NONLINEAR PROGRAMMING

K. Schittkowski, Ch. Zillober, Department of Mathematics, University of Bayreuth, D-95440 Bayreuth, Germany

Keywords: nonlinear programming, optimization, sequential quadratic programming, sequential linear programming, sequential convex programming, penalty method, barrier method, reduced gradient method, interior point method, optimality, Karush-Kuhn-Tucker condition, convergence, large scale optimization

Contents:

Glossary

Barrier function: Function that penalizes objective function when approaching boundary.
BFGS, DFP formula: Special quasi-Newton formulas. **Constraint qualification:** Regularity condition required for characterizing optimality.
Feasible region: Set of points satisfying all constraints.
Karush-Kuhn-Tucker conditions: Optimality conditions for smooth, constrained optimization problems based on the Lagrangian function.
Lagrangian function: Linear combination of objective and constraint functions for deriving

optimality conditions.
Line search: One-dimensional steplength calculation along a search direction.
Nonlinear program (NLP): Minimization of a nonlinear objective function subject to nonlinear equality and inequality constraints.
Penalty function: Function that penalizes violation of constraints.
Quasi-Newton-matrix: Approximation of Hessian matrix or its inverse by rank-2-updates.
Trust region: Limitation of new iterate for stabilization.

Summary

Nonlinear programming is a direct extension of linear programming, when we replace linear model functions by nonlinear ones. Numerical algorithms and computer programs are widely applicable and commercially available in form of *black box* software. However, to understand how optimization methods work, how corresponding programs are organized, how the results are to be interpreted, and, last not least, what are the limitations of the powerful mathematical technology, it is necessary to understand at least the basic terminology. Thus, we present a brief introduction into optimization theory, in particular we introduce optimality criteria for smooth problems. These conditions are extremely important to understand how mathematical algorithms work. SThe most popular classes of constrained nonlinear programming algorithms are introduced, i.e., penalty-barrier, interior point, augmented Lagrangian, sequential quadratic programming, sequential linear programming, generalized reduced gradient, and sequential convex programming methods. Common features and methodological differences are outlined. In particular we discuss extensions of these methods for solving large scale nonlinear programming problems.

1 Introduction

Whenever a mathematical model is available to simulate a *real-life* application, a straightforward idea is to apply mathematical optimization algorithms for minimizing a so-called cost function subject to constraints.

A typical example is the minimization of the weight of a mechanical structure under certain loads and constraints for admissible stresses, displacements, or dynamic responses. Highly complex industrial and academic design problems are solved today by means of nonlinear programming algorithms without any chance to get equally qualified results by traditional empirical approaches.

There exists a large variety of different types of optimization problems. Typically we distinguish between

- linear programming,
- quadratic programming,
- constrained nonlinear programming,
- dynamic programming,
- least squares, min-max, L_1-optimization,
- large scale optimization,
- semidefinite programming,
- non-smooth optimization,
- mixed-integer programming,
- optimal control,
- stochastic optimization,
- global optimization,
- multicriteria optimization,

to mention at least the main classes of problems for which the mathematical background is well understood and for which numerical algorithms are available. In this review, we consider only smooth, i.e. differentiable constrained nonlinear programming problems

$$\begin{aligned} &\min f(x) && x \in R^n \ , \\ &g_j(x) = 0 \ , && j = 1, \ldots, m_e \ , \\ &g_j(x) \geq 0 \ , && j = m_e + 1, \ldots, m \ , \\ &x_l \leq x \leq x_u \ . \end{aligned} \tag{1}$$

Here, x is a n-dimensional parameter vector, also called the vector of design variables, $f(x)$ the objective function or cost function to be minimized under nonlinear equality and inequality constraints given by $g_j(x)$, $j = 1, \ldots, m$. It is assumed that these functions are continuously differentiable in R^n. The above formulation implies that we do not allow any discrete or integer variables, respectively. But besides of this we do not require any further mathematical structure of the model functions. For a discussion of non-smooth optimization problems see: Non differentiable optimization.

To facilitate the subsequent notation, we assume that upper and lower bounds x_u and x_l are not handled separately, i.e. that they are considered as general inequality constraints. Then we get the nonlinear programming problem

$$\begin{aligned} &\min f(x) && x \in R^n \ , \\ &g_j(x) = 0, && j = 1, \ldots, m_e \ , \\ &g_j(x) \geq 0, && j = m_e + 1, \ldots, m \end{aligned} \tag{2}$$

called now NLP in abbreviated form.

Although optimization software can be used in form of a *black box*, it is highly desirable to understand at least the basic ideas of the mathematical analysis behind the problem. One reason is that there are a lot of situations, which could prevent an algorithm from approaching a solution in the correct way. Typically, an optimization algorithm breaks down with an error message and the corresponding documentation contains a lot of technical phrases that must be known to find a remedy. Another reason is that one would like to get an idea how accurate the solution is obtained and whether it is possible to improve or verify an existing approximation.

For these reasons, we present a very brief outline of the optimization theory behind the presented algorithms, on a very elementary level. First we need some notations for the first and second derivatives of a differentiable function. For mathematical basics see: Differential Calculus. The gradient of a real-valued function $f(x)$ is

$$\nabla f(x) := \left(\frac{\partial}{\partial x_1} f(x), \ldots, \frac{\partial}{\partial x_n} f(x) \right)^T \quad .$$

One further differentiation gives the Hessian matrix

$$\nabla^2 f(x) := \left(\frac{\partial^2}{\partial x_i \partial x_j} f(x) \right)_{i,j=1,\ldots,n}$$

of $f(x)$. The Jacobian matrix of a vector-valued function $F(x) = (f_1(x), \ldots, f_l(x))^T$ is

$$\nabla F(x) := \left(\frac{\partial}{\partial x_i} f_j(x) \right)_{i=1,\ldots,n; j=1,\ldots,l} \quad ,$$

also written in the form

$$\nabla F(x) = (\nabla f_1(x), \ldots, \nabla f_l(x)) \quad .$$

The fundamental tool for deriving optimality conditions and optimization algorithms is the so-called *Lagrangian function*

$$L(x, u) := f(x) - \sum_{j=1}^{m} u_j g_j(x) \tag{3}$$

defined for all $x \in R^n$ and $u = (u_1, \ldots, u_m)^T \in R^m$. The purpose of $L(x, u)$ is to link objective function $f(x)$ and constraints $g_j(x)$, $j = 1, \ldots, m$. The variables u_j are called the *Lagrangian multipliers* of the nonlinear programming problem.

