 1 using the two loop formulation

| Grid | Shape Iter. | Iterations QP solves | Execution Time(sec) |
|---|---|---|---|
| 8x8 | 30 | 31 | 3.8 |
| 16x16 | 60 | 897 | 29 |
| 32x32 | 225 | 5853 | 659 |
| 64x64 | 440 | 29462 | 10568 |

**Table 8.2** Numerical results for test problem # 1 using the single loop formulation

| Grid | Shape Iter. | Iterations Inner | Iterations Outer | Exec. (sec) |
|---|---|---|---|---|
| 8x8 | 12 | 79 | 80 | 2.9 |
| 16x16 | 16 | 114 | 121 | 7.0 |
| 32x32 | 120 | 1589 | 1908 | 1981 |
| 64x64 | 140 | 2176 | 4007 | 21347 |

**Table 8.3** Numerical results for test problem # 1 using the two loop formulation

| Grid | Shape Iter. | Iterations QP solves | Execution Time(sec) |
|---|---|---|---|
| 8x8 | 12 | 119 | 3.8 |
| 16x16 | 16 | 171 | 29 |
| 32x32 | 120 | 3095 | 359 |
| 64x64 | 140 | 9367 | 4834 |

**Table 8.4** Numerical results for test problem # 2 using the single loop formulation

| Grid | Power Iter. | Shape Iter. | Iterations Inner | Iterations Outer | Exec. (sec) |
|---|---|---|---|---|---|
| 8x8 | 64 | 30 | 121 | 62 | 3.1 |
| 16x16 | 125 | 60 | 358 | 544 | 7.3 |
| 32x32 | 563 | 225 | 3332 | 3620 | 3395 |
| 64x64 | 1767 | 440 | 9674 | 12191 | 67666 |

**Table 8.5** Numerical results for test problem # 3 using the two loop formulation

| Grid | Power Iter. | Shape Iter. | Iterations QP solves | Execution Time(sec) |
|---|---|---|---|---|
| 8x8 | 64 | 30 | 31 | 3.8 |
| 16x16 | 125 | 60 | 897 | 29 |
| 32x32 | 563 | 225 | 5853 | 659 |
| 64x64 | 1767 | 440 | 29462 | 10568 |

**Table 8.6** Numerical results for test problem # 3 using the single loop formulation

difference between the geometries of the problem here and in section 8.2. In this section, the differential equation is active on the exterior of the irregularly-shaped domain. The general problem formulation presented in section 8.1 still applies, however. There is still a state constraint of the form $L_\omega(u,p) = 0$, but we are particularly interested in the behavior of our state variables $(u,p)$ in the region between $\Omega$ and $\omega$, whereas in section 8.2 we were interested in the behavior of the state variables inside the region $\omega$.

We take $\Omega \in \Re^2$ and $x \in \Omega$ and consider an incompressible fluid flowing past an obstacle $\omega$. The fluid is modeled by the Navier-Stokes equation which we write in the form

$$\frac{\partial u}{\partial t} - -\nu \triangle u + (u \cdot \nabla)u + \nabla p = f \quad \text{in} \quad \Omega \tag{8.23}$$

with the condition for incompressibility of the fluid

$$\nabla \cdot u = 0 \quad \text{in} \quad \Omega. \tag{8.24}$$

This is a form of the Navier-Stokes equation which has been nondimensionalized in the customary way. In generating (8.23), a characteristic velocity $U$ and a characteristic length $d$ are chosen appropriate to the particular problem to be considered. These quantities, along with the kinematic viscosity, are used to nondimensionalize the physical form of the Navier-Stokes equation . The quantity $\nu$ is a dimensionless parameter referred to as the *viscosity parameter*. It is the reciprocal of the Reynolds number.

$$\nu = \frac{1}{R_e} = \frac{\mu}{\rho U d} \tag{8.25}$$

where $\rho$ and $\mu$ are density and viscosity, respectively, and $\mathbf{R_e}$ is the Reynolds number.

Let $n$ denote the outward-pointing unit vector normal to $\partial\omega$. The boundary conditions to be imposed are

$$\begin{aligned} &u - u_B = 0 \quad \text{on} \quad \partial\omega \\ &\int_{\partial\omega} u_B \cdot \mathbf{n} \; d\partial\omega = 0 \end{aligned} \tag{8.26}$$

and the initial conditions are

$$\begin{aligned} u(x,0) &= u_0(x) \\ \nabla \cdot u_0 &= 0 \quad \text{in} \quad \Omega. \end{aligned} \tag{8.27}$$

The boundary conditions are, in general, time-dependent.

We now introduce a space of functions, sufficiently smooth and periodic along the boundary of the fictitious domain, $\partial\Omega$. Define $\bar{V}$, such that

$$\bar{V} \equiv \left\{ v | v \in H^1(\Omega), \quad \text{and} \quad v \quad \text{is periodic at} \quad \partial\Omega \right\}. \tag{8.28}$$

We observe an equivalence between equations (8.23) and the weak formulation

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v dx - \nu \int_\Omega \nabla u \cdot \nabla v dx + \int_\Omega (u \cdot \nabla) u \cdot v dx + \int_\Omega p \nabla \cdot v dx = \int_\Omega f \cdot v dx + \int_{\partial\omega} \lambda \cdot v d\partial\omega \tag{8.29}$$

where $v \in \bar{V}$ and $u(t) \in \bar{V}$ for positive time $t$. The $\lambda$ in this equation can be viewed as a Lagrange multiplier enforcing the jump condition

$$\nu \frac{\partial u}{\partial n} - np \tag{8.30}$$

for points lying on the boundary $\partial\omega$. The square brackets in (8.30) denote the jump in the quantity $[\cdot]$ across the interface $\gamma$.

