Optimization in Industrial Engineering: SQP-Methods and Applications

Klaus Schittkowski

Department of Computer Science, University of Bayreuth
D - 95440 Bayreuth, Germany.
Email: klaus.schittkowski@uni-bayreuth.de

Abstract

Today, practical nonlinear programming problems are routinely solved by sequential quadratic programming (SQP) methods stabilized by a monotone line search procedure subject to a suitable merit function. To understand the mathematical background, we outline the optimality criteria from where the basic SQP step is easily derived and understood. In case of computational errors as for example caused by inaccurate function or gradient evaluations, however, the approach is unstable and often terminates with an error message. To prevent this situation, a non-monotone line search is proposed which allows the acceptance of a larger steplength. As a by-product, we consider also the possibility to adapt the line search to run under distributed systems. Some numerical results are included, which show that in case of very noisy function values a drastic improvement of the performance is achieved compared to the version with monotone line search. Most industrial applications of SQP methods are found in mechanical structural optimization. We present two alternative case studies from electrical engineering, the optimum design of satellite antennas and of surface acoustic wave filters.

Keywords: SQP, sequential quadratic programming, nonlinear programming, non-monotone line search, merit function, distributed computing, global convergence, numerical results

1 Introduction

For the beginning, we consider the general optimization problem to minimize an objective function under nonlinear equality and inequality constraints,

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

where x is an n-dimensional parameter vector containing the design variables. (1) is also called a nonlinear program or nonlinear programming problem, respectively. 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 treated as general inequality constraints. We then get the somewhat simpler program

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

It is assumed that all problem functions $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are continuously differentiable on the whole $\mathbb{R}^n$.

Although optimization software can be used in the form of a *black box* tool, it is highly desirable to understand at least the basic ideas of the mathematical analysis behind the problem. One reason is that there are many dangerous situations preventing 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 many technical phrases that must be understood to find a remedy. Another reason is that one would like to get an idea how accurate the solution is and whether it is possible to improve or verify a response.

For these reasons, we present a brief outline of the optimization theory behind the algorithms presented on a very elementary level. Optimality criteria identifying a solution of (2) are shown in Section 2 together with an illustrative example.

Sequential quadratic programming is the standard general purpose method to solve smooth nonlinear optimization problems, at least under the following assumptions:

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

There are numerous comparative numerical tests of implementations of SQP methods, e.g., for the code NLPQL of Schittkowski [44, 54]. Results of empirical comparative studies of NLPQL and related SQP methods are found in Schittkowski [39, 42, 41], Schittkowski et al. [55], and Hock and Schittkowski [25]. Further theoretical investigations published in [40, 43] and Dai and Schittkowski [12]. The algorithm is extended to solve also nonlinear least squares problems efficiently, see [46] or [50], and to handle problems with very many constraints, cf. [47].

To conduct the numerical tests, a random test problem generator is developed for a major comparative study, see [39]. Two collections with more than 300 academic and real-life test problems are published in Hock and Schittkowski [25] and in Schittkowski [45]. Corresponding Fortran source codes and a test frame are found in [49]. The test examples are part of the Cute test problem collection of Bongartz et al. [7]. About 80 test problems based on a Finite Element formulation are collected for the comparative evaluation in Schittkowski et al. [55]. A set of 1,170 least squares test problems solved by an extension of the code NLPQL to retain typical features of a Gauss-Newton algorithm, is described in [46, 50]. They are part of an interactive software system called EASY-FIT, see [51].

Moreover, there exist hundreds of commercial and academic applications of NLPQL, for example

1. mechanical structural optimization, see Schittkowski, Zillober, Zotemantel [55] and Kneppe, Krammer, Winkler [29],
2. data fitting and optimal control of transdermal pharmaceutical systems, see Boderke, Schittkowski, Wolf [3] or Blatt, Schittkowski [6],
3. computation of optimal feed rates for tubular reactors, see Birk, Liepelt, Schittkowski, and Vogel [5],
4. food drying in a convection oven, see Frias, Oliveira, and Schittkowski [23],
5. optimal design of horn radiators for satellite communication, see Hartwanger, Schittkowski, and Wolf [22],
6. receptor-ligand binding studies, see Schittkowski [48],
7. optimal design of surface acoustic wave filters for signal processing, see Bünner, Schittkowski, and van de Braak [8].

In Section 3, we outline the general mathematical structure of an SQP algorithm, especially the quadratic programming subproblem, the merit function, and the choice of penalty parameters. Moreover, we show that the implementation of an SQP method is more sophisticated and requires a deep understanding of the underlying mathematical structure of an SQP algorithm. Numerous modifications have been proposed during the past 30 years to improve the robustness, stability and efficiency of SQP

methods. The effective success in a practical environment under real-life conditions depends on a careful selection of tools to fix the numerous decision to be made at each design step of a code.

However, SQP methods are quite sensitive subject to round-off or approximation errors in function and especially gradient values. If objective or constraint functions cannot be computed within machine accuracy or if the accuracy by which gradients are approximated is above the termination tolerance, an SQP code often breaks down with an error message. In this situation, the line search cannot be terminated within a given number of iterations and the algorithm stops.

A new idea is proposed by Dai and Schittkowski [12] making use of non-monotone line search. The idea is to replace the reference value of the line search termination check by another one depending also on previous iterates. Thus, we accept larger stepsizes and are able to overcome situations where the quadratic programming subproblem yields insufficient search directions because of inaccurate gradients.

Moreover, the general availability of parallel computers and in particular of distributed computing motivates a careful redesign of SQP methods to allow simultaneous function evaluations. The line search procedure is further modified to allow parallel function calls, which can also be applied for approximating gradients by a difference formulae. Both extensions are described in Section 4 in more detail.