Moreover, we denote by P the *feasible region*, i.e., the set of all feasible solutions

$$P := \{x \in R^n : g_j(x) = 0,\ j = 1, \ldots, m_e,\ g_j(x) \geq 0,\ j = m_e + 1, \ldots, m\} \quad .$$

The *active inequality constraints* with respect to $x \in P$ are characterized by the index set

$$I(x) := \{j : g_j(x) = 0,\ m_e < j \leq m\} \quad .$$

We discuss very briefly the main strategies behind a few classes of nonlinear programming algorithms,

- penalty and barrier methods,

- augmented Lagrangian methods,

- interior point methods,

- sequential linear programming methods,

- sequential quadratic programming methods,

- generalized reduced gradient methods,

- sequential convex programming methods.

In particular, we will also discuss extensions of these methods to solve large scale optimization problems.

Each implementation of a method in one of these subclasses, requires additional decisions on a special variant or parameter selections, so that different codes of the same group may possess completely different performance in practice. Moreover, there exist combinations of the fundamental strategies making it even more difficult to classify nonlinear programming algorithms. Comparative studies of codes for the general model have been performed in the past. They proceed either from randomly generated test examples, or are based on artificial or simple application problems reflecting special mathematical properties.

2 Optimality Conditions

2.1 Convexity and Constraint Qualification

In general we can only expect that an optimization algorithm computes a local minimum and not a global one, i.e. a point $x^\star$ with $f(x^\star) \leq f(x)$ for all $x \in P \cap U(x^\star)$, where $U(x^\star)$ is a suitable neighborhood of $x^\star$. However, each local minimum of a nonlinear programming problem is a global one if the problem is convex, for example, if f is convex, g_j linear for $j = 1, \ldots, m_e$ and g_j concave for $j = m_e + 1, \ldots, m$. These conditions force the feasible region P to be a convex set.

Definition 1 *A function $f : R^n \to R$ is called convex, if*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

for all $x, y \in R^n$ and $\lambda \in (0,1)$, and concave, if we replace '$\leq$' by '$\geq$' in the above inequality.

For a twice differentiable function f convexity is equivalent to the the property that $\nabla^2 f(x)$ is positive semidefinite, i.e. $z^T \nabla^2 f(x) z \geq 0$ for all $z \in R^n$. Convexity of an optimization problem is important mainly from the theoretical point of view, since many convergence, duality or other theorems can be proved only for this special case. In practical situations, however, we have hardly a chance to test whether a numerical problem is convex or not.

To be able to formulate the subsequent optimality conditions, we need a special assumption to avoid irregular behavior of the feasible sets P at a local solution. We call it *constraint*

qualification, to be considered as some kind of *regularity* in more general form. In our situation it is sufficient to proceed from the following definition:

Definition 2 *A constraint qualification in $x^\star \in P$ is satisfied, if the gradients of active constraints, i.e. the vectors $\nabla g_j(x^\star)$ for $j \in \{1,\ldots,m_e\} \cup I(x^\star)$, are linearly independent.*

2.2 Karush-Kuhn-Tucker Conditions

For developing and understanding an optimization method, the subsequent theorems are essential. They characterize optimality and are therefore also important for testing a current iterate with respect to its convergence accuracy.

Theorem 1 (necessary second order optimality conditions) *Let f and g_j be twice continuously differentiable for $j = 1,\ldots,m$, $x^\star$ a local minimizer of (2) and the constraint qualification in $x^\star$ be satisfied. Then there exists a $u^\star \in R^m$, such that*

$$
\begin{aligned}
a) \qquad u_j^\star &\geq 0 \quad , \quad j = m_e+1,\ldots,m \quad , \\
g_j(x^\star) &= 0 \quad , \quad j = 1,\ldots,m_e \quad , \\
g_j(x^\star) &\geq 0 \quad , \quad j = m_e+1,\ldots,m \quad , \\
\nabla_x L(x^\star,u^\star) &= 0 \quad , \\
u_j^\star g_j(x^\star) &= 0 \quad , \quad j = m_e+1,\ldots,m
\end{aligned}
\tag{4}
$$

(first order condition),

$$
b) \; s^T \nabla_x^2 L(x^\star,u^\star) s \geq 0 \tag{5}
$$

for all $s \in R^n$ with $\nabla g_j(x^\star)^T s = 0$, $j \in \{1,\ldots,m_e\} \cup I(x^\star)$ (second order condition).

Statement a) of the theorem is called the *Karush-Kuhn-Tucker-condition*. It says that at a local solution the gradient of the objective function can be expressed by a linear combination of gradients of active constraints. Moreover statement b) implies that the Lagrangian function is positive semi-definite on the tangential space defined by the active constraints. For a discussion of general duality-based optimality conditions see: Duality Theory.

It is not possible to omit the constraint qualification, as shown by the subsequent example.

Example 1 *Let*

$$
\begin{aligned}
f(x_1,x_2) &:= x_1 \quad , \\
g_1(x_1,x_2) &:= -x_2 \qquad \geq 0 \quad , \\
g_2(x_1,x_2) &:= x_2 - x_1^2 \quad \geq 0 \quad .
\end{aligned}
$$

Since $P = \{(0,0)\}$, $x^\star = (0,0)$ is the optimal solution. However, we have

$$
\nabla_x L(x^\star,u^\star) = \begin{pmatrix} 1 \\ u_1^\star - u_2^\star \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \end{pmatrix}
$$

showing that the Karush-Kuhn-Tucker-condition cannot be satisfied.

It is also possible to derive a very similar reverse optimality condition, that does not require the constraint qualification.