For the solution of (8.29) with (8.26) and (8.27), operator-splitting methods should be considered. These methods (introduced in [115]) are popular and effective for the solution of time-discretization problems. Here, we describe the simple Peaceman-Rachford scheme, but one should be aware that more sophisticated operator-splitting schemes are often worthwhile.

Let $\Delta t$ denote a time step and $u^i$ the approximation to $u(x, i\Delta t)$. We obtain the following scheme

$$u^0 = u_0; \tag{8.31}$$

Assume that the $u^i$ is known up to some positive $i$. With this information we can compute approximations for velocity $u^{i+\frac{1}{2}}$, pressure $p^{i+\frac{1}{2}}$, and Lagrange multipliers $\lambda^{i+\frac{1}{2}}$ for the midpoint between $i\Delta t$ and $(i+1)\Delta t$. This is done by solving the problem

$$\begin{aligned} \frac{2}{\Delta t} \int_\Omega (u^{i+\frac{1}{2}} - u^i) \cdot v dx \quad & + \frac{\nu}{2} \int_\Omega \nabla u^{i+\frac{1}{2}} \cdot \nabla v dx - \int_\Omega p^{i+\frac{1}{2}} \nabla \cdot v dx - \int_{\partial\omega} \lambda^{i+\frac{1}{2}} \cdot v d\partial\omega \\ &= \int_\Omega f^{i+\frac{1}{2}} \cdot v dx - \frac{\nu}{2} \int_\Omega \nabla u^i \cdot \nabla v dx - \int_\Omega (u^i \cdot \nabla) u^i \cdot v dx \end{aligned} \tag{8.32}$$

for all functions $v \in \hat{V}$. The discrete boundary conditions are enforced in a straightforward manner.

$$\begin{aligned} \nabla u^{i+\frac{1}{2}} &= 0 \quad \text{in} \quad \Omega \\ \nabla u^{i+\frac{1}{2}} - u_B &= 0 \quad \text{on} \quad \partial\omega \end{aligned} \tag{8.33}$$

The functions $u^{i+\frac{1}{2}}$ and $p^{i+\frac{1}{2}}$ belong to $\hat{V}$ and $L^2(\Omega)$, respectively.

In the final step, $u^{i+1}$, $p^{i+1}$, and, $\lambda^{i+1}$ are calculated by solving

$$\begin{aligned}\frac{2}{\Delta t}\int_\Omega (u^{i+1}-u^{i+\frac{1}{2}})\cdot v dx \quad &+\frac{\nu}{2}\int_\Omega \nabla u^{i+1}\cdot\nabla v dx-\int_\Omega p^{i+\frac{1}{2}}\nabla\cdot v dx-\int_{\partial\omega}\lambda^{i+\frac{1}{2}}\cdot v d\partial\omega \\ &=\int_\Omega f^{i+1}\cdot v dx-\frac{\nu}{2}\int_\Omega \nabla u^{i+\frac{1}{2}}\cdot\nabla v dx-\int_\Omega (u^{i+\frac{1}{2}}\cdot\nabla)u^{i+1}\cdot v dx\end{aligned} \tag{8.34}$$

where $u^{i+1}\in\hat{V}$.

An important feature is that this splitting treats the advection term without modifications of the conditions on (8.26), the irregularly shaped boundary . The operator-splitting employed here is quite standard. It is a well documented scheme. We will refrain from further comment, referring the reader, instead, to Glowinski and Letallec [69] for a more in-depth discussion of operator-splitting schemes.

## 8.6 Shape optimization for flow problems

We now discuss an optimal-shape design problem which is solved with the numerical method of the previous section, used as a black box. It is a forward-flow simulation for an airfoil. What remains to be done is to define a shape functional for minimization and to choose a discretization (perhaps more appropriately described as the parameterization of the boundary of an irregularly shaped domain, $\partial\omega$).

The symmetric airfoil is defined by its boundary $\partial\omega$. The chord length of the airfoil is allowed to vary in the test problems; the location of both the head and the tail of the airfoil are control variables. We parameterize the shape of the airfoil by 7 values, $\kappa_i$, $i=1,\ldots 7$. Here, as in section 8.5, we parameterize only the upper portion of $\partial\omega$ and construct the lower portion by reflection in the $x_1$ axis. A simple polynomial is used

$$\partial\omega(\kappa)=\{(x_1,x_2)|x_2(x_1)=5\rho(\kappa_1 x_1^{\frac{1}{2}}+\kappa_2 x_1+\kappa_3 x_1^2+\kappa_4 x_1^3+\kappa_5 x_1^4)\}.$$

where $x_1$, $x_2$ are coordinates to the left of the closed trailing edge (see Figure 8.7). Here, the parameter $\rho$ is a thickness parameter (not to be confused with density as used in the previous section). For the NACA0012 airfoil we will take $\rho=.12$. The airfoil is extended by a closed trailing edge (marked by the three 'o' positions in the figure). Both the location of the head and the location of the tail (or, equivalently, the length of the closed trailing edge, denoted by 'C' in Figure 8.7) are control variables $\kappa_6$ and $\kappa_7$, respectively.

