NORTHWESTERN UNIVERSITY

Inexact Sequential Quadratic Programming Methods for Large-Scale Nonlinear Optimization

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Industrial Engineering and Management Sciences

By

Frank Edward Curtis

EVANSTON, ILLINOIS

June 2007

ABSTRACT

Inexact Sequential Quadratic Programming Methods for
Large-Scale Nonlinear Optimization

Frank Edward Curtis

This thesis concerns the development of robust algorithms for large-scale nonlinear programming. Despite recent advancements in high-performance computing power, classes of problems exist that continue to challenge the practical limits of contemporary optimization methods. The focus of this dissertation is the design and analysis of algorithms intended to achieve economy of computation for the solution of such problems, where, for ease of presentation, the discussion is framed in the context of equality constrained optimization.

The first part of this thesis concerns the development of a globally convergent inexact Sequential Quadratic Programming (SQP) framework. The novelty of the approach is that it is matrix-free (i.e., only mechanisms for computing products of vectors with Jacobian and Hessian matrices are required), thereby avoiding a need for the explicit formation of the arising iteration matrices. Iterative linear algebra techniques, for example, can then be used in place of factorization methods, allowing the introduction of inexactness into the step computation process to achieve further savings in computation. The algorithm automatically determines when a given inexact SQP step makes sufficient progress toward a solution of the nonlinear program, as measured by an exact penalty function, in order to ensure global convergence to a first order optimal point. An analysis of the global behavior of the algorithm is presented under common conditions, and numerical results are presented for a large collection of test problems and two realistic applications. Finally, algorithmic

enhancements are provided for cases where certain convexity assumptions about the problem formulation may fail to hold.

In the latter part of this thesis, a new globalization mechanism is proposed. The method expands the definition of a standard penalty function so that during each iteration the penalty parameter can be chosen as any number within a prescribed interval, rather than a fixed value. This increased flexibility in the step acceptance procedure is designed to promote long productive steps and fast convergence. An analysis of the global convergence properties of the mechanism in the context of a line search SQP method and numerical results for the `KNITRO` software package are presented.

ACKNOWLEDGMENTS

I would like to thank first, and above all, Jorge Nocedal. Far be it from me to try and capture, in only a few short lines, the level of guidance and support that he has provided for me over the past few years. Suffice it to say that I have been extremely privileged to have Jorge as an advisor and friend, and I would like nothing more than for us to maintain a close professional and personal relationship for many years to come.

I also thank the remaining members of my committee — Sanjay Mehrotra, Robert Fourer, and Richard Waltz — for sharing their knowledge and experience with me over the last few years. Along with Collette Coullard, Sanjay and Robert taught me the groundwork for much of the technical work in this thesis. They, along with my undergraduate advisors Chi-Kwong Li and Rex Kincaid, also provided me with inspirational models that have furthered my desire to pursue a career in academia. Richard has been a good friend and I would like to thank him and Todd Plantenga for access to, and technical support for, `KNITRO` source code.

Any scientific accomplishments contained in these pages would not have been possible without the knowledge and expertise of Richard Byrd. I am honored to have had the opportunity to work closely with Richard and would like to thank him for all of the stimulating conversations we have had about this and other work. I am also grateful to Eldad Haber for having confidence in me and taking the time to explain concepts that have greatly enhanced my understanding of a range of topics, and to Nick Gould for enlightening e-mail correspondence and much appreciated encouragement. Also, I would like to take this opportunity to especially thank my officemates and good friends, Gabriel López-Calva and Long Hei, for helping me out so much over the past few years, and Andreas Wächter and Sven Leyffer, for exhibiting to me the levels of creativity and intelligence in a researcher that I can only hope to achieve.

Further, I am forever grateful to the following people (arranged, naturally, from tallest to shortest): my father, Frank R. Curtis, for his constant interest and encouragement despite the fact that it has taken me all four of these years to explain what it is that I do; Michael Tampa, for his countless stories; my younger brother, Michael Curtis, and older sister, Lauren Maskin, for giving me people to look up to; Andrew Brincefield, for being as dependable a friend as you can get; Beth Hochman, for laughing at my stupid jokes when my mind has been too strained to be clever; my North Salem friends (whose average height, I'm guessing, fits in around here), for our annual reunions; Elisabeth Janßen, for knowing exactly what I've been going through; Meghan Davison, for helping me get here; and, of course, my mother, Cynthia Curtis, for, amongst so many other things, always making sure that my accomplishments over the past four years aren't limited to only the work in these pages.

Finally, I would like to make a special note to thank my lunch buddy, Kristin Sahyouni, for keeping me on track these last four years, but especially for helping to preserve my sanity during times when so many things seem so unbelievable!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A persistent challenge in the field of nonlinear optimization has been the design of algorithms for very large problems that test the limits of available computing machinery. Recent decades have witnessed significant advances in all types of mathematical programming algorithmic design that, along with the maturation of a variety of optimization software tools, have greatly expanded the range of applications that can be solved efficiently and reliably. However, certain classes of problems exist that continue to pose a number of computational obstacles. A prominent example is the class of problems where the constraints are given by a discretized set of partial differential equations (PDEs), as in such cases the dimension of the problem may increase ad infinitum with the fineness of a grid. For the solution of problems of this type, algorithm developers are faced with the challenge of designing methods that can achieve economy of computation when applied to problems of ever increasing complexity.

In this dissertation we present techniques for enhancing the computational efficiency of certain nonlinear programming techniques in two ways:

- First, we develop, analyze, and implement a globally convergent inexact Sequential Quadratic Programming (SQP) framework. The novelties of the approach are that it is

matrix-free (i.e., only mechanisms for calculating products of vectors with Hessian and Jacobian matrices are required) and that it allows for the introduction of inexactness into the step computation procedure.

- Second, we propose a new globalization strategy that extends the definition of a standard exact penalty function as a tool for guiding convergence. The method proposed is designed to inhibit the computed search direction as little as possible in order to promote long productive steps and fast convergence.

Overall, we are interested in methods that are able to compute steps in a relatively cheap manner, and in globalization strategies that allow the search to roam freely, yet quickly, to a solution point.

The computational benefits of inexact methods have been studied extensively in the contexts of unconstrained optimization and the solution of systems of nonlinear equations. Moreover, techniques of this type have been implemented in many popular software packages and have proved to be quite successful. The same cannot be said, however, about inexact methods for constrained optimization. This is due to the difficulty of designing algorithms that deal effectively with the paired goals of minimizing the objective and satisfying the constraints. In this dissertation, we provide methods that maintain the global convergence properties of contemporary algorithms while being economical enough for very large applications.

The structure of this dissertation is as follows. Chapter 2 includes relevant background on the theory of equality constrained optimization that forms the basis of the discussions in later chapters. In Chapter 3 we develop and analyze our proposed inexact SQP method and investigate its global behavior under common conditions. Chapter 4 presents some numerical results for an implementation of our inexact SQP method applied to a large set of test problems and a pair of realistic PDE-constrained applications. We present algorithmic

enhancements to the method in Chapter 5 for cases where certain convexity assumptions related to the problem formulation may not hold. Finally, in Chapter 6 we present, analyze, and provide numerical results for a new globalization strategy before presenting final remarks and comments on extending all of the methods in this dissertation to generally constrained problems and other algorithmic frameworks in Chapter 7.

Chapter 2

Equality Constrained Optimization

2.1 Problem Formulation

We frame this dissertation in the context of the equality constrained optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & c(x) = 0, \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^t$ are smooth nonlinear functions, but consider ways in which our methods can be extended to general nonlinear programming problems in Chapter 7. We are particularly interested in problems where the number of variables n and the number of constraints t, with $t \leq n$, are very large.

Algorithms for solving problem (2.1) often focus on producing first order optimal points, which can be defined in the following manner. First, the Lagrangian function corresponding to problem (2.1) is

$$\mathcal{L}(x, \lambda) \triangleq f(x) + \lambda^T c(x), \tag{2.2}$$

where $\lambda \in \mathbb{R}^t$ are Lagrange multipliers. If the functions f and c are continuously differen-

tiable, then a point x^* is first order optimal if there exist multipliers λ^* such that (x^*, λ^*) is a solution to the nonlinear system of equations

$$\nabla \mathcal{L}(x, \lambda) = \begin{bmatrix} g(x) + A(x)^T \lambda \\ c(x) \end{bmatrix} = 0, \tag{2.3}$$

where $g(x) \triangleq \nabla f(x)$ is the gradient of the objective function and $A(x)$ is the Jacobian of $c(x)$. The components in (x, λ) are referred to as the primal and dual variables, respectively.

In this dissertation, we are primarily concerned with globally convergent algorithms; i.e., methods that are guaranteed, under certain common assumptions, to converge to a first order optimal solution from remote starting points. Most globally convergent iterative algorithms for problem (2.1) have the following general form. First, at a given iterate x_k, a step is computed in either the primal or primal-dual space based on local and/or historical information of the problem functions. The step is then either accepted or rejected based on the reductions attained in the nonlinear objective $f(x)$, a constraint infeasibility measure $\|c(x)\|$, or some combination of both quantities. Here, $\|\cdot\|$ denotes a norm on $\mathbb{R}^t$. Such a globalization strategy, i.e., step acceptance method, is intended to direct the search toward solutions to (2.3) that correspond to local solutions of the nonlinear program (2.1). In the remainder of this chapter we describe one step computation method and the motivation behind two contemporary globalization mechanisms that form the basis of our proposed techniques.

Notation. In the remainder of this dissertation, we drop functional notation once values are clear from the context and use a subscript to delimit iteration number information of an optimization algorithm for functions and variables; i.e., we denote $f_k \triangleq f(x_k)$ as the objective function value corresponding to the kth iterate x_k, and similarly for other quantities. All norms are considered Euclidean (or l_2) norms unless otherwise indicated, though much of

our analysis will apply for any norm.

2.2 Sequential Quadratic Programming

One of the leading methods for solving constrained optimization problems is sequential quadratic programming (SQP). (In fact, modern interior point methods reduce to SQP when inequality constraints are not present in the problem formulation [28].) Algorithms in this class enjoy global convergence guarantees and typically require few iterations and function evaluations to locate a solution point.

Let us formalize a basic SQP approach for use throughout our discussion. From a given iterate x_k, the SQP methodology applied to problem (2.1) defines an appropriate displacement d_k in the primal space as the minimizer of a quadratic model of the objective subject to a linearization of the constraints. The quadratic program can be defined as

$$\min_{d \in \mathbb{R}^n} \quad f(x_k) + g(x_k)^T d + \tfrac{1}{2} d^T W(x_k, \lambda_k) d \tag{2.4a}$$

$$\text{s.t.} \quad c(x_k) + A(x_k) d = 0, \tag{2.4b}$$

where

$$W(x, \lambda) \approx \nabla^2_{xx} \mathcal{L}(x, \lambda) = \nabla^2_{xx} f(x) + \sum_{i=1}^{m} \lambda^i \nabla^2_{xx} c^i(x) \tag{2.5}$$

is equal to, or is a symmetric approximation for, the Hessian of the Lagrangian. Here, $c^i(x)$ and λ^i denote the ith constraint function and its corresponding dual variable, respectively. If the constraint Jacobian $A(x_k)$ has full row rank and $W(x_k, \lambda_k)$ is positive definite on the null space of $A(x_k)$, then a solution to (2.4) is well defined in this context. An alternative characterization of the SQP step d_k is given by the fact that it can equivalently be obtained

under similar assumptions as part of the solution to the primal-dual system (see [28]):

$$\begin{bmatrix} W(x_k, \lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} g(x_k) + A(x_k)^T \lambda_k \\ c(x_k) \end{bmatrix}. \tag{2.6}$$

We remark that the primal-dual matrix in the left-hand-side of this expression is symmetric and indefinite.

2.3 Globalization Techniques

Upon the calculation of an appropriate step (i.e., search direction), a globalization procedure must be invoked to ensure that the algorithm will converge to a first order optimal solution from remote starting points. We discuss the basic forms of two popular tools used for this purpose, penalty functions and filter mechanisms, for use throughout our discussion. We then present further details related to the implementation of a line search SQP approach that employs a penalty function to promote convergence — the method that forms the basis of the algorithms in this dissertation.

A penalty function combines the nonlinear objective and a constraint infeasibility measure into a function of the form

$$\phi_\pi(x) \triangleq f(x) + \pi \|c(x)\|, \tag{2.7}$$

where $\pi \geq 0$ is a penalty parameter. During iteration k, a step is deemed acceptable only if a sufficient reduction in ϕ_{π_k} is attained for a suitable value π_k of the penalty parameter. In current algorithms, the sequence $\{\pi_k\}$ is typically monotonically increasing throughout the run of the algorithm. Figure 2.1 illustrates the region of acceptable points from $p_k = (\|c(x_k)\|, f(x_k))$, corresponding to the current iterate x_k, in $\|c\|$-f space. A step d_k is acceptable if the resulting point $\bar{x} = x_k + d_k$ yields a pair $(\|c(\bar{x})\|, f(\bar{x}))$ lying sufficiently

below the solid line through p_k, where the slope of the line is defined by the current value of the penalty parameter π_k. The global convergence properties of such an approach were first shown in [22, 30].

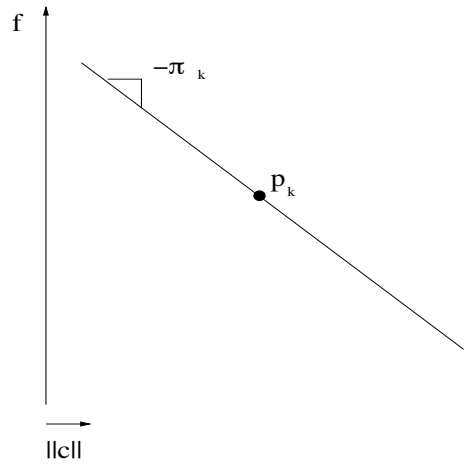

Figure 2.1: Region of acceptable points from p_k for the penalty function ϕ_{π_k}

A filter mechanism avoids the definition of a parameter to balance reductions in the objective with reductions in the constraints. In the spirit of multiobjective optimization, a filter considers pairs of values $(\|c(x)\|, f(x))$ obtained by evaluating the functions $\|c\|$ and f at all iterates preceding the current one. A pair $(\|c(x_i)\|, f(x_i))$ is said to dominate another pair $(\|c(x_j)\|, f(x_j))$ if and only if both $\|c(x_i)\| \leq \|c(x_j)\|$ and $f(x_i) \leq f(x_j)$. The filter $\mathcal{F}$ is then defined to be an index set corresponding to a list of pairs such that no pair dominates any other. A step d_k from x_k is considered acceptable if the resulting point $\bar{x} = x_k + d_k$ corresponds to a pair $(\|c(\bar{x})\|, f(\bar{x}))$ such that either

$$\|c(\bar{x})\| < \|c(x_i)\| \quad \text{or} \quad f(\bar{x}) < f(x_i) \tag{2.8}$$

for all $i \in \mathcal{F}$. Upon the acceptance of such a step, the pair $(\|c(\bar{x})\|, f(\bar{x}))$ may be added to the filter, in which case all points dominated by this pair are removed from $\mathcal{F}$. Figure 2.2 illustrates the region of acceptable points as that lying sufficiently below and to the left of

the piecewise linear function. The global convergence guarantees of such an approach have been shown when paired with certain types of step computation methods [15, 16, 18, 36].

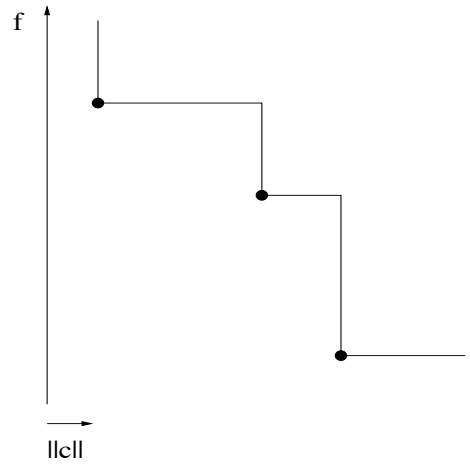

Figure 2.2: Region of acceptable points for a filter with three entries

Penalty functions and filters both have their own advantages and disadvantages. In Chapter 6 we consider globalization strategies in more detail, where we also propose a new technique designed to promote long steps and fast convergence to a solution point. In the majority of this dissertation, however, we exclusively consider penalty functions in the context of a line search SQP framework. We outline the main aspects of this framework now for use later in our discussion.

First, we note that ϕ_π is not differentiable. However, this penalty function is exact in the sense that if π is greater than a certain threshold, then a first order optimal point x^* of problem (2.1) is a stationary point of ϕ_π. That is, the directional derivative of ϕ_π in a direction d, denoted by $D\phi_\pi(d)$, is nonnegative at x^* for all $d \in \mathbb{R}^n$.

A critical component of an algorithm employing a penalty function to promote convergence is the technique implemented to set the penalty parameter π_k during each iteration k. For this purpose, we describe the general form of an effective approach used in some contemporary algorithms. Inspired by [11, 39], we begin by considering the following model

of ϕ_π around the current iterate x_k:

$$m_\pi(d) = f_k + g_k^T d + \tfrac{\omega(d)}{2} d^T W_k d + \pi \|c_k + A_k d\|, \tag{2.9}$$

where

$$\omega(d) \triangleq \begin{cases} 1 & \text{if } d^T W_k d \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

With this approximation, we can estimate the reduction in ϕ_π yielded by the computed primal step d_k by evaluating

$$\begin{aligned} mred_\pi(d_k) &\triangleq m_\pi(0) - m_\pi(d_k) \\ &= -g_k^T d_k - \tfrac{\omega_k}{2} d_k^T W_k d_k + \pi(\|c_k\| - \|c_k + A_k d_k\|) \end{aligned} \tag{2.11}$$

(where we note that $c_k + A_k d_k = 0$ for d_k in (2.6)). Following [6, 29, 39], we choose the penalty parameter so that the reduction in the model m_π obtained by d_k is sufficiently large with respect to the improvement in the infeasibility measure, as in

$$mred_\pi(d_k) \geq \sigma \pi \|c_k\| \tag{2.12}$$

for some constant $0 < \sigma < 1$. From (2.11) and (2.12), we have that if

$$\|c_k + A_k d_k\| \leq \epsilon \|c_k\| \tag{2.13}$$

for $0 < \epsilon < 1$, then

$$\pi_k \geq \frac{g_k^T d_k + \frac{\omega_k}{2} d_k^T W_k d_k}{(1-\tau)(\|c_k\| - \|c_k + A_k d_k\|)} \triangleq \chi_k \tag{2.14}$$

with $0 < \tau < 1$ satisfies this requirement for $\sigma = \tau(1-\epsilon)$. Further details related to

techniques for updating the penalty parameter will be considered extensively in Chapter 3.

Upon the calculation of a search direction d_k and penalty parameter π_k, we perform a backtracking line search to compute a steplength coefficient α_k satisfying the Armijo condition

$$\phi_{\pi_k}(x_k + \alpha_k d_k) \leq \phi_{\pi_k}(x_k) + \eta \alpha_k D\phi_{\pi_k}(d_k) \tag{2.15}$$

for some $0 < \eta < 1$. Accordingly, a primal-dual step will only be accepted if its primal component is a direction of nonincrease for the penalty function ϕ_{π_k}.

In summary, the algorithms proposed and analyzed in this dissertation are based on a standard line search SQP framework where during each iteration a search direction is computed as a (approximate) solution to the quadratic subproblem (2.4) and a line search is performed on the penalty function ϕ_π to ensure convergence to first order optimal points.

Chapter 3

An Inexact SQP Method for Equality Constrained Optimization

3.1 Background and Motivation

In this chapter we propose an algorithm for the equality constrained optimization problem (2.1). Our interest is in methods for very large problems for which the exact computation of steps in contemporary methods can be prohibitively expensive. One class of problems of this type that demands algorithmic improvements are those where the nonlinear constraint functions are defined by systems of partial differential equations (PDEs).

We consider a line search SQP framework as described in Chapter 2. Despite their strong theoretical properties and superior practical performance, a drawback of many contemporary SQP algorithms is that they require explicit representations of exact derivative information and the exact solution of one or more linear systems during every iteration. The acquisition of these quantities is particularly cumbersome in large-scale settings and the factorization of large iteration matrices is often impractical.

One way to overcome these difficulties is to solve the SQP subproblem (2.4) approximately using iterative linear algebra techniques. The main purpose of this chapter is to determine the accuracy with which the SQP subproblems must be solved in order to ensure global convergence in the context of a practical algorithm for problem (2.1). We both propose such a method and analyze its global behavior.

Our method resembles those in the class of inexact Newton methods for solving nonlinear systems of equations. There are, however, important differences between the two approaches. Inexact Newton methods for systems of equations are controlled by forcing parameters that ensure that the norm of the entire residual of the Newton equations decreases at every iteration [8]. Our approach, on the other hand, is based on requirements that the step decreases a model of a penalty function, while also satisfying bounds on the primal and dual components of the residual. We present sets of easily calculable conditions that handle these two residuals as separate quantities when determining if a given inexact solution is appropriate for the algorithm to follow. Such a solution may, for example, allow for an increase in the residual corresponding to primal feasibility provided it yields a substantial decrease in dual feasibility, or vice versa. The behavior of these components also helps determine when it is appropriate to increase the penalty parameter in the penalty function.

A variety of methods for constrained optimization with inexactness in step computations have been proposed recently. Jäger and Sachs [24] describe an inexact reduced SQP method in Hilbert space. Lalee, Nocedal, and Plantenga [26], Byrd, Hribar, and Nocedal [6] and Heinkenschloss and Vicente [23] propose composite-step approaches where the step is computed as an approximate solution to an SQP subproblem with a trust region constraint. Similarly, Walther [38] provides a composite-step method that allows incomplete constraint Jacobian information. Leibfritz and Sachs [27] analyze an interior point method that benefits from a reformulation of the quadratic programming subproblems as mixed linear complemen-

tarity problems. Our approach has some features in common with the algorithms of Biros and Ghattas [2, 3], Haber and Ascher [20], and Prudencio, Byrd and Cai [32] as we follow a full space SQP method and perform a line search to promote convergence. Unlike these papers, however, we present conditions that guarantee the global convergence of inexact SQP steps.