Theorem 2 (sufficient second order optimality conditions) *Let f and g_j be twice continuously differentiable for $j = 1, \ldots, m$ and $x^\star \in R^n$, $u^\star \in R^m$ be given, so that the following conditions are satisfied:*

a)
$$\begin{aligned} u_j^\star &\geq 0 \quad , \quad j = m_e + 1, \ldots, m \quad , \\ g_j(x^\star) &= 0 \quad , \quad j = 1, \ldots, m_e \quad , \\ g_j(x^\star) &\geq 0 \quad , \quad j = m_e + 1, \ldots, m \quad , \\ \nabla_x L(x^\star, u^\star) &= 0 \quad , \\ u_j^\star g_j(x^\star) &= 0 \quad , \quad j = m_e + 1, \ldots, m \end{aligned}$$

(first order condition),

b) $s^T \nabla_x^2 L(x^\star, u^\star) s > 0$ *for all* $s \in R^n$ *with* $s \neq 0$, $\nabla g_j(x^\star)^T s = 0$, $j = 1, \ldots, m_e$, *and for all* s *with* $\nabla g_j(x^\star)^T s = 0$, $j = m_e + 1, \ldots, m$, *and* $u_j^\star > 0$ *(second order condition).*

Then $x^\star$ is an isolated local minimum of f on P, i.e. there is a neighborhood $U(x^\star)$ of $x^\star$ with $f(x^\star) < f(x)$ for all $x \in U(x^\star) \cap P$, $x \neq x^\star$.

When reading a nonlinear programming textbook, one has to be aware of the fact that the optimality conditions are often stated in a slightly different way. The formulation of a NLP problem varies from author to author depending e.g. whether a minimum or a maximum is searched, whether the inequality constraints use $\leq$ instead of $\geq$, or whether upper and lower bounds are included. Also there exist different versions of the above statements, where the assumptions are either more general or more specialized, respectively.

Now let us consider a few examples.

Example 2 *Assume that $n = 2$, $m_e = 0$, $m = 2$, and that $x^\star$ is an optimal solution with active constraints g_1 and g_2. Then the gradient of the objective function must point into the cone spanned by the gradients $\nabla g_1(x^\star)$ and $\nabla g_2(x^\star)$. In other words, there must exist two multipliers $u_1^\star \geq 0$ and $u_2^\star \geq 0$ with*

$$\nabla f(x^\star) = u_1^\star \nabla g_1(x^\star) + u_2^\star \nabla g_2(x^\star).$$

Example 3 *Consider the simple NLP*

$$\begin{aligned} f(x) &:= x_1^2 + x_2 \quad , \\ g_1(x) &:= 9 - x_1^2 - x_2^2 \quad \geq 0 \quad , \\ g_2(x) &:= 1 - x_1 - x_2 \quad \geq 0 \quad . \end{aligned}$$

We observe immediately that $x^\star = (0, -3)^T$ *is the unique optimal solution of the convex optimization problem. From the Karush-Kuhn-Tucker condition*

$$\nabla_x L(x,u) = \begin{pmatrix} 2x_1 \\ 1 \end{pmatrix} - u_1 \begin{pmatrix} -2x_1 \\ -2x_2 \end{pmatrix} - u_2 \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2x_1(1+u_1) + u_2 \\ 1 + 2u_1 x_2 + u_2 \end{pmatrix} = 0$$

we get the multipliers $u_1^\star = 1/6$ *and* $u_2^\star = 0$*. Moreover, the Hessian matrix of the Lagrangian function*

$$\nabla_x^2 L(x^\star, u^\star) = \begin{pmatrix} 7/3 & 0 \\ 0 & 1/3 \end{pmatrix}$$

is positive definite.

3 Optimization Algorithms

3.1 Quasi-Newton Methods for Unconstrained Optimization

Historically all methods for constrained nonlinear programming are originated either from linear programming or unconstrained optimization. Since linear programming techniques are supposed to be well known (see: Linear Programming), we present a brief review on the most powerful technique for unconstrained optimization based on quasi-Newton updates, since the resulting tools are used by many of the more sophisticated algorithms for the constrained case.

We consider the unconstrained NLP

$$\begin{array}{l} \min \quad f(x) \\ x \in R^n \end{array} \tag{6}$$

with a differentiable function $f(x)$ defined on R^n. Starting from a given value $x_0 \in R^n$, we construct a sequence

$$x_{k+1} := x_k + \alpha_k d_k \tag{7}$$

of iterates for $k = 0, 1, 2, \ldots$, where α_k is a suitable steplength parameter discussed later.

In order to enforce convergence, we need a suitable search direction $d_k \in R^n$. The first idea could be, to apply the method of steepest descent

$$d_k := -\nabla f(x_k) \quad .$$

However, numerical experiments and also theoretical analysis show, that we have to expect a huge number of iterations even if $f(x)$ is a *simple*, for example a quadratic function (see: Numerical Analysis.

The next idea could be to apply Newton's method

$$d_k := -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \ ,$$

for which an excellent quadratic convergence speed can be expected. This approach, however, has the severe disadvantage that second derivatives must be computed. But we have to be aware of the fact, that in very many of the more serious practical application problems we have in mind, even the exact calculation of first derivatives is extremely expensive, inaccurate, or only possible in approximated form. Thus, we have to look for a certain compromise in form of an iterative algorithm

$$d_k := -H_k \nabla f(x_k) \ , \tag{8}$$

where $H_k \in R^{n \times n}$. We would like to find a formula using only first derivatives satisfying the following conditions:

1. H_k is positive definite, when starting from a positive definite matrix H_0.
2. H_k satisfies the so-called quasi-Newton condition

   $$H_{k+1}(\nabla f(x_{k+1}) - \nabla f(x_k)) = x_{k+1} - x_k \ . \tag{9}$$

3. H_{k+1} is computed from $H_k,\ x_{k+1},\ x_k,\ \nabla f(x_{k+1})$, and $\nabla f(x_k)$ by a rank-two-update, i.e.,

   $$H_{k+1} = H_k + \beta_k u_k u_k^T + \gamma_k v_k v_k^T \tag{10}$$

   with $\beta_k,\ \gamma_k \in R,\ u_k, v_k \in R^n$.
4. If $f(x) = \frac{1}{2} x^T A x + b^T x$, A positive definite, α_k exact steplength, the search directions $d_0, d_1, \ldots$ are conjugate w.r.t. A, i.e. $d_i^T A d_j = 0,\ 0 \leq i < j$.

