Chapter 11: Fundamentals of Algorithms for Nonlinear Constrained Optimization

Outline

1. Introduction
2. The Combinatorial Difficulty of Inequality-Constrained Problems
3. Elimination of Variables
4. Merit Functions and Filters

Outline

1. Introduction
2. The Combinatorial Difficulty of Inequality-Constrained Problems
3. Elimination of Variables
4. Merit Functions and Filters

Introduction

We now begin our discussion of algorithms for solving the general constrained optimization problem

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{1.1a}$$

$$s.t. \quad c_i(x) = 0, \quad i \in \mathcal{E}, \tag{1.1b}$$

$$c_i(x) \geq 0, \quad i \in \mathcal{I}, \tag{1.1c}$$

where the objective function f and the constraint functions c_i are all smooth, real-valued functions on a subset of $\mathbb{R}^n$, and $\mathcal{I}$ and $\mathcal{E}$ are finite index sets of inequality and equality constraints, respectively.

Categorizing Constrained Optimization Problems

Optimality conditions which characterize the solutions are useful for motivating algorithms, but to be truly efficient, the algorithm must take account of the particular properties and structure of the objective and constraint functions. There are many important special cases of (1.1) for which specialized algorithms are available. They include the following:

- *Linear programming*: the objective function f and all the constraints c_i are linear functions;
- *Quadratic programming*: the constraints c_i are linear and the objective function f is quadratic;
- *Nonlinear programming*: at least some of the constraints c_i are general nonlinear functions;
- *Linearly constrained optimization*: all the constraints c_i are linear;

Categorizing Constrained Optimization Problems

- *Bound-constrained optimization*: the only constraints in the problem have the form

$$x_i \geq l_i \quad \text{or} \quad x_i \leq u_i,$$

  where l_i and u_i are lower and upper bounds on the ith component of x;

- *Convex programming*: the objective function f is convex, the equality constraints $c_i(x) = 0$, $i \in \mathcal{E}$, are linear, and the inequality constraint functions $c_i(x)$, $i \in \mathcal{I}$, are concave.

These categories are neither mutually exclusive nor exhaustive, and some of the classes can be divided into important subclasses. This finer characterization is relevant to the discussion of algorithms.

Introduction

- The constrained optimization algorithms are iterative in nature.
- They generate a sequence of guesses for the solution x^* that, we hope, tend towards a solution. They may also generate a sequence of guesses for the Lagrange multipliers associated with the constraints.
- In deciding how to move from one iterate to the next, the methods use information about the objective and constraint functions and their derivatives, possibly combined with information gathered at earlier iterations and earlier stages of the algorithm.
- They terminate when they have either identified an approximate solution or when further progress seems to be impossible.

Introduction

In this chapter we discuss some of the fundamental issues in the design of algorithms for nonlinear constrained optimization problems. As in the chapters on unconstrained optimization, we will only study algorithms for finding local minimizers for (1.1); the problem of finding a global minimizer is outside our scope.

Initial Study of a Problem

Before solving a constrained optimization problem by means of one of the algorithms we will describe, it is useful first to study the problem to see whether a simplification is possible. In some cases, it is possible to find a solution without use of a computer. For example, an examination of the constraints may show that the feasible region is empty or that the objective function is not bounded in the feasible region.

Initial Study of a Problem

It may also be possible to solve the KKT conditions

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && && (1.2a)\\
c_i(x^*) &= 0, && \forall i \in \mathcal{E}, && (1.2b)\\
c_i(x^*) &\geq 0, && \forall i \in \mathcal{I}, && (1.2c)\\
\lambda_i^* &\geq 0, && \forall i \in \mathcal{I}, && (1.2d)\\
\lambda_i^* c_i(x^*) &= 0, && \forall i \in \mathcal{E} \cup \mathcal{I}. && (1.2e)
\end{aligned}$$

directly by guessing which of the inequality constraints are active at the solution. Knowledge of the active constraints reduces the KKT conditions to a system of equations that can be solved directly. This approach, however, is rarely practical. Even if the active constraints can be identified-normally the most difficult issue facing a constrained optimization algorithm-we would still need to solve the resulting system of equations numerically.

Initial Study of a Problem

Attention should also be given to the nature of the constraints. Some may be considered "hard" and others "soft" constraints. From the algorithmic point of view, hard constraints are those that must be satisfied in order that the functions and constraints in (1.1) be meaningful. Some of these functions may not even be defined at infeasible points.

- For instance, a variable is constrained to be positive because its square root is required in the calculation of the objective function.
- Another example is a problem in which all the variables must sum to zero to satisfy some conservation law.