The constraint functions, $G(\kappa)$, are purely heuristic. We require that $(\epsilon_w-x_2|_{\partial\omega})\leq 0$ for any $x_2$ not an endpoint of the airfoil. Here, $\epsilon_w$ is a positive parameter, a minimum thickness of the airfoil. (We found that setting $\epsilon_w=10^{-6}$ works well.) This constraint ensures that the upper and lower surface intersect at no more than two places (at

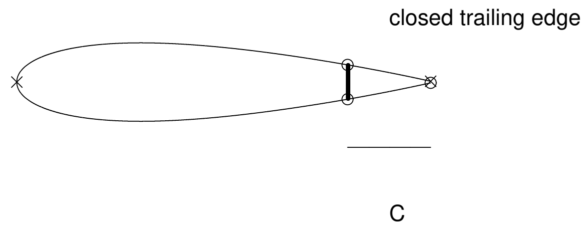


**Figure 8.7** Construction of the closed trailing edge

the two endpoints of the airfoil). Secondly, upper and lower bounds on the physical cross-sectional area of the airfoil are imposed. In practice, the overall effect of these constraints is to trim computational work by keeping the minimization procedure away from obviously impossible airfoil shapes that yield very small but non–optimal values of the objective function. These constraints are important when a large number of design variables are used and the initial guesses for the design variables result in a shape that is not close to optimal. In the case of the results presented here, the constraints caused somewhat shorter run–times because they reduced the number of simulations tested, but they did not drastically affect the minimization process. In part, the short run–times were due to the small number of design variables and the "semi–local" starting value for the design variables (see Figure 8.8 for the initial shape and optimal shapes). It is worth noting that, for this particular test problem, there were no constraints active at the solution (i.e. $G(\kappa^*) < 0$).

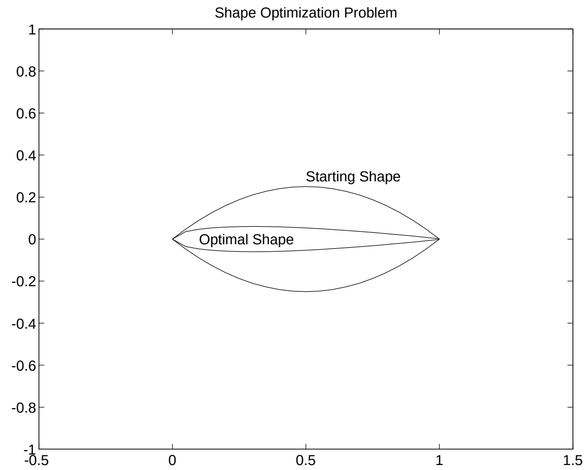


**Figure 8.8** Initial guess shape and solution

An explicit example of this problem is the inverse-pressure identification problem for Navier–Stokes. We will present some results closely related to the results of Glowinski, Kearsley, Pan and Periaux ([67] and [68]). The parameterizations used here are more general than in the cited papers but otherwise the results are very comparable.

Consider a pressure distribution, $p_T$ say, that has been generated by fluid-flow around an obstacle, $\omega_*$. We think of $p_T$ as a target pressure and $\omega_*$ as an optimal shape. The aim is to recover the shape $\omega_*$ by minimizing the shape functional, $S\left(u, p, (\partial\omega)\right) = \|p - p_T\|$, for some well-chosen norm $\|\cdot\|$.

The equations (8.23), (8.24), and (8.26) of the previous section apply here also. The Navier-Stokes equations and boundary conditions which determine $(u, p)$ are

$$\begin{aligned} &\frac{\partial u}{\partial t} - -\nu\triangle u + (u\cdot\nabla)u + \nabla p = f \quad \text{in} \quad \Omega \\ &\nabla\cdot u = 0 \quad \text{in} \quad \Omega \\ &u - u_B = 0 \quad \text{on} \quad \partial\omega \\ &\int_{\partial\omega} u_B \cdot \mathbf{n} d\partial\omega \end{aligned} \tag{8.35}$$

A particular $u_B$ is chosen for this problem to produce a model of the start-up of a wind tunnel in which the upstream fluid velocity starts at zero and approaches a steady value exponentially in time. The details of this $u_B$ are found in ([67]).

We can write our problem in output least-squares form,

$$\begin{aligned} &\min_{\kappa} \|p_T - p\|^2_{L^2(\partial\omega} \\ &\quad \text{subject to:} \\ &G(\kappa) \leq 0. \end{aligned} \tag{8.36}$$

The shape minimization was performed using the algorithm described in section 3 (as was done in the previous section). The parallel nature of the algorithm in section 3 is an important feature here because the Navier-Stokes simulations are expensive calculations. We used 16 processors of the Touchstone Delta machine to perform the minimization in parallel. A two stage coarse/fine strategy was employed to solve the shape-optimization minimization. Initially, in order to locate a rough neighborhood of the solution, 10 points per processor were sampled with a very large search strategy. Later, 25 points per processor were sampled with a smaller pattern search strategy to polish the quality of the solution.

Numerical results are summarized in the table below.

## 8.7 Conclusions

It is clear that the fictitious domain technique, both for the solution of elliptic problems and for the solution of more complicated partial differential equations, is an

| $h_p$ | $h_v$ | $\Delta t$ | Reynolds | $\angle$ of Attack | Function evaluations | Cost Function |
|---|---|---|---|---|---|---|
| 1/64 | 1/128 | 2.5e-3 | 1000 | $0^o$ | 2.5E2 | 4.9E-1 |
| 1/64 | 1/128 | 2.5e-3 | 500 | $0^o$ | 2.5E2 | 1.1E-1 |
| 1/32 | 1/64 | 5.0e-3 | 1000 | $0^o$ | 1.8E2 | 5.7E-1 |
| 1/32 | 1/64 | 5.0e-3 | 500 | $0^o$ | 1.8E2 | 1.3E-1 |
| 1/64 | 1/128 | 2.5e-3 | 1000 | $5^o$ | 1.9E2 | 9.8E-2 |
| 1/64 | 1/128 | 2.5e-3 | 500 | $5^o$ | 1.9E2 | 9.4E-2 |
| 1/32 | 1/64 | 5.0e-3 | 1000 | $5^o$ | 1.5E2 | 1.5E-1 |
| 1/32 | 1/64 | 5.0e-3 | 500 | $5^o$ | 1.5E2 | 1.4E-1 |