Since the Hessian $f(x^\star)$ is positive definite at a strict local minimizer $x^\star$, and since H_k is supposed to approximate its inverse $\nabla^2 f(x^\star)^{-1}$, we require positive definite matrices H_k. A particular advantage is, that we get descent directions d_k because of

$$d_k^T \nabla f(x_k) = -\nabla f(x_k)^T H_k \nabla f(x_k) < 0 \ . \tag{11}$$

To motivate the quasi-Newton condition, consider a quadratic function

$$f(x) := \frac{1}{2} x^T A x - b^T x + c$$

with a positive definite matrix A. Then we get for $H_{k+1} := A^{-1}$

$$\begin{aligned} H_{k+1}(\nabla f(x_{k+1}) - \nabla f(x_k)) &= A^{-1}(A x_{k+1} - b - A x_k + b) \\ &= x_{k+1} - x_k \ . \end{aligned} \tag{12}$$

Thus, the condition is satisfied showing that quasi-Newton matrices approximate the inverse Hessian of f in the above sense. The rank-two update guarantees in addition a numerically cheap procedure, since the complexity is in the order of n^2. From the last condition we get even finite convergence within n iterations in case of minimizing a quadratic function with conjugate directions and exact steplength obtained by minimizing $f(x_k + \alpha d_k)$ over all α.

It is possible to get a whole family of formulas satisfying the above conditions besides of the last one, since the exact steplength parameters are too expensive to be computed. The most popular quasi-Newton-updates are the Davidon-Fletcher-Powell (DFP) formula

$$H_{k+1} = H_k + \frac{p_k p_k^T}{p_k^T q_k} - \frac{H_k q_k q_k^T H_k}{q_k^T H_k q_k} \tag{13}$$

and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula

$$H_{k+1} = H_k + \left(1 + \frac{q_k^T H_k q_k}{p_k^T q_k}\right) \frac{p_k p_k^T}{p_k^T q_k} - \frac{1}{p_k^T q_k} \left(p_k q_k^T H_k + H_k q_k p_k^T\right) \quad , \tag{14}$$

where $q_k := \nabla f(x_{k+1}) - \nabla f(x_k)$ and $p_k := x_{k+1} - x_k$. Under certain regularity assumptions, local superlinear convergence can be shown, i.e.

$$\|x_{k+1} - x^\star\| \leq \gamma_k \|x_k - x^\star\| \tag{15}$$

for a sequence $\{\gamma_k\}$ tending to zero, if we start sufficiently close to an optimal solution of (6). Thus, the convergence speed is faster than the linear one of the steepest descent method, but not as fast as the quadratic one of Newton's method.

There exists a very interesting relationship between the DFP and the BFGS formula. If $B_k = H_k^{-1}$, the inverse BFGS formula looks very similar to the DFP formula,

$$B_{k+1} = B_k + \frac{q_k q_k^T}{p_k^T q_k} - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k} \quad , \tag{16}$$

where $B_k = H_k^{-1}$ for all k. Thus, both methods are also said to be dual in this sense. To improve numerical stability, practical quasi-Newton algorithms often proceed from the inverse BFGS formula or any stabilized variant, where d_k is then computed from

$$B_k d = -\nabla f(x_k) \quad . \tag{17}$$

The next question is how to find a suitable steplength α_k in (7). An iterative approximation of the exact minimizer

$$f(\alpha_k^\star) = \min\{f(x_k + \alpha d_k) : 0 < \alpha < \infty\}$$

is too expensive, so that we accept α_k as soon as a sufficient decrease along the descent direction d_k is obtained. By combining quadratic or cubic interpolation with some kind of steplength reduction, the iterative process is stopped as soon as for example the Goldstein condition

$$0 < -\mu_1 \alpha \nabla f(x_k)^T d_k \leq f(x_k) - f(x_k + \alpha d_k) \leq -\mu_2 \alpha \nabla f(x_k)^T d_k \tag{18}$$

or the Armijo condition

$$f(x_k) - f(x_k + \alpha d_k) \geq -\mu_1 \alpha \nabla f(x_k)^T d_k \tag{19}$$

with $0 < \mu_1 < \mu_2 < 1$ is satisfied. Also many other algorithms have been derived in the past.

3.2 Penalty and Barrier Methods

Penalty and barrier methods belong to the first attempts to solve constrained optimization problems satisfactorily. The basic idea is to construct a sequence of unconstrained optimization problems and to solve them by any standard minimization method, so that the minimizers of the unconstrained problems converge to the solution of the constrained one. To simplify the notation, we suppress equality constraints within this section.

To construct the unconstrained problems, so-called penalty terms are added to the objective function which *penalize* $f(x)$ whenever the feasible region is left. A factor r_k controls the degree of penalizing f. Proceeding from a sequence $\{r_k\}$ with $r_k \to \infty$ for $k = 0, 1, 2, \ldots$, penalty functions can be defined e.g. by

$$\mathcal{L}(x, r) = f(x) + r_k \sum_{j=1}^{m} (\min(0, g_j(x)))^2 \ , \tag{20}$$

$$\mathcal{L}(x, r) = f(x) + r_k \sum_{j=1}^{m} |\min(0, g_j(x))| \ . \tag{21}$$

These penalty functions allow violation of constraints and are called external ones.

Is is also possible to define barriers at the border of the feasible region. Then only strictly feasible iterates are allowed in contrast to penalty functions. Barrier methods construct a sequence of unconstrained problems and a factor r_k controls the weight of the barriers. However, barrier terms cannot be defined for equality constraints. The most popular barrier functions are

$$\mathcal{L}(x, r) = f(x) - \frac{1}{r_k} \sum_{j=1}^{m} \log(g_j(x)) \ , \tag{22}$$

$$\mathcal{L}(x, r) = f(x) + \frac{1}{r_k} \sum_{j=1}^{m} \frac{1}{g_j(x)} \ . \tag{23}$$

The barrier terms tend to infinity if a feasible iterate converges to the border of the feasible region. By increasing the barrier parameter r_k, the weight of the constraints is then decreased step by step. A subsequent stepsize procedure has to ensure the feasibility of the iterates. Moreover, certain shift terms can be added with the goal to attain a border more accurately allowing also slightly violated constraints.

The unconstrained nonlinear programming problems are solved by any standard technique, e.g. a quasi-Newton method combined with a line search. However, the line search must be performed quite accurately due to the steep, narrow valleys created by the penalty or barrier terms, respectively. The main disadvantage is, that the condition number of the Hessian matrix of the penalty or barrier function tends to infinity when the parameter r_k becomes too large.