Initial Study of a Problem

- Constrained optimization problems with soft constraints are sometimes recast by the modeler as an unconstrained problem in which a penalty term including the constraints is added to the objective function.
- This penalty approach usually introduces ill-conditioning, which may or may not be harmful depending on the algorithm used for the unconstrained optimization.
- The user of optimization algorithms must decide whether it is preferable to use one of the approaches in which the constraints are treated explicitly or whether a penalty approach is adequate.

Initial Study of a Problem

- For problems with hard constraints that must be satisfied at all iterates, we must use feasible algorithms. Usually not all the constraints are hard, and therefore these algorithms choose an initial point that satisfies the hard constraints and produce a new iterate that is also feasible for these constraints.
- Feasible algorithms are usually slower and more expensive than algorithms that allow the iterates to be infeasible, since they cannot follow shortcuts to the solution that cross infeasible territory.
- However, they have the advantage that the objective function f can be used to judge the merit of each point. Since the constraints are always satisfied, there is no need to introduce a more complex merit function that takes account of the constraint violations.

Outline

1 Introduction

2 The Combinatorial Difficulty of Inequality-Constrained Problems

3 Elimination of Variables

4 Merit Functions and Filters

The Combinatorial Difficulty of Inequality-Constrained Problems

One of the main challenges in solving nonlinear programming problems lies in dealing with inequality constraints - in particular, in deciding which of these constraints are active at the solution and which are not.

One approach, which is the essence of active-set methods,

- starts by giving a *working set* $\mathcal{W}$ as a guess of the optimal active set $\mathcal{A}^*$, that is, the set of constraints that are satisfied as equalities at a solution;
- then solve a problem in which the constraints in $\mathcal{W}$ are imposed as equalities and the constraints not in $\mathcal{W}$ are ignored;
- then check to see if there is a choice of Lagrangian multipliers such that the solution x^* as a local solution of (1.1). Otherwise, we make a different choice of $\mathcal{W}$ and repeat the process.

This approach is based on the observation that, in general, it is much simpler to solve equality-constrained problems than to solve nonlinear programs.

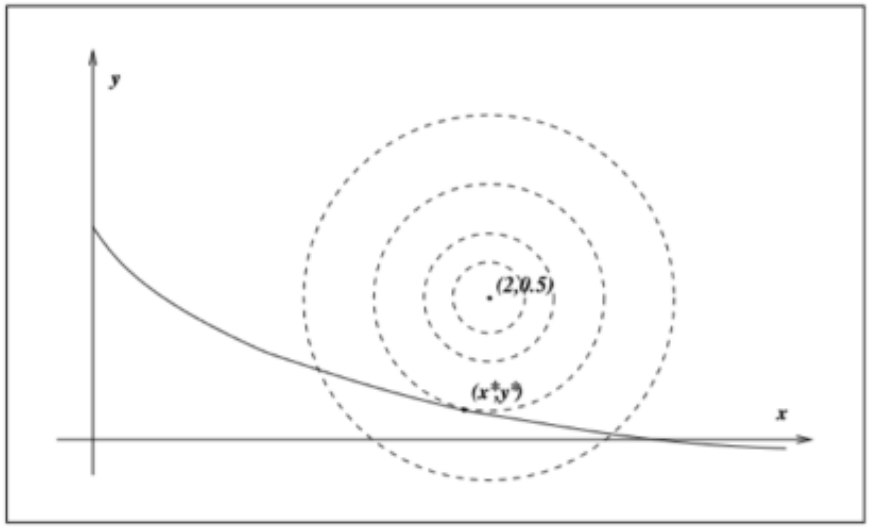
The Combinatorial Difficulty of Inequality-Constrained Problems

- The number of choices for working set $\mathcal{W}$ may be very large - up to $2^{|\mathcal{I}|}$, where $|\mathcal{I}|$ is the number of inequality constraints.
- Since the number of possible working sets grows exponentially with the number of inequalities - a phenomenon which we refer to as the *combinatorial difficulty* of nonlinear programming - we cannot hope to design a practical algorithms by considering all possible choices for $\mathcal{W}$.

The Combinatorial Difficulty of Inequality-Constrained Problems

Consider the problem

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \equiv \tfrac{1}{2}(x-2)^2 + \tfrac{1}{2}(y-\tfrac{1}{2})^2 \\ s.t. \quad & (x+1)^{-1} - y - \tfrac{1}{4} \geq 0, \\ & x \geq 0, \quad y \geq 0. \end{aligned}$$



Illustration

- This example suggests that even for a small number of inequality constraints, determine of the optimal active set is not a simple task.
- However, that some choices of $\mathcal{W}$ can be eliminated from consideration if we make use of knowledge of the functions that define the problem, and their derivatives.
- In fact, the active set methods use this kind of information to make a series of educated guesses for the working set, avoiding choices of $\mathcal{W}$ that obviously will not lead to a solution of (1.1).