**Table 8.7** Coarse minimization: 10 points per processor

| $h_p$ | $h_v$ | $\Delta t$ | Reynolds | $\angle$ of Attack | Function evaluation | Cost Function |
|---|---|---|---|---|---|---|
| 1/64 | 1/128 | 2.5e-3 | 1000 | $0^o$ | 2.1E2 | 1.8E-4 |
| 1/64 | 1/128 | 2.5e-3 | 500 | $0^o$ | 2.1E2 | 2.4E-4 |
| 1/32 | 1/64 | 5.0e-3 | 1000 | $0^o$ | 1.5E2 | 5.7E-4 |
| 1/32 | 1/64 | 5.0e-3 | 500 | $0^o$ | 1.5E2 | 1.3E-4 |
| 1/64 | 1/128 | 2.5e-3 | 1000 | $5^o$ | 2.0E2 | 1.2E-4 |
| 1/64 | 1/128 | 2.5e-3 | 500 | $5^o$ | 2.0E2 | 1.6E-4 |
| 1/32 | 1/64 | 5.0e-3 | 1000 | $5^o$ | 1.6E2 | 1.4E-4 |
| 1/32 | 1/64 | 5.0e-3 | 500 | $5^o$ | 1.6E2 | 1.1E-4 |

**Table 8.8** Fine minimization: 25 points per processor

attractive alternative for use with irregularly-shaped domains. It has become a solution method of choice in a number of different fields.

The fictitious domain method provides an effective way of evaluating the objective function for the Stokes and Navier–Stokes problems of shape-optimization. In particular, the computational cost of regridding a mesh for every trial shape can be a prohibitively expensive. Furthermore, the combination of constrained PDS with FD is effective.

Ideas for future work include extending our shape-optimization algorithm to the solution of more complicated objective functions and to the shape-optimization problems associated with drag reduction. Work that lies ahead includes the proper synthesis of Domain Decomposition ideas with Fictitious Domain techniques for the solution of nonlinear partial differential equations. The combination of FDM with shape-optimization may lead to particularly effective use of parallel computation.

# Chapter 9

# Conclusions

A large class of computational problems arising in industry, science, and engineering can be posed as Optimal Control Problems. Many of these are discretizations of partial differential equations and thus are very large systems. It often happens that these problems are ill-conditioned, limited by feasible regions of difficult shapes or have other complications precluding the use of the fast but delicate algorithms commonly proposed. In this thesis, we have formulated two robust algorithms which will help to fill the consequent need.

The first robust algorithm is a hybrid SQP algorithm which addresses an exigency often encountered when iterating far from the solution, i.e., inconsistency among the linearized constraints. This is a difficulty which must be anticipated when using algorithms equivalent to Newton's method. The algorithm here is based on a set of perturbed problems whose solutions form a sequence converging to the original problem. The effect of the perturbation is to bypass some of the difficulties of the original problem. In Chapter 2, conditions sufficient for global convergence of this hybrid algorithm are proven.

The second robust algorithm addresses the problem encountered when derivatives of functions are not available. This situation can arise when derivatives are not computable or are noisy and ill-conditioned, or just unreasonably expensive to compute, quite a common occurrence in practical applications. Most importantly, the hybrid algorithm presented here is conveniently programmed to fully utilize parallel computation. It avoids premature numerical degeneracy of the simplex as infeasible iterates approach a feasible point and it minimizes chattering or feasible/infeasible cycling which may occur when near the constraint set.

Increasingly, as optimization algorithms are applied to more and industrial problems, research will be needed to devise and elucidate robust algorithms. Invented test problems will no longer be adequate proving grounds for numerical schemes. The ugly systems of the "real world" are appearing in the academic setting. The algorithms here are demonstrated in five examples relating to models of practical problems: boundary control of the heat equation, a hierarchical control problem, A Stefan problem, control with Coulomb friction, and a shape optimization problem. In each case, the formulation is tailored to some special structure of the problem. The variety of successfully treated problems suggests that there is a wide range of useful applications for these algorithms.

# Bibliography

[1] K. M. Anstreicher. On long step path following and SUMT for linear and quadratic programming. Technical Report 32, Yale School of Organization and Management, August 1990.

[2] K. M. Anstreicher, D. den Hertog, and C. Ross. A long step barrier method for convex quadratic programming. *Algorithmica*, 10(5):365, 1993.

[3] C. Atamian, G. V. Dinh, R. Glowinski, J. He, and J. Periaux. On some imbedding methods applied to fluid dynamics and electro–magnetics. *Computer methods in applied mechanics and engineering*, 12:1271–1299, 1993.

[4] M. Avriel. *Nonlinear Programming: Analysis and Methods.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[5] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, New York, 1994.

[6] M. C. Bartholomew-Biggs. Opsqp and opalqp - new implementations of the sequential quadtratic programming approach to constrained optimisation. Technical Report 208, Hatfield Polytechnic, Numerical Optimisation Centre, 1989.

[7] D. Begis and R. Glowinski. Application de la méthod des éléments finis à l'approximation d'un probléme de domaine optimal. méthodes de résolution des problémes approchés. *Applied Mathematics and Optimization*, 2:130–169, 1975.

[8] P. T. Boggs, P. D. Domich, J. R. Donaldson, and C. Witzgall. Algorithmic enhancements to the method of centers for linear programming. *ORSA Journal on Computing*, 1(3):159–171, 1989.

[9] P. T. Boggs, P. D. Domich, J. E. Rogers, and C. Witzgall. An interior point method for linear and quadratic programming problems. *Mathematical Programming Society Committee on Algorithms Newsletter*, 19:32–40, 1991.

[10] P. T. Boggs, P. D. Domich, J. E. Rogers, and C. Witzgall. An interior point method for general large scale quadratic programming problems. Technical Report 5406, National Institute of Standards and Technology, Gaithersburg, MD. 20899, 1994.

[11] P. T. Boggs and J. Tolle. A family of descent functions for constrained optimization. *SIAM Journal of Numerical Analysis*, (21), 1984.