3.3 Augmented Lagrangian Methods

Multiplier or augmented Lagrangian methods, respectively, try to avoid the disadvantage of penalty and barrier algorithms, that too large penalty parameters lead to ill-conditioned unconstrained subproblems. Thus, the objective function is augmented by a term including information about the Lagrangian function,

$$\mathcal{L}(x,u,r) := f(x) + \frac{1}{2}\sum_{i=1}^{m} r_j (g_j(x) - v_j)_-^2 \tag{24}$$

where $a_- = \min(0,a)$ for $a \in R$, $v \in R^m$, and $r \in R^m$. Multipliers are approximated by $r_j v_j$. A similar augmented Lagrangian function is

$$\mathcal{L}(x,v,r) := f(x) - \sum_{j=1}^{m} \begin{cases} (v_j g_j(x) - \frac{1}{2} r_j g_j(x)^2), & \text{if } g_j(x) \le v_j/r_j \\ \frac{1}{2} v_j^2 / r_j, & \text{otherwise} \end{cases} \tag{25}$$

After solving an unconstrained minimization problem with one of the above objective functions, the multiplier estimates are updated according to certain rules, e.g. by

$$\overline{v_j} := v_j - \min(g_j(x), v_j)$$

in the first case or

$$\overline{v_j} := v_j - \min(r_j g_j(x), v_j)$$

in the second one for $j = 1, \ldots, m$. If there is no sufficient reduction of constraint violation, then the penalty parameter vector r is increased as well, typically by a constant factor.

3.4 Interior Point Methods

Motivated by the successful development of new methods for linear programming (see: Linear Programming), modern variants of barrier methods have been derived, so-called interior point methods. Under mild conditions, the set of unconstrained minima of a barrier function forms a smooth curve $x(\mu)$ for $\mu \in (0;\infty)$ for convex optimization problems, the so-called central path. A search direction d_k at an iterate x_k is computed by linear extrapolation along the tangent of the central path.

To motivate the basic idea, we omit equality constraints for the matter of simplicity, and introduce slack variables for the inequality restrictions, i.e. we proceed from an extended problem

$$\begin{aligned} &\min f(x)\ , && x \in R^n, y \in R^m \\ &g(x) - y = 0\ , \\ &y \ge 0\ , \end{aligned} \tag{26}$$

where $g(x) = (g_1(x), \ldots, g_m(x))^T$. The corresponding Karush-Kuhn-Tucker conditions subject to the Lagrangian

$$L(x, y, u, v) = f(x) - (g(x) - y)^T u - v^T y \tag{27}$$

are

$$\begin{aligned} \nabla f(x) - \nabla g(x) u &= 0 \ , \\ u - v &= 0 \ , \\ g(x) - y &= 0 \ , \\ y &\geq 0 \ , \\ v &\geq 0 \ , \\ v_j y_j &= 0 \ , \ \ j = 1, \ldots, m \ , \end{aligned} \tag{28}$$

where $u = (u_1, \ldots, u_m)^T$ and $v = (v_1, \ldots, v_m)^T$ are the corresponding multiplier vectors. Unfortunately, the application of Newton's method to solve this set of nonlinear equations directly, is prevented by the complementary condition $v_j y_j = 0$, which states that either a slack variable must be zero or the corresponding multiplier value. Thus, we modify (28) somewhat, replace the complementary condition by a weaker one of the form $v_j y_j = \mu$ with a suitable positive regularity parameter μ and multiply the third equation with -1. Assuming strict feasibility of v and y, i.e. $v > 0$ and $y > 0$, we obtain the equations

$$\begin{aligned} \nabla f(x) - \nabla g(x) v &= 0 \ , \\ y - g(x) &= 0 \ , \\ v_j y_j &= \mu \ , \ j = 1, \ldots, m \ . \end{aligned} \tag{29}$$

On the other hand, we obtain exactly the same equations characterizing an optimal solution, if we compute a stationary point of the logarithmic barrier function (22)

$$\mathcal{L}(x, y, v, r) = f(x) - (g(x) - y)^T v - \frac{1}{r} \sum_{j=1}^{m} \log y_j \ , \tag{30}$$

with $r := 1/\mu$ and perform some trivial algebraic transformations. Here we penalize the Lagrangian function and consider (30) as an augmented Lagrangian function, see also the previous section.

Now let x_k and y_k be a current iterate, $y_k = (y_1^k, \ldots, y_m^k) > 0$ the current slack variables, $v_k = (v_1^k, \ldots, v_m^k) > 0$ a multiplier estimate, and $\{\mu_k\}$ a series of given positive parameters tending to zero. Then we apply Newton's method to (29), the third set of equations written as $\mu_k - v_j^k y_j^k = 0$, leading to

$$\begin{pmatrix} \nabla_x^2 L(x_k, v_k) & -\nabla g(x_k) \\ -\nabla g(x_k)^T & -V_k^{-1} Y_k \end{pmatrix} \begin{pmatrix} d \\ p \end{pmatrix} = \begin{pmatrix} a_k \\ b_k \end{pmatrix}. \tag{31}$$

In this case, $L(x_k, v_k)$ denotes again the Lagrangian function of the original problem NLP. V_k and Y_k are diagonal matrices containing the coefficients of v_k and y_k, and the right-hand side

is defined by $a_k := -(\nabla f(x_k) - \nabla g(x_k)v_k)$ and $b_k := -(\mu_k V_k^{-1} e - g(x_k))$, $e \in R^m$ a vector containing the value one in each component. If d_k and p_k denote the solution of the linear system (31), a new iterate is obtained by

$$x_{k+1} = x_k + d_k \ , \quad v_{k+1} = v_k + p_k \ .$$

The corresponding slack variables are computed from $y_{k+1} = -V_k^{-1}(Y_k p_k - \mu_k e)$.

Assuming that $\nabla_x^2 L^{-1}$ exists, (31) can be reduced to

$$(\nabla g(x_k)^T \nabla_x^2 L^{-1}(x_k, v_k) \nabla g(x_k) + V_k^{-1} Y_k)p = -\nabla g(x_k)^T \nabla_x^2 L^{-1}(x_k, v_k) a_k - b_k \tag{32}$$

or to

$$(\nabla g(x_k) Y_k^{-1} V_k \nabla g(x_k)^T + \nabla_x^2 L(x_k, v_k))d = -\nabla g(x_k) Y_k^{-1} V_k b_k + a_k \ , \tag{33}$$

respectively, by eliminating either d or p. System (32) is of dimension $m \times m$, system (33) of dimension $n \times n$. Depending on the structure of (26), we solve either (31), (32), or (33).