Section 5 contains some numerical results obtained for a set of more than 300 standard test problems of the collections published in Hock and Schittkowski [25] and in Schittkowski [45]. They show the sensitivity of the new version with respect to the number of parallel machines and the influence of gradient approximations by forward differences under uncertainty. However, it must be emphasized that the distributed computation of function values is only simulated. It is up to the user to adopt the code to a particular parallel environment.

An important advantage od SQP methods is their numerical stability and robustness. Only very few parameters must be set to call the corresponding code, typically only the maximum number of iterations and a termination tolerance. Nevertheless numerous pitfalls and traps exist which can occur during the modelling process and which can prevent a successful and efficient solution. The most important ones are

1. inappropriate termination accuracy,
2. violation of constraint qualification,
3. non-differentiable model functions,
4. empty feasible domains,
5. undefined model function values,
6. wrong function and variable scaling.

In Section 6 we show some remedies and outline how to avoid them that at least in certain situations.

Finally, we briefly introduce two complex industrial applications in Sections 7 and 8, where SQP methods are in daily use, i.e., the optimal design of horn radiators for satellite communication and the optimal design of surface acoustic wave filters.

2 Necessary and Sufficient Optimality Criteria

First we introduce some notations used throughout this paper.

- The gradient of a scalar function $f(x)$ is

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

- The Hessian matrix of a scalar function $f(x)$ is

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

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

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

The Jacobian matrix of $F(x) = (f_1(x), \ldots, f_l(x))^T$ is also written in the form

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

A twice continuously differentiable function is called smooth. The optimization theory for smooth problems is based on the Lagrangian function that combines objective function $f(x)$ and constraints $g_j(x)$, $j = 1, \ldots, m$, in a proper way. In particular, the Lagrangian function allows us to state necessary and sufficient optimality conditions.

Definition 2.1 *Let problem (2) be given.*

a) The feasible region P is the set of all feasible solutions

$$P \doteq \{x \in I\!R^n : g_j(x) = 0,\ j = 1, \ldots, m_e,\ g_j(x) \geq 0,\ j = m_e + 1, \ldots, m\} \quad . \tag{3}$$

b) The active constraints with respect to $p \in P$ are characterized by the index set

$$I(x) \doteq \{j : g_j(x) = 0,\ m_e < j \leq m\} \quad . \tag{4}$$

c) The Lagrangian function of (2) is defined by

$$L(x, u) \doteq f(x) - \sum_{j=1}^{m} u_j g_j(x) \tag{5}$$

for all $x \in I\!R^n$ and $u = (u_1, \ldots, u_m)^T \in I\!R^m$. The variables u_j are called the Lagrangian multipliers of the nonlinear programming problem.

x is also called the primal and u the dual variable of the nonlinear program (2). To become familiar with the notation of a Lagrangian function, we consider a very simple example that we use throughout this chapter to illustrate the theory.

Example 2.1 *An optimization problem is defined by the functions*

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

The Lagrangian function of the problem is

$$L(x, u) = {x_1}^2 + x_2 - u_1(9 - {x_1}^2 - {x_2}^2) - u_2(1 - x_1 - x_2) \quad .$$

It is easy to see in Figure 1 that the optimal solution $x^\star$ and the corresponding active set are

$$x^\star = (0, -3)^T, \quad I(x^\star) = \{1\} \quad .$$

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

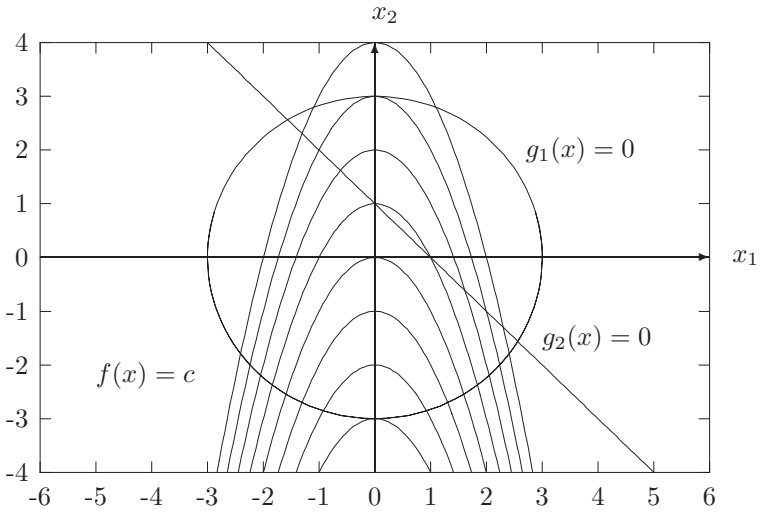

Figure 1: Feasible Domain and Objective Function

Definition 2.2 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if*

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

for all $x, y \in \mathbb{R}^n$ and $\lambda \in (0,1)$, and concave if we replace $\leq$ by $\geq$.

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

To formulate the subsequent optimality conditions, we need a special assumption to avoid irregular behavior of the feasible set P at a local solution. We call it *constraint qualification*, also denoted as *Slater-condition* or *regularity* in more general form. In our situation, it is sufficient to proceed from the following definition.

Definition 2.3 *The nonlinear program (2) satisfies a constraint qualification in $x^\star \in P$, if the gradients of active constraints, the vectors $\bigtriangledown g_j(x^\star)$ for $j \in \{1, \ldots, m_e\} \cup I(x^\star)$, are linearly independent.*

Example 2.2 *Assume that constraints are given by*

$$\begin{aligned} g_1(x_1, x_2) &= x_2 \geq 0 \ , \\ g_2(x_1, x_2) &= -x_2 + {x_1}^2 \geq 0 \ , \end{aligned}$$

and let $x^\star = (0,0)^T$. Since $\bigtriangledown g_1(x^\star) = (0,1)^T$, $\bigtriangledown g_2(x^\star) = (0,-1)^T$, we get

$$\bigtriangledown g_1(x^\star) + \bigtriangledown g_2(x^\star) = 0 \ .$$

Thus, the constraint qualification is not satisfied, see Figure 2.