Interior-Point Methods

A different approach is followed by interior-point (or barrier) methods. These methods generate iterates that stay away from the boundary of the feasible region defined by the inequality constraints. As the solution of the nonlinear program is approached, the barrier effects are weakened to permit an increasing accurate estimate of the solution. In this manner, interior-point methods avoids the combinatorial difficulty of nonlinear programming.

Outline

1. Introduction
2. The Combinatorial Difficulty of Inequality-Constrained Problems
3. Elimination of Variables
4. Merit Functions and Filters

Simple Elimination using Linear Constraints

When dealing with constrained optimization problems, it is natural to try to use the constraints to eliminate some of the variables from the problem, to obtain a simpler problem with fewer degrees of freedom. Consider the problem

$$\begin{aligned} \min \quad & f(x_1, x_2, x_3, x_4, x_5) \\ s.t. \quad & x_1 + x_3^2 - x_4 x_5 = 0, \\ & -x_2 + x_4 + x_3^2 = 0, \end{aligned}$$

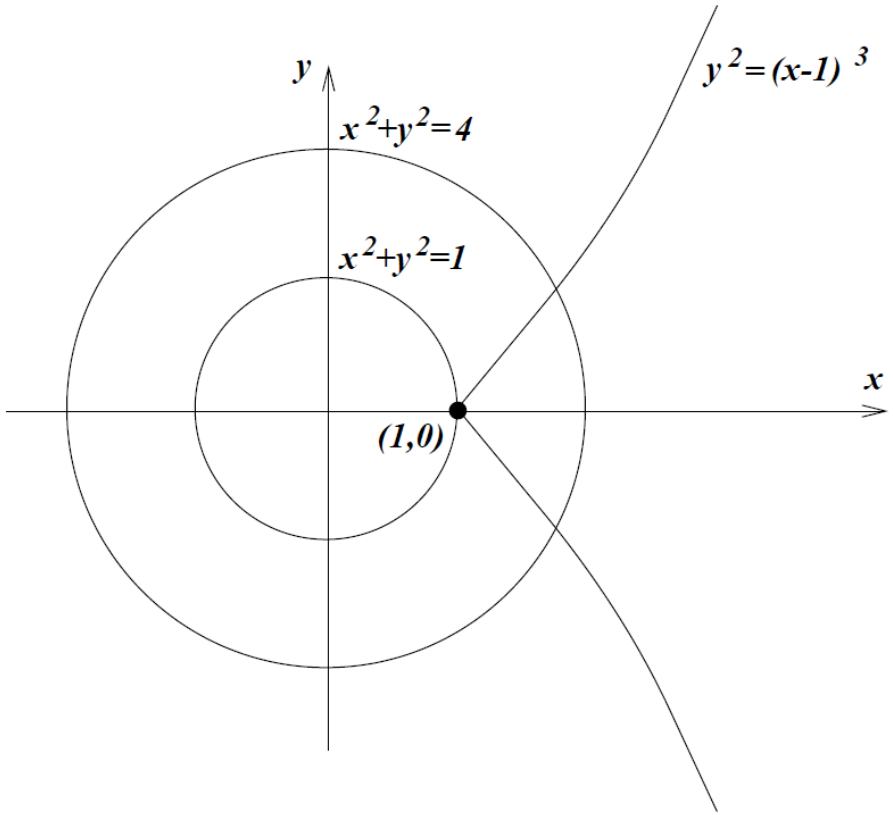
However, elimination techniques must be used with care, as they may alter the problem or introduce ill conditioning.

The Dangers of Nonlinear elimination

Consider the problem

$$\min x^2 + y^2 \text{ s.t. } (x-1)^3 = y^2.$$

The contours of the objective function and the constraints are illustrated in the following figure, which shows that the solution is $(x, y) = (1, 0)$.



The Dangers of Nonlinear elimination

We attempt to solve this problem by eliminating y. By doing so, we obtain

$$h(x) = x^2 + (x-1)^3.$$

Clearly, $h(x) \to -\infty$ as $x \to \infty$. By blindly applying this transformation we may conclude that the problem is in fact unbounded, but this view ignores the fact that the constraint $(x-1)^3 = y^2$ implicitly imposes the bound

$$x \geq 1$$

that is active at the solution. This bound should therefore be explicitly introduced into the problem if we are to perform the elimination of y.