The assumption that slack variables y_k and their dual variables v_k must remain strictly positive during the iteration, is enforced by an adopted line search. The main advantage is, that feasibility of the original variable x_k is not required. Some of the coefficients converge to zero, if the corresponding constraints become active. The barrier parameter μ_k is updated after each step, not only after a complete unconstrained minimization cycle as for penalty methods.

3.5 Sequential Linear Programming

In many situations, sequential linear programming methods, respectively, are quite powerful, for instance when trying to exploit special sparsity structures of the Jacobian of the constraints, when solving problems with very many active constraints, and, in particular, when round-off errors prevent the usage of higher order methods, say sequential quadratic programming methods.

The idea is to approximate the nonlinear problem (2) by a linear one to get a new iterate. Thus, the next iterate $x_{k+1} = x_k + d_k$ is formulated with respect to solution d_k of the following linear programming problem

$$\begin{array}{ll} \min \ \nabla f(x_k)^T d & , \ d \in R^n \\ \nabla g_j(x_k)^T d + g_j(x_k) = 0 & , \ j = 1, \ldots, m_e \ , \\ \nabla g_j(x_k)^T d + g_j(x_k) \geq 0 & , \ j = m_e + 1, \ldots, m \ , \\ \| \ d \ \|_\infty \leq \delta_k & . \end{array} \tag{34}$$

The principle advantage is that the above problem can be solved by any standard linear programming software, when we use the maximum norm $\| \ . \ \|_\infty$ to restrict the choice of d. However, we cannot guarantee to obtain a descent direction w.r.t. a suitable merit function, and have to add certain trust regions or move limits. Additional bounds for the computation

of d_k are set to avoid bad estimates particularly in the beginning of the algorithm, when the linearization is too inaccurate. The bound δ_k has to be adapted during the algorithm. One possible way is to consider the exact penalty function (21)

$$p(x,r) := f(x) + \sum_{i=1}^{m_e} r_j |g_j(x)| + \sum_{i=m_e+1}^{m} r_j |\min(0, g_j(x))| \tag{35}$$

defined for each $x \in R^n$ and $r = (r_1, \ldots, r_m)^T$. Moreover, we need its first order Taylor approximation given by

$$p_a(x,d,r) := f(x) + \nabla f(x)^T d + \sum_{i=1}^{m_e} r_j |g_j(x) + \nabla g_j(x)^T d)| + \sum_{i=1}^{m} r_j |\min(0, g_j(x) + \nabla g_j(x)^T d)| \tag{36}$$

Then we consider the quotient of the actual and predicted change at an iterate x_k and a solution d_k of the linear programming subproblem

$$q_k := \frac{p(x_k, r) - p(x_k + d_k, r)}{p(x_k, r) - p_a(x_k, d_k, r)} \tag{37}$$

where the penalty parameters are predetermined and must be sufficiently large, e.g. larger than the expected multiplier values at an optimal solution. The update of δ_k is then performed by

$$\delta_{k+1} := \begin{cases} \delta_k/\sigma & , \text{ if } q_k < \rho_1 \\ \delta_k \sigma & , \text{ if } q_k > \rho_2 \\ \delta_k & , \text{ otherwise} \end{cases} \tag{38}$$

Here $\sigma > 1$ and $0 < \rho_1 < \rho_2 < 1$ are constant numbers.

3.6 Sequential Quadratic Programming

Sequential quadratic programming methods are the standard general purpose algorithms for solving smooth nonlinear optimization problems under the following assumptions:

- The problem is not too big.
- The functions and gradients can be evaluated with sufficiently high precision.
- The problem is smooth and well-scaled.

The key idea is to approximate also second order information to get a fast final convergence speed. Thus, we define a quadratic approximation of the Lagrangian function $L(x,u)$ and an approximation of the Hessian matrix $\nabla_x^2 L(x_k, u_k)$ by a quasi-Newton matrix B_k. Then we get the subproblem

$$\begin{array}{ll} \min \ \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d & , \ d \in R^n : \\ \nabla g_j(x_k)^T d + g_j(x_k) = 0 & , \ j = 1, \ldots, m_e \ , \\ \nabla g_j(x_k)^T d + g_j(x_k) \geq 0 & , \ j = m_e + 1, \ldots, m \ . \end{array} \tag{39}$$

Usually, convergence is ensured by performing a line search, i.e. a steplength computation to accept a new iterate $x_{k+1} := x_k + \alpha_k d_k$ for an $\alpha_k \in (0, 1]$ only if x_{k+1} satisfies a descent property with respect to a solution d_k of (39). In this case, we need also a simultaneous line search with respect to the multiplier approximations called v_k and define $v_{k+1} := v_k + \alpha_k(u_k - v_k)$ where u_k denotes the optimal Lagrangian multiplier of the quadratic programming subproblem (39).

The line search is performed with respect to a merit function

$$\phi_{r_k}(\alpha) := \mathcal{L}(x_k + \alpha d_k, v_k + \alpha(u_k - v_k), r_k)$$

and a Goldstein (18) or Armijo (19) sufficient descent condition, where $\mathcal{L}(x, v, r)$ is a suitable exact penalty or augmented Lagrangian function, e.g. of the type (21) or (25), respectively. In this case also equality constraints must be taken into account accordingly. r_k is a suitable penalty parameter which must be sufficiently big to ensure that d_k and v_k are descent directions for $\phi_{r_k}(\alpha)$.

The update of the matrix B_k can be performed by standard techniques known from unconstrained optimization. In most cases, the BFGS-method is applied, see (16), starting from the identity or any other positive definite matrix B_0, and using

$$\begin{array}{lcl} q_k & := & \nabla_x L(x_{k+1}, u_k) - \nabla_x L(x_k, u_k) \ , \\ p_k & := & x_{k+1} - x_k \ . \end{array} \tag{40}$$

Under some safeguards it is possible to guarantee that all matrices B_k remain positive definite. Among the most attractive features of sequential quadratic programming methods is the superlinear convergence speed in the neighborhood of a solution, see (15).