[12] P. T. Boggs and J. Tolle. Sequential quadratic programming. *Acta Numerica*, pages 1–54, 1995.

[13] P. T. Boggs, J. W. Tolle, and A. J. Kearsley. A merit function for inequality constrained nonlinear programming problems. Technical Report 4702, National Institute of Standards and Technology, Gaithersburg, MD. 20899, 1991.

[14] P. T. Boggs, J. W. Tolle, and A. J. Kearsley. A practical algorithm for general large scale nonlinear optimization problems. Technical Report 5704, National Institute of Standards and Technology, Gaithersburg, MD. 20899, 1994. To appear in SIAM Journal of Optimization.

[15] P. T. Boggs, J. W. Tolle, and A. J. Kearsley. A truncated sqp algorithm for large scale nonlinear programming problems. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis: Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, pages 69–77, Boston, MA, 1994. Kluwer Academic Publishers.

[16] P. T. Boggs, J. W. Tolle, and A. J. Kearsley. A basic global convergence analysis of a large scale algorithm for general nonlinear programming problems. Internal report, National Institute of Standards and Technology, 1995.

[17] P. T. Boggs, J. W. Tolle, and A. J. Kearsley. On the convergence of a trust region SQP algorithm for nonlinearly constrained optimization problems. In J. Dolezal, editor, *Proceedings of the 17th IFIP TC7 Conference on System Modeling and Optimization*, page (to appear), London, 1995. Chapman.

[18] G. E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, 6:81–101, 1957.

[19] J. A. Burns and S. Kang. A control problem for Burgers equation with bounded input/output. Technical Report 90–45, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Va. 23665, 1990.

[20] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi–newton matrices and their use in limited memory methods. *Math Prog.*, 63:129–156, 1994.

[21] H. Cabannes. Study of the motions of a vibrating string subject to solid friction. *Math. Meth. of Appl. Sci.*, 3:287–300, 1981.

[22] L. T. Campos, J. T. Oden, and N. Kikuchi. A numerical analysis of a class of contact problems with friction in elastodynamics. *Comp. Meth. Appl. Eng.*, 34:821–845, 1982.

[23] C. Carthel, R. Glowinski, and J. L. Lions. On exact and approximate boundary controllabilities for the heat equation. a numerical approach. *Jounal of Optimization Theory and Applications*, 82(3):429–484, 1994.

[24] J. Céa. *Optimisation: Théorie et Algorithmes*. Dunod, Paris, 1971.

[25] M. Celis, J. E. Dennis, and R. A. Tapia. A trust region strategy for nonlinear equality constrained optimization. In P. T. Boggs, R. Byrd, and R. Schnabel, editors, *Numerical Optimization 1984*, pages 71–82. SIAM, Philadelphia, PA, 1985.

[26] R. M. Chamberlain. Some examples of cycling in variable metric algorithms for constrained minimization. *Mathematical Programming*, 16:378–383, 1979.

[27] R. M. Chamberlain, C. Lemarechal, H. C. Pedersen, and M. J. D. Powell. The watchdog technique for forcing convergence in algorithms for constrained optimization. *Mathematical Programming Studies*, 16:1–17, 1982.

[28] T. Chan and T. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[29] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1978.

[30] T. F. Coleman. Large scale numerical optimization: Introduction and overview. In *Encyclopedia of Computer Science and Technology*, New York, 1991. Marcel Dekker, Inc.

[31] T. F. Coleman and J. Liu. An interior Newton method for quadratic programming. Technical Report TR 93–1388, Cornell University, Ithaca, NY, 1993.

[32] A. R. Conn. Nonlinear programming, exact penalty functions and programming techniques for non-smooth functions. In P. T. Boggs, R. Byrd, and R. Schnabel, editors, *Numerical Optimization 1984*, pages 3–25, Philidelphia, PA, 1985. SIAM.

[33] A. R. Conn, N. I. M. Gould, and P. L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization*. Springer–Verlag, Berlin, Heidelberg, New-York, Tokyo, 1992.

[34] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience Publishers, New York, 1953.

[35] L. C. Cowsar, R. Glowinski, A. J. Kearsley, R. A. Tapia, M. F. Wheeler, and I. Yotov. A novel optimization approach to multiphase flow. Subsurface Modeling Group Industrial Affiliates Meeting, August 1994.

[36] L. C. Cowsar and M. F. Wheeler. Parallel domain decomposition method for mixed finite elements for elliptic partial differential equations. In R. Glowinski, Y. Kuznetsov, G. Meurant, and J. Périaux, editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philidelphia, PA, 1991. SIAM.

[37] S. J. Cox and M. Ross. Extremal eigenvalue problems for starlike drums. In M.P. Bendsoe and C.A. Mota Soares, editors, *Proceedings of the NATO Advanced Research Workshop on Topology Design of Structures*. Kluwer, 1993.

[38] J. Crank. *Free and Moving Boundary Problems*. Oxford University Press, Oxford, 1984.

[39] E. J. Dean, R. Glowinski, Y. M. Kuo, and G. Nasser. Multiplier techniques for some dynamical systems with dry friction. *C. R. Acad. Sci. (Paris)*, 314:153–159, 1992.

[40] E. J. Dean, R. Glowinski, Y. M. Kuo, and M. G. Nasser. On the discretization of some second order in time differential equations. applications to nonlinear wave problems. In A.V. Balakrishnan, editor, *Computational Techniques in Identification and Control of Flexible Flight Structures*, pages 199–246. Optimization Sofware Inc., 1990.

[41] E.J. Dean and P. Gubernatis. Pointwise control of Burgers' equation—a numerical approach. *Computers & Mathematics, with Applications*, 22(7):93–100, 1991.

[42] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.