This chapter is organized as follows. In Section 3.2 we provide an overview of our approach and globalization strategy. Section 3.3 contains details about the most crucial aspect of our algorithm, namely, the sets of conditions used to determine if a given inexact SQP solution is considered an acceptable step. The well-posedness of our approach is also discussed, the accountability of which allows us to present global convergence guarantees under common conditions in Section 3.4. Closing remarks and issues related to extensions of this work are presented in Section 3.5, and numerical experience for a particular implementation of the method on a wide range of problems is presented in Chapter 4.

3.2 Outline of the Algorithm

Recall that the SQP search direction d_k from an iterate x_k can be obtained as part of the solution of the primal-dual system (2.6). An explicit representation of the primal-dual matrix

$$\begin{bmatrix} W(x_k, \lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} \tag{3.1}$$

and an exact solution of (2.6) can be expensive to obtain, particularly when the factors of (3.1) are not very sparse. We are interested, therefore, in identifying inexact solutions of (2.6) that can also be considered appropriate steps for the algorithm to accept during a given iteration. Such inexact solutions can be obtained in a variety of ways, such as by applying an

iterative linear system solver to the primal-dual system. Regardless of the method chosen, for an inexact solution (d_k, δ_k) we define the residual vectors (ρ_k, r_k) by the equation

$$\begin{bmatrix} W(x_k, \lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} g(x_k) + A(x_k)^T \lambda_k \\ c(x_k) \end{bmatrix} + \begin{bmatrix} \rho_k \\ r_k \end{bmatrix}. \tag{3.2}$$

The step can then be appraised based on properties of the residual vector and other quantities related to the SQP subproblem formulation (2.4). For convex problems, an inexact Newton method intended for nonlinear equations will suffice [8]. That is, the norm of the right-hand-side vector in (2.6) can serve as a merit function, and convergence can be guaranteed by systematically decreasing this value (provided that $W(x_k, \lambda_k)$ is the exact Hessian of the Lagrangian). For nonconvex problems, however, a step that decreases the first order optimality error may move away from a minimizer, or may be trapped near a stationary point of the Lagrangian. Thus, merit functions more appropriate to constrained optimization, such as the penalty function ϕ_π defined in Section 2.3, should be considered. The challenge is to compute inexact SQP steps and a value for π that ensure progress in the penalty function ϕ_π during every iteration.

In summary, our approach follows a standard line search SQP framework. During each iteration, a step is computed as an inexact solution to the primal-dual system (2.6) satisfying appropriate conditions that deem the step acceptable. The penalty parameter is then set based on properties of the computed step, after which a backtracking line search is performed to compute a steplength coefficient α_k satisfying the Armijo condition (2.15). Finally, the iterate is updated along with function and derivative information at the new point. The novelty of our approach, i.e., the precise definition of what constitutes an acceptable step, and the convergence properties of this algorithm are considered in the remainder of this chapter.

3.3 Step Computation and Selection

This section contains the details of our algorithm related to the computation of an acceptable inexact SQP step. We present conditions that the steps of the algorithm must satisfy so that the method is well-posed and will converge to a local solution of (2.1) from remote starting points under common conditions.

An intuitive condition that one may impose on an inexact SQP step is that the directional derivative of the penalty function $D\phi_{\pi_k}(d_k)$ along the primal component d_k must be sufficiently negative. Such a condition could be used in the development of a globally convergent SQP approach, but quantifying an appropriate steepness of the directional derivative is a difficult task in practice.

As an alternative, we borrow from the ideas outlined in Section 2.3 related to the computation of steps that sufficiently reduce the model m_π of the penalty function ϕ_π. At the heart of our approach is the claim that a given primal-dual step is often beneficial for the algorithm to follow provided the following condition is satisfied.

Model Reduction Condition. *A step (d_k, δ_k) computed in an inexact SQP algorithm must satisfy*

$$mred_{\pi_k}(d_k) \geq \sigma\pi_k \max\{\|c_k\|, \|r_k\| - \|c_k\|\} \tag{3.3}$$

for a given constant $0 < \sigma < 1$ and appropriate $\pi_k > 0$.

We will see the effects of this condition below and in Section 3.4. In particular, (3.3) will indeed ensure that the directional derivative of the penalty function is sufficiently negative along the primal step component d_k while also providing a mechanism for determining appropriate values of the penalty parameter. We note that conditions similar to the model reduction condition (3.3) are presented in the context of the inexact SQP algorithm proposed by Heinkenschloss and Vicente [23]. However, their conditions relate to a composite-step

strategy while (3.3) relates to ensuring sufficient decrease in a model of a penalty function for the full primal step d_k.

3.3.1 Step Acceptance Conditions

An acceptable step will be required to satisfy one of two sets of conditions. Although the step computations can be performed using almost any technique, we refer to the conditions as "termination tests" in reference to algorithms that apply an iterative solver to the primal-dual system (2.6), as in this framework the conditions are used to determine when to terminate the iteration. Each termination test ensures that the step satisfies (3.3) for a sufficiently large value of the penalty parameter and enforces requirements on the residuals (ρ_k, r_k) to ensure convergence to a local solution of (2.1). In particular, for both termination tests we require that the relative residual of the entire primal-dual system remains bounded; i.e., we require

$$\left\| \begin{bmatrix} \rho_k \\ r_k \end{bmatrix} \right\| \leq \kappa \left\| \begin{bmatrix} g_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \right\| \tag{3.4}$$

for some constant $\kappa > 0$ over all k. In addition, the tests impose restrictions on when it is allowed to increase the penalty parameter in order to satisfy the model reduction condition (3.3).

The first termination test addresses those steps providing a sufficiently large reduction in the model of the penalty function for the most recent value of the penalty parameter. We assume that an initial value $\pi_{-1} > 0$ is given.

Termination Test I. *Let $0 < \epsilon, \sigma < 1$ and $\beta, \kappa > 0$ be given constants. A step (d_k, δ_k) computed in an inexact SQP algorithm is acceptable if* (3.4) *holds, the model reduction*

condition (3.3) *holds for* $\pi_k = \pi_{k-1}$, *and*

$$\|\rho_k\| \leq \max\{\beta\|c_k\|, \epsilon\|g_k + A_k^T\lambda\|\}, \tag{3.5}$$

where the residuals (ρ_k, r_k) *are defined by* (3.2).

We claim that Termination Test I allows for productive steps to be taken that may have been computed in a relatively cheap manner, say after only a few iterations of an iterative solver applied to the primal-dual system (2.6). For steps satisfying this test, given that a sufficient reduction in the model of the penalty function has been obtained we need only enforce a generally loose bound on the dual infeasibility measure $\|\rho_k\|$.

The second termination test addresses those steps providing a sufficiently large reduction in the linear model of the constraints.

Termination Test II. *Let* ϵ, β, *and* κ *be given by Termination Test I. A step* (d_k, δ_k) *computed in an inexact SQP algorithm is acceptable if* (3.4) *holds,*

$$\|r_k\| \leq \epsilon\|c_k\|, \tag{3.6a}$$

$$\textit{and} \quad \|\rho_k\| \leq \beta\|c_k\|, \tag{3.6b}$$

where the residuals (ρ_k, r_k) *are defined by* (3.2).

A step satisfying Termination Test II may not satisfy the model reduction condition (3.3) for $\pi_k = \pi_{k-1}$. However, for such steps we require that the penalty parameter be increased to satisfy (2.14), which we rewrite for convenience in our updated notation as

$$\pi_k \geq \frac{g_k^T d_k + \frac{\omega_k}{2} d_k^T W_k d_k}{(1-\tau)(\|c_k\| - \|r_k\|)} \triangleq \chi_k \tag{3.7}$$

with $0 < \tau < 1$. Notice from (3.6a) that the denominator in the above expression is positive

and along with (2.11) the rule (3.7) implies

$$mred_{\pi_k}(d_k) \geq \tau\pi_k(\|c_k\| - \|r_k\|) \geq \tau(1-\epsilon)\pi_k\|c_k\|. \tag{3.8}$$

Thus, when (3.6a) is satisfied, the model reduction condition (3.3) holds with $\sigma = \tau(1-\epsilon)$.

In summary, a step (d_k, δ_k) will be required to satisfy Termination Test I or II. In each case, the model reduction condition (3.3) will hold; Termination Test I demands it explicitly and the rule (3.7) is used to enforce it when Termination Test II is satisfied. For consistency between Termination Test I and II and (3.7), we may set $\sigma = \tau(1-\epsilon)$ for Termination Test I. In this case, it can be seen that for any step satisfying Termination Test II that does not satisfy Termination Test I, the rule (3.7) will increase the penalty parameter beyond its most recent value. This follows from the fact that under (3.4) the bounds (3.6) are tighter than (3.5), and so any step satisfying Termination Test II and (3.7) for $\pi_k = \pi_{k-1}$ would satisfy (3.3) for this same value of the penalty parameter and thus would satisfy Termination Test I.

Algorithm 3.1. Inexact SQP Method

Given parameters $0 < \epsilon, \tau, \sigma, \eta < 1$ *and* $0 < \beta, \kappa, \varepsilon$

Initialize x_0, λ_0, *and* $\pi_{-1} > 0$

for $k = 0, 1, 2, \ldots,$ *until a convergence test for* (2.1) *is satisfied*

Compute $f_k, g_k, c_k, W_k,$ *and* A_k *and set* $\pi_k \leftarrow \pi_{k-1}$ *and* $\alpha_k \leftarrow 1$

Compute a step (d_k, δ_k) *satisfying Termination Test I or II*

if *Termination Test II is satisfied and* (3.7) *does not hold, set* $\pi_k \leftarrow \chi_k + \varepsilon$

Perform a backtracking line search to obtain α_k *satisfying* (2.15)

Set $(x_{k+1}, \lambda_{k+1}) \leftarrow (x_k, \lambda_k) + \alpha_k(d_k, \delta_k)$

endfor

In practice, the step can be computed by producing a sequence of candidate steps $\{(d, \delta)\}$

via the application of an iterative solver to (2.6). The corresponding residuals $\{(\rho, r)\}$ can then be computed and Termination Tests I and II can be evaluated during each iteration or after a few steps of the iterative solver. The constants (ϵ, β) should be tuned for a specific application and can significantly influence the practical performance of the algorithm. In particular, the value for β should be chosen to reflect the relationship between the scales of the primal and dual feasibility measures. The scale dependence of such a parameter is not ideal, but a bound similar to (3.6b) is used to ensure the boundedness of the penalty parameter π_k (as we show in Lemma 3.11) if the rule (3.7) is enforced. Since such a method for setting the penalty parameter has proved to work well in practice [39], we employ this update rule in Algorithm 3.1 and define β and (3.6b) as given. The remaining constants can generally be set to default values. Further discussion of appropriate values for the constants and an example implementation of Algorithm 3.1 are given in Chapter 4.

3.3.2 Well-posedness of the Algorithm

We conclude this section with a few observations to show that Algorithm 3.1 is well-defined under common conditions.

Suppose, during iteration k, that Algorithm 3.1 has not terminated. We may then assume that the optimality conditions (2.3) evaluated at the current iterate are nonzero and we have not reached a stationary point for the penalty function ϕ_{π_k} for a sufficiently large value of the penalty parameter π_k. Moreover, assume that for any $\kappa > 0$ the step computation procedure will eventually produce a step with corresponding residual satisfying (3.4) whenever (2.6) is consistent. Then, (3.5) or (3.6) will eventually be satisfied and an acceptable step satisfying Termination Test I or II will be computed.

Once an acceptable step is obtained, we must ensure that a positive steplength coefficient α_k can be calculated to satisfy the Armijo condition (2.15). We consider this issue by first

presenting the following result.

Lemma 3.2. *The directional derivative of the penalty function ϕ_π along a step d satisfies*

$$D\phi_\pi(d) \leq g^T d - \pi(\|c\| - \|r\|).$$

Proof. Applying Taylor's theorem, we find for some constant $\gamma_1 > 0$

$$\begin{aligned}
\phi(x+\alpha d) - \phi(x) = &\ f(x+\alpha d) - f(x) + \pi(\|c(x+\alpha d)\| - \|c(x)\|) \\
\leq &\ \alpha g^T d + \gamma_1 \alpha^2 \|d\|^2 + \pi(\|c(x) + \alpha A d\| - \|c(x)\|) \\
= &\ \alpha g^T d + \gamma_1 \alpha^2 \|d\|^2 + \pi(\|(1-\alpha)c(x) + \alpha r\| - \|c(x)\|) \\
\leq &\ \alpha(g^T d - \pi(\|c(x)\| - \|r\|)) + \gamma_1 \alpha^2 \|d\|^2,
\end{aligned}$$

where $r = c(x) + Ad$ as in (3.2). Dividing both sides by α and taking the limit as $\alpha \to 0$ yields the result. □

Given this result, we present the following consequence of our model reduction condition. (A stronger result will be given as Lemma 3.9.)

Lemma 3.3. *If the model reduction condition* (3.3) *holds for a step (d_k, δ_k) and penalty parameter π_k, then the directional derivative of the penalty function satisfies $D\phi_{\pi_k}(d_k) \leq 0$.*

Proof. Observe from (2.11) that the inequality (3.3) can be rewritten as

$$g_k^T d_k - \pi_k(\|c_k\| - \|r_k\|) \leq -\tfrac{\omega_k}{2} d_k^T W_k d_k - \sigma \pi_k \max\{\|c_k\|, \|r_k\| - \|c_k\|\},$$

so, by Lemma 3.2, a step (d_k, δ_k) satisfying (3.3) yields

$$\begin{aligned} D\phi_{\pi_k}(d_k) &\leq g_k^T d_k - \pi_k(\|c_k\| - \|r_k\|) \\ &\leq -\tfrac{\omega_k}{2} d_k^T W_k d_k - \sigma\pi_k \max\{\|c_k\|, \|r_k\| - \|c_k\|\}. \end{aligned} \tag{3.9}$$

The result follows from the above inequality and (2.10). □

We have shown under common conditions that an acceptable inexact SQP step (d_k, δ_k) can always be computed by Algorithm 3.1 and that steps satisfying the model reduction condition (3.3) correspond to directions of nonincrease for the penalty function ϕ_{π_k}. These results allow us to show that the Armijo condition (2.15) is satisfied by some positive α_k (see Lemma 3.10), and so Algorithm 3.1 is well-posed.

We mention in passing that, as a corollary to Lemma 3.2, we may avoid the exact computation of the directional derivative of the penalty function along a step d by defining the estimate

$$\widetilde{D}\phi_\pi(d) \triangleq g^T d - \pi(\|c\| - \|r\|). \tag{3.10}$$

As such, the Armijo condition (2.15) can be substituted by

$$\phi_{\pi_k}(x_k + \alpha_k d_k) \leq \phi_{\pi_k}(x_k) + \eta\alpha_k \widetilde{D}\phi_{\pi_k}(d_k). \tag{3.11}$$

All of the analysis in this chapter holds when either (2.15) or (3.11) is observed in the line search procedure of Algorithm 3.1.

3.4 Global Analysis

Let us begin our investigation of the global behavior of the algorithm by making the following assumptions about the problem and the set of computed iterates.

Assumptions 3.4. *The sequence* $\{(x_k, \lambda_k)\}$ *generated by Algorithm 3.1 is contained in a convex set* Ω *and the following properties hold:*

(a) The functions f *and* c *and their first and second derivatives are bounded on* Ω.

(b) The sequence $\{\lambda_k\}$ *is bounded.*

(c) The constraint Jacobians A_k *have full row rank and their smallest singular values are bounded below by a positive constant.*

(d) The sequence $\{W_k\}$ *is bounded.*

(e) There exists a constant $\mu > 0$ *such that for any* $u \in \mathbb{R}^n$ *with* $u \neq 0$ *and* $A_k u = 0$ *we have* $u^T W_k u \geq \mu \|u\|^2$.

These assumptions are fairly standard [22, 31]. Assumption 3.4 is a little weaker than the common assumption that the iterates are contained in a compact set. Assumptions 3.4(b) and (c) are strong; we use them to simplify the analysis in order to focus on the issues related to inexactness. It would be of interest in future studies of inexact SQP methods to relax these assumptions. Assuming that W_k is positive definite on the null space of the constraints is natural for line search algorithms, for otherwise there would be no guarantee of descent unless certain enhancements to the algorithm are implemented (see Chapter 5). We comment on the validity of Assumption 3.4(b) in Section 3.5.

We now assume that during iteration k we have obtained an acceptable step (d_k, δ_k) with residuals (ρ_k, r_k) defined by (3.2). We consider the decomposition

$$d_k = u_k + v_k, \tag{3.12}$$

where u_k lies in the null space of the constraint Jacobian A_k and v_k lies in the range space of A_k^T. We do not intend to compute the components explicitly; the decomposition is only

for analytical purposes [5, 7]. We refer to u_k, which by definition satisfies $A_k u_k = 0$, as the tangential component and v_k as the normal component of the step. In this manner we may rewrite

$$\begin{aligned} mred_{\pi_k}(d_k) = & -g_k^T(u_k + v_k) - \tfrac{\omega_k}{2}(u_k + v_k)^T W_k(u_k + v_k) + \pi_k(\|c_k\| - \|r_k\|) \\ = & -g_k^T u_k - \tfrac{\omega_k}{2} u_k^T W_k u_k \\ & - \omega_k u_k^T W_k v_k - g_k^T v_k - \tfrac{\omega_k}{2} v_k^T W_k v_k + \pi_k(\|c_k\| - \|r_k\|). \end{aligned} \tag{3.13}$$

Our analysis hinges on our ability to classify the effects of two types of steps: those lying sufficiently in the null space of the constraints and those sufficiently orthogonal to the linearized feasible region. We show that such a distinction can be made by observing the relative magnitudes of the normal and tangential components of the primal component d_k.

We first present a result related to the magnitude of the normal step.

Lemma 3.5. *For all k, the normal component v_k is bounded in norm and for some $\gamma_2 > 0$ satisfies*

$$\|v_k\|^2 \leq \gamma_2 \max\{\|c_k\|, \|r_k\|\}. \tag{3.14}$$

Furthermore, for all k such that Termination Test II is satisfied, there exists $\gamma_3 > 0$ such that

$$\|v_k\| \leq \gamma_3(\|c_k\| - \|r_k\|). \tag{3.15}$$

Proof. From $A_k v_k = -c_k + r_k$ and the fact that v_k lies in the range space of A_k^T, it follows that

$$v_k = A_k^T(A_k A_k^T)^{-1}(-c_k + r_k),$$

and so

$$\|v_k\| \leq \|A_k^T(A_k A_k^T)^{-1}\|(\|c_k\| + \|r_k\|). \tag{3.16}$$

This, along with (3.4), the fact that Assumptions 3.4(a) and (b) imply that $\|c_k\|$ and $\|g_k + A_k^T\lambda_k\|$ are bounded, and the fact that Assumptions 3.4(a) and (c) imply that $\|A_k^T(A_kA_k^T)^{-1}\|$ is bounded, implies v_k is bounded in norm for all k. The inequality (3.16) also yields

$$\begin{aligned}\|v_k\|^2 \leq & \left(\|A_k^T(A_kA_k^T)^{-1}\|(\|c_k\| + \|r_k\|)\right)^2 \\ \leq & \left(2\|A_k^T(A_kA_k^T)^{-1}\| \max\{\|c_k\|, \|r_k\|\}\right)^2 \\ = & \left[4\|A_k^T(A_kA_k^T)^{-1}\|^2 \max\{\|c_k\|, \|r_k\|\}\right] \max\{\|c_k\|, \|r_k\|\}, \end{aligned} \tag{3.17}$$

where (3.4) and Assumptions 3.4(a), (b), and (c) also imply that the bracketed expression in (3.17) is bounded. Thus, (3.14) holds. Finally, if Termination Test II is satisfied, then from (3.6a) and (3.16) we have

$$\begin{aligned}\|v_k\| \leq & \|A_k^T(A_kA_k^T)^{-1}\|(1+\epsilon)\|c_k\| \\ \leq & \|A_k^T(A_kA_k^T)^{-1}\| \left(\tfrac{1+\epsilon}{1-\epsilon}\right) (\|c_k\| - \|r_k\|),\end{aligned}$$

and so (3.15) holds. □

A similar result can be proved for the tangential component.

Lemma 3.6. *The tangential components $\{u_k\}$ are bounded in norm.*

Proof. Assumption 3.4(e), the fact that u_k lies in the null space of the constraint Jacobian A_k, and the first block equation of (3.2) yield

$$\begin{aligned}\mu\|u_k\|^2 \leq & \; u_k^T W_k u_k \\ = & -g_k^T u_k + \rho_k^T u_k - u_k^T W_k v_k \\ \leq & \; (\|g_k\| + \|\rho_k\| + \|W_k v_k\|)\|u_k\|,\end{aligned}$$

and so

$$\|u_k\| \leq (\|g_k\| + \|\rho_k\| + \|W_k v_k\|)/\mu.$$

The result follows from the facts that Assumptions 3.4, Lemma 3.5, and the bounds (3.4), (3.5), and (3.6) imply that all terms in the right-hand-side of this inequality are bounded. □

We now turn to the following result addressing the relative magnitudes of the normal and tangential components of a given step.