3.7 Generalized Reduced Gradient Methods

By introducing artificial slack variables, the original nonlinear programming problem is converted into a problem with nonlinear equality constraints and lower bounds for the slack variables only. Thus, we proceed from a slightly more general problem

$$\begin{array}{ll} \min \ \overline{f}(z) & , \ z \in R^{\overline{n}}, \\ \overline{g}(z) = 0 & , \\ z_l \leq z \leq z_u & , \end{array} \tag{41}$$

where $\overline{n} = n + m$. As in linear programming, the variables of z are classified into basic and non-basic ones, now called y and x, respectively.

The basic idea is to satisfy the system of equations $\overline{g}(z) = 0$ for all possible iterates. Let $z = (x, y)$ and $y(x)$ be a solution of this system with respect to given variables x, i.e. $\overline{g}(x, y(x)) = 0$, and let $F(x) := \overline{f}(x, y(x))$ be the so-called reduced objective function. Starting from a feasible iterate and an initial set of basic variables, the algorithm performs a search step with respect to the free variables, e.g. by a conjugate gradient or a quasi-Newton method. If the new iterate violates constraints, then it is projected onto the feasible domain by a Newton-type technique. If necessary, a line search is performed also combined with a restoration phase to

get feasible iterates. However, if the projection on the feasible domain P is not possible, or if a basic variable violates a bound, one has to perform an exchange of basic variables as in linear programming.

For evaluating a search direction in the reduced space, we need the gradient of the reduced objective function $F(x)$ with respect to the non-basic variables x, which is computed from

$$\nabla F(x) = \nabla_x \overline{f}(x, y(x)) - \nabla_x \overline{g}(x, y(x)) \nabla_y \overline{g}(x, y(x))^{-1} \nabla_y \overline{f}(x, y(x)) \quad . \tag{42}$$

Generalized reduced gradient methods can be extended easily to problems with a special structure in the constraints or very large problems. Moreover, they are related to sequential quadratic programming methods and there exist combinations of both approaches. Their particular advantage is that they try to follow the feasible region as closely as possible.

3.8 Sequential Convex Programming

We assume for simplicity that only inequality constraints exist, i.e. that $m_e = 0$. Sequential convex programming methods replace the nonlinear program NLP by a sequence of convex subproblems. Given an iterate x_k, the model functions of (2), i.e. f and g_j, $j = 1, \ldots, m$, are approximated by some functions f_k and g_{jk} at x_k, $j = 1, \ldots, m$. The basic idea is to linearize f and g_j with respect to transformed variables $(U_i - x_i)^{-1}$ and $(x_i - L_i)^{-1}$ depending on the sign of the corresponding first partial derivative. U_i and L_i are reasonable bounds and adopted by the algorithm after each successful step. Also several other transformations have been developed in the past.

The approach is motivated by applications in structural mechanical optimization, where these methods have been invented. Especially certain displacement constraints become *more linear* after the approximation.

The objective function f is replaced by

$$f_k(x) := \alpha_k^0 + \sum_{i \in I_k^+} \frac{\alpha_k^i}{U_i - x_i} - \sum_{i \in I_k^-} \frac{\alpha_k^i}{x_i - L_i} \tag{43}$$

and constraints g_j are approximated by

$$g_{jk}(x) := \beta_{jk}^0 + \sum_{i \in J_{jk}^-} \frac{\beta_{jk}^i}{U_i - x_i} - \sum_{i \in J_{jk}^+} \frac{\beta_{jk}^i}{x_i - L_i} \tag{44}$$

where α_k^i and β_{jk}^i are suitable constants depending on x_k for $j = 1,\ldots,m$ and $i = 0,\ldots,n$. The index sets are defined by

$$I_k^+ = \{i : 1 \le i \le n, \frac{\partial}{\partial x_i} f(x_k) \ge 0\}$$

and

$$I_k^- = \{i : 1 \le i \le n, \frac{\partial}{\partial x_i} f(x_k) < 0\} \ .$$

In a similar way J_{jk}^- and J_{jk}^+ are defined. The approximations of f and g_j, respectively, are of first-order, i.e.

$$f_k(x_k) = f(x_k) \ , \ \ g_{jk}(x_k) = g_j(x_k)$$

and

$$\nabla f_k(x_k) = \nabla f(x_k) \ , \ \ \nabla g_{jk}(x_k) = \nabla g_j(x_k)$$

for all $j = 1, \ldots, m$.

By an appropriate adoption of the objective function, strict convexity of $f_k(x)$ is guaranteed. Thus, we get a convex and separable subproblem

$$\begin{array}{lll} \min \ f_k(x) & , \ x \in R^n, & \\ g_{jk}(x) \ge 0 & , & (45) \\ \underline{x}_k \le x \le \overline{x}_k & , & \end{array}$$

where $\underline{x}_k = x_k - \omega(x_k - L)$ and $\overline{x}_k = x_k - \omega(x_k - U)$, $\omega \in (0, 1)$, $L := (L_1, \ldots, L_n)^T$, and $U := (U_1, \ldots, U_n)^T$. The parameter ω is introduced to keep the solution away from the bounds. The solution x_{k+1} of (45) is unique provided that the feasible domain is non-empty.

To stabilize convergence, a line search procedure can be added as e.g. for sequential quadratic programming methods. Usually the subproblems are solved very efficiently by a dual approach, where dense linear systems of dimension $m \times m$ must be considered.

4 Large Scale Optimization

The expression *large scale nonlinear programming* cannot be defined via explicit figures defining n and m. We denote a problem to be large, if it cannot be solved by any of the standard approaches of the previous section because of storage requirements, round-off errors, or excessive computing times. Large scale problems, however, often posses special structures as e.g. sparse Jacobian or Hessian matrices. The exploitation of these properties is the main challenge of large scale nonlinear programming.

For most nonlinear programming methods the ability to solve large scale problems depends on the question how certain subproblems are defined and whether they can be solved efficiently. Basically, all the approaches lead to the solution of large linear systems. The efficiency of a nonlinear programming method for large scale applications depends mainly on the formulation of these systems, for example whether they are sparse and whether sparsity in the Jacobian and the Hessian matrices of the original problem can be exploited when solving the subproblems. In this sense, large scale nonlinear programming is closely related to large scale linear algebra (see: Numerical Linear Algebra).

4.1 Limited Memory Quasi-Newton Updates

We consider briefly so-called limited memory BFGS updates for unconstrained optimization, which can be used also for solving large constrained problems.