The Dangers of Nonlinear elimination

- This example shows that elimination of nonlinear equations may result in errors that can be difficult to trace. For this reason, nonlinear elimination is not used by most optimization algorithms.
- Instead, many algorithms linearize the constraints and apply elimination techniques to the simplified problem. We will now describe systematic procedures for performing this elimination of linear constraints.

Simple Elimination Using Linear Constraints

Let us consider the minimization of a nonlinear function subject to a set of linear equality constraints,

$$\min f(x) \quad \text{s.t. } Ax = b \tag{3.1}$$

where A is an $m \times n$ matrix with $m \leq n$. Suppose for simplicity that A has full row rank. Under this assumption, we can find a subset of m columns of A that is linearly independent. If we gather these columns into an $m \times m$ matrix B, and define an $n \times n$ permutation matrix P that swaps these columns to the first m column positions in A, we can write

$$AP = [B|N],$$

where N denotes the $n - m$ remaining columns of A. We define the subvectors $x_B \in \mathbb{R}^m$ and $x_N \in \mathbb{R}^{n-m}$ in such a way that

$$\begin{bmatrix} x_B \\ x_N \end{bmatrix} = P^T x,$$

and call x_B the basic variables and B the basis matrix.

Simple Elimination Using Linear Constraints

Noting that $PP^T = I$, we can rewrite the constraint $Ax = b$ as

$$b = Ax = AP(P^T x) = Bx_B + Nx_N.$$

From this formula we deduce that the basic variables are given by

$$x_B = B^{-1}b - B^{-1}Nx_N. \tag{3.2}$$

We can therefore compute a feasible point for the constraints $Ax = b$ by choosing any value of x_N, and then setting x_B according to the formula (3.2). The problem (3.1) is therefore equivalent to the unconstrained problem

$$\min_{x_N} h(x_N) \equiv f\left(P\begin{bmatrix} B^{-1}b - B^{-1}Nx_N \\ x_N \end{bmatrix}\right).$$

We refer to (3.2) as *simple elimination of variables*.

This discussion shows that **a nonlinear optimization problem with linear equality constraints is, from a mathematical point of view, the same as an unconstrained problem.**

Example

Consider the problem

$$\begin{aligned} \min \quad & \sin(x_1 + x_2) + x_3^2 + \tfrac{1}{3}(x_4 + x_5^4 + x_6/2) \\ \text{s.t.} \quad & 8x_1 - 6x_2 + x_3 + 9x_4 + 4x_5 = 6 \\ & 3x_1 + 2x_2 - x_4 + 6x_5 + 4x_6 = -4. \end{aligned}$$

By simple elimination of variables, we have

$$\begin{aligned} \min_{x_1, x_2, x_4, x_5} \quad & \sin(x_1 + x_2) + (8x_1 - 6x_2 + 9x_4 + 4x_5 - 6)^2 \\ & + \tfrac{1}{3}(x_4 + x_5^4 - [(1/2) + (3/8)x_1 + (1/4)x_2 - (1/8)x_4 + (3/4)x_5]). \end{aligned}$$

General Reduction Strategies for Linear Constraints

Suppose A has full row rank. Choose matrices $Y \in \mathbb{R}^{n\times m}$ and $Z \in \mathbb{R}^{n\times(n-m)}$ with the following properties:

$$[Y, Z] \in \mathbb{R}^{n\times n} \text{ is nonsingular, } AZ = 0. \tag{3.3}$$

These properties indicates that the columns of Z are a basis for the null space of A. Since

$$x = Yx_Y + Zx_Z,$$

for some vectors $x_Y \in \mathbb{R}^m$ and $x_Z \in \mathbb{R}^{n-m}$. By substituting it back into the constraints $Ax = b$, we obtain

$$Ax = (A\,Y)x_Y = b;$$

since $A\,Y$ is nonsingular, x_Y can be written explicitly as $x_Y = (A\,Y)^{-1}\,b$. Then we conclude that any vector x of the form