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

Theorem 2.1 (necessary 2nd order optimality conditions) *Let f and g_j be twice continuously differentiable for $j = 1, \ldots, m$, $x^\star$ a local solution of (2), and the constraint qualification in $x^\star$ be satisfied. Then there exists $u^\star \in \mathbb{R}^m$ with*

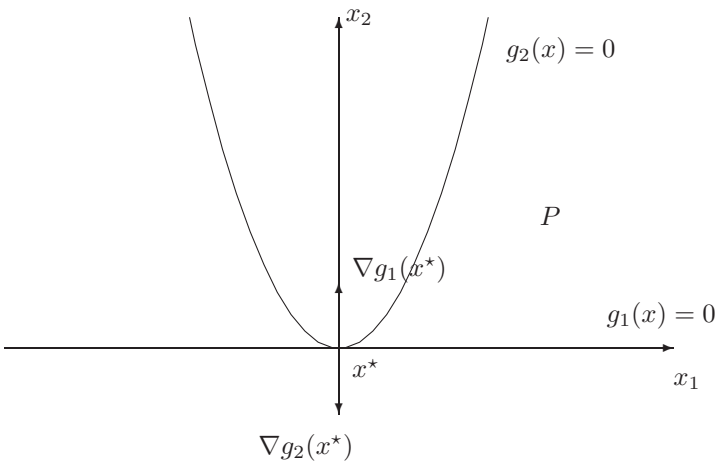

Figure 2: Constraint Qualification

a) (first-order condition)

$$\begin{aligned} & u_j^\star \geq 0 \ , \ \ j = m_e + 1, \ldots, m \ \ , \\ & g_j(x^\star) = 0 \ , \ \ j = 1, \ldots, m_e \ \ , \\ & g_j(x^\star) \geq 0 \ , \ \ j = m_e + 1, \ldots, m \ \ , \\ & \bigtriangledown_x L(x^\star, u^\star) = 0 \ , \\ & u_j^\star \, g_j(x^\star) = 0 \ , \ \ j = m_e + 1, \ldots, m \ \ , \end{aligned} \tag{6}$$

b) (second-order condition)

$$s^T \bigtriangledown_x^2 L(x^\star, u^\star) s \geq 0 \tag{7}$$

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

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. Statement b) implies that the Lagrangian function is positive semi-definite on the tangential space defined by the active constraints.

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

Example 2.3 *Let*

$$\begin{array}{lcll} f(x_1, x_2) & = & x_1 \ , & \\ g_1(x_1, x_2) & = & -x_2 & \geq 0 \ , \\ g_2(x_1, x_2) & = & x_2 - {x_1}^2 & \geq 0 \ . \end{array}$$

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

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

indicating 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 2.2 (sufficient 2nd order optimality conditions) *Let* f *and* g_j *be twice continuously differentiable for* $j = 1, \ldots, m$ *and* $x^\star \in \mathbb{R}^n$, $u^\star \in \mathbb{R}^m$ *be given, so that the following conditions are satisfied:*

a) (first-order condition)

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

b) (second-order condition)

$$
s^T \bigtriangledown_x^2 L(x^\star, u^\star) s > 0
$$

for all $s \in I\!R^n$ *with* $s \neq 0$, $\bigtriangledown g_j(x^\star)^T s = 0$, $j = 1, \ldots, m_e$, *and for all* s *with* $\bigtriangledown g_j(x^\star)^T s = 0$, $j = m_e + 1, \ldots, m$, *and* $u_j^\star > 0$.

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 nonlinear programming textbooks, one has to be aware that optimality conditions are often stated in a slightly different way. The formulation of a nonlinear programming problem varies from author to author, for example depending on a minimum or a maximum formulation, whether the inequality constraints use $\leq$ instead of $\geq$, or whether upper and lower bounds are included or not. There exist different versions of the above theorems, where assumptions are either more general or more specialized, respectively. To illustrate the optimality criteria, we consider a few examples.

Example 2.4 *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* $\bigtriangledown g_1(x^\star)$ *and* $\bigtriangledown g_2(x^\star)$. *In other words, there must exist two multipliers* $u_1^\star \geq 0$ *and* $u_2^\star \geq 0$ *with*

$$
\bigtriangledown f(x^\star) = u_1^\star \bigtriangledown g_1(x^\star) + u_2^\star \bigtriangledown g_2(x^\star) \ ,
$$

see Figure 3.

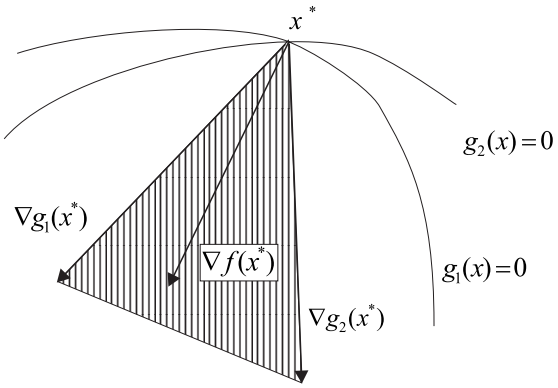

Figure 3: Optimality Condition

Example 2.5 *Consider again Example 2.1,*

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

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

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

we get the multipliers $u_1^\star = 1/6$ *und* $u_2^\star = 0$. *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 The Quadratic Programming Subproblem and the Augmented Lagrangian Merit Function

Sequential quadratic programming or SQP methods belong to the most powerful nonlinear programming algorithms we know today for solving differentiable nonlinear programming problems of the form (2). The theoretical background is described e.g. in Stoer [58] in form of a review, or in Spellucci [57] in form of an extensive text book. From the more practical point of view, SQP methods are also introduced in the books of Papalambros, Wilde [33] and Edgar, Himmelblau [14]. Their excellent numerical performance was tested and compared with other methods in Schittkowski [39], and since many years they belong to the most frequently used algorithms to solve practical optimization problems.