Lemma 3.7. *There exists $\gamma_4 > 0$ such that $\|u_k\|^2 \geq \gamma_4 \|v_k\|^2$ implies $\frac{1}{2} d_k^T W_k d_k \geq \frac{\mu}{4} \|u_k\|^2$.*

Proof. Assumption 3.4(e) implies that for any γ_4 such that $\|u_k\|^2 \geq \gamma_4 \|v_k\|^2$ we have

$$\begin{aligned} \tfrac{1}{2} d_k^T W_k d_k = & \ \tfrac{1}{2} u_k^T W_k u_k + u_k^T W_k v_k + \tfrac{1}{2} v_k^T W_k v_k \\ \geq & \ \tfrac{\mu}{2} \|u_k\|^2 - \|u_k\| \|W_k\| \|v_k\| - \tfrac{1}{2} \|W_k\| \|v_k\|^2 \\ \geq & \ \left(\frac{\mu}{2} - \frac{\|W_k\|}{\sqrt{\gamma_4}} - \frac{\|W_k\|}{2\gamma_4} \right) \|u_k\|^2. \end{aligned}$$

Thus, Assumption 3.4(d) implies that the result holds for some sufficiently large $\gamma_4 > 0$. □

With the above results, we can now formalize a distinction between two types of steps. Let $\gamma_4 > 0$ be chosen large enough as described in Lemma 3.7 and consider the sets of indices

$$\begin{aligned} K_1 \triangleq & \ \{k : \|u_k\|^2 \geq \gamma_4 \|v_k\|^2\} \\ \text{and} \quad K_2 \triangleq & \ \{k : \|u_k\|^2 < \gamma_4 \|v_k\|^2\}. \end{aligned}$$

Most of the remainder of our analysis will be dependent on these sets and the corresponding quantity

$$\Theta_k \triangleq \begin{cases} \|u_k\|^2 + \|c_k\|, & k \in K_1, \\ \max\{\|c_k\|, \|r_k\|\}, & k \in K_2. \end{cases} \tag{3.18}$$

The relevance of Θ_k will be seen in the following three lemmas as a quantity that can be used for bounding the length of the primal step and the directional derivative of the penalty function, which will then provide a lower bound for the sequence of steplength coefficients $\{\alpha_k\}$.

Lemma 3.8. *There exists $\gamma_5 > 1$ such that, for all k,*

$$\|d_k\|^2 \leq \gamma_5 \Theta_k,$$

and hence

$$\|d_k\|^2 + \|c_k\| \leq 2\gamma_5 \Theta_k. \tag{3.19}$$

Proof. For $k \in K_1$, we find

$$\begin{aligned} \|d_k\|^2 = & \ \|u_k\|^2 + \|v_k\|^2 \\ \leq & \ \left(1 + \tfrac{1}{\gamma_4}\right) \|u_k\|^2 \\ \leq & \ \left(1 + \tfrac{1}{\gamma_4}\right) (\|u_k\|^2 + \|c_k\|). \end{aligned}$$

Similarly, Lemma 3.5 implies that for $k \in K_2$

$$\begin{aligned} \|d_k\|^2 = & \ \|u_k\|^2 + \|v_k\|^2 \\ < & \ (\gamma_4 + 1)\|v_k\|^2 \\ \leq & \ (\gamma_4 + 1)\gamma_2 \max\{\|c_k\|, \|r_k\|\}. \end{aligned}$$

To establish (3.19) we note that $\Theta_k + \|c_k\| \leq 2\Theta_k$ for all k. □

The directional derivative of the penalty function ϕ_{π_k} can be bounded in a similar manner.

Lemma 3.9. *There exists $\gamma_6 > 0$ such that, for all k,*

$$\widetilde{D}\phi_{\pi_k}(d_k) \leq -\gamma_6 \Theta_k.$$

Proof. Recalling (3.9) and (3.10) we have

$$\widetilde{D}\phi_{\pi_k}(d_k) \leq -\tfrac{\omega_k}{2} d_k^T W_k d_k - \sigma\pi_k \max\{\|c_k\|, \|r_k\| - \|c_k\|\}.$$

By Lemma 3.7 and (2.10), we have that $\omega_k = 1$ for $k \in K_1$ and thus

$$\widetilde{D}\phi_{\pi_k}(d_k) \leq -\tfrac{\mu}{4}\|u_k\|^2 - \sigma\pi_k\|c_k\|.$$

Similarly, for $k \in K_2$ we have from (2.10)

$$\begin{aligned} \widetilde{D}\phi_{\pi_k}(d_k) \leq & \; -\sigma\pi_k \max\{\|c_k\|, \|r_k\| - \|c_k\|\} \\ \leq & \; -\tfrac{1}{2}\sigma\pi_k \max\{\|c_k\|, \|r_k\|\}. \end{aligned}$$

The result holds for $\gamma_6 = \min\{\frac{\mu}{4}, \frac{1}{2}\sigma\pi_k\}$, which is bounded away from zero as $\{\pi_k\}$ is nondecreasing. □

Together, the previous two lemmas can be used to bound the sequence of steplength parameters.

Lemma 3.10. *The sequence $\{\alpha_k\}$ is bounded below and away from zero.*

Proof. Recall that the line search condition requires (3.11), which we rewrite for convenience as

$$\phi_{\pi_k}(x_k + \alpha_k d_k) - \phi_{\pi_k}(x_k) \leq \eta\alpha_k \widetilde{D}\phi_{\pi_k}(d_k).$$

Suppose that the line search condition fails for some $\bar{\alpha} > 0$, so

$$\phi_{\pi_k}(x_k + \bar{\alpha} d_k) - \phi_{\pi_k}(x_k) > \eta \bar{\alpha} \widetilde{D} \phi_{\pi_k}(d_k).$$

From the proof of Lemma 3.2 and (3.10) we have

$$\phi_{\pi_k}(x_k + \bar{\alpha} d_k) - \phi_{\pi_k}(x_k) \leq \bar{\alpha} \widetilde{D} \phi_{\pi_k}(d_k) + \bar{\alpha}^2 \gamma_1 \|d_k\|^2,$$

so

$$(\eta - 1) \widetilde{D} \phi_{\pi_k}(d_k) \leq \bar{\alpha} \gamma_1 \|d_k\|^2.$$

Lemmas 3.8 and 3.9 then yield

$$(1 - \eta) \gamma_6 \Theta_k < \bar{\alpha} \gamma_1 \gamma_5 \Theta_k,$$

so

$$\bar{\alpha} > (1 - \eta) \gamma_6 / (\gamma_1 \gamma_5).$$

Thus, α_k need never be set below $(1 - \eta)\gamma_6/(\gamma_1\gamma_5)$ for the line search condition (3.11) to be satisfied. □

Another important property of Algorithm 3.1 is that under Assumptions 3.4 the penalty parameter remains bounded. We prove this result in the following lemma, illustrating the importance of the bound (3.6b).

Lemma 3.11. *The sequence of penalty parameters $\{\pi_k\}$ is bounded above and $\pi_k = \pi_{\bar{k}}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq 0$.*

Proof. Recall that the penalty parameter is increased during iteration k of Algorithm 3.1 only if Termination Test II is satisfied. Therefore, for the remainder of this proof let us

assume that Termination Test II is satisfied and so the inequalities in (3.6) hold. By (3.7) the parameter π_k is chosen to satisfy the first inequality in (3.8), namely

$$mred_{\pi_k}(d_k) \geq \tau\pi_k(\|c_k\| - \|r_k\|). \tag{3.20}$$

Let us also recall equation (3.13):

$$\begin{aligned} mred_{\pi_k}(d_k) = \ & \left[-g_k^T v_k - \tfrac{\omega_k}{2} v_k^T W_k v_k\right] \\ & + \left[-g_k^T u_k - \tfrac{\omega_k}{2} u_k^T W_k u_k - \omega_k u_k^T W_k v_k\right] + \pi_k(\|c_k\| - \|r_k\|). \end{aligned}$$

The result follows from our ability to bound the terms inside the brackets with respect to the constraint reduction. First, by Lemma 3.5 and Assumptions 3.4 there exists γ_3' such that

$$-g_k^T v_k - \tfrac{\omega_k}{2} v_k^T W_k v_k \geq -\gamma_3'(\|c_k\| - \|r_k\|).$$

Second, the first block equation of (3.2) yields

$$\begin{aligned} & - g_k^T u_k - \tfrac{\omega_k}{2} u_k^T W_k u_k - \omega_k u_k^T W_k v_k \\ & \qquad = \begin{cases} -\rho_k^T u_k + \frac{1}{2} u_k^T W_k u_k & \text{if } d_k^T W_k d_k \geq 0 \\ -(\rho_k - W_k v_k)^T u_k + u_k^T W_k u_k & \text{otherwise.} \end{cases} \end{aligned}$$

If $d_k^T W_k d_k \geq 0$, then Lemma 3.6, Assumption 3.4(e), and the bounds (3.6) on the residuals

(ρ_k, r_k) imply that there exists $\gamma_7, \gamma_7' > 0$ such that

$$\begin{aligned} -g_k^T u_k - \tfrac{\omega_k}{2} u_k^T W_k u_k - \omega_k u_k^T W_k v_k &\geq -\|\rho_k\|\|u_k\| + \tfrac{1}{2} u_k^T W_k u_k \\ &\geq -\gamma_7 \|\rho_k\| \\ &\geq -\gamma_7 \beta \|c_k\| \\ &\geq -\gamma_7 \left(\tfrac{\beta}{1-\epsilon}\right) (\|c_k\| - \|r_k\|) \\ &= -\gamma_7' (\|c_k\| - \|r_k\|). \end{aligned}$$

If $d_k^T W_k d_k < 0$, Lemma 3.6, Assumptions 3.4(d) and (e), the bounds (3.6) on the residuals (ρ_k, r_k), and Lemma 3.5 imply that there exists $\gamma_8, \gamma_8' > 0$ such that

$$\begin{aligned} -g_k^T u_k - \tfrac{\omega_k}{2} u_k^T W_k u_k - \omega_k u_k^T W_k v_k &\geq -\|\rho_k\|\|u_k\| - \|W_k u_k\|\|v_k\| + u_k^T W_k u_k \\ &\geq -\gamma_8 (\|\rho_k\| + \|v_k\|) \\ &\geq -\gamma_8 \left(\tfrac{\beta}{1-\epsilon} + \gamma_3\right) (\|c_k\| - \|r_k\|) \\ &= -\gamma_8' (\|c_k\| - \|r_k\|). \end{aligned}$$

These results together imply

$$mred_{\pi_k}(d_k) \geq (-\max\{\gamma_7', \gamma_8'\} - \gamma_3' + \pi_k)(\|c_k\| - \|r_k\|),$$

and so (3.20) is always satisfied for

$$\pi_k \geq (\max\{\gamma_7', \gamma_8'\} + \gamma_3') / (1 - \tau).$$

Therefore, if $\pi_{\bar{k}} \geq (\max\{\gamma_7', \gamma_8'\} + \gamma_3')/(1-\tau)$ for some iteration number $\bar{k} \geq 0$, then $\pi_k = \pi_{\bar{k}}$ for $k \geq \bar{k}$. This, together with the fact that whenever Algorithm 3.1 increases the penalty

parameter it does so by at least a positive finite amount, proves the result. □

We can now present the following result related to the lengths of the primal components of the steps computed in Algorithm 3.1 and the convergence of the iterates toward the feasible region of problem (2.1).

Lemma 3.12. *Algorithm 3.1 yields*

$$\lim_{k\to\infty} \|c_k\| = 0 \quad \textit{and} \quad \lim_{k\to\infty} \|d_k\| = 0.$$

Proof. For all k, it can easily be seen that

$$\phi_{\pi_k}(x_k) - \phi_{\pi_k}(x_k + \alpha_k d_k) \geq \gamma_9 \Theta_k$$

for some $\gamma_9 > 0$ follows from (3.11) and Lemmas 3.9 and 3.10. By Lemma 3.11 the algorithm eventually computes, during iteration $\bar{k}$, a finite value $\pi_{\bar{k}}$ beyond which the penalty parameter will never be increased. Therefore, the penalty parameter π_k remains fixed for $k \geq \bar{k}$, i.e., $\phi_{\pi_k} = \phi_{\pi_{\bar{k}}}$ for $k \geq \bar{k}$, and (3.19) implies

$$\begin{aligned}
\phi_{\pi_{\bar{k}}}(x_{\bar{k}}) - \phi_{\pi_{\bar{k}}}(x_k) &= \sum_{j=\bar{k}}^{k-1} (\phi_{\pi_{\bar{k}}}(x_j) - \phi_{\pi_{\bar{k}}}(x_{j+1})) \\
&\geq \gamma_9 \sum_{j=\bar{k}}^{k-1} \Theta_j \\
&\geq \frac{\gamma_9}{2\gamma_5} \sum_{j=\bar{k}}^{k-1} (\|d_j\|^2 + \|c_j\|).
\end{aligned}$$

The result follows from the above and the fact that Assumption 3.4(a) implies $\phi_{\pi_{\bar{k}}}$ is bounded below. □

We are now ready to present the main result of this section.

Theorem 3.13. *Algorithm 3.1 yields*

$$\lim_{k\to\infty} \left\| \begin{bmatrix} g_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \right\| = 0.$$

Proof. Recall that $\alpha_k \leq 1$ for all k and from Lemma 3.10 we have that $\{\alpha_k\}$ is bounded below and away from zero. An expansion of the first block of the optimality conditions (2.3) yields

$$\|g_{k+1} + A_{k+1}^T \lambda_{k+1}\| \leq \|g_k + A_k^T \lambda_k + \alpha_k(\nabla_{xx}^2 L_k d_k + A_k^T \delta_k)\| + \alpha_k^2 E(d_k, \delta_k),$$

where

$$E(d_k, \delta_k) = O(\|d_k\|^2 + \|d_k \cdot \delta_k\|).$$

This, along with the first block equation in (3.2) and Assumptions 3.4, implies

$$\begin{aligned} &\|g_{k+1} + A_{k+1}^T \lambda_{k+1}\| \\ &\quad \leq \|g_k + A_k^T \lambda_k + \alpha_k(W_k d_k + A_k^T \delta_k) + \alpha_k(\nabla_{xx}^2 L_k - W_k) d_k\| + \alpha_k^2 E(d_k, \delta_k) \\ &\quad \leq \|g_k + A_k^T \lambda_k + \alpha_k(\rho_k - g_k - A_k^T \lambda_k)\| + \alpha_k \|(\nabla_{xx}^2 L_k - W_k) d_k\| + \alpha_k^2 E(d_k, \delta_k) \\ &\quad \leq (1 - \alpha_k)\|g_k + A_k^T \lambda_k\| + \alpha_k \|\rho_k\| + \alpha_k E'(d_k, \delta_k) \end{aligned} \tag{3.21}$$

where

$$E'(d_k, \delta_k) = O(\|d_k\| + \|d_k\|^2 + \|d_k \cdot \delta_k\|). \tag{3.22}$$

The bounds (3.5) and (3.6b) imply

$$(1-\alpha_k)\|g_k + A_k^T\lambda_k\| + \alpha_k\|\rho_k\| \\ \leq \max\{(1-(1-\epsilon)\alpha_k)\|g_k + A_k^T\lambda_k\|, (1-\alpha_k)\|g_k + A_k^T\lambda_k\| + \alpha_k\beta\|c_k\|\}$$

which, along with (3.21) and the boundedness of $\{\alpha_k\}$, implies that for some $0 < \gamma_{10} < 1$ and $\gamma_{11} > 0$ we have

$$\|g_{k+1} + A_{k+1}^T\lambda_{k+1}\| \leq \max\{(1-\gamma_{10})\|g_k + A_k^T\lambda_k\|, \gamma_{11}\|c_k\|\} + \alpha_k E'(d_k, \delta_k). \tag{3.23}$$

The boundedness of $\{\alpha_k\}$, Lemma 3.12, and the fact that Assumption 3.4(b) implies δ_k is bounded in norm imply, together with (3.22), that

$$\lim_{k\to\infty} \alpha_k E'(d_k, \delta_k) = 0. \tag{3.24}$$

Consider an arbitrary $\hat{\gamma} > 0$. Lemma 3.12 and the limit (3.24) imply that there exists $k' \geq 0$ such that for all $k \geq k'$ we have

$$\gamma_{11}\|c_k\| < (1-\gamma_{10})\hat{\gamma} \quad \text{and} \quad \alpha_k E'(d_k, \delta_k) < \tfrac{1}{2}\gamma_{10}\hat{\gamma}. \tag{3.25}$$

Suppose $k \geq k'$ and $\|g_k + A_k^T\lambda_k\| > \hat{\gamma}$. We find from (3.23) that

$$\begin{aligned} \|g_{k+1} + A_{k+1}^T\lambda_{k+1}\| &\leq (1-\gamma_{10})\|g_k + A_k^T\lambda_k\| + \tfrac{1}{2}\gamma_{10}\hat{\gamma} \\ &\leq \|g_k + A_k^T\lambda_k\| - \tfrac{1}{2}\gamma_{10}\hat{\gamma}. \end{aligned}$$

Therefore, $\{\|g_k + A_k^T\lambda_k\|\}$ decreases monotonically by at least a constant amount for $k \geq k'$ while $\{\|g_k + A_k^T\lambda_k\|\} > \hat{\gamma}$, so we eventually find $\|g_k + A_k^T\lambda_k\| \leq \hat{\gamma}$ for some $k = k'' \geq k'$.

Then, for $k \geq k''$ we find from (3.23) and (3.25) that

$$\begin{aligned} \|g_{k+1} + A_{k+1}^T \lambda_{k+1}\| &\leq (1 - \gamma_{10})\hat{\gamma} + \tfrac{1}{2}\gamma_{10}\hat{\gamma} \\ &\leq (1 - \tfrac{1}{2}\gamma_{10})\hat{\gamma}, \end{aligned}$$

so $\|g_k + A_k^T \lambda_k\| \leq \hat{\gamma}$ for all $k \geq k''$. Since the above holds for any $\hat{\gamma} > 0$, we have

$$\lim_{k \to \infty} \|g_k + A_k^T \lambda_k\| = 0,$$

and so the result follows with the above and the result of Lemma 3.12. □

3.5 Final Remarks

In this chapter we have developed an inexact SQP algorithm for equality constrained optimization that is globally convergent under common conditions. We close with some remarks about the assumptions used in our analysis, the rate of convergence of our approach, and possible extensions of this work.

First, let us recall Assumption 3.4(b), related to the boundedness of the multipliers. Our analysis does not guarantee that the multipliers remain bounded in general; in fact, Algorithm 3.1 does not exert direct control over them. We can ensure that $\{\lambda_k\}$ remains bounded, however, by adding to Termination Test I a requirement of the form

$$\|\rho_k\| \leq \kappa' \max\{\|g_k\|, \|A_k\|\}$$

for a constant $\kappa' > 0$. Such a condition ensures that ρ_k is bounded independently of the multipliers λ_k, so then (3.2) and Assumptions 3.4 will imply that λ_{k+1} is bounded. An

alternative would be to include a safeguard in the algorithm by which the multiplier estimate λ_k is set to a nominal value, say $\lambda_k = 0$, if $\|g_k + A_k^T \lambda_k\|$ is larger than a given constant.

Second, the rate of convergence of Algorithm 3.1 may be slow for a given problem. One can ensure a fast convergence rate, however, by imposing at each step a requirement of the form

$$\left\| \begin{bmatrix} \rho_k \\ r_k \end{bmatrix} \right\| \leq \kappa_k \left\| \begin{bmatrix} g_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \right\| \tag{3.26}$$

where $0 < \{\kappa_k\} < 1$ [8]. Then, tightening the values of κ_k during any point of a run of Algorithm 3.1 will influence the convergence rate if unit steplengths are taken. For example, if $0 \leq \kappa_k \leq \hat{\kappa} < 1$ for all large k, then the rate of convergence is linear with rate $\hat{\kappa}$. If, in addition, $\kappa_k \to 0$, then the rate of convergence is superlinear [8]. In practice, the penalty function ϕ_π can reject unit steps even close to the solution, but this difficulty can be overcome by the use of a second order correction or non-monotone techniques [28]. In this manner, we can be sure that the rate of convergence of Algorithm 3.1 will be fast once the penalty parameter is stabilized.

In addition it is worth noting that imposing condition (3.26) with sufficiently small κ_k implies that the bound (3.5) in Termination Test I is automatically satisfied, and the bounds (3.6) of Termination Test II are satisfied in the case where $\|c_k\|$ is greater than some constant times $\|g_k + A_k^T \lambda_k\|$.

Finally, it would be of interest to analyze the behavior of inexact SQP methods in the presence of Jacobian singularities. However, such an analysis can be complex and would have brought the focus away from the intended scope of this chapter. Therefore, we chose to discuss the design of inexact SQP methods in the benign context of Assumptions 3.4. However, we do consider algorithmic enhancements for cases where W_k is not positive definite on the null space of A_k in Chapter 5.

Chapter 4

Numerical Experience with an Inexact SQP Method for Equality Constrained Optimization

This chapter contains a description of a particular implementation of Algorithm 3.1 and corresponding numerical results to illustrate the robustness of our approach and its effectiveness on two instances of a realistic PDE-constrained model problem. We begin by providing some general comments that may be useful for any implementation before presenting our chosen approach.

In terms of the input parameters required for Algorithm 3.1, we make the following general comments on their practical effects. First, the values (ϵ, β) should receive special attention as they may greatly affect the ease with which Termination Tests I and II, and therefore the model reduction condition (3.3), are satisfied; larger values for (ϵ, β) allow more steps to satisfy at least one of the tests at a given point. In general, looser bounds in Termination Tests I and II will result in cheaper step computations, but these savings

must be balanced against possible increases in the number of outer iterations required to find a solution. The parameters may also influence the magnitude of the penalty parameter computed in the algorithm; larger values may yield larger values for π_k. A similar effect on the penalty parameter will be seen for smaller values of the input τ in (3.7) and larger values of σ in (3.3), and so together all of the parameters may affect the number of iterations required until the penalty parameter stabilizes, an important phenomenon in the analysis of Section 3.4. In general, however, we claim that the parameters (σ, τ), as well as the remaining constants, can be set to default values or to promote consistency between the two termination tests, as we do in (4.3) below.

An appropriate stopping condition for the overall nonlinear program is given by

$$\|g_k + A_k^T \lambda_k\|_\infty \leq \max\{\|g_k\|_\infty, 1\}\epsilon_{opt}, \tag{4.1}$$

$$\|c_k\|_\infty \leq \max\{\|c(x_0)\|_\infty, 1\}\epsilon_{feas}, \tag{4.2}$$

where $0 < \epsilon_{opt}, \epsilon_{feas} < 1$ and x_0 is the starting point (e.g., see [39]).

The following algorithm was implemented in a stand-alone Matlab code and incorporated into the Matlab environment for benchmarking PDE-constrained optimization solvers provided by Haber and Hanson [21]. We use the generalized minimum residual (GMRES) method [33] in the stand-alone implementation for the step computations, for which we adapted the implementation by Kelley [25]. The GMRES method does not exploit the symmetry of the matrix (3.1) in the primal-dual system (2.6), but the stability of the approach is ideal for illustrating the robustness of Algorithm 3.1. In the benchmarking environment, we use an implementation of the symmetric quasi-minimum residual (SymQMR) method [17] to exploit the symmetry of the primal-dual matrix and perform fast step computations. The `termination` variable is used to indicate the successful or unsuccessful termination of the solver near a local solution of problem (2.1).