[43] R. S. Dembo and U. Tulowitzki. Sequential truncated quadratic programming methods in numerical optimization. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 83–101. SIAM, Philadelphia, PA, 1985.

[44] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ., 1983.

[45] J. E. Dennis and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1:448–474, 1991.

[46] P. D. Domich, P. T. Boggs, J. E. Rogers, and C. Witzgall. Optimizing over three-dimensional subspaces in an interior-point method for linear programming. *Linear Algebra and its Applications*, 152, July 1991.

[47] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, 1986.

[48] G. Duvaut. Résolution d'un problème de stefan. *C. R. Acad. Sci. Paris*, 276:1461–1463, 1973.

[49] G. Duvaut and J. L. Lions. *Inequalities in Mechanics and Physics.* Springer-Verlag, Berlin, 1976.

[50] S. C. Eisenstadt and H. F. Walker. Globally convergent inexact newton methods. *SIAM Journal on Optimization*, 4:393–422, 1994.

[51] I. Ekeland and R. Temam. *Convex analysis and variational problems.* American Elsevier Pub. Co., 1976.

[52] M. M. El-Alem. A robust trust region algorithm with nonmonotonic penalty parameter scheme for constrained optimization. Technical Report TR-92-30, Department of Computational and Applied Mathematics, Rice University, P.O. Box 1892, Houston, Tx 77251-1892, 1992.

[53] A. El-Bakry, T. Tsuchyia, R. A. Tapia, and Y. Zhang. On the formulation and theory of the primal-dual Newton interior-point method for nonlinear programming. Technical Report # 92-40, Department of Computational and Applied Mathematics, Rice University, P.O. Box 1892, Houston, Tx 77251-1892, 1992.

[54] S. K. Eldersveld. Large-scale sequential quadratic programming algorithms. Technical Report SOL 92-4, Dept. Operations Research, Stanford University, 1992.

[55] C. M. Elliot and J. R. Ockendon. *Weak and Variational Methods for Moving Boundary Problems.* Pitman Pub., Boston, 1982.

[56] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Academic Press, New York, 1983.

[57] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, Inc., New York, 1968.

[58] R. Fletcher. Function minimization without evaluating derivatives– a review. *The Computer Journal*, 8:33–41, 1965.

[59] R. Fletcher. An exact penalty function for nonlinear programming with inequalities. *Mathematical Programming*, 5:129–150, 1973.

[60] R. Fletcher. An $\ell_1$ penalty function method for nonlinear constraints. In P. T. Boggs, R. Byrd, and R. Schnabel, editors, *Numerical Optimization 1984*, pages 26–40, Philidelphia, PA, 1985. SIAM.

[61] C. A. Floudas and P. M. Pardalos. *A collection of test problems for constrained global optimization algorithms.* Springer-Verlag, New York, 1990.

[62] J. Gauvin and J. W. Tolle. Differential stability in nonlinear programming. *SIAM Journal on Control and Optimization*, 15:294–311, 1977.

[63] P. E. Gill, W. Murray, M. Saunders, and M. H. Wright. Some theoretical properties of an augmented Lagrangian merit function. Technical Report 86-6, Dept. Operations Research, Stanford University, Stanford, California 94305, 1986.

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

[65] R. Glowinski. *Numerical Methods for Nonlinear Variational Problems.* Springer-Verlag, New York, 1984.

[66] R. Glowinski and A. J. Kearsley. On the simulation and control of some friction constrained motions. Technical Report # 93-21, Department Computational & Applied Mathematics, Rice Unversity, P. O. Box 1982, Houston TX. 77251-1892, 1993. to appear *SIAM Journal on Optimization.*

[67] R. Glowinski, A. J. Kearsley, T-W. Pan, and J. Periaux. Numerical simulation and optimal shape for viscous flow by a fictitious domain method. *International Journal for Numerical Methods in Fluids*, 20:695–711, 1995.

[68] R. Glowinski, A. J. Kearsley, T-W. Pan, and J. Periaux. Numerical simulation and optimal shape for viscous flow by a fictitious domain method. Technical Report TR95–16, Center for Research on Parallel Computation, Rice Unversity, P. O. Box 1982, Houston TX. 77251-1892, 1995. to appear *Computational Fluid Dynamics Review.*

[69] R. Glowinski and P. LeTallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics.* SIAM, Philadelphia, Pa, 1989.

[70] R. Glowinski and J.L. Lions. Exact and approximate controllability for distributed parameter systems i. *Acta Numerica*, pages 269–378, 1994.

[71] R. Glowinski and J.L. Lions. Exact and approximate controllability for distributed parameter systems ii. *Acta Numerica*, pages 159–333, 1995.

[72] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain method for external incompressible viscous flow modeled by Navier–Stokes equations. *Comp. Meth. Appl. Mech. Eng.*, 112:133–148, 1994.

[73] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain method for the Dirichlet problem and applications. *Comp. Meth. Appl. Mech. Eng.*, 111:282–303, 1994.

[74] R. Glowinski, T-W. Pan, and J. Periaux. A fictitious domain/domain decomposition methods for partial differential equations. In *Domain–based parallelism and problem decomposition methods in computautional science and engineering*, Philadelphia, PA, 1995. SIAM.

[75] R. Glowinski and M. F. Wheeler. Domain decomposition and mixed finite element methods for elliptic problems. In R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 144–172, Philadelphia, 1988. SIAM.

[76] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1989.

[77] C. C. Gonzaga. Large-steps path-following methods for linear programming, part i: Barrier function method. *SIAM Journal on Optimization*, 1(2):268–279, May 1991.

[78] C. C. Gonzaga. Large-steps path-following methods for linear programming, part ii: Potential reduction method. *SIAM Journal on Optimization*, 1(2):280–292, May 1991.

[79] M. D. Gunzburger, L. S. Hou, and T. P. Svobodny. Numerical approximation of an optimal control problem associated with the Navier-Stokes equation. *Appl. Math. Lett. 2*, 1:29–31, 1989.