The basic idea is to formulate and solve a quadratic programming subproblem in each iteration which is obtained by linearizing the constraints and approximating the Lagrangian function $L(x,u)$ (5) quadratically, where $x \in I\!R^n$ is the primal and $u = (u_1, \ldots, u_m)^T \in I\!R^m$ the dual variable, i.e., the multiplier vector. Assume that $x_k \in I\!R^n$ is an actual approximation of the solution, $v_k \in I\!R^m$ an approximation of the multipliers, and $B_k \in I\!R^{n\times n}$ an approximation of the Hessian of the Lagrangian function all identified by an iteration index k. Then a quadratic programming subproblem of the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d \\ d \in I\!R^n: & \nabla g_j(x_k)^T d + g_j(x_k) = 0 \;\;,\;\; j \in E, \\ & \nabla g_j(x_k)^T d + g_j(x_k) \geq 0 \;\;,\;\; j \in I \end{array} \tag{8}$$

is formulated and must be solved in each iteration. Here we introduce index sets $E \doteq \{1, \ldots, m_e\}$ and $I \doteq \{m_e+1, \ldots, m\}$. Let d_k be the optimal solution, u_k the corresponding multiplier of this subproblem, and denote by

$$z_k \doteq \begin{pmatrix} x_k \\ v_k \end{pmatrix} \quad , \quad p_k \doteq \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \tag{9}$$

the composed iterate z_k and search direction x_k. A new iterate is obtained by

$$z_{k+1} \doteq z_k + \alpha_k p_k \;\;, \tag{10}$$

where $\alpha_k \in (0,1]$ is a suitable steplength parameter.

To motivate the formulation of the particular subproblem, let $m_e = m$ for simplicity, i.e., we assume for a moment that there are no inequality constraints. The Karush-Kuhn-Tucker optimality conditions (6) are then written in the form

$$\begin{pmatrix} \nabla_x L(x,v) \\ g(x) \end{pmatrix} = 0$$

with $g(x) = (g_1(x), \ldots, g_m(x))^T$. In other words, the optimal solution and the corresponding multipliers are the solution of a system of $n+m$ nonlinear equations $F(z) = 0$ with $n+m$ unknowns $z = (x,v)$, where

$$F(z) = \begin{pmatrix} \nabla_x L(x,v) \\ g(x) \end{pmatrix} \;\;.$$

Let $z_k = (x_k, v_k)$ be an approximation of the solution. We apply Newton's method and get an estimate for the next iterate by

$$\nabla F(z_k) q_k + F(z_k) = 0 \;\;.$$

After insertion, we obtain the equation

$$\begin{pmatrix} B_k & : & -\nabla g(x_k) \\ \nabla g(x_k)^T & : & 0 \end{pmatrix} \begin{pmatrix} d_k \\ y_k \end{pmatrix} + \begin{pmatrix} \nabla f(x_k) - \nabla g(x_k) v_k \\ g(x_k) \end{pmatrix} = 0$$

with $B_k = \nabla_x^2 L(x_k, v_k)$, where $q_k = (d_k, y_k)$. Defining now $u_k = y_k + v_k$, we get

$$B_k d_k - \nabla g(x_k) u_k + \nabla f(x_k) = 0$$

and

$$\nabla g(x_k)^T d_k + g(x_k) = 0 \ .$$

But these equations are exactly the optimality conditions for the quadratic programming subproblem. To sum up, we come to the following conclusion:

A sequential quadratic programming method is identical to Newton's method of solving the necessary optimality conditions, if B_k is the Hessian of the Lagrangian function and if we start sufficiently close to a solution.

Now we assume that inequality constraints are again permitted, i.e., $m_e \leq m$. A straightforward analysis shows that if $d_k = 0$ is an optimal solution of (8) and u_k the corresponding multiplier vector, then x_k and u_k satisfy the necessary optimality conditions of (2).

However, the linear constraints in (8) can become inconsistent even if we assume that the original problem (2) is solvable. As in Powell [34], we add an additional variable δ to (8) and solve an $(n+1)$-dimensional subproblem with consistent constraints.

Another numerical drawback of (8) is that gradients of all constraints must be reevaluated in each iteration step. But if x_k is close to the solution, the calculation of the gradients of inactive nonlinear constraints is redundant. Given a constant $\varepsilon > 0$, we define the sets

$$\overline{I}_1^{(k)} = \{j \in I : g_j(x_k) \leq \varepsilon \text{ or } v_j^{(k)} > 0\}, \quad \overline{I}_2^{(k)} = I \setminus \overline{I}_1^{(k)} \tag{11}$$

and solve the following subproblem at each step,

$$\begin{array}{rl} \text{minimize} & \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d + \frac{1}{2} \varrho_k \delta^2 \\ d \in I\!R^n, \delta \in [0,1]: & \nabla g_j(x_k)^T d + (1-\delta) g_j(x_k) = 0, \quad j \in E, \\ & \nabla g_j(x_k)^T d + (1-\delta) g_j(x_k) \geq 0, \quad j \in \overline{I}_1^{(k)}, \\ & \nabla g_j(x_{k(j)})^T d + g_j(x_k) \geq 0, \ j \in \overline{I}_2^{(k)}. \end{array} \tag{12}$$

The indices $k(j) \leq k$ denote previous iterates where the corresponding gradient has been evaluated the last time. We start with $\overline{I}_1^{(0)} \doteq I$ and $\overline{I}_2^{(0)} \doteq \emptyset$ and reevaluate constraint gradients in subsequent iterations only if the constraint belongs to the active set $\overline{I}_1^{(k)}$. The remaining rows of the Jacobian matrix remain filled with previously computed gradients.