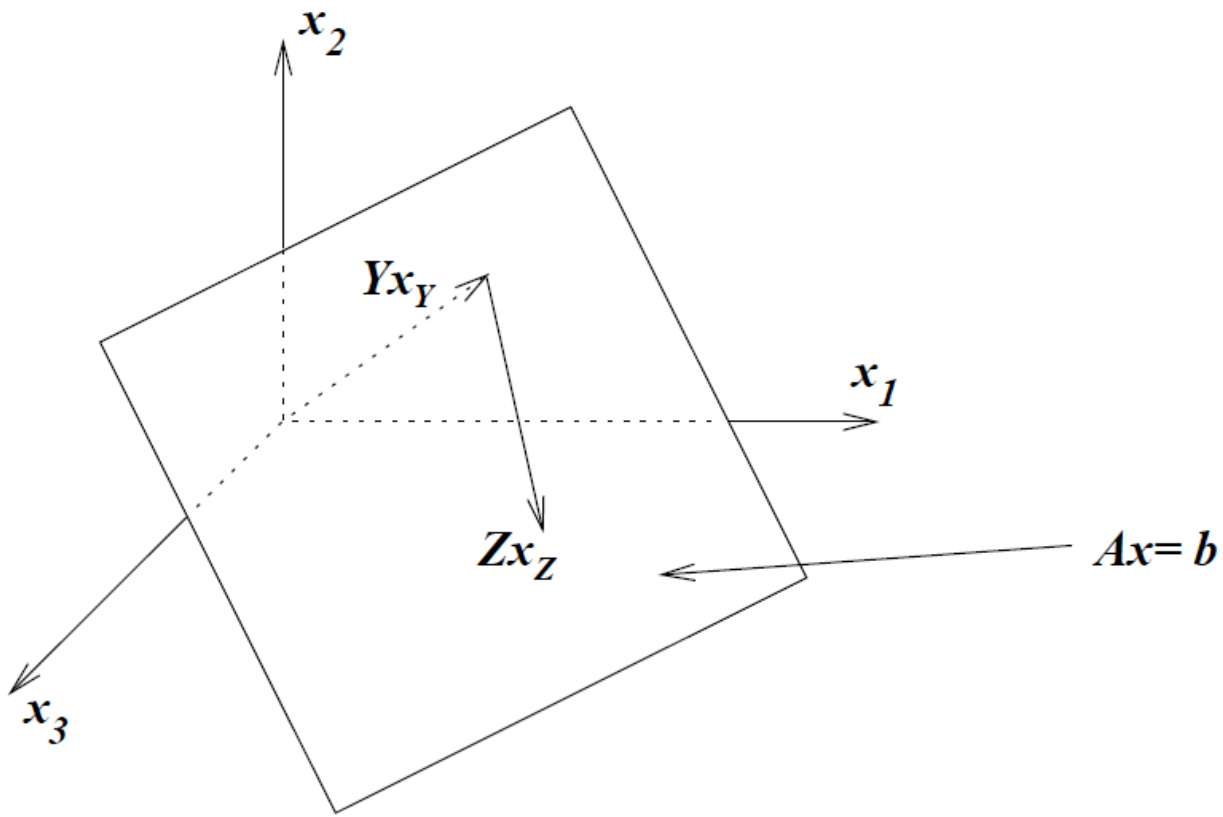
$$x = Y(A\,Y)^{-1}\,b + Zx_Z$$

satisfies the constraints $Ax = b$ for any choice of $x_Z \in \mathbb{R}^{n-m}$. Therefore, (3.1) can be restated equivalently as the following unconstrained problem

$$\min_{x_Z} \quad h(x_Z) \equiv f(\,Y(A\,Y)^{-1}\,b + Zx_Z)$$

.

General elimination



Case in which $A \in \mathbb{R}^{l \times 3}$, showing the particular solution and a step in the null space of A.

Computation of Y and Z

Computing QR factorization of A^T, which has the form

$$A^T\Pi = [Q_1, Q_2]\begin{pmatrix} R \\ 0 \end{pmatrix}$$

where $Q_1 \in \mathbb{R}^{n\times m}$, $Q_2 \in \mathbb{R}^{n\times(n-m)}$ have orthonormal columns, $R \in \mathbb{R}^{m\times m}$ is upper triangular and nonsingular and $\Pi \in \mathbb{R}^{m\times m}$ is a permutation matrix. We define