[80] S. B. Hamma. *Etude de Methodes Numeriques d'Optimisation Globale*. Phd thesis, L'Universite Paul Sabatier de Toulouse (sciences), 1992.

[81] S. P. Han. Variable metric methods for minimizing a class of nondifferentiable functions. *Mathematical Programming*, 20:1–13, 1981.

[82] J. Haslinger and P. Neittaanmáki. *Finite Element Approximation for optimal shape design*. John Wiley & Sons, New York, 1988.

[83] M. Heinkenschloss and E. W. Sachs. Numerical solution of a constrained control problem for a phase field model. Technical Report ICAM Report 94-05-02, Interdisciplinary Center for Applied Mathematics ICAM, Virginia Polytechnic Institute and State University, Blacksburg, Va. 24061-0531, 1994.

[84] N. J. Higham. Optimization by direct search methods in matix computations. *SIAM Journal on Matrix Analysis and Applications*, 14(2):317–333, 1993.

[85] W. Hock and K. Schittkowski. *Test examples for nonlinear programming codes.* Springer–Verlag, Berlin, 1981.

[86] K. H. Hoffmann and W. Krabbs. *Optimal Control of Partial Differential Equations: Proceedings of the IFIP WG 7.2 International Conference, Irsee.* Springer-Verlag, New York, 1991.

[87] K. H. Hoffmann and J. Sprekels. Real–time control in a free boundary problem connected with the continuous casting of steel. In Hoffmann and Krabs, editors, *Optimal Control of PDE's.* Birkhauser, 1984.

[88] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *J. Assoc. Comput. Mach.*, 8:212–229, 1961.

[89] P. Huard. Resolution of mathematical programming with nonlinear constraints by the method of centers. In J. Abadie, editor, *Nonlinear Programming*, pages 209–219. North-Holland Pub. Co., Amsterdam; Wiley, New York, 1967.

[90] F. Jarre. Interior point methods for convex programming. Technical Report Report SOL 90–16, Department of Operations Research, Stanford University, Stanford, California, 1990.

[91] D. D. Joseph. Parameter and domain dependence of eigenvalues of elliptic partial differential equations. *Arch. of Rat. Mech. Anal.*, 24:35–58, 1967.

[92] A. J. Kearsley, R. A. Tapia, and V. J. Torczon. On the use of parallel direct search methods for nonlinear programming problems. Technical Report # 93-33, Department of Computational & Applied Mathematics, Rice University, P. O. Box 1892, Houston TX. 77251–1892, 1993.

[93] C. T. Kelley and E. W. Sachs. Solution of optimal control problems by a pointwise projected Newton method. *SIAM Journal on Control and Optimization*, 33:1731–1757, 1995.

[94] C. T. Kelley and S. J. Wright. Sequential quadratic programming for certain parameter identification problems. *Mathematical Programming*, 51:281–305, 1991.

[95] N. Kikuchi and J. T. Oden. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods.* SIAM, Philadelphia, Pa, 1988.

[96] P. LeTallec. Domain decomposition methods in computational mechanics. *Computational Mechanics Advances*, 1:121–220, 1994.

[97] J. L. Lions. *Optimal Control of Systems Governed by Partial Differential Equations.* Springer-Verlag, Berlin, 1971.

[98] J. L. Lions. Exact controllability, stabilization and perturbations for distributed systems. *SIAM Review*, 30:1–68, 1988.

[99] J. L. Lions. Hierarchical control. *Proceedings of the Indian Academy of Sciences (Mathematical Sciences)*, 104(1):295–304, February 1994.

[100] J. L. Lions and E. Magenes. *Non-homogeneouss Boundary Value Problems and Applications I.* Springer-Verlag, New York, 1972.

[101] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Prog*, 45:503–528, 1989.

[102] D. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, Massachusetts, 1989.

[103] N. Maratos. *Exact penalty function algorithms for finite dimensional and control optimization problems.* P.h.d. thesis, University of London, 1978.

[104] R. D. C. Monteiro. A globally convergent primal–dual interior point algorithm for convex programming. *Mathematical Programming*, 64(2):123–147, 1994.

[105] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4:553 – 572, 1983.

[106] B. T. Murray, A. A. Wheeler, and M. E. Glicksman. Simulations of experimentally observed dendritic growth behavior using a phase-field model. *Journal of Crystal Growth*, 2:38–56, 1995.

[107] W. Murray and F. J. Prieto. A sequential quadtratic programming algorithm using an incomplete solution of the subproblem. Technical Report SOL 93–3, Stanford University, Dept. Operations Research, 1994.

[108] S. G. Nash and J. Nocedal. A numerical study of the limited memory BFGS method and the truncated–Newton method for large scale optimization. *SIAM Journal on Optimization*, 1:358–372, 1991.

[109] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[110] A. J. Silva Neto and R. E. White. Numerical control of the Stefan problem: maximum melting. *Computer Meth. in Appl. Mech. and Engineering*, 113:351–362, 1994.

[111] R. Nickel and J. W. Tolle. A sparse sequential quadratic programming algorithm. *Journal of Optimization Theory and Applications*, 60(3):453–473, 1989.

[112] J. Nocedal. Updating quasi-newton matrices with limited storage. *Math of Computation*, 35(151):773–782, 1980.

[113] P. M. Pardalos and J. B. Rosen. *Constrained global optimization : algorithms and applications*. Springer–Verlag, New York, 1987.

[114] I. Pawlow. Optimal control of two–phase Steffan problems – numerical solutions. In Hoffmann and Krabs, editors, *Optimal Control of PDE's II*. Birkhauser, 1987.

[115] D. H. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *J. Soc. Ind. Appl. Math.*, 3:28–41, 1955.