We denote by $(d_k,\ u_k)$ the solution of (12), where u_k is the multiplier vector, and by δ_k the additional variable to prevent inconsistent linear constraints. Under a standard regularity assumption, i.e., the constraint qualification, it is easy to see that $\delta_k < 1$.

B_k is a positive-definite approximation of the Hessian of the Lagrange function. For the global convergence analysis presented in this paper, any choice of B_k is appropriate as long as the eigenvalues are bounded away from zero. However, to guarantee a superlinear convergence rate, we update B_k by the BFGS quasi-Newton method

$$B_{k+1} \doteq B_k + \frac{a_k a_k^T}{b_k^T a_k} - \frac{B_k b_k b_k^T B_k}{b_k^T B_k b_k} \tag{13}$$

with

$$\begin{array}{rcl} a_k & \doteq & \nabla_x L(x_{k+1}, u_k) - \nabla_x L(x_k, u_k), \\ b_k & \doteq & x_{k+1} - x_k. \end{array} \tag{14}$$

Usually, we start with the unit matrix for B_0 and stabilize the formula by requiring that $a_k^T b_k \geq 0.2\ b_k^T B_k b_k$, see Powell [34], Stoer [58], or Schittkowski [41]. A penalty parameter ϱ_k is required to reduce the perturbation of the search direction by the additional variable δ as much as possible, see Schittkowski [43] for a motivation and a formula.

To enforce global convergence of the SQP method, we have to select a suitable penalty parameter r_k and to select a steplength α_k, see (10), subject to a merit function $\phi_k(\alpha)$. To avoid the Maratos effect,

a certain irregularity leading to infinite cycles, we use the differentiable augmented Lagrange function of Rockafellar [38],

$$\Phi_r(x,v) \doteq f(x) - \sum_{j\in E\cup I_1} (v_j g_j(x) - \frac{1}{2} r_j g_j(x)^2) - \frac{1}{2}\sum_{j\in I_2} v_j^2/r_j \ , \tag{15}$$

with $r = (r_1,\ldots,r_m)^T$, $I_1 \doteq \{j \in I : g_j(x) \le v_j/r_j\}$ and $I_2 \doteq I\setminus I_1$, cf. Schittkowski [43]. The merit function is then defined by

$$\phi_k(\alpha) \doteq \Phi_{r_{k+1}}(z_k + \alpha p_k), \tag{16}$$

see also (9). To ensure that p_k is a descent direction of $\Phi_{r_{k+1}}(z_k)$, i.e., that

$$\phi_k'(0) = \nabla\Phi_{r_{k+1}}(z_k)^T p_k < 0 \ , \tag{17}$$

the new penalty parameter r_{k+1} must be selected carefully. Each coefficient $r_j^{(k)}$ of r_k is updated by

$$r_j^{(k+1)} \doteq \max\left(\sigma_j^{(k)} r_j^{(k)}, \frac{2m(u_j^{(k)} - v_j^{(k)})}{(1-\delta_k) d_k^T B_k d_k}\right) \tag{18}$$

for $j = 1,\ldots,m$. The sequence $\{\sigma_j^{(k)}\}$ is introduced to allow decreasing penalty parameters at least in the beginning of the algorithm by assuming that $\sigma_j^{(k)} \le 1$. A a sufficient condition to guarantee convergence of $\{r_j^{(k)}\}$ is that there exists a positive constant ζ with

$$\sum_{k=0}^{\infty}\left[1 - (\sigma_j^{(k)})^{\zeta}\right] < \infty \ . \tag{19}$$

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

Example 3.1 *Consider Example 2.1, where the optimization problem is given by*

$$\begin{array}{rl} & \min\ {x_1}^2 + x_2 \\ x_1, x_2: & 9 - {x_1}^2 - {x_2}^2 \ge 0 \ , \\ & 1 - x_1 - x_2 \ge 0 \ . \end{array}$$

Proceeding from the starting values $x_0 = (2,0)^T$ and the identity matrix for B_0, we get the quadratic programming subproblem

$$\begin{array}{rl} & \min\ \frac{1}{2}{d_1}^2 + \frac{1}{2}{d_2}^2 + 4d_1 + d_2 \\ d_1, d_2: & -4d_1 + 5 \ge 0 \ , \\ & -d_1 - d_2 - 1 \ge 0 \ . \end{array}$$

Since none of the linear constraints is active at the unconstrained minimum, the solution is $d_0 = (-4,-1)^T$ with multiplier vector $u_0 = (0,0)^T$. Assuming that $\alpha_0 = 1$, $x_1 = (-2,-1)^T$ is the next iterate. The new approximation of the Hessian of the Lagrangian function, B_1, is computed by one update of the BFGS method

$$B_1 = B_0 + \frac{a_0 a_0^T}{b_0^T a_0} - \frac{B_0 b_0 b_0^T B_0}{b_0^T B_0 b_0} \ ,$$

$$a_0 = \bigtriangledown_x L(x_1,u_0) - \bigtriangledown_x L(x_0,u_0) = \bigtriangledown f(x_1) - \bigtriangledown f(x_0) = \begin{pmatrix} -8 \\ 0 \end{pmatrix} \ ,$$

$$b_0 = x_1 - x_0 = \begin{pmatrix} -4 \\ -1 \end{pmatrix} \ .$$

Then

$$B_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{32}\begin{pmatrix} 64 & 0 \\ 0 & 0 \end{pmatrix} - \frac{1}{17}\begin{pmatrix} 16 & 4 \\ 4 & 1 \end{pmatrix} \approx \begin{pmatrix} 2 & -\frac{1}{4} \\ -\frac{1}{4} & 1 \end{pmatrix} \ .$$