$$Y = Q_1, \quad Z = Q_2.$$

Then

$$AY = \Pi R^T, \quad AZ = 0.$$

Therefore, any solution of $Ax = b$ can be expressed as

$$x = Q_1 R^{-T}\Pi^T b + Q_2 x_Z.$$

The minimum-norm step

Due to $A^T\Pi = Q_1 R$, we have

$$\Pi^T A A^T \Pi = R^T R.$$

Then $(AA^T)^{-1} = \Pi R^{-1} R^{-T} \Pi^T$. Therefore,

$$A^T(AA^T)^{-1} = A^T \Pi R^{-1} R^{-T} \Pi^T = Q_1 R^{-T} \Pi^T,$$

which indicates that any solution of $Ax = b$ can be expressed as

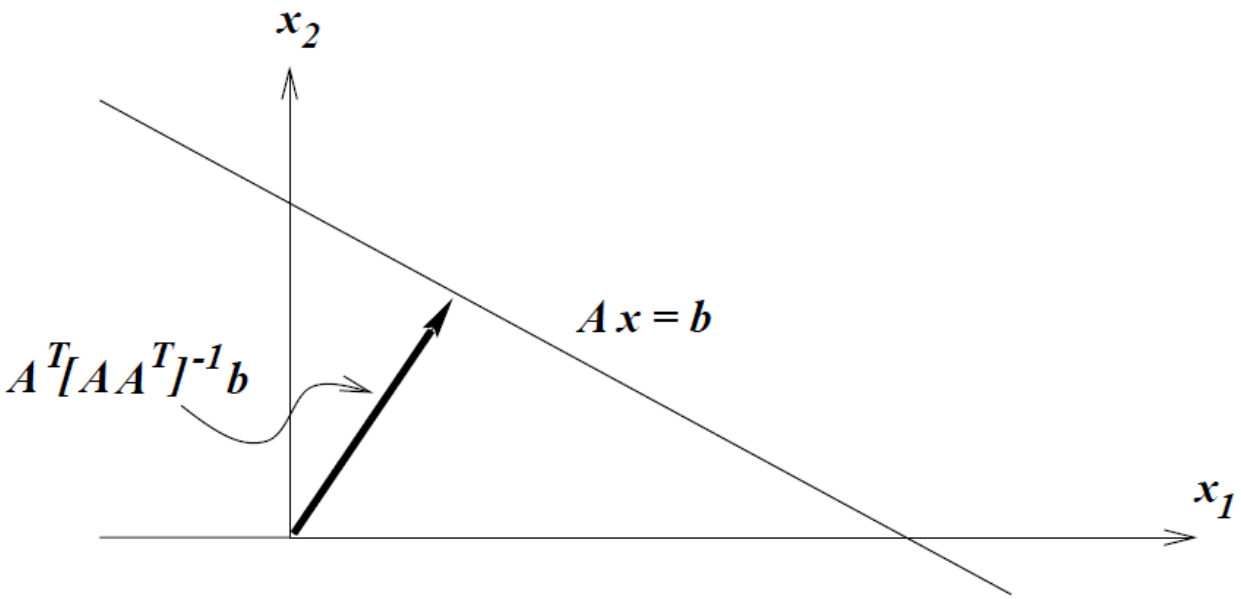
$$x = A^T(AA^T)^{-1} b + Q_2 x_Z.$$

And the particular solution $A^T(AA^T)^{-1} b$ is the solution of the following problem:

$$\min \|x\|^2 \quad \text{s.t. } Ax = b;$$

That is, it is the minimum-norm solution of $Ax = b$.

The minimum-norm step



Effect of Inequality Constraints

Elimination of variables is not always beneficial if inequality constraints are present alongside the equalities. For instance, consider problem

$$\begin{aligned} \min \quad & \sin(x_1 + x_2) + x_3^2 + \tfrac{1}{3}(x_4 + x_5^4 + x_6/2) \\ \text{s.t.} \quad & 8x_1 - 6x_2 + x_3 + 9x_4 + 4x_5 = 6 \\ & 3x_1 + 2x_2 - x_4 + 6x_5 + 4x_6 = -4 \\ & x \geq 0. \end{aligned}$$

Effect of Inequality Constraints

$$
\begin{aligned}
\min_{x_1,x_2,x_4,x_5} \quad & \sin(x_1+x_2)+(8x_1-6x_2+9x_4+4x_5-6)^2 \\
& +\tfrac{1}{3}(x_4+x_5^4-[(1/2)+(3/8)x_1+(1/4)x_2-(1/8)x_4+(3/4)x_5]) \\
s.t. \quad & 8x_1-6x_2+9x_4+4x_5 \le 6 \\
& (3/4)x_1+(1/2)x_2-(1/4)x_4+(3/2)x_5 \le -1 \\
& (x_1,x_2,x_4,x_5) \ge 0.
\end{aligned}
$$

Hence, the cost of eliminating the equality constraints is to make the inequalities more complicated than the simple bounds $x \ge 0$. For many algorithms, this transformation will not yield any benefit.

Outline

1. Introduction
2. The Combinatorial Difficulty of Inequality-Constrained Problems
3. Elimination of Variables
4. Merit Functions and Filters

Introduction

- Suppose that an algorithm for solving the nonlinear programming problem (1.1) generates a step that gives a substantial reduction in the objective function but leads us farther away from the feasible region than the current iterate. Should we accept this step?
- This question is not easy to answer. We must look for a way to balance the twin (of competing) goals of reducing the objective and satisfying the constraints.
- Merit functions and filters are two approaches for achieving this balance. In a typical constrained optimization algorithm, a step p will be accepted only if it is leads to a sufficient reduction in the merit function ϕ or if it is acceptable to the filter.