Algorithm 4.1. Inexact SQP Method (`isqp`)

Given parameters $0 < \epsilon_{feas}, \epsilon_{opt}, \epsilon, \tau, \sigma, \eta, \alpha_{\min} < 1$ *and* $0 < k_{\max}, \beta, \kappa, \varepsilon$

Initialize x_0, λ_0, *and* $\pi_{-1} > 0$

Set `termination` $\leftarrow$ `success`

for $k = 0, 1, 2, \ldots, k_{max}$, *or until* (4.1) *and* (4.2) *are satisfied*

Compute f_k, g_k, c_k, W_k, *and* A_k *and set* $\pi_k \leftarrow \pi_{k-1}$ *and* $\alpha_k \leftarrow 1$

for $j = 0, 1, 2, \ldots, n + t$, *or until Termination Test I or II is satisfied*

Set (d_k, δ_k) *as the* j*th iterative solver solution*

endfor

if $\widetilde{D}\phi_\pi(d_k) > 0$ *for all* $\pi \geq \pi_k$, *set* `termination` $\leftarrow$ `failure` *and* **break**

if *Termination Test II is satisfied and* (3.7) *does not hold, set* $\pi_k \leftarrow \chi_k + \varepsilon$

while (3.11) *is not satisfied and* $\alpha_k \geq \alpha_{\min}$, *set* $\alpha_k \leftarrow \alpha_k/2$

if $\alpha_k < \alpha_{\min}$, *set* `termination` $\leftarrow$ `failure` *and* **break**

Set $(x_{k+1}, \lambda_{k+1}) \leftarrow (x_k, \lambda_k) + \alpha_k(d_k, \delta_k)$

endfor

if (4.1) *or* (4.2) *is not satisfied, set* `termination` $\leftarrow$ `failure`

return `termination`

We recognize three types of failures in the above approach. First, due to the iteration limit imposed on the inner **for** loop, or if the positive definiteness of Assumption 4.1(e) is violated, the iterative linear system solver may not provide a solution satisfying Termination Test I or II. In this case, we will try to use the last computed step d_k anyway, and, if necessary, we will try increasing π_k to yield a negative value for the directional derivative $D\phi_{\pi_k}(d_k)$. However, if the directional derivative is nonnegative for any value $\pi \geq \pi_{k-1}$ of the penalty parameter, then the step is deemed an ascent direction for the penalty function and the algorithm terminates. Second, if the steplength coefficient must be cut below a

given $\alpha_{\min}$ in order to obtain a step satisfying the Armijo condition (3.11), then the search direction is deemed unsuitable and the algorithm fails. Even if we have a descent direction, this failure can occur due to finite precision arithmetic errors or if $\alpha_{\min}$ is too large relative to the curvature of the functions. Finally, if the algorithm terminates without satisfying the nonlinear program stopping conditions (4.1) and (4.2), then the maximum number of iterations has been reached. Though there exist techniques for continuing a stagnated run of the algorithm when an ascent direction for the penalty function or a short steplength coefficient is computed, we implement naïve failure tests in Algorithm 4.1 to aggressively challenge the robustness of our approach.

4.1 The CUTEr and COPS Collections

We first applied our stand-alone Matlab implementation of Algorithm 4.1 to a set of 44 equality constrained problems from the CUTEr [4, 19] and COPS [10] collections. Problems from these sets, for which AMPL models were available, were selected based on size — fewer than 10,000 variables — and were only considered if the code was able to provide a solution in the case where steps are computed via an exact solution of the primal-dual system (2.6) during each iteration. We note that W_k was set to the exact Hessian of the Lagrangian and that a multiple of the identity matrix was added to W_k, when necessary, to satisfy the positive definiteness of Assumption 3.4(e).

Table 4.1 contains a listing of the input parameters implemented in our code.
For the remaining parameters, we set, as is generally appropriate,

$$\sigma \leftarrow \tau(1-\epsilon) \tag{4.3}$$

$$\text{and} \quad \beta \leftarrow \max\left\{\frac{\|g_0 + A_0^T\lambda_0\|}{\|c_0\|+1}, 1\right\}. \tag{4.4}$$

Input	Value	Input	Value
ϵ_{feas}	10^{-6}	ε	10^{-4}
ϵ_{opt}	10^{-6}	$\alpha_{\min}$	10^{-8}
ϵ	0.1	$k_{\max}$	1000
τ	0.1	κ	1
η	10^{-8}	π_{-1}	1

Table 4.1: Input values for Algorithm 4.1 for CUTEr and COPS problems

As previously mentioned, this value for σ promotes consistency between Termination Tests I and II and (3.7). Such a value for β aims to reflect the relationship in scale between the primal and dual feasibility measures.

We illustrate the robustness of Algorithm 4.1 for problems in our test set by comparing it to a naïve inexact method that only enforces a reduction in the entire primal-dual residual. Our implementation of this approach, also done in Matlab, is identical to Algorithm 4.1 except that the stopping test

$$\textbf{for } j = 0, 1, 2, \ldots, n + t, \textit{ or until Termination Test I or II is satisfied}$$

for the inner **for** loop is replaced by

$$\textbf{for } j = 0, 1, 2, \ldots, n + t, \textit{ or until } (3.4) \textit{ is satisfied}$$

with $0 < \kappa < 1$. We performed multiple runs of this algorithm, which we call `ires`, for each problem in the test set and will refer to each run by the particular value of κ used.

It is important to note that as the results provided in this section are intended only as a simple illustration of the robustness of our approach, we did not implement a preconditioner for the primal-dual system for these numerical experiments. We stress, however, that preconditioning is an essential part of any implementation for many large-scale problems, as can be seen in Section 4.2.

Table 4.2 provides the percentage of problems successfully solved for each of the solvers. All of the failures for the `ires` algorithm occurred due to the fact that either the directional

Algorithm	`ires`					`isqp`
κ	2^{-1}	2^{-3}	2^{-5}	2^{-7}	2^{-9}	10
% Solved	50%	68%	80%	82%	86%	100%

Table 4.2: Success rates for algorithms `ires` and `isqp`

derivative $D\phi_{\pi_k}(d_k)$ of the penalty function was nonnegative for all allowable values of the penalty parameter π_k or the backtracking line search reduced the steplength coefficient α_k below the given tolerance $\alpha_{\min}$. Thus, we find that even for relatively small values of the tolerance parameter κ, the primal component d_k provided by GMRES can yield a value for the directional derivative $D\phi_{\pi_k}(d_k)$ of the penalty function that is not sufficiently negative for any $\pi \geq \pi_{k-1}$. In other words, `ires` runs the risk of computing nearly-exact solutions of the primal-dual system (2.6) that correspond to directions of insufficient decrease for the penalty function ϕ_{π_k}. Detailed results for algorithms `ires` and `isqp` are provided in Appendix A.

We close this section with a remark related to an extension of Algorithm 4.1 described in Section 3.5. Specifically, we claim that the rate of convergence can be controlled by requiring, within Termination Tests I and II, a condition of the form (3.26) for $0 < \{\kappa_k\} < 1$. Incidentally, by implementing such an approach, one can directly observe the extra cost associated with evolving the `ires` algorithm described above into a robust method. We performed addition tests of this type and found this extra cost to be minimal for the problems in our test set. For example, we compared `ires` with a hybrid approach, call it `ires-isqp`, that imposes inequality (3.26) within the step computation of Algorithm 4.1, where κ for the `ires` algorithm was set equal to $\kappa_k = 2^{-5}$ for all k. For the 35 problems solved by both of these algorithms, an average of only 0.5 extra total GMRES iterations over the entire run of the algorithm were required by `ires-isqp`. Moreover, by observing Termination Tests I and II within the iterative solver, the 9 problems left unsolved by `ires` (approximately 20%

of the total number of 44 problems) were all solved successfully by `ires-isqp`. Indeed, the extra cost is minimal with respect to the added robustness.

4.2 A Model PDE-constrained Problem

This section provides numerical results for Algorithm 4.1 applied to two instances of a model PDE-constrained problem. We present an overview of the model problem formulation, briefly describe a few details of the implementation, and present results related to tuning the input parameters κ, ϵ, and β to achieve fast convergence. Further details of the problem formulation and Matlab implementation can be found in [21].

PDE-constrained optimization problems represent a large class of difficult applications for which mathematical programming techniques are actively being applied and specialized. Types of problems in this class include optimal design, optimal control, shape optimization, and parameter estimation of PDE systems. Examples of specific applications include weather forecasting [12], aerodynamic design [41], optimal flow control [2, 3, 32], chemical engineering [1], and environmental engineering [21]. The computational requirements of Algorithm 4.1 are well-suited for many large-scale PDE-constrained problems due to its matrix-free nature. Thus, we are particularly interested in the efficiency of the algorithm for problems in this class.

The benchmarking environment of Haber and Hanson [21] considers a number of instances of the inverse problem of recovering an approximation for a model $x_m(z)$ based on measurement data $q_d(z)$ on the solution of a forward problem $x_f(z)$. The forward problem is assumed to be linear with respect to $x_f(z)$, so that the PDE — which is assumed to be defined on an appropriate domain Ω and equipped with suitable boundary conditions — can be written as

$$\mathcal{B}(x_m(z))x_f(z) = q_r(z), \tag{4.5}$$

where $\mathcal{B}$ is a differential operator depending on the model $x_m(z)$ and $q_r(z)$ is an appropriate right-hand-side for the (nonlinear) constraints. The problem to be solved is chosen as the minimization of a Tikhonov functional to find $x_m(z)$, which, after appropriate discretizations of the PDE and regularization functional have been applied, can be formulated as the constrained optimization problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \tfrac{1}{2}\|Qx_f - q_d\|^2 + \tfrac{\nu}{2}\|L(x_m - x_{ref})\|^2 \\ &\text{s.t. } B(x_m)x_f = q_r, \end{aligned} \tag{4.6}$$

where $x = (x_m, x_f)$. Here, x_m is the grid function approximating $x_m(z)$ and arranged as a vector, and similarly for q_r, x_f and q_d. Q is a projection operator for x_f onto the locations in Ω to which the data q_d are associated, L is a regularization matrix not dependent on x_m, $\nu > 0$ is a regularization parameter, x_{ref} is a given reference model, and B is a nonsingular matrix dependent on x_m.

We briefly describe the basic components of a SQP method for (4.6) before providing numerical results for two particular instances of the problem. In particular, let us derive the elements of the primal-dual system (2.6) used in the Matlab implementation described in [21]. The Lagrangian function (see (2.2)) for the finite-dimensional problem (4.6) is

$$\mathcal{L}(x,\lambda) = \tfrac{1}{2}\|Qx_f - q_d\|^2 + \tfrac{\nu}{2}\|L(x_m - x_{ref})\|^2 + \lambda^T V(B(x_m)x_f - q_r) \tag{4.7}$$

where $\lambda \in \mathbb{R}^t$ are Lagrange multipliers and $V \in \mathbb{R}^{t\times t}$ is a diagonal matrix defined to maintain the meaning of λ as a discretization of a continuous Lagrange multiplier $\lambda(z)$. Thus, the

first order optimality conditions for problem (4.6) are

$$\nabla \mathcal{L}(x, \lambda) = \begin{bmatrix} Q^T(Qx_f - q_d) + B_f^T V\lambda \\ \nu L^T L(x_m - x_{ref}) + B_m^T V\lambda \\ V(B(x_m)x_f - q_r) \end{bmatrix} = 0 \tag{4.8}$$

where $B_f \triangleq \partial(B(x_m)x_f)/\partial x_f = B(x_m)$ and $B_m = \partial(B(x_m)x_f)/\partial x_m$. As the computation of second order information is impractical for the given problem instances, the code uses a Gauss-Newton method (e.g., see [28]) to approximate the Hessian. Thus, for our test results we define the primal-dual system (see (2.6)) as

$$\begin{bmatrix} Q^TQ & 0 & B_f^TV \\ 0 & \nu L^TL & B_m^TV \\ VB_f & VB_m & 0 \end{bmatrix} \begin{bmatrix} d_m \\ d_f \\ \delta \end{bmatrix} = - \begin{bmatrix} Q^T(Qx_f - q_d) + B_f^T V\lambda \\ \nu L^T L(x_m - x_{ref}) + B_m^T V\lambda \\ V(B_f x_f - q_r) \end{bmatrix}. \tag{4.9}$$

It is worth noting that Assumptions 3.4(c) and (e) hold for the quantities in (4.9). Moreover, the code applies a preconditioner to (4.9) that uses a few steps of a Jacobi iteration for the forward problem and a limited memory BFGS approximation of the inverse of the reduced Hessian; again, details can be found in [21].

We apply Algorithm 4.1 to two instances of problem (4.6). For the first instance, which we refer to as `Elliptic`, the forward problem (4.5) takes on the form of an elliptic PDE and has applications in groundwater modeling and DC resistivity. In the second instance, call it `Parabolic`, the PDE is parabolic and time-dependent and the optimization problem (4.6) arises in fields such as optimal tomography and electromagnetic imaging. In general, the goal is to solve each problem on a relatively fine grid for a small value of the regularization parameter ν. However, for illustrative purposes we simply set $\nu \leftarrow 0.1$ and consider easily-managed grid sizes: 16 grid points in each of the three spatial dimensions for a total of

$n = 8192$ variables and $t = 4096$ constraints for problem `Elliptic`, and 8 grid points in each of three spatial and one time dimension for a total of $n = 4608$ variables and $t = 4096$ constraints for problem `Parabolic`.

Our goal in the following numerical experiments is to determine values for the parameters $(\kappa, \epsilon, \beta)$ that provide the best algorithmic performance; i.e., we are interested in appropriate values for three critical inputs so that the optimization process requires the least amount of computational effort. With respect to the remaining inputs, Table 4.3 contains a listing of the values implemented in our code, where again the value for σ depends on the input ϵ.

Input	Value	Input	Value
ϵ_{feas}	10^{-4}	ε	10^{-4}
ϵ_{opt}	10^{-4}	$\alpha_{\min}$	10^{-8}
τ	0.1	$k_{\max}$	100
σ	$\tau(1-\epsilon)$	π_{-1}	10^{-8}
η	10^{-8}		

Table 4.3: Input values for Algorithm 4.1 for two PDE-constrained problems

In terms of the three critical parameters $(\kappa, \epsilon, \beta)$, let us first claim (similarly to (4.4)) that an appropriate form for β is given by

$$\beta \leftarrow \beta' \left(\frac{\|g_0 + A_0^T \lambda_0\|}{\|c_0\| + 1} \right) \tag{4.10}$$

with $\beta' > 0$. In our tests, we set β according to (4.10) for various values of β'. Overall, we consider all possible combinations of the values in Table 4.4.

Input	Values
κ	$\{1.0, 0.5, 0.1\}$
ϵ	$\{0.9, 0.5, 0.1\}$
β'	$\{100, 10, 1\}$

Table 4.4: Critical inputs for Algorithm 4.1 for two PDE-constrained problems

Data was accumulated for three performance measures: the numbers of optimization

iterations (k), total SymQMR iterations (j), and matrix-vector products (Mv) with the constraint Jacobian. Matrix-vector products with the constraint Jacobian are performed both for the application of the preconditioner for the primal-dual system and within the iterative linear system solver itself. In fact, as the preconditioner can be seen to be highly effective for these problem instances — which is reflected in the fact that only small numbers of optimization and SymQMR iterations are required to find a solution for most combinations of the input parameters — we consider the required number of matrix-vector products with the constraint Jacobian as the primary measure of performance.

Results for all combinations of the input values $(\kappa, \epsilon, \beta)$ can be found in Appendix A. In particular, Tables 4.5 and 4.6 provide the two best input combinations in terms of required matrix-vector products with the constraint Jacobian (corresponding to the bold entries in Table A.4) for problems `Elliptic` and `Parabolic`, respectively.

κ	ϵ	β'	k	j	Mv
1.0	0.9	10	11	17	3768
1.0	0.5	10	8	19	3599

Table 4.5: Results for Algorithm 4.1 on problem `Elliptic`

κ	ϵ	β'	k	j	Mv
1.0	0.1	10	6	9	5635
0.5	0.1	10	5	11	5305

Table 4.6: Results for Algorithm 4.1 on problem `Parabolic`

Overall, the results provided in Tables 4.5, 4.6, and A.4 suggest the following general guidelines for the inputs $(\kappa, \epsilon, \beta)$. First, the failures illustrated in Table A.4 can be directly attributed to the fact that a low value for the input κ can hinder an otherwise convergent approach in the presence of an ill-conditioned primal-dual iteration matrix. That is, for problem `Parabolic`, the algorithm failed when SymQMR was unable to provide a solution satisfying (3.4) for $\kappa = 0.1$, and so the implementation was forced to use whatever final

primal step d_k was provided by the iteration, which in many cases turned out to be an ascent direction for the penalty function for all allowable values of the penalty parameter. Even without these failures, however, the results for problem `Elliptic` for $\kappa = 0.1$ suggest that the algorithm may spend an unnecessarily large amount of computational effort to compute nearly-exact solutions of the primal-dual system (4.9), despite the fact that steps satisfying Termination Test I or II for a larger κ may have been encountered during the solution process. Thus, the results suggest that relatively large values of the input κ may be appropriate, at least during early iterations, and fast convergence will instead be attained via careful selection of the parameters (ϵ, β). With respect to these values, the results imply that setting β by (4.10) with $\beta' \leftarrow 10$ may be appropriate. A specific value for ϵ did not entirely dominate the other tested values, though by examining Table A.4 in detail we find that ϵ in the range of 0.5 may be a reasonable initial choice.

In the end, if it is computationally tractable to do so, values of the inputs $(\kappa, \epsilon, \beta)$ should be tuned separately for each application, or should be set based on practical experience with related problem instances.

Chapter 5

Negative Curvature and Equality Constrained Optimization

5.1 Background and Motivation

In this chapter we consider issues related to the extension of the inexact line search SQP method for solving problem (2.1) described in Chapter 3 for applications where certain convexity assumptions about the problem formulation may fail to hold at some of the algorithm iterates. We present some preliminary strategies for this case and verify their usefulness, for clarity and simplicity, in the case where exact solutions of the primal-dual system are computed.

As previously mentioned, a standard assumption for line search SQP methods is that during each iteration, W_k is positive definite on the null space of the constraint Jacobian A_k (see Assumption 3.4(e)), in which case the SQP subproblem (2.4), rewritten here for

convenience as

$$\min_{d \in \mathbb{R}^n} \quad f_k + g_k^T d + \tfrac{1}{2} d^T W_k d \tag{5.1a}$$

$$\text{s.t.} \quad c_k + A_k d = 0, \tag{5.1b}$$

is well defined. For many applications, however, such an assumption is invalid in the sense that during iteration k there may exist $u \in \mathbb{R}^n$ with

$$A_k u = 0 \quad \text{and} \quad u^T W_k u < 0. \tag{5.2}$$

We refer to such a u as a *direction of negative curvature* for the subproblem (5.1) and note that the presence of directions of this type may thwart the progress of an SQP approach unless certain precautions are made. A solution to (5.1) does not exist in this setting, and so if a solution to the primal-dual system

$$\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \delta_k \end{bmatrix} = - \begin{bmatrix} g_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \tag{5.3}$$

exists, then it corresponds to a saddle point or perhaps a maximizer of the quadratic model of the objective (5.1a) over the linearized constraints (5.1b). Thus, (d_k, δ_k) obtained via (5.3) may not be an appropriate search direction with respect to the nonlinear program (2.1) and algorithmic modifications may be necessary to ensure progress and guarantee convergence.

Despite the difficulties that arise in this framework, a number of methods for constrained optimization that provide global convergence guarantees despite the presence of negative curvature directions have been proposed and implemented. Methods that impose a trust region boundary in the step computation procedure, such as those implemented in `Filter-SQP`

[13, 14, 16] and KNITRO-CG [6], are generally not disrupted in this setting as a solution to the SQP subproblem becomes well defined with the additional constraint. Line search methods, on the other hand, such as those implemented in LOQO [34] and IPOPT [35, 36, 37], often cope with negative curvature directions by first obtaining spectral information of the primal-dual matrix (3.1) via factorization techniques. It is known that if (3.1) has exactly n positive and t negative eigenvalues, then Assumption 3.4(e) holds at the current iterate (e.g., see [28]). Thus, an iterative procedure can be implemented to modify W_k in such a way that after a finite number of modifications the matrix is positive definite on the null space of A_k, and so usual SQP techniques can be applied in an unadulterated manner. The methods we develop in this chapter have some features in common with this approach, except that we may not always modify W_k if negative curvature directions are known to be present.

In summary, we are interested in the extension of our inexact SQP method for problems where negative curvature directions may be present; i.e., cases where Assumption 3.4(e) may not hold. A significant challenge in this context is that, without access to spectral information of the primal-dual matrix, it may not even be known whether directions of negative curvature are present at the current iterate for the given W_k. Thus, we begin by returning to the relatively benign case where exact solutions of the primal-dual system are obtained during each iteration. In this setting, we can more clearly present an enhancement to contemporary matrix modification strategies that should prove to be useful in an inexact environment, in ways that will be described in the last section.

This chapter is organized as follows. In Section 5.2, we consider an SQP algorithm where steps are computed via an exact solution to the primal-dual system and motivate an enhancement to common matrix modification strategies normally used to eliminate the presence of negative curvature directions. The global behavior of the approach will be discussed in Section 5.3, after which we present some numerical results for a set of test

problems in Section 5.4. Closing remarks and issues related to extending the ideas of this chapter to an inexact SQP framework are presented in Section 5.5.

5.2 Step Computation and Selection

In this section, we develop new ideas for handling the presence of negative curvature directions in a line search SQP framework. As in Chapter 3, we suggest that a powerful tool for ensuring global convergence is the reinforcement of certain model reductions through appropriately chosen conditions, as opposed to relying on the accurate solution of a series of well defined subproblems.