The new quadratic programming subproblem is

$$\begin{array}{rl} & \min\ {d_1}^2 + \frac{1}{2}{d_2}^2 - \frac{1}{4}d_1 d_2 - 4d_1 + d_2 \\ d_1, d_2: & 4d_1 + 2d_2 + 4 \geq 0\ , \\ & -d_1 - d_2 + 4 \geq 0\ . \end{array}$$

Again, the unconstrained solution is feasible and

$$d_1 = \frac{1}{31}\begin{pmatrix} 60 \\ -16 \end{pmatrix}\ ,\quad u_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

is the optimal solution of the subproblem. Assuming that the steplength is 1, the new iterate is

$$x_2 = \frac{1}{31}\begin{pmatrix} -2 \\ -47 \end{pmatrix}\ .$$

4 An SQP Algorithm with Non-monotone Line Search and Distributed Function Evaluations

The implementation of a line search algorithm is a critical issue when implementing a nonlinear programming algorithm, and has significant effect on the overall efficiency of the resulting code. On the one hand we need a line search to stabilize the algorithm, on the other hand it is not advisable to waste too many function calls. Moreover, the behavior of the merit function becomes irregular in case of constrained optimization because of very steep slopes at the border caused by penalty terms. Even the implementation is more complex than shown below, if linear constraints or bounds for variables are to be satisfied during the line search.

Typically, the steplength α_k is chosen to satisfy the Armijo [1] condition

$$\phi_k(\alpha_k) \leq \phi_k(0) + \mu\alpha_k\phi_k'(0)\ , \tag{20}$$

see for example Ortega and Rheinboldt [31], or any other related stopping rule. Since p_k is a descent direction, i.e., $\phi_k'(0) < 0$, we achieve at least a sufficient decrease of the merit function at the next iterate. The test parameter μ must be positive and not greater than 0.5.

The parameter α_k is chosen by a separate algorithm which should take the curvature of the merit function into account. If α_k is too small, the line search terminates very fast, but the resulting stepsizes might become too small leading to a higher number of outer iterations. On the other hand, a larger value close to one requires too many function calls during the line search. Thus, we need some kind of compromise which is obtained by applying first a polynomial interpolation, typically a quadratic one, combined with a bisection strategy in irregular situations, if the interpolation scheme does not lead to a reasonable new guess. (20) is then used as a stopping criterion.

However, practical experience shows that monotonicity requirement of (20) is often too restrictive especially in case of very small values of $\phi_r'(0)$, which are caused by numerical instabilities during the solution of the quadratic programming subproblem or, more frequently, by inaccurate gradients. To avoid interruption of the whole iteration process, the idea is to conduct a line search with a more relaxed stopping criterion. Instead of testing (20), we accept a stepsize α_k as soon as the inequality

$$\phi_k(\alpha_k) \leq \max_{k-l(k)\leq j\leq k} \phi_j(0) + \mu\alpha_k\phi_k'(0) \tag{21}$$

is satisfied, where $l(k)$ is a predetermined parameter with $l(k) \in \{0, \ldots, \min(k, L)\}$, L a given tolerance. Thus, we allow an increase of the reference value $\phi_{r_{j_k}}(0)$, i.e. an increase of the merit function value. For $L = 0$, we get back the original criterion (20).

To implement the non-monotone line search, we need a queue consisting of merit function values at previous iterates. We allow a variable queue length $l(k)$ which can be adapted by the algorithm, for example, if we want to apply a standard monotone line search as long as it terminates successfully within a given number of steps and to switch to the non-monotone one otherwise.

The idea to store reference function values and to replace the sufficient descent property by a sufficient 'ascent' property, is for example described in Dai [11], where a general convergence proof for the

unconstrained case is presented. The general idea goes back to Grippo, Lampariello, and Lucidi [17], and was extended to constrained optimization and trust region methods in a series of subsequent papers, see Bonnans et al. [4], Grippo et al. [18, 19], Ke et al. [28], Panier and Tits [32], Raydan [37], and Toint [60, 61].

To summarize, we obtain the following non-monotone line search algorithm based on quadratic interpolation and an Armijo-type bisection rule which can be applied in the k-th iteration step of an SQP algorithm:

Algorithm 4.1 *Let β, μ with $0 < \beta < 1$ and $0 < \mu < 0.5$ be given, further an integer $l(k) \geq 0$.*

Start: $\alpha_{k,0} \doteq 1$

For $i = 0, 1, 2, \ldots$ do:

1) If $\phi_k(\alpha_{k,i}) < \max_{k-l(k)\leq j\leq k} \phi_j(0) + \mu\ \alpha_{k,i}\ \phi_k'(0)$, let $i_k \doteq i$, $\alpha_k \doteq \alpha_{k,i_k}$ and stop.

2) Compute $\bar{\alpha}_{k,i} \doteq \dfrac{0.5\ \alpha_{k,i}^2\ \phi_r'(0)}{\alpha_{k,i}\phi_r'(0) - \phi_r(\alpha_{k,i}) + \phi_r(0)}$.