Merit Functions

- In unconstrained optimization, the objective function f is the natural choice for the merit function. Usually, the unconstrained optimization methods require that f be decreased at each step (or at least after a certain number of iterations).
- In feasible methods, in which the starting point and all subsequent iterates satisfy all the constraints in the problem, the objective function is still an appropriate merit function.
- On the other hand, algorithms that allow iterates to violate at least some constraints require some way to assess the quality of the steps and iterates. The most common way to make this assessment is with a merit function.

Merit Functions

A widely used merit function for the general nonlinear programming problem (1.1) is the ℓ_1 penalty function defined by

$$\phi_1(x;\mu) = f(x) + \mu \sum_{i\in\mathcal{E}} |c_i(x)| + \mu \sum_{i\in\mathcal{I}} |c_i(x)|^- \tag{4.1}$$

where we use the notation $[z]^- = \max\{0, -z\}$. The positive scalar μ is the penalty parameter, which determines the weight that we assign to constraint satisfaction relative to minimization of the objective.

Definition 1

A merit function $\phi(x;\mu)$ is exact if there is a positive scalar μ^* such that for any $\mu > \mu^*$, any local solution of the nonlinear programming problem (1.1) is a local minimizer of $\phi(x;\mu)$.

l_1 merit function

- Under certain assumptions, the l_1 merit function $\phi_1(x;\mu)$ is exact and that the threshold value μ^* is given by

$$\mu^* = \max\{|\lambda_i^*|, i \in \mathcal{E} \cup \mathcal{I}\},$$

  where the λ_i^* denote the Lagrange multipliers associated with an optimal solution x^*.

- Since the optimal Lagrange multipliers are, however, not known in advance, algorithms based on the l_1 merit function contain rules for adjusting the penalty parameter whenever there is reason to believe that it is not large enough (or is excessively large). These rules depend on the choice of optimization algorithm and are discussed in the next chapters.

- Another exact merit function is the l_2 merit function, which for equality-constrained problems takes the form

$$\phi_2(x;\mu) = f(x) + \mu\|c(x)\|_2.$$

Fletcher's Augmented Lagrangian

It is both smooth and exact. For equality-constrained problems, it is given by

$$\phi_{\mathrm{F}}(x;\mu) = f(x) - \lambda(x)^T c(x) + \frac{1}{2}\mu \sum_{i\in\mathcal{E}} c_i(x)^2,$$

where $\mu > 0$ is the penalty parameter and

$$\lambda(x) = [A(x)A(x)^T]^{-1} A(x)\nabla f(x).$$

(Here $A(x)$ denotes the Jacobian of c(x).) Although this merit function has some interesting theoretical properties, it has practical limitations, including the expense of solving for $\lambda(x)$.

Augmented Lagrangian

For equality-constrained problems it has the form

$$\mathcal{L}_A(x, \lambda; \mu) = f(x) - \lambda^T c(x) + \frac{1}{2}\mu \| c(x) \|^2$$

We assess the acceptability of a trial point (x^+, λ^+) by comparing the value of $\mathcal{L}_A(x^+, \lambda^+; \mu)$ with the value at the current iterate, (x, λ).

Sufficient Decrease Condition

- In a line search method, the sufficient decrease condition requires the step length parameter $\alpha > 0$ to be small enough that the inequality

$$\phi(x + \alpha p; \mu) \leq \phi(x; \mu) + \eta \alpha D(\phi(x; \mu); p),$$

  is satisfied for some $\eta \in (0, 1)$.

- In a trust region method, a quadratic model $q(p)$ is used to estimate the value of the merit function ϕ after a step p. The sufficient decrease condition can be stated in terms of a decrease in this model, as follows

$$\phi(x + p; \mu) \leq \phi(x; \mu) - \eta(q(0) - q(p)),$$

  for some $\eta \in (0, 1)$.

Filters

Nonlinear programming has two goals: minimization of the objective function and the satisfaction of the constraints. If we define a measure of infeasibility as

$$h(x) = \sum_{i \in \mathcal{E}} |c_i(x)| + \sum_{i \in \mathcal{I}} |c_i(x)|^-,$$

we can write these two goals as

$$\min_x f(x) \text{ and } \min_x h(x).$$

Unlike merit functions, which combine both problems into a single minimization problem, filter methods keep the two goals separate. Filter methods accept a trial step x^+ as a new iterate if the pair $(f(x^+), h(x^+))$ is not *dominated* by a previous pair $(f_l, h_l) = (f(x_l), h(x_l))$ generated by the algorithm.