[116] O. Pirroneau. *Shape Optimization*. Springer–Verlag, New York, 1984.

[117] R. Polak. Shifted barrier functions (theory and methods). *Mathematical Programming*, 54:177–222, 1992.

[118] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Proceedings of the 1977 Dundee Biennial Conference on Numerical Analysis*, Berlin, 1977. Springer–Verlag.

[119] M. J. D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In Susana Gomez and Jean-Pierre Hennart, editors, *Advances in Optimization and Numerical Analysis: Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, pages 51–67, Boston, MA, 1994. Kluwer Academic Publishers.

[120] M. Renardy. Ill-possedness at the boundary for elastic solids sliding under coulomb friction. *Journal of Elasticity*, 27:281–287, 1992.

[121] J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

[122] S. Robinson. Perturbed Kuhn–Tucker points and rates of convergence for a class of nonlinear-programming algorithms. *Mathematical Programming*, 7:1–16, 1974.

[123] S. Robinson. Stability theory for systems of inequalities. part I: Linear systems. *SIAM Journal on Numerical Analysis*, 12(5):754–769, 1975.

[124] R. T. Rockafellar. *Convex analysis.* Princeton University Press, 1970.

[125] B. Rousselet. Optimal design and eigenvalue problems. In J. Stoer, editor, *Optimization techniques : proceedings of the 8th IFIP Conference on Optimization Techniques, Wurzburg, September 5-9, 1977*, pages 343–352, New York, 1978. Springer-Verlag.

[126] B. Rousselet. Shape design sensitivity of a membrane. *Journal of Optimization Theory and Applications*, 40(4), 1983.

[127] L. I. Rubinstein. *The Stefan Problem.* American Mathematical Society, 1971. translation from Russian by A. D. Solomon.

[128] J. Simon. Differentiation with respect to the domain in boundary value problems. *Numerical Functional Analysis and Optimization*, 2(7–8):649–687, 1980.

[129] J. Sokolowski and J. P. Zolesio. *Introduction to Shape Optimization: Shape Sensitivity Analysis*, volume 16 of *Springer Series in Computational Mathematics.* Springer-Verlag, 1992.

[130] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, San Diego, California, 1973.

[131] M. B. Subrahmanyam. An extension of the simplex method to constrained nonlinear optimization. *J.O.T.A.*, 62(2):311–320, 1889.

[132] W. H. Swann. Direct search methods. In W. Murray, editor, *Numerical Methods for Unconstrained Optimization*, pages 13–28. Academic Press, New York, 1972.

[133] W. W. Symes. A differential semblance criterion for inversion of multioffset seismic reflection data. *J. Geophys. Res.*, 98:2061–2073, 1993.

[134] K. Tanabe. An algorithm for the constrained maximizatioon in nonlinear programming. *J. Operations Research Soc. of Japan*, 17(4):184–201, 1974.

[135] R. A. Tapia. Augmented Lagrangian methods for constrained optimization: the role of the penalty constant. In S. Inceri and G. Treccani, editors, *Ottimizzazioni non lineare e applicazioni*, Bologna, Italy, 1980. The International Conference on Nonlinear Optimization and Applications in L'Aquuila, Italy, Pitagora Editrice.

[136] R. A. Tapia. On the role of slack variables in quasi-Newton methods for constrained optimization. In L. C. W. Dixon and G. P. Szegó, editors, *Numerical Optimization of Dynamical Systems*, pages 235–246. North-Holland, Amsterdam, 1980.

[137] R. A. Tapia and J. R. Thompson. *Nonparametric Probability Density Estimation.* Johns Hopkins, Baltimore, 1978.

[138] J. R. Thompson and R. A. Tapia. *Nonparametric function estimation, modeling, and simulation.* SIAM, Philadelphia, 1990.

[139] K. Tone. Revisions of constraint approximations in the successive QP method for nonlinear programming problems. *Mathematical Programming*, 26:144–152, 1983.

[140] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines.* Ph.d. thesis, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston TX. 77251–1892, 1989.

[141] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1:123–145, 1991.

[142] V. Torczon. PDS: Direct search methods for unconstrained opti mization on either sequential or parallel machines. Technical Report 92–9, Department of Mathematical Sciences, Rice University, Houston, TX 77251–1892, 1992. Submitted to *acm Transactions on Mathematical Software.*

[143] V. Torczon. On the convergence of pattern search methods. Technical Report TR93-10, Department of Mathematical Sciences, Rice University, P. O. Box 1892, Houston TX. 77251–1892, 1993.

[144] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior-point methods. *Mathematical Programming*, 58:1–32, 1993.

[145] A. Vardi. A trust region algorithm for equality constrained minimization: convergence properties and implementation. *SIAM Journal on Numerical Analysis*, 22:575–591, 1985.

[146] H. von Stackelberg. *Marktform und Gleichgewicht.* J. Springer, Vienna, 1934.

[147] J. A. Warren and W. J. Boettinger. Prediction of dendtritic growth and microsegregation patterns in a binary alloy using the phase-field method. *Acta Metall. Mater.*, 43(2):689–703, 1995.

[148] A. A. Wheeler, N. A. Ahmad, W. J. Boettinger, R. J. Braun, G. B. McFadden, and B. T. Murray. Recent developments in phase-field models of solidification. *Advances in Space Research*, 16:163–172, 1994.

[149] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, pages 341–407, 1992.

[150] S. J. Wright. Interior point methods for optimal control of discrete-time systems. *J. O. T. A.*, 77:161–187, 1993.

[151] H. Yabe, N. Yamaki, and S. Takahashi. Global convergence of sequential inexact qp method for constrained optimization. *S.U.T. Journal of Mathematics*, 27(1):127–138, 1991.

[152] W. I. Zangwill. Minimizing a function without calculating derivatives. *Comput. J.*, 10(3):293–296, 1967.