3) Let $\alpha_{k,i+1} \doteq \max(\beta\ \alpha_{k,i}, \bar{\alpha}_{k,i})$.

Corresponding convergence results for the monotone case, i.e., $L = 0$, are found in Schittkowski [43]. $\bar{\alpha}_{k,i}$ is the minimizer of the quadratic interpolation and we use a relaxed Armijo-type descent property for checking termination. Step 3) is required to avoid irregular situations. For example, the minimizer of the quadratic interpolation could be outside of the feasible domain $(0, 1]$. The line search algorithm must be implemented together with additional safeguards, for example to prevent violation of bounds and to limit the number of iterations.

The increasing numerical complexity of practical optimization problems requires a careful redesign of SQP methods to allow simultaneous function evaluations. On the one hand, distributed function calls can be used to approximate derivatives by difference formulae without influencing the nonlinear programming algorithm at all. However, the line search algorithm requires the serial computation of $\phi_k(\alpha_{k,i})$, $i = 1, 2, \ldots$. Thus, Algorithm 4.1 is further modified to allow parallel function calls.

To outline the approach, let us assume that functions can be computed simultaneously on P different machines, $P \geq 1$. Then P test values $\alpha_{k,i} \doteq \beta^{i-1}$ with $\beta = \varepsilon^{1/(P-1)}$ are selected, $i = 1, \ldots, P$, where ε is a guess for the machine precision. We require P parallel function calls to get the corresponding model function values. The largest $\alpha_{k,i}$ satisfying a sufficient descent property (21), say for $i = i_k$, is accepted as the new steplength for getting the subsequent iterate with $\alpha_k \doteq \alpha_{k,i_k}$. For an alternative approach based on pattern search, see Hough, Kolda, and Torczon [26].

The proposed parallel line search will work efficiently, if the number of parallel machines P is sufficiently large, and is summarized as follows:

Algorithm 4.2 *Let β, μ with $0 < \beta < 1$ and $0 < \mu < 0.5$ be given, further two integers $P \geq 1$ and $l(k) \geq 0$.*

Start: For $\alpha_{k,i} \doteq \beta^i$ compute $\phi_k(\alpha_{k,i})$ for $i = 0, \ldots, P-1$.

For $i = 0, 1, 2, \ldots$ do:

If $\phi_k(\alpha_{k,i}) < \max_{k-l(k)\leq j\leq k} \phi_j(0) + \mu\ \alpha_{k,i}\ \phi_k'(0)$, let $i_k \doteq i$, $\alpha_k \doteq \alpha_{k,i_k}$ and stop.

To precalculate P candidates at log-distributed points between a small tolerance $\alpha = \tau$ and $\alpha = 1$, $0 < \tau << 1$, we propose $\beta = \tau^{1/(P-1)}$.

The paradigm of parallelism is SPMD, i.e., Single Program Multiple Data. In a typical situation we suppose that there is a complex application code providing simulation data, for example by an expensive Finite Element calculation in mechanical structural optimization. It is supposed that various instances of the simulation code providing function values are executable on a series of different machines controlled by a master program that executes an optimization algorithm. By a message passing system, for example PVM, see Geist et al. [15], only very few data need to be transferred from the master to the slaves. Typically only a set of design parameters of length n must to be passed. On return, the master accepts new model responses for objective function and constraints, at most $m+1$ double precision numbers. All

massive numerical calculations and model data, for example to compute the stiffness matrix of a Finite Element model in a mechanical engineering application, remain on the slave processors of the distributed system.

Now we are able to formulate the SQP algorithm for solving the constrained nonlinear programming problem (2), see Schittkowski [40, 41, 43] for further details. First, we have to select a couple of real constants ε, β, μ, $\bar{\delta}$, $\bar{\varrho}$, ε and of integer constants L and P, that are not changed within the algorithm and that satisfy

$$\varepsilon \geq 0\ ,\quad 0 \leq \beta \leq 1\ ,\quad 0 \leq \bar{\delta} < 1\ ,\quad \bar{\varrho} > 1\ ,\quad \varepsilon > 0,\ ,\quad L \geq 0\ ,\quad P \geq 1.$$

Choose starting values $x_0 \in I\!R^n$, $v_0 \in I\!R^m$ with $v_j^{(0)} \geq 0$ for $j \in I$, $B_0 \in I\!R^{n\times n}$ positive definite, $\varrho \in I\!R$ with $\varrho > 0$, and $r_0 \in I\!R^m$ with $r_j^{(0)} > 0$ for $j = 1, \ldots, m$. Moreover, we set $\overline{I}_1^{(0)} \doteq I$ and $\overline{I}_2^{(0)} \doteq \emptyset$.

The main steps consist of the following instructions:

Algorithm 4.3 *Start: Evaluate $f(x_0)$, $\nabla f(x_0)$, $g_j(x_0)$, and $\nabla g_j(x_0)$, $j = 1, \ldots, m$.*
For $k = 0, 1, 2, \ldots$ compute x_{k+1}, v_{k+1}, B_{k+1}, r_{k+1}, ϱ_{k+1}, and $I_1^{(k+1)}$ as follows:

Step 1. *Solve the quadratic programming subproblem (12) and denote by d_k, δ_k the optimal solution and by u_k the optimal multiplier vector. If $\delta_k \geq \bar{\delta}$, let $\varrho_k \doteq \bar{\varrho}\varrho_k$ and solve (12) again*

Step 2. *Determine a new penalty parameter r_{k+1} by (18).*

Step 3. *If $\phi_k'(0) \geq 0$, let $\varrho_k \doteq \bar{\varrho}\varrho_k$ and go to Step 1.*