Definitions

(a) A pair (f_k, h_k) is said to dominate another pair (f_l, h_l) if both $f_k \leq f_l$ and $h_k \leq h_l$.

(b) A filter is a list of pairs (f_l, h_l) such that no pair dominates any other.

(c) An iterate x_k is said to be acceptable to the filter if (f_k, h_k) is not dominated by any other pair in the filter.

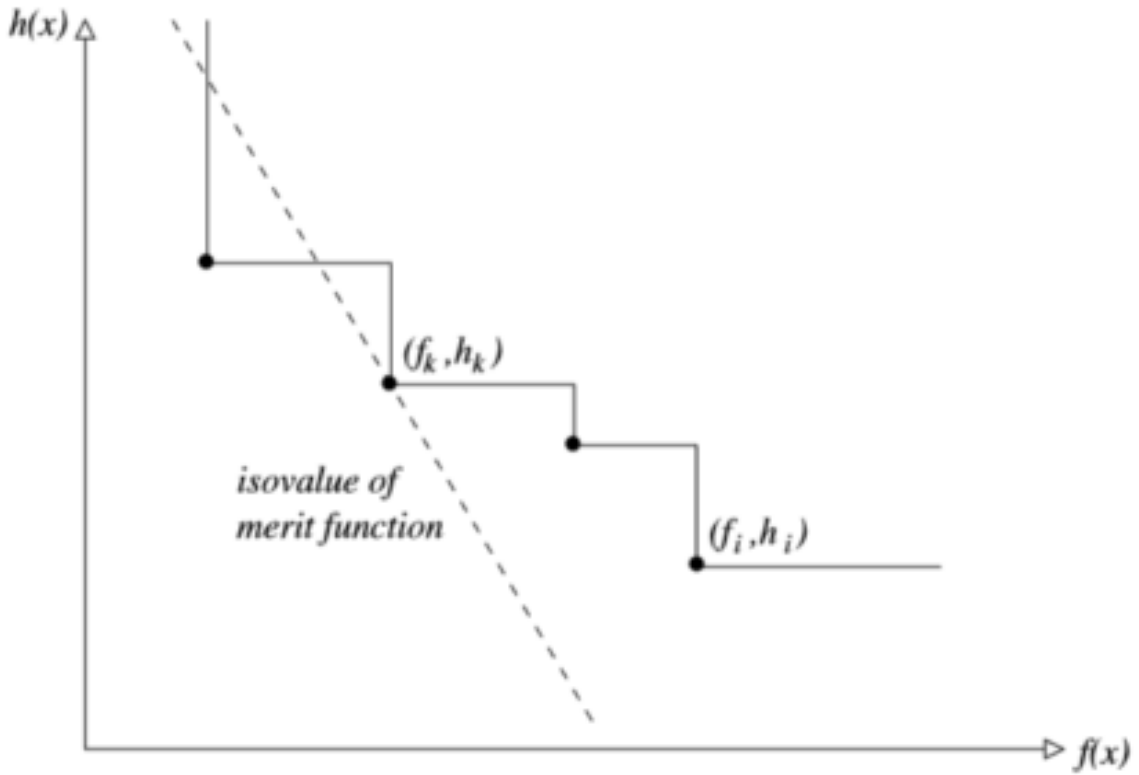
Iterates Update

If a trial step $x^+ = x_k + \alpha_k p_k$ generated by a line search method gives a pair (f^+, h^+) that is acceptable to the filter, we set $x_{k+1} = x^+$; otherwise, a backtracking line search is performed. In a trust-region method, if the step is not acceptable to the filter, the trust region is reduced, and a new step is computed.

General Filter Method

Choose a starting point x_0 and an initial trust-region radius Δ_0;
Set $k \leftarrow 0$;
repeat until a convergence test is satisfied
    if the step-generation subproblem is infeasible
        Compute x_{k+1} using the feasibility restoration phase;
    else
        Compute a trial iterate $x^+ = x_k + p_k$;
        if (f^+, h^+) is acceptable to the filter
            Set $x_{k+1} = x^+$ and add (f_{k+1}, h_{k+1}) to the filter;
            Choose Δ_{k+1} such that $\Delta_{k+1} \geq \Delta_k$;
            Remove all pairs from the filter that are dominated
                by (f_{k+1}, h_{k+1});
        else
            Reject the step, set $x_{k+1} = x_k$;
            Choose $\Delta_{k+1} < \Delta_k$;
        end if
    end if
    $k \leftarrow k + 1$;
end repeat

Comparison between filter and merit function



Thanks for your attention!