The BFGS update scheme (14) suggests an update of a matrix H_k by storing a dense matrix of dimension $n \times n$. However, it is possible to store only the update vectors p_k and q_k and to compute the search direction (8) from previous iterates. To limit the memory needed, we store the last l of these vector pairs, where l is relatively small.

Equation (14) is equivalent to

$$H_{k+1} = V_k^T H_k V_k + \rho_k p_k p_k^T \tag{46}$$

where $\rho_k := (p_k^T q_k)^{-1}$ and $V_k := I - \rho_k p_k q_k^T$. Thus, the limited memory BFGS update can be written as

$$\begin{aligned} H_{k+1} = \quad & (V_k^T \dots V_{k-l+1}^T) H_{k-l+1} (V_{k-l+1} \dots V_k) \\ & + \rho_{k-l+1} (V_k^T \dots V_{k-l+2}^T) p_{k-l+1} p_{k-l+1}^T (V_{k-l+2} \dots V_k) \\ & + \rho_{k-l+2} (V_k^T \dots V_{k-l+3}^T) p_{k-l+2} p_{k-l+2}^T (V_{k-l+3} \dots V_k) \\ & + \dots + \rho_k p_k p_k^T \quad . \end{aligned} \tag{47}$$

To compute H_{k+1}, one needs to evaluate only vector-vector products and to store

$$(p_{k-l+1}, q_{k-l+1}), \dots, (p_k, q_k) \quad .$$

A similar approach exists if the inverse BFGS formula (16) is preferred.

4.2 Sequential Linear Programming

In many situations the sequential linear programming method is very suitable for solving large scale nonlinear programming problems, since large linear programs can be solved very efficiently today for example by interior point methods. Because of the linearity of the approximation, special sparsity patterns of the Jacobian matrix $\nabla g(x_k)$ are passed to the constraint matrix of the linear program directly.

A particular advantage of sequential linear programming methods is, that second order information is neither used nor updated. Thus, there is no hereditary of round-off errors in case of inaccurate gradients, for example implied by numerical approximations. On the other hand, convergence is at most linear making the method inefficient in case of highly nonlinear problems with only a few active constraints.

4.3 Interior Point Methods

We have seen in the previous section, that the interior point method allows the formulation of three different systems of linear equations of dimension $n + m$, m, or n, respectively. Thus we

choose (32) if there are more constraints than variables, (33) if there are more variables than constraints, or (31) if we want to exploit sparsity patterns.

4.4 Sequential Quadratic Programming

The sequential quadratic programming method generates a sequence of quadratic problems with matrices B_k of dimension $n \times n$, which are usually dense. There are, however, extensions to make the method also applicable for large problems. One possibility is to apply limited memory BFGS updates as outlined before, combined with an iterative solver for the quadratic subproblem.

Other possibilities are the application of the so-called range space method that is attractive for problems with few constraints, or the null space method vice versa. All these approaches can be combined with sparse update techniques and iterative linear system solvers to avoid the computation and storage of decomposition matrices.

Sequential quadratic programming methods for problems with a large number of nonlinear equality constraints are also available, known under the name *reduced SQP method*.

4.5 Sequential Convex Programming

The standard sequential convex programming method requires the solution of a sequence of nonlinear convex and separable subproblems with dense matrices of dimension $m \times m$.

Another approach to solve large convex subproblems is to apply the so-called primal-dual interior point method as outlined in Section 3.8. First we note that the Hessian matrix of the Lagrangian of the approximated subproblem is known analytically and is diagonal. Moreover, a sparsity pattern of the Jacobian matrix of the original constraints is passed to the convex subproblem (45) immediately and can be exploited.

As for sequential linear programming methods, there is no hereditary of round-off errors and the algorithm solves large nonlinear problems that take advantage of the inverse approximations, very efficiently.

Bibliography

Biegler, L.T., Coleman, T.F., Conn, A.R. and Santosa, F.N. eds. (1997). *Large-Scale Optimization with Applications, Part I, II, III.* Springer, Heidelberg, New York. [Series on papers about large scale optimization and applications]

Coleman, T.F. (1984). *Large Sparse Numerical Optimization.* Lecture Notes in Computer Science, Vol. 165. Springer, Heidelberg, New York. [An overview of basic techniques for large scale nonlinear programming]

Edgar, T.F. and Himmelblau, D.M. (1988). *Optimization of Chemical Processes.* McGraw Hill, New York. [Optimization models and algorithms especially for chemical engineering]

Fiacco, A.V. and McCormick, G.P. (1968). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* Wiley, New York. Reprinted by SIAM Publications (1990), Philadel-

phia. [The classical introduction to penalty and barrier methods]
Fletcher, R. (1987). *Practical Methods of Optimization.* John Wiley and Sons, New York. [Introduction to theory and classical methods]
Gill, P.E., Murray, W. and Wright, M.H. (1992). *Practical Optimization.* Academic Press, New York. [An overview of optimization methods from a numerical point of view]
Kaplan, W. (1999). *Maxima and Minima with Applications.* John Wiley, New York [Introduction into more theoretical aspects of nonlinear programming]
Luenberger, D. (1984). *Linear and Nonlinear Programming.* Addison-Wesley, Reading (MA). [Introduction to theory and classical methods]
Mangasarian, O.L. (1969). *Nonlinear Programming.* McGraw-Hill, New York. Reprinted by SIAM Publications (1994), Philadelphia. [An overview of the theoretical background of nonlinear programming, esp. optimality conditions]
Moré, J.J. and Wright, S.J. (1993). *Optimization Software Guide.* SIAM Publications, Philadelphia. [A review on available nonlinear programming software]
Nesterov, Y.E. and Nemirovskii, A.S. (1994). *Interior Point Polynomial Methods in Convex Programming.* SIAM Publications, Philadelphia. [A comprehensive study of the theoretical background of interior point methods]
Nocedal, J. and Wright, S.J. (1999). *Numerical Optimization.* Springer, Heidelberg, New York. [More modern introduction and review on nonlinear programming techniques]
Papalambros, P. and Wilde, D.J. (1988). *Principles of Optimal Design.* Cambridge University Press, Cambridge. [Introduction with applications from mechanical engineering]
Wright, S.J. (1997). *Primal-Dual Interior-Point Methods.* SIAM publications, Philadelphia. [An introduction to interior point methods for linear programming with extensions to nonlinear programming]