Let us begin by presenting the general form of a particular component of the approach that will be developed in detail in the remainder of this section. As mentioned above, some contemporary line search methods [34, 37] will iteratively modify W_k until the matrix is positive definite on the null space of A_k, which can be verified if the primal-dual matrix (3.1) has n positive, t negative eigenvalues, and no zero eigenvalues; i.e., the *inertia* of the primal-dual matrix is *correct* in that

$$inertia\left(\begin{bmatrix} W_k & A_k^T \\ A_k & 0 \end{bmatrix}\right) = (n, t, 0). \tag{5.4}$$

We frame our techniques around this type of procedure, and still assume that after a finite number of modifications the matrix W_k will be positive definite on the null space of A_k, but emphasize that a step may be accepted before (5.4) holds. The algorithm is to be used as the step computation procedure in a line search SQP framework, resulting in the primal-dual step (d_k, δ_k).

Algorithm 5.1. Outline of a Matrix Modification Strategy

for $j = 0, 1, 2, \ldots$

Compute (d_k, δ_k) *via* (5.3)

if (5.4) *holds, then* **break**

if (d_k, δ_k) *is acceptable, then* **break**

Modify W_k

endfor

The key difference between Algorithm 5.1 and standard matrix modification strategies is that we intend to characterize certain properties of the solution (d_k, δ_k) that may deem the step acceptable despite the fact that negative curvature may be present. Modifications of W_k may still be necessary in the step computation procedure, though excessive modifications may be avoided if an acceptable search direction is encountered early and recognized appropriately.

We develop the notion of an acceptable step in the following manner. First, let us recall the decomposition

$$d_k = u_k + v_k \tag{5.5}$$

with the tangential component u_k lying in the null space of A_k and the normal component v_k lying in the range space of A_k^T. As part of a solution to the primal-dual system (5.3), the component v_k can be viewed as a productive move toward satisfying the constraints. Then, in conjunction with this step, the component u_k would ideally correspond to a step towards optimality; i.e., minimizing the quadratic model of the objective from the point $x_k + v_k$ over the null space of A_k.

An unfortunate consequence of the presence of negative curvature directions, however, is that the tangential component may be unbounded, or may not correspond to a productive step for minimizing the objective. Still, despite any undesirable properties of u_k, we claim

that the full step d_k can be considered acceptable if the behavior of this tangential step can be observed and subsequently controlled. This can be done in one of two ways. First, if the curvature along the complete primal step d_k is sufficiently positive with respect to the length of u_k, as in

$$\xi_1 \|u_k\|^2 \leq d_k^T W_k d_k \tag{5.6}$$

with $\xi_1 > 0$, then we may assume that u_k is an appropriate step towards optimality and the step may be acceptable. Here, $\|\cdot\|$ denotes a norm on $\mathbb{R}^n$. Alternatively, if the length of the tangential component is not too large with respect to the length of the normal component, as in

$$\xi_2 \|u_k\|^2 \leq \|v_k\|^2 \tag{5.7}$$

with $\xi_2 > 0$, then regardless of the quality of u_k the full step d_k may be considered acceptable in that it at least provides a sufficient move towards satisfying the constraints. Overall, a given step d_k may be deemed acceptable based on properties of its corresponding tangential and normal components, which, in the presence of negative curvature directions, must correspond to having a bounded tangential step with respect to appropriate quantities.

Later on, in our analysis in Section 5.3, we find that an alternative motivation for the inequalities (5.6) and (5.7) relates to our ability to label the resulting steps as corresponding to one of two index sets, K_1' and K_2', defined similar to K_1 and K_2 from Section 3.4.

A difficulty, of course, of relying on the inequalities (5.6) and (5.7) is that we do not have access to the components u_k and v_k during the solution of the primal-dual system (5.3). We never compute these components explicitly. Practically, however, we can exploit available information in order to verify that one or both of the conditions hold for a given d_k. We consider this issue with the following lemma.

Lemma 5.2. *If a step d_k satisfies*

$$\theta_1 \left(\|d_k\|^2 - \|A_k d_k\|^2 / \|A_k\|^2 \right) \leq d_k^T W_k d_k \tag{5.8}$$

for $\theta_1 > 0$, then (5.6) holds for some $\xi_1 > 0$. Similarly, if

$$\theta_2 \|d_k\|^2 \leq \|A_k d_k\|^2 / \|A_k\|^2 \tag{5.9}$$

for $0 < \theta_2 < 1$, then (5.7) holds for some $\xi_2 > 0$.

Proof. First, by the fact that $A_k u_k = 0$, we have

$$\|A_k d_k\| = \|A_k v_k\| \leq \|A_k\| \|v_k\|,$$

and so

$$\|v_k\|^2 \geq \|A_k d_k\|^2 / \|A_k\|^2. \tag{5.10}$$

Moreover, $A_k u_k = 0$ and the fact that v_k lies in the range space of A_k^T implies

$$\|d_k\|^2 = \|u_k\|^2 + \|v_k\|^2. \tag{5.11}$$

By (5.10) and (5.11),

$$\begin{aligned} \|u_k\|^2 = & \ \|d_k\|^2 - \|v_k\|^2 \\ \leq & \ \|d_k\|^2 - \|A_k d_k\|^2 / \|A_k\|^2, \end{aligned}$$

and so a step satisfying (5.8) satisfies (5.6) for $\xi_1 = \theta_1$. Similarly, by (5.10) and (5.11), a

step satisfying (5.9) has

$$\begin{aligned} \|u_k\|^2 = & \ \|d_k\|^2 - \|v_k\|^2 \\ \leq & \ \tfrac{1}{\theta_2}\|A_k d_k\|^2/\|A_k\|^2 - \|v_k\|^2 \\ \leq & \ \left(\tfrac{1-\theta_2}{\theta_2}\right)\|v_k\|^2, \end{aligned}$$

and so (5.7) holds for $\xi_2 = \theta_2/(1-\theta_2)$. □

We are now ready to present a complete algorithm for this framework.

Algorithm 5.3. SQP with Negative Curvature Safeguards

Given parameters $0 < \tau, \eta, \theta_2 < 1$ *and* $0 < \theta_1, \varepsilon$

Initialize x_0, λ_0, *and* $\pi_{-1} > 0$

for $k = 0, 1, 2, \ldots,$ *until a convergence test for* (2.1) *is satisfied*

Compute f_k, g_k, c_k, W_k, *and* A_k *and set* $\pi_k \leftarrow \pi_{k-1}$ *and* $\alpha_k \leftarrow 1$

for $j = 0, 1, 2, \ldots$

Compute (d_k, δ_k) *via* (5.3)

if (5.4) *holds, then* **break**

if (5.8) *or* (5.9) *holds, then* **break**

Modify W_k

endfor

if (3.7) *does not hold, set* $\pi_k \leftarrow \chi_k + \varepsilon$

Perform a backtracking line search to obtain α_k *satisfying* (2.15)

Set $(x_{k+1}, \lambda_{k+1}) \leftarrow (x_k, \lambda_k) + \alpha_k(d_k, \delta_k)$

endfor

5.3 Global Analysis

This section contains an outline of a global convergence proof for Algorithm 5.3 under Assumptions 3.4(a)-(d). We assume that Assumption 3.4(e) holds, but only for iterations where (5.4) is found to be satisfied; i.e., we tighten our notion of the primal-dual matrix (3.1) having the correct inertia by only including cases where Assumption 3.4(e) is satisfied. We borrow the results from Section 3.4 that remain relevant in this context and intend only to pinpoint changes in the theory required to confront the presence of negative curvature.

Again, our analysis hinges on our ability to classify different types of productive steps. In Section 3.4, such a distinction was made implicitly by relying on our assumptions of the problem formulation and the required qualities of an accepted step. In the presence of negative curvature directions, however, this sort of analysis breaks down as certain desirable properties of the tangential component u_k are lost. Thus, the importance of enforcing the inequalities (5.8) and (5.9) in the definition of an appropriate search direction when (5.4) fails to hold can be seen almost immediately when trying to extend analysis similar to that of Section 3.4 to the negative curvature context. Through these conditions, we are able to impose an explicit distinction between two types of desirable steps when an implicit distinction cannot be made.

The analysis proceeds as follows. First, we can begin by noting that under Assumptions 3.4(a)-(d), the results of Lemma 3.5 are unaffected by negative curvature directions. Thus, we may refer to Lemma 3.5 in the results that follow. Unfortunately, the results of Lemmas 3.6 and 3.7 hold only for those steps computed when (5.4) is satisfied, but along

with inequalities (5.8) and (5.9) we may consider the sets of indices

$$\begin{aligned} K_1 &\triangleq \{k : (5.4) \text{ holds and } \|u_k\|^2 \geq \gamma_4 \|v_k\|^2\}, \\ K_2 &\triangleq \{k : (5.4) \text{ holds and } \|u_k\|^2 < \gamma_4 \|v_k\|^2\}, \\ K_1' &\triangleq \{k : (5.8) \text{ holds when (5.4) does not}\}, \\ \text{and } K_2' &\triangleq \{k : (5.9) \text{ holds when (5.4) and (5.8) do not}\}, \end{aligned}$$

where it can be seen that $k \in K_1 \cup K_2 \cup K_1' \cup K_2'$ for all k in Algorithm 5.3. Here, $\gamma_4 > 0$ is chosen large enough as described in Lemma 3.7 and, similar to the analysis of Section 3.4, the quantity

$$\Theta_k' \triangleq \begin{cases} \|u_k\|^2 + \|c_k\|, & k \in K_1 \cup K_1', \\ \|c_k\|, & k \in K_2 \cup K_2'. \end{cases} \tag{5.12}$$

can be used for bounding the length of the primal step and the directional derivative of the penalty function.

The next result has a flavor similar to Lemma 3.8.

Lemma 5.4. *There exists $\gamma_5 > 1$ such that, for all k,*

$$\|d_k\|^2 \leq \gamma_5 \Theta_k',$$

and hence

$$\|d_k\|^2 + \|c_k\| \leq 2\gamma_5 \Theta_k'. \tag{5.13}$$

Proof. For $k \in K_1 \cup K_1'$, we find by Lemma 3.5 that

$$\begin{aligned} \|d_k\|^2 &= \|u_k\|^2 + \|v_k\|^2 \\ &\leq \|u_k\|^2 + \gamma_2 \|c_k\|. \end{aligned}$$

Similarly, Lemmas 3.5 and 5.2 and (5.9) imply that for $k \in K_2 \cup K_2'$

$$\begin{aligned} \|d_k\|^2 = & \|u_k\|^2 + \|v_k\|^2 \\ \leq & (\max\{(1-\theta_2)/\theta_2, \gamma_4\} + 1)\|v_k\|^2 \\ \leq & (\max\{(1-\theta_2)/\theta_2, \gamma_4\} + 1)\gamma_2\|c_k\|. \end{aligned}$$

To establish (5.13) we note that $\Theta_k' + \|c_k\| \leq 2\Theta_k'$ for all k. □

Similarly, the following resembles Lemma 3.9.

Lemma 5.5. *The directional derivative of ϕ_π along a step d satisfies*

$$D\phi_\pi(d) = g^T d - \pi\|c\|. \tag{5.14}$$

Moreover, there exists $\gamma_6 > 0$ such that, for all k,

$$D\phi_{\pi_k}(d_k) \leq -\gamma_6\Theta_k'.$$

Proof. The proof of equality (5.14) can be found in [28].

From the penalty parameter update (3.7) we have

$$D\phi_{\pi_k}(d_k) \leq -\tfrac{\omega_k}{2} d_k^T W_k d_k - \tau\pi_k\|c_k\|.$$

By Lemma 3.7, (5.8), and (2.10), we have that $\omega_k = 1$ for $k \in K_1 \cup K_1'$ and thus

$$D\phi_{\pi_k}(d_k) \leq -\min\left\{\tfrac{\theta_1}{2}, \tfrac{\mu}{4}\right\}\|u_k\|^2 - \tau\pi_k\|c_k\|.$$

Similarly, for $k \in K_2 \cup K_2'$ we have from (2.10)

$$D\phi_{\pi_k}(d_k) \leq -\tau\pi_k\|c_k\|.$$

The result holds for $\gamma_6 = \min\left\{\frac{\theta_1}{2}, \frac{\mu}{4}, \tau\pi_k\right\}$, which is bounded away from zero as $\{\pi_k\}$ is nondecreasing. □

Together, the previous two lemmas can be used to bound the sequence of steplength coefficients in a manner similar to that of Lemma 3.10; details of the proof are omitted here. However, the fact that the penalty parameter remains bounded relies heavily on our definition of the index sets K_1' and K_2'.

Lemma 5.6. *The sequence of penalty parameters $\{\pi_k\}$ is bounded above and $\pi_k = \pi_{\bar{k}}$ for all $k \geq \bar{k}$ for some $\bar{k} \geq 0$.*

Proof. By Lemma 3.11, we claim that there exists some penalty parameter value, call it $\hat{\pi}$, beyond which π_k will never be increased during at iteration when (5.4) holds. Thus, in the remainder of this proof let us assume that for the current iteration we have $k \in K_1' \cup K_2'$.

In Algorithm 5.3, the parameter π_k is chosen to satisfy (3.7), and so

$$\left[-g_k^T d_k - \tfrac{\omega_k}{2} d_k^T W_k d_k\right] + (1-\tau)\pi_k\|c_k\| \geq 0. \tag{5.15}$$

The result follows from our ability to bound the term inside the square brackets with respect to the infeasibility measure. First, by (2.10), $\omega_k = 1$ for $k \in K_1'$, and so (5.3) and

Assumption 3.4(b) imply that for some $\gamma_7 > 0$ we have

$$\begin{aligned} -g_k^T d_k - \tfrac{1}{2} d_k^T W_k d_k \geq & \; -g_k^T d_k - d_k^T W_k d_k \\ = & \; -d_k^T A_k^T (\lambda_k + \delta_k) \\ = & \; -c_k^T (\lambda_k + \delta_k) \\ \geq & \; -\gamma_7 \|c_k\|. \end{aligned}$$

Similarly, for $k \in K_2'$, Assumptions 3.4 and the analysis of Lemma 3.8 imply that for some $\gamma_8, \gamma_8' > 0$ we have

$$\begin{aligned} -g_k^T d_k - \tfrac{\omega_k}{2} d_k^T W_k d_k \geq & \; -\gamma_8 \|d_k\|^2 \\ \geq & \; -\gamma_8' \|v_k\|^2 \\ \geq & \; -\gamma_8' \gamma_2 \|c_k\|. \end{aligned}$$

These results together imply

$$-g_k^T d_k - \tfrac{\omega_k}{2} d_k^T W_k d_k \geq -\max\{\gamma_7, \gamma_8' \gamma_2\} \|c_k\|,$$

and so (5.15) is always satisfied for

$$\pi_k \geq \max\{\gamma_7, \gamma_8' \gamma_2\} / (1 - \tau).$$

Therefore, if $\pi_{\bar{k}} \geq \max\{\hat{\pi}, \max\{\gamma_7, \gamma_8' \gamma_2\}/(1-\tau)\}$ for some iteration number $\bar{k} \geq 0$, then $\pi_k = \pi_{\bar{k}}$ for $k \geq \bar{k}$. This, together with the fact that whenever Algorithm 5.3 increases the penalty parameter it does so by at least a positive finite amount, proves the result. $\square$

Finally, we claim that results similar to those of Lemma 3.12 and Theorem 3.13 fol-

low from the lemmas above and so Algorithm 5.3 is globally convergent under Assumptions 3.4(a)-(d).

5.4 Numerical Results

This section contains numerical results for a particular implementation of Algorithm 5.3 based on the `KNITRO-Direct` algorithm from the `KNITRO 5.0` software package [40]. The goal of these numerical experiments is to investigate the computational tradeoffs between a standard matrix modification strategy and new approach described in this chapter. In fact, the default `KNITRO-Direct` algorithm usually reverts to a trust region iteration in the presence of negative curvature directions. For our tests, however, we enabled internal options to ensure that our implementation runs as a pure line search algorithm. The matrix modifications are performed by adding to W_k, which is initially set to the Hessian of the Lagrangian, a constant ν times the identity matrix, where ν is initialized to zero for each new iterate, set to 10^{-15} if a first modification is needed, and increased by a factor of 10^2 for each remaining modification iteration.

We tested our code using a set of 36 equality constrained problems from the CUTEr [4, 19] and COPS [10] test set collections. The set is composed of all of the problems considered in the numerical experiments of Section 4.1 for which the inertia of the primal-dual matrix was found to be incorrect during at least one iteration. For evaluating our new step acceptance conditions, we computed l_2-norms for all components in inequalities (5.8) and (5.9) except for A_k, for which we computed the l_1-norm and observed the inequality $\|A_k\|_1^2 \leq t\|A_k\|_2^2$. Table 5.1 contains a listing of the parameters used in our code. The remaining parameters, such as those related to the termination test for solving the nonlinear program and the line search procedure, are equal to those in Table 4.1.

We compare the results of a line search SQP method using a standard matrix modification

Parameter	Value	Parameter	Value
τ	0.1	θ_1	1
η	10^{-8}	θ_2	0.75
π_{-1}	1	ε	1

Table 5.1: Input values for Algorithm 5.3

approach (i.e., an approach similar to Algorithm 5.3 except that inequalities (5.8) and (5.9) are never observed), call it algorithm `matmod`, and our implementation of Algorithm 5.3, which we refer to as `matmod_new` in the results that follow. It is clear from the description of Algorithm 5.3 that during a single iteration for the same iterate, `matmod_new` will perform at most, but never more than the number of matrix modifications performed by `matmod`. However, as the algorithms may compute different search directions in the solution process, the total number of iterations for solving the nonlinear program may differ between the two approaches. For these reasons, and since a majority of the computational effort for each algorithm is often spent in the step computation procedure, we suggest that the total number of iterations and the total number of primal-dual matrix factorizations over a complete run of each algorithm are appropriate measures for comparison.

Results for the two algorithms are summarized in Figures 5.1 and 5.2 in terms of logarithmic performance profiles, as described in [9]. Here, the leftmost values indicate the percentage of times each algorithm solves a given problem using the least value of the given measure, i.e., number of iterations or matrix factorizations. The values fail to add to one as ties are present. The rightmost function values illustrate the robustness of each approach; i.e., the percentage of times that a given problem is solved.

The results are quite good for the new approach. Not only does an algorithm that employs inequalities (5.8) and (5.9) often require fewer iterations to find a solution, but a considerable amount of savings is often experienced in terms of factorizations of a primal-dual matrix. Possible explanations for the impressive performance of `matmod_new` are that

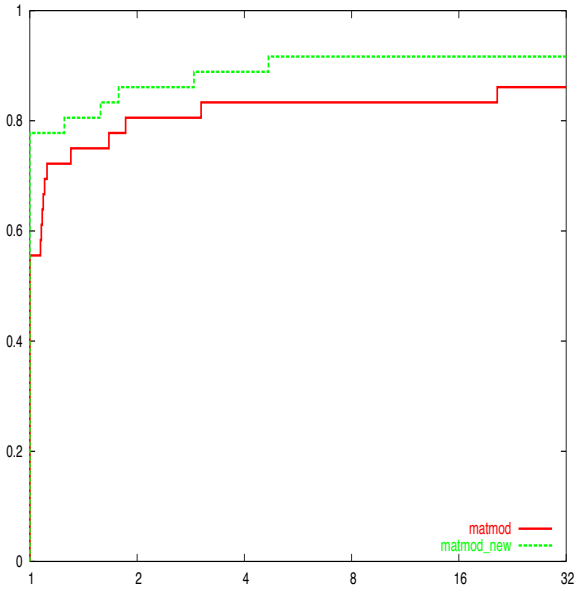

Figure 5.1: Performance profile for iterations in Algorithm 5.3

good search directions are indeed computed despite the presence of negative curvature or that the algorithm benefits by simply escaping troubling regions quickly. In any case, we believe that these results are an encouraging sign for extending the ideas of this chapter to an inexact SQP framework, for reasons described in the following section. Detailed results for both algorithm `matmod` and `matmod_new` can be found in Appendix B.

5.5 Final Remarks

In this chapter we have motivated and presented an enhancement to standard matrix modification strategies for ensuring the global convergence of a line search SQP method in the presence of negative curvature directions. The main idea of the approach is to quantify specific properties that may deem a solution to the primal-dual system (5.3) acceptable despite the fact that a solution to the corresponding SQP subproblem (5.1) may not exist. In this manner, we found that global convergence guarantees can be maintained and for a set of test

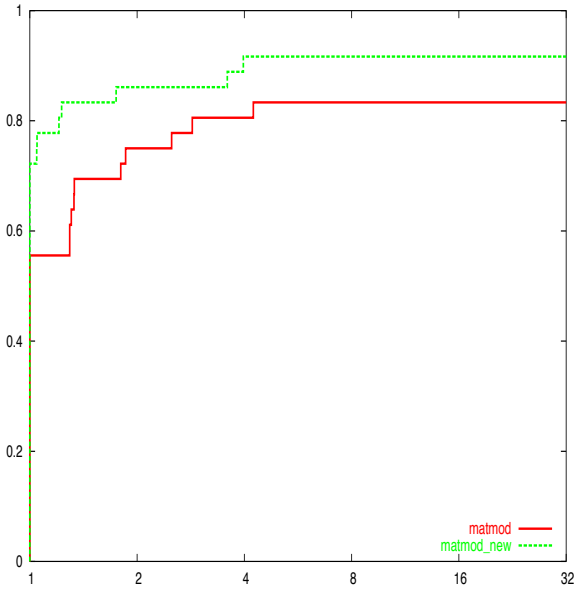

Figure 5.2: Performance profile for matrix factorizations in Algorithm 5.3

problems the total number of primal-dual matrix factorizations – and in many cases even the number of total iterations for solving the nonlinear program – was consistently reduced.

We believe that the ideas and analysis of this chapter will prove to be particularly useful in an inexact SQP framework. Certainly, in cases where spectral information of the primal-dual matrices is known, a reduction in the number of factorizations required to find a solution can correspond to significant savings in computational effort. However, in such cases one can still be comforted in the fact that as soon as W_k has been modified sufficiently in order for the matrix to be positive definite on the null space of A_k, the solution to the primal-dual system can instantly be recognized as acceptable.

In an inexact setting, we do not have this luxury and the question of when to modify W_k becomes a very difficult one. For example, consider a situation where the algorithm has performed a significant amount of work without yet computing an acceptable step. Without spectral information of the current primal-dual matrix, it is unknown whether W_k must be

modified before an acceptable solution could be found or if simply a more exact solution to the primal-dual system is required. Thus, the establishment of conditions that may deem a step acceptable even before the correct inertia has been obtained can yield significant gains. Matrix modifications may still be necessary, though we hope that the number of times that the algorithm must confront the difficult choice between modifying the matrix and solving the system more exactly will be reduced. Thus, inequalities (5.8) and (5.9) may prove quite useful in an inexact environment.

Finally, we note that one disadvantage of Algorithm 5.3 is that the solution (d_k, δ_k) must be computed before inequalities (5.8) and (5.9) can be verified. Computationally, this may negate the benefits attained by having to perform fewer factorizations. In an inexact SQP framework, however, such as when an iterative linear system solver is applied to the primal-dual system, this may not be an issue as the (inexact) solution (d_k, δ_k) is often available throughout the step computation process.

Chapter 6

Flexible Penalty Functions for Equality Constrained Optimization

6.1 Introduction

In this chapter we focus on step acceptance mechanisms for nonlinear constrained optimization. We discuss some of the main advantages and disadvantages of contemporary strategies and present a new globalization approach designed to promote long productive steps and fast convergence. The method is supported by global convergence guarantees to first order optimal points under common conditions.

As previously mentioned (see Section 2.3), the two most popular step acceptance tools in use today are penalty functions and filter mechanisms. One disadvantage of a penalty function relates to the monotonicity required when updating the penalty parameter π throughout a run of the algorithm. In fact, nonmonotone updates for the penalty parameter are available that maintain global convergence guarantees, but such methods often rely on ad hoc heuristics that eventually fall back on the convergence properties of monotone strategies,

and so we do not discuss them here. Depending on the specific update strategy used, π may at some point be set to an excessively large value, even at a point that is relatively far from a solution. As a result, a large priority will be placed on computing steps that produce sufficient reductions in constraint infeasibility, effectively "blocking" steps that move away from the feasible region. This can be detrimental as empirical evidence has shown that accepting steps that temporarily increase infeasibility can often lead to fast convergence. Figure 6.1 illustrates this blocking behavior of a penalty function, where we highlight the region where the step would have produced a reduction in the objective f (and so may have been acceptable to a filter). Here, the point $p_k = (\|c(x_k)\|, f(x_k))$ corresponds to the constraint infeasibility measure and objective function evaluated at the current iterate x_k.

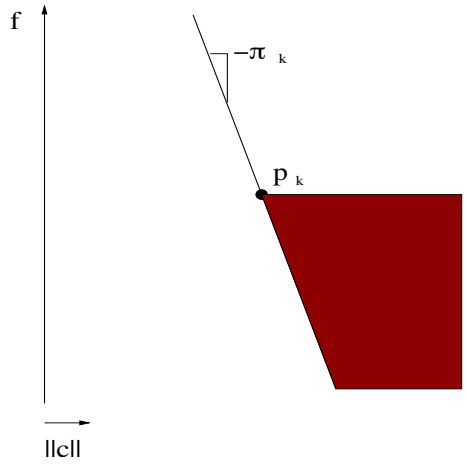

Figure 6.1: A region of points blocked by the penalty function ϕ_{π_k}

We note that a second disadvantage of a penalty function is that a low value of π may block steps that improve feasibility but increase f. However, modern step acceptance strategies effectively deal with this problem by defining local models of ϕ_π, such as the model m_k defined in Section 2.3, with which an adequately large value of π can be determined to avoid excessive blocking. In summary, our view is that the main weakness of penalty-based strategies is the blocking effect illustrated in Figure 6.1.

One disadvantage of a filter mechanism is that a step can be blocked by a filter entry, i.e., historical information of the problem functions, when in fact the step is a productive move toward a solution from the current iterate. This is particularly worrisome when steps are blocked that would amount to a sufficient reduction in constraint infeasibility. Figure 6.2 depicts a filter with a single element, call it a, where the point p_k corresponding to the current iterate is shown as the isolated point with an objective value sufficiently less than the filter entry. The shaded portion illustrates one region of points that are blocked by the filter, despite the fact that a step into this region would correspond to a reduction in constraint infeasibility from the current iterate (and so would be accepted by a penalty function for a sufficiently large value of the penalty parameter).

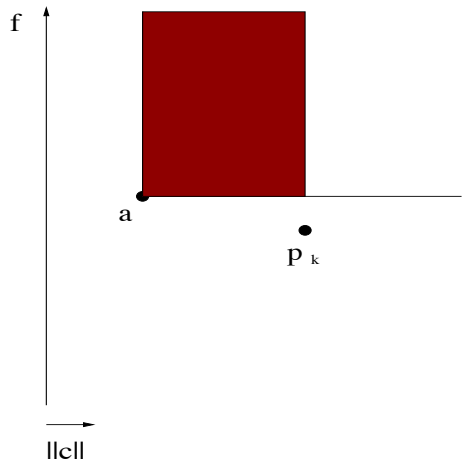

Figure 6.2: A region of points blocked by a filter with entry a

In an extreme example, consider the case where the filter entry a in Figure 6.2 is a Pareto optimal solution to the multiobjective optimization problem of minimizing the pair $(\|c(x)\|, f(x))$ over all $x \in \mathbb{R}^n$. A point is Pareto optimal if it cannot be dominated by any other point. Thus, if the current iterate again corresponds to the point p_k in Figure 6.2, then all paths from p_k to the feasible region must pass through a region of points dominated by a. Feasibility can only be attained if a single computed step were to fall beyond the region

dominated by the filter entry or if a backup mechanism, such as a feasibility restoration phase, were implemented.

In summary, both penalty functions and filters can be shown to block different types of productive steps. A penalty function suffers from high priority being placed on improving feasibility and convergence can be slowed by forcing the algorithm to hug the feasible region. A filter mechanism, on the other hand, suffers from handling problem (2.1) too much like a multiobjective optimization problem, when in fact a certain priority on converging to the feasible region may be appropriate, especially as the algorithm progresses.

This chapter is organized as follows. In Section 6.2 we develop our new globalization mechanism, which will be used in the presentation of a complete line search SQP algorithm presented in Section 6.3. An analysis of the global behavior of this approach is provided in Section 6.4, after which we present some numerical results for a set of test problems in Section 6.5.

6.2 A Flexible Penalty Function

In this section, we outline a new step acceptance strategy for iterative algorithms for solving problem (2.1). By observing the strengths and weaknesses of penalty functions and filters, we hope to emulate the accepting behavior of both methods while attempting to avoid any blocking of productive steps.

We make the following remarks related to the appropriateness of using a filter or a penalty function during different stages of an algorithm for problem (2.1). First, during early iterations, the filter mechanism has the benefit that a variety of types of steps are considered acceptable. For example, for a one-element filter, i.e., a filter containing only an entry corresponding to the current iterate, a step will be accepted as long as a sufficient reduction in the objective, constraint infeasibility, or both, is attained. This may be of use

to promote long steps during early iterations when an appropriate value for the penalty parameter may not yet be known. However, during later iterations, it may be reasonable to assume that an appropriate value for the penalty parameter may be determinable based on information computed throughout the run of the algorithm. This value can be used to correctly block certain types of steps from increasing constraint infeasibility. The use of a penalty function in later iterations may also avoid the risk of blocking steps that decrease constraint infeasibility when convergence toward the feasible region should be prioritized.

In an attempt to define a single mechanism that will capture all of the these characteristics, and given that the penalty function approach appears to be more flexible than a filter in that it permits a reweighting of objective and constraint infeasibility measures, we present an improvement to the penalty strategies.

Let us begin by commenting on an enhancement of a penalty function approach implemented in some current strategies (e.g., see [39]) in order to motivate our method. At the start of iteration k, a specific value π_{k-1} of the penalty parameter is carried over from the previous iteration. If the algorithm were to maintain this value, then only a step corresponding to a move into the region sufficiently below the solid line in Figure 6.3 would be acceptable. Upon the calculation of d_k, the strategy may determine that an increase of the penalty parameter to some value $\bar{\pi}_k > \pi_{k-1}$ may be appropriate, in which case only a step corresponding to a move into the region sufficiently below the dashed line in Figure 6.3 would be acceptable. Rather than automatically set $\pi_k \leftarrow \bar{\pi}_k$, however, a simple heuristic that maintains the global convergence properties of the algorithm is to first compute the function values for $\bar{x} = x_k + d_k$, namely $\|c(\bar{x})\|$ and $f(\bar{x})$. If $(\|c(\bar{x})\|, f(\bar{x}))$ lies sufficiently below the dashed line in Figure 6.3, then we may accept the step and indeed set $\pi_k \leftarrow \bar{\pi}_k$. However, if $(\|c(\bar{x})\|, f(\bar{x}))$ lies sufficiently below the solid line, then the step could be considered acceptable for $\pi_k \leftarrow \pi_{k-1}$, effectively avoiding an increase in the penalty parameter.

In summary, these strategies do not consider a single value of π at x_k, but rather may select from a pair of values depending on the actual reductions attained by the step. Thus, we can view the region of acceptable points as that lying below the solid *or* dashed line in Figure 6.3.

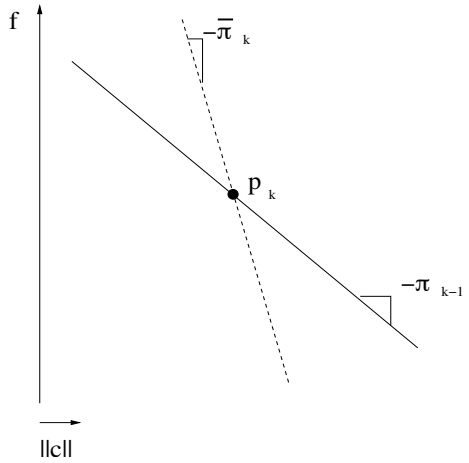

Figure 6.3: Illustration of the iterative nature of penalty parameter updates

An extension of this idea forms the basis of the method we now propose. Consider the collection of penalty functions

$$\begin{aligned} \phi_\pi(x) &\triangleq f(x) + \pi\|c(x)\|, \\ \pi &\in [\pi^l, \pi^u], \end{aligned} \tag{6.1}$$

for $0 \leq \pi^l \leq \pi^u$. We define a step to be acceptable if a sufficient reduction in ϕ_π has been attained for at least one $\pi \in [\pi^l, \pi^u]$. Clearly, if π^l is always chosen to equal π^u, then this approach is equivalent to using a penalty function with a fixed π during each iteration. Alternatively, if $\pi^l = 0$ and $\pi^u = \infty$, then this approach has the form of a one-element filter. In general, the region of acceptable points is that given by the region down and to the left of the piecewise linear function illustrated in Figure 6.4, where the "kink" in the function always occurs at $p_k = (\|c(x_k)\|, f(x_k))$, corresponding to the current iterate x_k. As the penalty parameter π is allowed to fluctuate in the interval $[\pi^l, \pi^u]$, we refer to (6.1) as a

"flexible" penalty function.

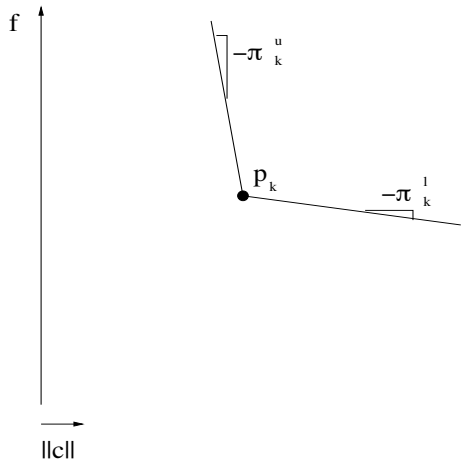

Figure 6.4: Region of acceptable points from p_k for a flexible penalty function

The practical behavior of an algorithm employing a standard penalty function ϕ_π depends heavily on the update strategy of the single penalty parameter π. For our method, we now need to consider the update strategies for two parameters, π^l and π^u. However, as different requirements in terms of convergence guarantees are necessary for each of these two boundary values, and as they have significantly different practical effects, we have the ability to design their updates in a manner suitable for accepting long productive steps within a globally convergent framework.

6.3 A Line Search SQP Method

In this section we propose a technique for employing a flexible penalty function in the context of a line search SQP method for problem (2.1). An analysis of the global behavior of the approach under common conditions is the topic of Section 6.4.

In order to motivate our handling of π^l and π^u and clarify the necessary features of the algorithm, let us recall the general form of a convergence proof for a SQP method employing

a penalty function (see Chapters 3 and 5). We claim that convergence can be guaranteed under common assumptions in the following manner.

(a) First, consider a penalty function ϕ_π for some given $\pi > 0$ and suppose that an infinite sequence of steps are computed such that each step yields a sufficient reduction in ϕ_π. If the reduction attained in each step is large with respect to appropriate measures, including the infeasibility measure $\|c\|$, then we claim that these steps will eventually correspond to reductions in the first order optimality conditions of (2.1), thus driving the search toward a solution point of the nonlinear program. This phenomenon does not require that π takes on any specific value — only that the penalty parameter eventually remains fixed. In our method, ϕ_{π^l} will play the role of this penalty function.

(b) At a given point x, it may not be possible to compute a step that will yield a reduction in ϕ_{π^l} that is large enough with respect to the current infeasibility measure $\|c(x)\|$. Such can be the case, for example, if x is a stationary point of ϕ_{π^l} and is infeasible to problem (2.1). In this situation, π^l must be set to a higher value, call it $\bar{\pi}^l$, so that reductions in $\phi_{\bar{\pi}^l}$ are attainable from x.

(c) Increases in π^l must be performed in such a way that the parameter eventually becomes fixed at a finite value, or else the algorithm may not adequately drive the optimality conditions of problem (2.1) to zero. Rather than control π^l directly in this manner, however, we propose to control an upper bound for the penalty parameter, π^u, that can be shown to remain bounded under common conditions if updated using an appropriate strategy. By maintaining a separate quantity π^u as an upper bound on π^l in (6.1), we allow the algorithm to be more accepting of points that reduce constraint infeasibility while being able to maintain a value of π^l that is as small as possible so that the algorithm can also move freely away from the feasible region. As π^u will remain

bounded, so will π^l, and so eventually the algorithm will compute an infinite number of steps that reduce ϕ_{π^l} and move toward a first order optimal point of problem (2.1).

With this general form of a convergence theory in mind, we present an algorithm with the following features. For simplicity, we assume that the step is computed via the exact solution to the primal-dual system (2.6), after which we perform a backtracking line search to compute a steplength coefficient α_k satisfying a condition similar to the Armijo condition (2.15). The specific quantities involved in the line search procedure are presented later on as they are better described once further details of the approach have been established. Overall, the complete globalization strategy requires effective methods for handling three parameters. First, we require a concrete strategy for updating the values π^l and π^u defined in the previous section. Second, we require a mechanism for choosing a value, call it $\dot{\pi}$, in the establishment of an appropriate line search condition. Since π^u and $\dot{\pi}$ will be set before the line search and π^l will be updated after the line search, we define strategies to set π^u_k, $\dot{\pi}_k$, and π^l_{k+1} during iteration k (where $\pi^l_0 > 0$ is assumed to be provided as a small initial value). We consider each of these parameters in turn.

First consider the parameter π^u. A large value of π^u indicates that the algorithm considers almost any step that provides a sufficiently large reduction in constraint infeasibility to be acceptable. As approaching the feasible region is a necessity for any algorithm for solving problem (2.1), we may choose to initialize π^u to such a large value and increase it only when necessary. In fact, an increase may be necessary if at the current iterate we are unable to compute a step that reduces the constraint infeasibility measure $\|c\|$ without significantly increasing the objective f. Thus, we propose to increase π^u if and only if the computed step indicates that a large increase in the objective will result from a reduction in constraint infeasibility.

In fact, we can establish such an update for π^u by setting its value according to current

methods for updating π for a standard penalty function. In particular, we can emulate [39] and set

$$\pi_k^u \leftarrow \begin{cases} \pi_{k-1}^u & \text{if } \pi_{k-1}^u \geq \chi_k \\ \chi_k + \varepsilon & \text{otherwise,} \end{cases} \tag{6.2}$$

for some small constant $\varepsilon > 0$, where we recall from (2.14) the quantity

$$\chi_k \triangleq \frac{g_k^T d_k + \frac{\omega_k}{2} d_k^T W_k d_k}{(1-\tau)(\|c_k\| - \|c_k + A_k d_k\|)} \tag{6.3}$$

with $0 < \tau < 1$ and ω_k defined by (2.10). It can be seen that in this case π^u will be increased during an iteration if and only if the model m_π (see (2.9)) of the penalty function ϕ_π indicates that an increase in the objective, reflected by a positive numerator in (6.3), will be produced by a step toward the feasible region, implied by the fact that the step satisfies the linearized constraints $c_k + A_k d_k = 0$ in (2.4).

Once the step has been computed and π_k^u has been set, we must perform a backtracking line search to compute a steplength α_k. With $D\phi_\pi(d_k)$ denoted as the directional derivative of ϕ_π along d_k, we require that α_k satisfy the Armijo condition

$$\begin{aligned} \phi_\pi(x_k + \alpha_k d_k) \leq\ & \phi_\pi(x_k) + \eta \alpha_k D\phi_{\dot\pi_k}(d_k) \\ \text{for some } \pi \in\ & [\pi_k^l, \pi_k^u] \end{aligned} \tag{6.4}$$

where $0 < \eta < 1$ and $\dot\pi \in [\pi_k^l, \pi_k^u]$. Observe that the more negative the value of $D\phi_{\dot\pi}(d_k)$, the fewer the number of values of α_k satisfying this condition for a given d_k. Thus, we would like this term to be negative enough to ensure sufficient descent, while also being as small as possible so as to allow the largest number of acceptable steplengths. We consider the issue of choosing $\dot\pi$ by presenting the following lemma.

Lemma 6.1. *The directional derivative of ϕ_π along a step d satisfies*

$$D\phi_\pi(d) = g^T d - \pi\|c\|. \tag{6.5}$$

Moreover, for iteration k, $\pi \geq \chi_k$ implies $D\phi_\pi(d_k) \leq 0$.

Proof. The proof of equality (6.5) can be found in [28]. Then, from (6.3) and the fact that $c_k + A_k d_k = 0$ from (2.6), we have during iteration k

$$\begin{aligned} D\phi_{\chi_k}(d_k) = &\ g_k^T d_k - \chi_k\|c_k\| \\ \leq &\ -\frac{\omega_k}{2} d_k^T W_k d_k - \tau\chi_k\|c_k\|, \end{aligned}$$

and so $D\phi_{\chi_k}(d_k) \leq 0$ follows from (2.10). This, along with the fact that $\pi \geq \chi_k$ yields

$$D\phi_\pi(d) = g^T d - \pi\|c\| \leq g^T d - \chi_k\|c\| = D\phi_{\chi_k}(d),$$

implies that $D\phi_\pi(d) \leq 0$ for all $\pi \geq \chi_k$. □

Thus, the line search will be performed with

$$\begin{aligned} D\phi_{\dot\pi_k}(d_k) = &\ g_k^T d_k - \dot\pi_k\|c_k\|, \\ \text{where } \dot\pi_k \triangleq &\ \max\{\pi_k^l, \chi_k + \varepsilon\}, \end{aligned} \tag{6.6}$$

which along with (6.2) and the fact that $\pi_k^l \leq \pi_{k-1}^u$ ensures $\dot\pi_k \in [\pi_k^l, \pi_k^u]$.

Finally, a good update strategy for π^l is perhaps the most difficult issue. On the one hand, one can simply set π^l equal to π^u during each iteration so that, along with the update strategy (6.2) for π^u, global convergence can be guaranteed. As previously mentioned, however, this can be overly restrictive and block productive steps that move away from the feasible region. At the other extreme, a strategy that does not sufficiently increase π^l may not yield a

convergent method. In particular, a fixed small value for π^l may allow an infinite number of steps to be accepted that do not approach the feasible region of problem (2.1).

To motivate the update strategy for π^l that we propose, consider the numbered regions illustrated in Figure 6.5, where the position and shape of each portion depends on the parameters π_k^l (set during iteration $k-1$) and π_k^u, and the location of the point $p_k = (\|c(x_k)\|, f(x_k))$. A step into region I would not be acceptable to the flexible penalty function (6.1), as opposed to a step into region II, III, or IV, which would be acceptable.

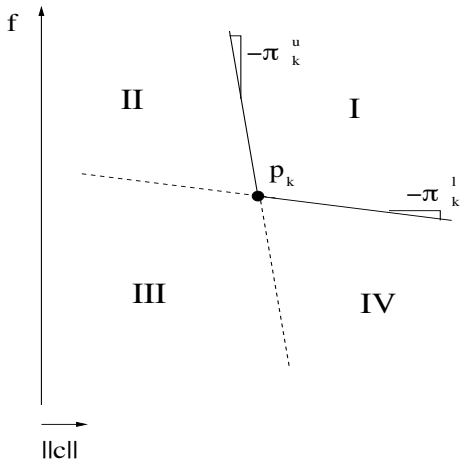

Figure 6.5: Regions defined by the current state of a flexible penalty function

One technique for updating π^l is based on the observation that steps into region IV correspond to an increase in constraint infeasibility. A large number of steps of this type have the potential to drive the search far from the feasible region. Thus, as a certain priority should be placed on the algorithm eventually approaching the feasible region, it may be appropriate to increase π^l after any step into region IV.

Despite the obvious logic behind an approach that increases π^l in this manner, a few objections should be raised. First, an update of this type instantly reduces the area composing region IV. Thus, by increasing π^l precisely at the time when steps into this region are being produced by the step computation procedure, we may be blocking productive steps toward

a desirable region of the search space. In addition, consider the case where π^l is greater than or equal to the threshold value such that stationary points of the penalty function are first order optimal points of the nonlinear program. An increase of π^l in this case may not only slow convergence, but may be completely unnecessary to ensure that the feasible region will eventually be reached.

Thus, we motivate our proposed method as follows. First, recall our observation that for a fixed π^l, an infinite number of steps that sufficiently reduce ϕ_{π^l} (i.e., that lead exclusively into regions III or IV in Figure 6.5) would converge to an optimal solution. Thus, we consider an increase in π^l to be appropriate only after a step into region II has been accepted. This has the logical interpretation that we only become more restrictive by blocking steps that increase infeasibility when the algorithm is confronted with steps that indicate that moves toward the feasible region correspond to significant increases in the objective.

Thus, our update rule for π^l depends on the region in the $\|c\|$-f space to which the step $\alpha_k d_k$ moved upon the conclusion of the line search. If the Armijo condition (6.4) was satisfied for $\pi = \pi^l_k$ (i.e., the step was into region III or IV in Figure 6.5), then we set $\pi^l_{k+1} \leftarrow \pi^l_k$. Otherwise (i.e., if the step was into region II), π^l will be increased. Since we would prefer to block as few future steps as possible, we propose to increase π^l gradually, but sufficiently in order to ensure convergence. We propose an update of the form

$$\pi^l_{k+1} \leftarrow \min\{\pi^u_k, \pi^l_k + \max\{0.1(\pi - \pi^l_k), \varepsilon^l\}\} \tag{6.7}$$

for some $\pi \in [\pi^l_k, \pi^u_k]$ and some small constant $\varepsilon^l > 0$. The implications of rule (6.7) can be seen in Section 6.4. In particular, the update guarantees that $\pi^l = \pi^u$ after a finite number of steps into region II have been taken for a fixed π^u. We propose a method for choosing an appropriate value of π for (6.7) in Section 6.5.

Overall, we have described the following algorithm.

Algorithm 6.2. SQP Method with a Flexible Penalty Function

Initialize x_0, λ_0, $0 \leq \pi_0^l \leq \pi_{-1}^u$, $0 < \varepsilon, \varepsilon^l$, *and* $0 < \eta, \tau < 1$

for $k = 0, 1, 2, \ldots,$ *until a convergence test for problem* (2.1) *is satisfied*

Compute $f_k, g_k, c_k, W_k,$ *and* A_k *and set* $\pi_{k+1}^l \leftarrow \pi_k^l$ *and* $\alpha_k \leftarrow 1$

Compute (d_k, δ_k) *via* (2.6)

Set π_k^u *according to* (6.2) *and* $\dot{\pi}_k$ *by* (6.6)

Perform a backtracking line search to obtain α_k *satisfying* (6.4)

If the Armijo condition (6.4) *does not hold for* $\pi = \pi_k^l$, *set* π_{k+1}^l *by* (6.7)

Set $(x_{k+1}, \lambda_{k+1}) \leftarrow (x_k, \lambda_k) + \alpha_k(d_k, \delta_k)$

endfor

In the backtracking line search, we have in mind that the algorithm first fixes $\alpha_k = 1$ and then tries to find a $\pi \in [\pi_k^l, \pi_k^u]$ satisfying the Armijo condition (6.4). If no such value is found, then α_k is iteratively decreased, where for each value all $\pi \in [\pi_k^l, \pi_k^u]$ are tested, and the procedure terminates once a pair (α_k, π) satisfying (6.4) is encountered. Further details related to a tractable implementation of the line search procedure are given in Section 6.5.

6.4 Global Analysis

In this section we explore the global convergence properties of Algorithm 6.2 where we again assume that the problem formulation and the set of computed iterates satisfy Assumptions 3.4. We show that it is easy to extend the analysis of Section 3.4 to our approach employing a flexible penalty function, where here we argue that the algorithm will eventually compute an infinite sequence of steps that sufficiently reduce the penalty function ϕ_{π^l} for a fixed $\pi^l > 0$, thus driving the first order optimality conditions (2.3) of the nonlinear program (2.1) to zero.

Our analysis is again facilitated by the notion of a decomposition $d_k = u_k + v_k$ (see (3.12)), where the tangential component u_k lies in the null space of the constraint Jacobian A_k and the normal component v_k lies in the range space of A_k^T. Many of the first results provided in Section 3.4 are unrelated to the step acceptance procedure, and thus still apply for Algorithm 6.2 under Assumptions 3.4. In particular, Lemmas 3.5 through 3.8 can be proved in a similar manner, and so we can refer to those results, the sets of indices K_1 and K_2, and the quantity Θ_k defined in (3.18), in the analysis that follows.

The first result we need, related to the step acceptance procedure, bounds the quantity $D\phi_{\dot{\pi}_k}(d_k)$ with $\dot{\pi}$ defined by (6.6).

Lemma 6.3. *There exists $\gamma_6 > 0$ such that, for all k,*

$$D\phi_{\dot{\pi}_k}(d_k) \leq -\gamma_6 \Theta_k.$$

Proof. By (6.6), we have that $\dot{\pi}_k \geq \chi_k$. Thus, combining (6.3) and (6.5) we have that

$$D\phi_{\dot{\pi}_k}(d_k) \leq -\frac{\omega_k}{2} d_k^T W_k d_k - \sigma \dot{\pi}_k \|c_k\|. \tag{6.8}$$

By Lemma 3.7 and (2.10), we have that $\omega_k = 1$ for $k \in K_1$ and thus

$$D\phi_{\dot{\pi}_k}(d_k) \leq -\frac{\mu}{4}\|u_k\|^2 - \sigma \dot{\pi}_k \|c_k\|.$$

Similarly, for $k \in K_2$ we have from (6.8) and (2.10)

$$D\phi_{\dot{\pi}_k}(d_k) \leq -\sigma \dot{\pi}_k \|c_k\|.$$

The result holds for $\gamma_6 = \min\{\frac{\mu}{4}, \sigma\dot{\pi}_k\}$, which is positive as $\dot{\pi}_k \geq \pi_k^l > 0$. □

The next result bounds the sequence of steplength coefficients.

Lemma 6.4. *The sequence $\{\alpha_k\}$ is bounded below by a positive constant.*

Proof. Let us rewrite the Armijo condition (6.4) for convenience as

$$\phi_\pi(x_k + \alpha_k d_k) - \phi_\pi(x_k) \leq \eta \alpha_k D\phi_{\dot{\pi}_k}(d_k) \tag{6.9}$$

for $\pi \in [\pi_k^l, \pi_k^u]$. Suppose that the line search fails for some $\bar{\alpha} > 0$, which means that (6.9) does not hold for any $\pi \in [\pi_k^l, \pi_k^u]$. In particular,

$$\phi_{\dot{\pi}_k}(x_k + \bar{\alpha} d_k) - \phi_{\dot{\pi}_k}(x_k) > \eta \bar{\alpha} D\phi_{\dot{\pi}_k}(d_k),$$

where we recall that $\dot{\pi} \in [\pi_k^l, \pi_k^u]$. As seen in [28], it can be shown under Assumptions 3.4 that for some $\gamma_7 > 0$ we have

$$\phi_{\dot{\pi}_k}(x_k + \bar{\alpha} d_k) - \phi_{\dot{\pi}_k}(x_k) \leq \bar{\alpha} D\phi_{\dot{\pi}_k}(d_k) + \bar{\alpha}^2 \gamma_7 \|d_k\|^2,$$

so

$$(\eta - 1) D\phi_{\dot{\pi}_k}(d_k) \leq \bar{\alpha} \gamma_7 \|d_k\|^2.$$

Lemmas 3.8 and 6.3 then yield

$$(1 - \eta)\gamma_4 \Theta_k < \bar{\alpha} \gamma_3 \gamma_7 \Theta_k,$$

so

$$\bar{\alpha} > (1 - \eta)\gamma_4 / (\gamma_3 \gamma_7).$$

Thus, α_k need never be set below $(1 - \eta)\gamma_4/(\gamma_3\gamma_7)$ for the Armijo condition (6.4) to be satisfied for some $\pi \in [\pi_k^l, \pi_k^u]$. □

Another important property of Algorithm 6.2 is that under Assumptions 3.4 the sequence

$\{\pi_k^u\}$ remains bounded. In fact, this result can be proved in a manner similar to that of Lemma 3.11 as in Algorithm 6.2 the parameter π^u is updated in a manner similar to that of π in Chapter 3. Thus, π_k^u remains bounded, which can be used to prove the following similar result for $\{\pi_k^l\}$.

Lemma 6.5. *The sequence $\{\pi_k^l\}$ is bounded above and π_k^l remains constant for all sufficiently large k.*

Proof. First note that π_k^u remains fixed for all sufficiently large k. By (6.7) we have that if π^l is increased, then it is done so by at least a finite constant amount, or it is set equal to the current value of π^u. Thus, the result follows from (6.7) and the fact that there can only be a finite number of increases of π^l. □

We can now present the following result related to the lengths of the primal components of the steps computed in Algorithm 6.2 and the convergence of the iterates toward the feasible region of problem (2.1).

Lemma 6.6. *Algorithm 6.2 yields*

$$\lim_{k\to\infty} \|c_k\| = 0 \quad \textit{and} \quad \lim_{k\to\infty} \|d_k\| = 0.$$

Proof. By Lemma 6.5 the algorithm eventually computes, during a certain iteration $k^* \geq 0$, a finite value π^* beyond which the value of the parameter π^l will never be increased. This means that for all sufficiently large k, the Armijo condition (6.4) is satisfied for $\pi^l = \pi^*$ (or else π^l would be increased). From Lemmas 6.3 and 6.4, we then have that for all $k \geq k^*$

$$\phi_{\pi^*}(x_k) - \phi_{\pi^*}(x_k + \alpha_k d_k) \geq \gamma_8 \Theta_k$$

for some $\gamma_8 > 0$. Therefore, (3.19) implies

$$\begin{aligned}\phi_{\pi^*}(x_{k^*}) - \phi_{\pi^*}(x_k) &= \sum_{j=k^*}^{k-1} (\phi_{\pi^*}(x_j) - \phi_{\pi^*}(x_{j+1})) \\ &\geq \gamma_8 \sum_{j=k^*}^{k-1} \Theta_j \\ &\geq \frac{\gamma_8}{2\gamma_5} \sum_{j=k^*}^{k-1} (\|d_j\|^2 + \|c_j\|).\end{aligned}$$

The result follows from the above and the fact that Assumption 3.4(a) implies ϕ_{π^*} is bounded below. □

Finally, with this result we can again apply the analysis in Theorem 3.13 to show that for Algorithm 6.2 we have

$$\lim_{k\to\infty} \left\| \begin{bmatrix} g_k + A_k^T \lambda_k \\ c_k \end{bmatrix} \right\| = 0.$$

Thus, Algorithm 6.2 is globally convergent to first order optimal solutions from remote starting points.

6.5 Numerical Results

In this section we present numerical results for a particular implementation of Algorithm 6.2 based on the `KNITRO-Direct` algorithm from the `KNITRO 5.0` software package [40]. We tested the code using a set of 85 equality constrained problems from the CUTEr [4, 19] and COPS [10] collections. The default `KNITRO-Direct` algorithm may revert to a trust region iteration to handle negative curvature and to ensure global convergence. In our tests, we enabled internal options — including one that modifies W_k if necessary to ensure that the resulting matrix is positive definite on the null space of A_k — to ensure that our

implementation performs as a pure line search algorithm.

Let us make a few remarks related to the implementation of our step acceptance strategy before illustrating the numerical results in detail. First, we claim that during iteration k the Armijo condition (6.4) is satisfied for $\pi \in [\pi_k^l, \pi_k^u]$ if and only if it is satisfied for either $\pi = \pi_k^l$ or $\pi = \pi_k^u$. Thus, the line search for a given step d_k can be performed simply by evaluating the reductions attained in $\phi_{\pi_k^l}$ and $\phi_{\pi_k^u}$. Second, in the update rule (6.7) we propose to use

$$\pi = \frac{f(x_k + \alpha_k d_k) - f(x_k) - \eta \alpha_k D\phi_{\hat{\pi}_k}(d_k)}{\|c(x_k)\| - \|c(x_k + \alpha_k d_k)\|}, \tag{6.10}$$

which can be defined as the smallest value such that the step $\alpha_k d_k$ would have been accepted. Here, it can easily be seen that for a step into region II (see Figure 6.5) this value for π lies in the interval $[\pi_k^l, \pi_k^u]$, as desired.

As the globalization strategy described in this chapter is designed to promote long steps for fast convergence, we propose that the numbers of iterations and function evaluations required to find a solution are appropriate measures for comparison with other methods. We compare the results of the algorithm using the standard penalty function approach in `KNITRO-Direct`, call it `pi_default`, with the results using a flexible penalty function. For `pi_default` and the algorithm with a flexible penalty function, we initialize π and π^l to 10^{-8}, respectively. We consider the four initial values 1, 10, 100, and 1000 for π^u, which correspond to the algorithms we refer to as `pi_flex_1`, `pi_flex_10`, `pi_flex_100`, and `pi_flex_1000`, respectively. Table 6.1 contains a complete listing of the input parameters for our implementation of Algorithm 6.2; further details of the implementation can be found in [40].

The results for the five algorithms are summarized in Figures 6.6 and 6.7 in terms of logarithmic performance profiles, as described in [9]. Here, the leftmost values indicate the percentage of times each algorithm solves a given problem using the least value of the given

Parameter	Value
π_0^l	10^{-8}
π_{-1}^u	$\{1, 10, 100, 1000\}$
ε	10^{-4}
ε^l	10^{-4}
η	10^{-8}
τ	10^{-1}

Table 6.1: Input values for Algorithm 6.2

measure, i.e., number of iterations or function evaluations. The values fail to add to one as ties are present. The rightmost function values illustrate the robustness of each approach; i.e., the percentage of times that a given problem is solved.

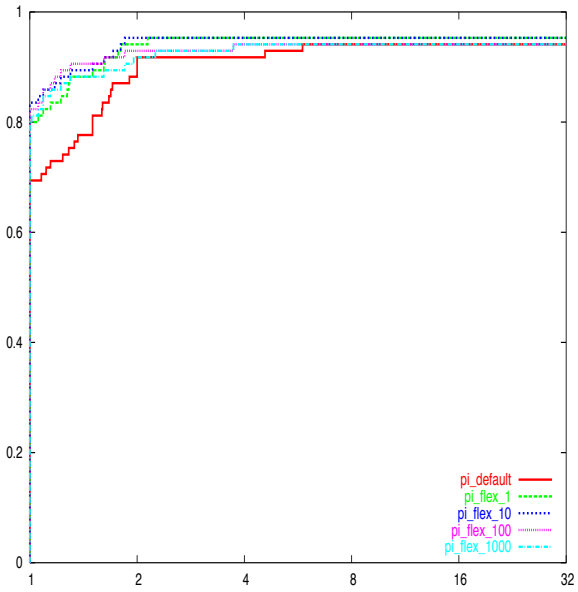

Figure 6.6: Performance profile for iterations in Algorithm 6.2

The results are encouraging. Not only does an algorithm with a flexible penalty function approach often require fewer iterations to find a solution, but a considerable amount of savings is often experienced in terms of function evaluations. This can be understood as the line search procedure generally has to perform fewer backtracks for a given step, leading to longer

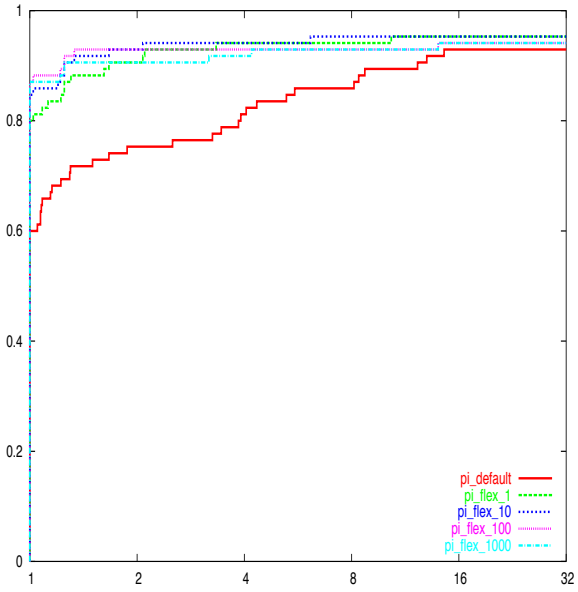

Figure 6.7: Performance profile for function evaluations in Algorithm 6.2

steps and a higher percentage of unit steplengths (i.e., full Newton steps). Detailed results for algorithms `pi_default`, `pi_flex_10`, and `pi_flex_100` can be found in Appendix C.

Chapter 7

Conclusion

In this dissertation, we have presented a robust and efficient inexact method for the solution of large-scale nonlinear optimization problems, as well as a new globalization scheme intended to allow uninhibited and fast convergence. The global behavior of each approach under common conditions has been studied and numerical results have been provided to demonstrate the practical performance of the techniques on a wide range of test problems and realistic applications.

A major theme in this work has been the importance of tying various aspects of the optimization process into one cohesive unit. For example, in the case of our inexact SQP method we found that a certain model of a penalty function, which is normally observed solely within the step acceptance mechanism, could be incorporated into the step computation procedure as a meaningful and effective tool for controlling inexactness. Likewise, with our new globalization strategy we found that a mathematical device could be defined that allows for relatively unrestricted movement during early iterations, but can also automatically tighten itself to forcefully guide convergence when necessary, thus manipulating the search appropriately throughout a run of the algorithm.

We close this dissertation with a few remarks related to the extension of our methods to

general nonlinear programming problems and other algorithmic frameworks. We claim that many of the ideas and much of the analysis described in earlier chapters can be applied, and predict that the strong theoretical properties and impressive practical performance illustrated by our work can be carried over to other settings.

A general nonlinear programming problem has the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) \\ \text{s.t. } & c^{\mathcal{E}}(x) = 0, \\ & c^{\mathcal{I}}(x) \leq 0, \end{aligned} \tag{7.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $c^{\mathcal{E}} : \mathbb{R}^n \to \mathbb{R}^{t^{\mathcal{E}}}$, and $c^{\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{t^{\mathcal{I}}}$ are smooth nonlinear functions. In comparison to the equality constrained problem (2.1), the presence of inequality constraints in (7.1) introduces a combinatorial aspect to the problem as at an optimal point any of the bounds in $c^{\mathcal{I}}(x) \leq 0$ may or may not be tight. Still, algorithms for solving (7.1) have been studied extensively and the methods described in this dissertation may prove useful for problems of this type.

For example, an interior-point method applied to problem (7.1) introduces a log-barrier term with parameter $\mu > 0$ for the inequalities into the objective to form the perturbed subproblem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \; & f(x) - \mu \sum_{i \in \mathcal{I}} \ln s^i \\ \text{s.t. } & c^{\mathcal{E}}(x) = 0, \\ & c^{\mathcal{I}}(x) + s = 0. \end{aligned} \tag{7.2}$$

A solution for problem (7.1) is then found via the (approximate) solution of a sequence of problems of the form (7.2) for $\mu \to 0$, where throughout the process the vector of slack variables $s = (s^1, \ldots, s^{t^{\mathcal{I}}}) \in \mathbb{R}^{t^{\mathcal{I}}}$ is implicitly assumed to be positive. As in Chapter 2, we

may define for a given μ the Lagrangian function

$$\mathcal{L}(x, s, \lambda) \triangleq f(x) - \mu \sum_{i \in \mathcal{I}} \ln s^i + \lambda^{\mathcal{E}^T} c^{\mathcal{E}}(x) + \lambda^{\mathcal{I}^T} (c^{\mathcal{I}}(x) + s) \tag{7.3}$$

with Lagrange multipliers $\lambda = (\lambda^{\mathcal{E}}, \lambda^{\mathcal{I}}) \in \mathbb{R}^{t^{\mathcal{E}} + t^{\mathcal{I}}}$, and derive the first-order optimality conditions

$$\nabla \mathcal{L}(x, s, \lambda) = \begin{bmatrix} g(x) + A^{\mathcal{E}}(x)^T \lambda^{\mathcal{E}} + A^{\mathcal{I}}(x)^T \lambda^{\mathcal{I}} \\ -\mu S^{-1} e + \lambda^{\mathcal{I}} \\ c^{\mathcal{E}}(x) \\ c^{\mathcal{I}}(x) + s \end{bmatrix} = 0. \tag{7.4}$$

Here, $A^{\mathcal{E}}(x)$ and $A^{\mathcal{I}}(x)$ represent the Jacobians of the equality and inequality constraints, respectively, with respect to x, S is a diagonal matrix with elements corresponding to the entries in s, and $e \in \mathbb{R}^{t^{\mathcal{I}}}$ is the vector of ones. Thus, the primal-dual system arising in a line search SQP approach for problem (7.2) can be written as

$$\begin{bmatrix} W_k & 0 & A_k^{\mathcal{E}^T} & A_k^{\mathcal{I}^T} \\ 0 & \mu S_k^{-2} & 0 & I \\ A_k^{\mathcal{E}} & 0 & 0 & 0 \\ A_k^{\mathcal{I}} & I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_k^x \\ d_k^s \\ \delta_k^{\mathcal{E}} \\ \delta_k^{\mathcal{I}} \end{bmatrix} = - \begin{bmatrix} g_k + A_k^{\mathcal{E}^T} \lambda_k^{\mathcal{E}} + A_k^{\mathcal{I}^T} \lambda_k^{\mathcal{I}} \\ -\mu S_k^{-1} e + \lambda_k^{\mathcal{I}} \\ c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} + s_k \end{bmatrix} \tag{7.5}$$

with $W_k \approx \nabla^2_{xx} \mathcal{L}_k$.

We claim that inexactness can be introduced into the step computation procedure in this setting as in that of Chapter 3. In particular, we can again define a penalty function of the form

$$\varphi_\pi(x, s) \triangleq f(x) - \mu \sum_{i \in \mathcal{I}} \ln s^i + \pi \left\| \begin{bmatrix} c^{\mathcal{E}}(x) \\ c^{\mathcal{I}}(x) + s \end{bmatrix} \right\| \tag{7.6}$$

with penalty parameter $\pi > 0$ as our globalization mechanism. A model for φ_π about the iterate (x_k, s_k) is given by

$$m_\pi(d^x, d^s) \triangleq f_k + g_k^T d^x + \tfrac{\omega(d^x)}{2} d^{xT} W_k d^x + \pi \left\| \begin{bmatrix} c_k^{\mathcal{E}} \\ c_k^{\mathcal{I}} + s_k \end{bmatrix} + \begin{bmatrix} A_k^{\mathcal{E}} & 0 \\ A_k^{\mathcal{I}} & I \end{bmatrix} \begin{bmatrix} d^x \\ d^s \end{bmatrix} \right\| - \mu \left(\sum_{i \in \mathcal{I}} \ln s_k^i + e^T S_k^{-1} d^s - \tfrac{1}{2} d^{sT} S_k^{-2} d^s \right) \tag{7.7}$$

with $\omega(d)$ defined in (2.10), with which we can estimate the reduction in φ_π given by an inexact solution (d_k^x, d_k^s) to (7.5) by evaluating $mred_\pi(d_k^x, d_k^s) = m_\pi(0,0) - m_\pi(d_k^x, d_k^s)$. Acceptance conditions for a given inexact solution, similar to Termination Tests I and II provided in Chapter 2, can then be defined in order to ensure that this model reduction is sufficiently large for an appropriate choice of π. The main challenge in the development of such an approach will be careful observations related to the values of the variables in s. As mentioned above, these slack variables must remain positive throughout the run of the algorithm, which must be strictly enforced, for example, via a *fraction to the boundary rule*. If the algorithm is to perform effectively in practice, algorithmic features must be set in place, such as an appropriate scaling of the slack variables or extra conditions imposed in the termination tests, to ensure that the maintanence of positive slacks does not result in series of stagnated steps.

Similarly, the ideas outlined in Chapter 6 can be extended to this context via the definition of the flexible penalty function

$$\begin{aligned} \varphi_\pi(x) \triangleq\ & f(x) - \mu \sum_{i \in \mathcal{I}} \ln s^i + \pi \left\| \begin{bmatrix} c^{\mathcal{E}}(x) \\ c^{\mathcal{I}}(x) + s \end{bmatrix} \right\|, \\ \pi \in\ & [\pi^l, \pi^u], \end{aligned}$$

for a given $\mu > 0$, where $0 \leq \pi^l \leq \pi^u$. An algorithm similar to Algorithm 6.2 can be applied to each barrier subproblem (7.2), where again careful consideration must be made for the updates of the parameters π^l and π^u and for the selection of an appropriate line search condition. A mechanism must also be set for choosing appropriate values for π^l and π^u for each new value of the barrier parameter μ.

Finally, we make the following brief remarks related to the application of a flexible penalty function to other algorithmic frameworks, such as a trust region SQP approach, where, for simplicity, we again consider the equality constrained formulation (2.1). In contrast to satisfying the Armijo condition for some appropriate penalty parameter, in a trust region method a step d_k from x_k is typically accepted if and only if the actual reduction in the penalty function ϕ_π, defined by

$$\phi\text{red}_\pi(d_k) \triangleq \phi_\pi(x_k) - \phi_\pi(x_k + d_k),$$

is large with respect to the reduction obtained in a model such as m_π, defined in Section 2.3. This condition can be written as

$$\frac{\phi\text{red}_\pi(d_k)}{mred_\pi(d_k)} \geq \eta$$

for some $0 < \eta < 1$, where it should be noted that we may necessarily have $\|c_k + A_k d_k\| > 0$ due to the imposed trust region boundary. Instead of restricting the step acceptance criteria to this inequality for a fixed $\pi > 0$ during each iteration k, however, we claim that an effect similar to that expressed in Chapter 6 can be achieved if instead a step is considered acceptable if

$$\frac{\phi\text{red}_{\pi_k^l}(d_k)}{mred_{\dot\pi_k}(d_k)} \geq \eta \quad \text{or} \quad \frac{\phi\text{red}_{\pi_k^u}(d_k)}{mred_{\dot\pi_k}(d_k)} \geq \eta,$$

where $[\pi_k^l, \pi_k^u]$ is a prescribed interval and $\dot\pi_k \in [\pi_k^l, \pi_k^u]$ is chosen carefully so that $mred_{\dot\pi_k}(d_k)$ is sufficiently positive. All of the quantities π_k^l, π_k^u, and $\dot\pi_k$ can be defined and updated in a

manner similar to that described in Chapter 6.

Bibliography

[1] L. T. Biegler and A. Wächter. SQP SAND strategies that link to existing modeling systems. In L. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders, editor, *Large-scale PDE-constrained optimization*, Lecture notes in computational science and engineering, Heidelberg, Berlin, New York, 2003. Springer Verlag.

[2] G. Biros and O. Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: the Krylov-Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713, 2005.

[3] G. Biros and O. Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: the Lagrange-Newton solver, and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 27(2):714–739, 2005.

[4] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.

[5] R. H. Byrd, J.-Ch. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[6] R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[7] R. H. Byrd and J. Nocedal. An analysis of reduced Hessian methods for constrained optimization. *Mathematical Programming*, 63(4):129–156, 1994.

[8] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact-Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[9] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.

[10] E. D. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory, Argonne, Illinois, USA, 2004.

[11] M. El-Hallabi. A hybrid algorithm for nonlinear equality constrained optimization problems: global and local convergence theory. Technical Report TR4-99, Mathematics and Computer Science Department, Institut National des Postes et Télécommunications, Rabat, Morocco, 1999.

[12] M. Fisher, J. Nocedal, Y. Trémolet, and S. J. Wright. Data assimilation in weather forecasting: A case study in PDE-constrained optimization. Technical report, European Centre for Medium-Range Weather Forecasts, 2007.

[13] R. Fletcher and S. Leyffer. User manual for filterSQP. Numerical Analysis Report NA/181, Department of Mathematics, University of Dundee, Dundee, Scotland, 1998.

[14] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–269, 2002.

[15] R. Fletcher, S. Leyffer, and Ph. L. Toint. On the global convergence of an SLP-filter algorithm. Technical Report 98/13, Department of Mathematics, University of Namur, Namur, Belgium, 1998.

[16] R. Fletcher, S. Leyffer, and Ph. L. Toint. On the global convergence of a filter-SQP algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[17] R. W. Freund and N. M. Nachtigal. A new Krylov-subspace method for symmetric indefinite linear systems. In W. F. Ames, editor, *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, pages 1253–1256. IMACS, 1994.

[18] C. C. Gonzaga, E. Karas, and M. Vanti. A globally convergent filter method for nonlinear programming. *SIAM Journal on Optimization*, 14(3):646–669, 2003.

[19] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and sifdec: A Constrained and Unconstrained Testing Environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, 2003.

[20] E. Haber and U. M. Ascher. Preconditioned all-at-once methods for large, sparse parameter estimation problems. *Inverse Problems*, 17:1847–1864, 2001.

[21] E. Haber and L. Hanson. Model problems in PDE-constrained optimization. Technical report, Emory University, 2007.

[22] S. P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977.

[23] M. Heinkenschloss and L. N. Vicente. Analysis of inexact trust-region SQP algorithms. *SIAM Journal on Optimization*, 12:283–302, 2001.

[24] H. Jäger and E. W. Sachs. Global convergence of inexact reduced SQP methods. *Optimization Methods and Software*, 7:83–110, 1997.

[25] C. T. Kelley. `http://www4.ncsu.edu/~ctk/matlab_roots.html`, 2006. Iterative Methods for Linear and Nonlinear Equations: Matlab Codes.

[26] M. Lalee, J. Nocedal, and T. D. Plantenga. On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, 8(3):682–706, 1998.

[27] F. Leibfritz and E. W. Sachs. Inexact SQP interior point methods and large scale optimal control problems. *SIAM Journal on Control and Optimization*, 38(1):272–293, 1999.

[28] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, second edition, 2006.

[29] E. O. Omojokun. *Trust region algorithms for optimization with nonlinear equality and inequality constraints*. PhD thesis, University of Colorado, Boulder, Colorado, USA, 1989.

[30] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis, Dundee 1977*, number 630 in Lecture Notes in Mathematics, pages 144–157, Heidelberg, Berlin, New York, 1978. Springer Verlag.

[31] M. J. D. Powell. Variable metric methods for constrained optimization. In Bachem, A., Grötschel, M., and Korte, B., editors, *Mathematical Programming : The State of the Art, Bonn 1982*. Springer-Verlag, 1983.

[32] E. E. Prudencio, R. Byrd, and X. C. Cai. Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems. *SIAM Journal on Scientific Computing*, 27:1305–1328, 2006.

[33] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[34] R. J. Vanderbei and D. F. Shanno. An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[35] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. Technical Report RC 23033, IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, 2003.

[36] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005.

[37] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[38] A. Walther. A first-order convergence analysis of trust-region methods with inexact Jacobians. Technical Report MATH-WR-01-2005, Institute of Scientific Computing, Technische Universitat Dresden, Dresden, Germany, 2005.

[39] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming, Series A*, 107:391–408, 2006.

[40] R. A. Waltz and J. Nocedal. KNITRO user's manual. Technical Report OTC 2003/05, Optimization Technology Center, Northwestern University, Evanston, IL, USA, April 2003.

[41] D. P. Young, W. P. Huffman, R. G. Melvin, C. L. Hilmes, and F. T. Johnson. Nonlinear elimination in aerodynamic analysis and design optimization. In *Proceedings of the First Sandia Workship on Large-scale PDE Constrained Optimization*, Lecture Notes in Computational Science and Engineering, Heidelberg, Berlin, New York, 2002. Springer Verlag.

Appendix A. Algorithm 4.1

Table A.1: Key for Table A.2

Symbol	Meaning
Name	Name of problem
n	Number of variables
t	Number of constraints
κ	Value of input κ
k	Number of optimization iterations
j	Number of iterative solver iterations
—	Algorithm failure

Table A.2: Results for Algorithm 4.1 on CUTEr and COPS problems

Algorithm			ires										isqp	
κ			2^{-1}		2^{-3}		2^{-5}		2^{-7}		2^{-9}		1	
Name	n	t	k	j	k	j	k	j	k	j	k	j	k	j
aug3dc	3873	1000	20	149	8	143	5	129	4	121	3	107	6	106
bt1	2	1	—	—	—	—	—	—	—	—	—	—	38	83
bt2	3	1	—	—	14	46	12	43	10	36	12	46	24	66
bt3	5	3	8	28	7	45	2	12	1	8	1	8	31	62
bt4	3	2	—	—	7	30	6	30	6	30	6	30	17	61
bt5	3	2	15	49	7	29	6	26	7	35	7	35	9	30
bt6	5	2	—	—	12	67	9	55	8	53	9	61	28	75
bt7	5	3	—	—	—	—	—	—	—	—	18	135	37	145
bt9	4	2	13	47	10	46	9	46	8	42	8	45	19	58
bt10	2	2	—	—	6	21	6	22	6	24	6	24	15	42
bt11	5	3	20	99	—	—	8	55	7	52	7	55	20	80
bt12	5	3	—	—	8	43	7	46	5	35	5	35	27	116
dtoc1nd	735	490	—	—	—	—	25	2558	22	2604	23	3422	77	2275
eigena2	110	55	—	—	—	—	—	—	19	917	16	1071	72	991
eigenaco	110	55	—	—	—	—	—	—	—	—	8	163	79	539
eigencco	30	15	39	502	19	437	16	304	14	331	14	398	116	2502
elec1	150	50	—	—	—	—	—	—	—	—	—	—	142	10214
fccu	19	8	18	100	8	75	5	65	4	61	3	48	54	118
genhs28	10	8	18	114	7	65	5	66	3	41	3	44	42	132
hs006	2	1	—	—	—	—	5	15	5	15	5	15	30	60
hs007	2	1	—	—	10	22	7	18	7	19	7	21	14	24
hs008	2	2	10	20	5	16	4	14	4	16	4	16	10	20
hs027	3	1	—	—	—	—	25	92	22	83	10	37	27	90
hs028	3	1	4	9	1	4	1	4	1	4	1	4	1	4
hs039	4	2	13	47	10	46	9	46	8	42	8	45	19	58
hs040	4	3	16	64	4	23	4	26	4	28	4	28	13	45
hs046	5	2	471	3257	303	2113	16	109	18	126	18	126	316	2095
hs047	5	3	22	119	18	123	16	122	16	128	16	128	57	265
hs048	5	2	19	60	6	24	5	23	1	7	1	7	24	35
hs049	5	2	—	—	17	93	16	92	16	95	16	98	43	138
hs050	5	3	26	107	11	50	9	56	9	65	9	72	47	94
hs051	5	3	18	61	5	18	3	19	2	8	2	8	35	89
hs052	5	3	6	25	2	11	2	13	1	8	1	8	11	30
hs061	3	2	11	25	—	—	5	22	—	—	—	—	12	28
hs077	5	2	17	65	11	62	10	61	9	60	9	63	21	74
hs078	5	3	19	79	7	35	4	28	4	28	4	28	36	84
hs079	5	3	18	83	6	42	4	30	4	31	4	31	36	89
hs100lnp	7	2	—	—	9	53	7	50	6	50	7	61	16	54
hs111lnp	10	3	—	—	—	—	—	—	13	143	15	169	42	301
maratos	2	1	—	—	4	11	5	14	5	14	7	20	6	15
mwright	5	3	—	—	10	59	7	51	7	52	7	54	28	132
orthrega	517	256	—	—	—	—	—	—	—	—	—	—	105	4657
orthregb	27	6	—	—	—	—	—	—	—	—	—	—	31	120
robot	7	2	—	—	—	—	—	—	—	—	—	—	15	80

Table A.3: Key for Table A.4

Symbol	Meaning
n	Number of variables
t	Number of constraints
κ	Value of input κ
ϵ	Value of input ϵ
β'	Value of input β'
k	Number of optimization iterations
j	Number of iterative solver iterations
Mv	Number of matrix-vector products with constraint Jacobian
—	Algorithm failure

Table A.4: Results for Algorithm 4.1 on two PDE-constrained problems

			`Elliptic` ($n = 8192, t = 4096$)			`Parabolic` ($n = 4608, t = 4096$)		
κ	ϵ	β'	k	j	Mv	k	j	Mv
1.0	0.9	100	10	1250	142158	9	9	6785
1.0	0.9	10	**11**	**17**	**3768**	9	9	6785
1.0	0.9	1	8	41	5438	9	9	6785
1.0	0.5	100	9	25	4243	9	9	6785
1.0	0.5	10	**8**	**19**	**3599**	7	10	6479
1.0	0.5	1	7	49	6192	6	14	6956
1.0	0.1	100	9	3704	477477	9	9	6785
1.0	0.1	10	6	38	5178	**6**	**9**	**5635**
1.0	0.1	1	7	1456	197349	5	883	319735
0.5	0.9	100	8	26	4440	6	11	5640
0.5	0.9	10	7	38	5372	6	11	5640
0.5	0.9	1	9	1278	177352	6	14	6956
0.5	0.5	100	8	26	4440	6	11	5640
0.5	0.5	10	7	38	5372	6	11	5640
0.5	0.5	1	8	67	7975	6	14	6956
0.5	0.1	100	9	3704	477477	6	11	5640
0.5	0.1	10	6	38	5178	**5**	**11**	**5305**
0.5	0.1	1	7	1456	197349	5	883	319735
0.1	0.9	100	6	24	4064	—	—	—
0.1	0.9	10	6	1355	107576	—	—	—
0.1	0.9	1	6	188	23837	5	18	8030
0.1	0.5	100	6	24	4064	—	—	—
0.1	0.5	10	6	1355	107576	—	—	—
0.1	0.5	1	6	188	23837	5	18	8030
0.1	0.1	100	6	24	4064	—	—	—
0.1	0.1	10	6	1355	107576	—	—	—
0.1	0.1	1	5	186	23442	5	18	8030

Appendix B. Algorithm 5.3

Table B.5: Key for Table B.6

Symbol	Meaning
Name	Name of problem
n	Number of variables
t	Number of constraints
k	Number of optimization iterations
j	Number of primal-dual matrix factorizations
—	Algorithm failure

Table B.6: Iteration and matrix modification counts for Algorithm 5.3

			matmod		matmod_new	
Name	n	t	k	j	k	j
bt1	2	1	15	40	14	14
bt4	3	2	11	17	10	13
bt7	5	3	13	27	7	15
bt9	4	2	9	12	9	12
catenary	496	166	43	57	43	57
chain1	799	600	6	8	6	8
chain2	1599	1200	9	12	9	12
chain3	3199	2400	8	10	8	10
dtoc1nc	1485	990	9	12	26	43
dtoc1nd	735	490	28	57	35	70
dtoc2	5994	3996	10	15	6	6
eigena2	110	55	—	—	27	57
eigenb2	110	55	—	—	—	—
eigenbco	110	55	41	101	2	2
eigenc2	462	231	6717	6733	20	35
eigencco	30	15	12	22	11	17
elec1	150	50	31	67	55	117
elec2	300	100	73	157	56	118
elec3	600	200	86	195	402	775
gilbert	1000	1	19	22	17	23
hs006	2	1	9	17	9	17
hs007	2	1	8	14	8	14
hs027	3	1	19	29	30	35
hs039	4	2	9	12	9	12
hs047	5	3	17	21	17	21
hs061	3	2	27	48	27	48
hs100lnp	7	2	7	13	7	7
hs111lnp	10	3	14	21	13	22
lch	600	1	38	91	1	1
mwright	5	3	8	14	8	14
orthrds2	203	100	—	—	—	—
orthrega	517	256	124	284	41	67
orthregb	27	6	2	4	2	4
orthregc	10005	5000	13	22	12	17
orthrgds	10003	5000	87	187	—	—
robot	7	2	9	28	9	21

Appendix C. Algorithm 6.2

Table C.7: Key for Tables C.8 and C.9

Symbol	Meaning
Name	Name of problem
n	Number of variables
t	Number of constraints
k	Number of iterations
$f$eval	Number of function evaluations
—	Algorithm failure

Table C.8: Iteration and function evaluation counts for Algorithm 6.2

			pi_default		pi_flex_10		pi_flex_100	
Name	n	t	k	$f$eval	k	$f$eval	k	$f$eval
aug2d	20192	9996	3	5	3	5	2	3
aug3dc	3873	1000	2	3	2	3	2	3
aug3d	3873	1000	2	3	2	3	2	3
bt1	2	1	32	678	12	92	7	15
bt2	3	1	11	14	11	13	12	13
bt3	5	3	2	3	2	3	2	3
bt4	3	2	11	15	12	13	12	13
bt5	3	2	6	7	6	7	6	7
bt6	5	2	9	11	9	11	9	11
bt7	5	3	32	189	16	36	16	36
bt8	5	2	10	11	10	11	10	11
bt9	4	2	13	15	13	14	13	14
bt10	2	2	6	7	6	7	6	7
bt11	5	3	9	15	7	8	7	8
bt12	5	3	4	6	3	4	3	4
byrdsphr	3	2	62	63	62	63	62	63
catena	32	11	6	7	6	7	6	7
catenary	496	166	—	—	—	—	—	—
chain1	799	600	6	7	6	7	6	7
chain2	1599	1200	8	11	7	9	7	9
chain3	3199	2400	13	23	24	28	24	28
channel1	1598	1598	3	4	3	4	3	4
channel2	3198	3198	3	4	3	4	3	4
channel3	6398	6398	3	4	3	4	3	4
dixchlng	10	5	10	13	9	10	9	10
dtoc1l	14985	9990	6	7	6	7	6	7
dtoc1na	1485	990	6	7	6	7	6	7
dtoc1nb	1485	990	5	6	5	6	5	6
dtoc1nc	1485	990	9	12	11	15	11	15
dtoc1nd	735	490	59	174	31	43	31	43
dtoc2	5994	3996	8	43	8	11	8	11
dtoc3	14996	9997	2	3	2	3	2	3
dtoc4	14996	9997	6	52	3	4	3	4
dtoc5	9998	4999	6	49	3	4	3	4
dtoc6	10000	5000	11	13	11	12	11	12
eigena2	110	55	—	—	—	—	—	—
eigenaco	110	55	3	4	3	4	3	4
eigenb2	110	55	2602	49273	—	—	—	—
eigenbco	110	55	41	42	41	42	41	42
eigenc2	462	231	4486	50451	8067	104728	—	—
eigencco	30	15	12	14	12	14	12	14
elec1	150	50	31	66	50	51	50	51
elec2	300	100	73	171	59	68	59	68

Table C.9: Iteration and function evaluation counts for Algorithm 6.2

			pi_default		pi_flex_10		pi_flex_100	
Name	n	t	k	$f$eval	k	$f$eval	k	$f$eval
elec3	600	200	86	231	54	60	54	60
fccu	19	8	2	3	2	3	2	3
genhs28	10	8	2	3	2	3	2	3
gilbert	1000	1	18	21	19	20	19	20
gridnetb	13284	6724	2	3	2	3	2	3
hager1	10000	5000	2	3	2	3	2	3
hager2	10000	5000	2	3	2	3	2	3
hager3	10000	5000	2	3	2	3	2	3
hs006	2	1	5	6	5	6	5	6
hs007	2	1	20	32	26	40	26	40
hs008	2	2	5	6	5	6	5	6
hs009	2	1	4	30	4	30	4	30
hs026	3	1	19	20	19	20	19	20
hs027	3	1	134	916	27	84	27	84
hs028	3	1	2	3	2	3	2	3
hs039	4	2	13	15	13	14	13	14
hs040	4	3	3	4	3	4	3	4
hs046	5	2	13	14	13	14	13	14
hs047	5	3	18	21	18	21	18	21
hs048	5	2	2	3	2	3	2	3
hs049	5	2	16	17	16	17	16	17
hs050	5	3	8	9	8	9	8	9
hs051	5	3	2	3	2	3	2	3
hs052	5	3	2	3	2	3	2	3
hs061	3	2	27	62	16	18	16	18
hs077	5	2	8	10	8	10	8	10
hs078	5	3	4	5	4	5	4	5
hs079	5	3	4	5	4	5	4	5
hs100lnp	7	2	8	9	8	9	8	9
hs111lnp	10	3	14	16	13	14	13	14
lch	600	1	38	39	38	39	38	39
maratos	2	1	3	4	3	4	3	4
mwright	5	3	8	9	8	9	8	9
orthrdm2	4003	2000	8	61	5	7	5	7
orthrds2	203	100	—	—	—	—	—	—
orthrega	517	256	133	436	78	137	78	137
orthregb	27	6	3	13	2	3	2	3
orthregc	10005	5000	15	72	11	13	41	182
orthregd	10003	5000	9	65	6	8	6	8
orthrgdm	10003	5000	10	67	6	8	6	8
orthrgds	10003	5000	752	7469	19	31	19	31
robot	7	2	—	—	7	12	8	10