Step 4. *Define the new penalty parameter ϱ_{k+1}.*

Step 5. *Choose a queue length $l(k)$, $0 \leq l(k) \leq L$, to apply Algorithm 4.1 in case of $P = 1$ or Algorithm 4.2 otherwise with respect to the merit function $\phi_k(\alpha)$, see (16), to get a steplength α_k.*

Step 6. *Let $x_{k+1} \doteq x_k + \alpha_k d_k$, $v_{k+1} \doteq v_k + \alpha_k(u_k - v_k)$ be new iterates and evaluate $f(x_{k+1})$, $g_j(x_{k+1})$, $j = 1, \ldots, m$, $\nabla f(x_{k+1})$, $\overline{I}_1^{(k+1)}$ by (11), and $\nabla g_j(x_{k+1})$, $j \in E \cup \overline{I}_1^{(k+1)}$.*

Step 7. *Compute a suitable new positive-definite approximation of the Hessian of the Lagrange function B_{k+1} by (13), set $k \doteq k + 1$, and repeat the iteration.*

To implement Algorithm 4.3, various modifications are necessary. Despite of a theoretically well-defined procedure, the practical code might fail because of round-off errors or violations of some assumptions. For example, we need additional bounds to limit the number of cycles in Step 1, between Step 3 and Step 1, in the line search, and to limit the number of outer iteration steps. The tolerance ε which is used to determine active constraints by (11), serves also as a stopping criterion. The implementation NLPQL of Schittkowski [44, 54] applies several criteria, either simultaneously or in an alternative way, e.g.,

$$\begin{aligned} & d_k^T B_k d_k < \varepsilon^2\ , \\ & |\bigtriangledown f(x_k)^T d_k| + \textstyle\sum_{j=1}^m |u_j^{(k)} g_j(x_k)| < \varepsilon\ , \\ & \|\bigtriangledown_x L(x_k, u_k)\| < \sqrt{\varepsilon}\ , \\ & \textstyle\sum_{j=1}^m |g_j(x_k)| + \sum_{j=m_e+1}^m |\min(0, g_j(x_k))| < \sqrt{\varepsilon}\ . \end{aligned} \tag{22}$$

There remain a few comments to illustrate some further algorithmic details. Most of them are based on empirical investigations or some experience obtained from a large number of practical applications.

1. Although it is sometimes possible to find a reasonable good starting point x_0, it is often impossible to get an initial guess for the Hessian the Lagrange function and the multipliers. Thus, the choice of $B_0 \doteq I$, where I denotes the n by n identity matrix, and $v \doteq 0$ is often used in practice.

2. The BFGS formula (13) can be replaced by an equivalent formula for the factors of a Cholesky decomposition $B_k = L_k L_k^T$, where L_k is a lower triangular matrix.

3. The quadratic programming subproblem can be solved by any available *black-box* algorithm. If a Cholesky decomposition is updated as outlined before, the choice of a primal-dual algorithm is recommended, see e.g. Powell [36]. The additional variable ρ requires the introduction of an additional column and row to B_k, where the lowest diagonal element contains the penalty parameter ρ_k.

4. Since the introduction of the additional variable δ leads to an undesired perturbation of the search direction, it is recommended to solve first problem (8) without the additional variable and to introduce ρ only in case of a non-successful return.

5. It is recommended to start the line search procedure with $l(k) \doteq 0$ and to switch to the non-monotone case $l(k) \doteq L$ only if α_k cannot be computed within a given number of steps.

6. The parallel line search with a limited number of available merit function values seems to be restrictive and seems to require additional theoretical convergence assumptions, since the analysis of the subsequent section is not able to predetermine the required number of cycles or sub-iterations, see also the above comments. However, a practical implementation of the non-monotone line search procedure of Algorithm 4.1 also needs a fixed limitation of the number of iterations. Thus, we assume for simplicity that P is sufficiently big.

Example 4.1 *Consider Example 2.1 again, where the optimization problem is given by*

$$\begin{aligned} & \min \ {x_1}^2 + x_2 \\ x_1, x_2 : \quad & 9 - {x_1}^2 - {x_2}^2 \ \geq \ 0 \ , \\ & 1 - x_1 - x_2 \ \geq \ 0 \ . \end{aligned}$$

If we apply an existing implementation, for example the code NLPQL of Schittkowski [44], we get the iteration sequence of Table 4, also plotted in Figure 4. $r(x_k)$ is the sum of all constraint violations and $s(x_k)$ is one of the internal stopping criteria,

$$s(x_k) = \mid \bigtriangledown f(x_k)^T d_k \mid + \sum_{i=1}^{m} \mid u_i^k \, g_i(x_k) \mid \quad . \tag{23}$$

k	x_1^k	x_2^k	$f(x_k)$	$r(x_k)$	$s(x_k)$
0	2.0000000	0.0000000	4.0000000	1.0	17.02
1	-2.0000000	-1.0000000	3.0000000	0.0	8.13
2	-0.1250000	-1.5937500	-1.5781250	0.0	1.38
3	0.1583333	-2.9065278	-2.8814583	0.0	0.22
4	$0.4648261 \cdot 10^{-1}$	-3.0032835	-3.0011228	$0.22 \cdot 10^{-1}$	$0.54 \cdot 10^{-2}$
5	$-0.3564753 \cdot 10^{-2}$	-3.0004167	-3.0004040	$0.25 \cdot 10^{-2}$	$0.81 \cdot 10^{-3}$
6	$0.8267424 \cdot 10^{-5}$	-3.0000022	-3.0000022	$0.13 \cdot 10^{-4}$	$0.43 \cdot 10^{-5}$
7	$-0.3494870 \cdot 10^{-7}$	-3.0000000	-3.0000000	$0.74 \cdot 10^{-10}$	$0.25 \cdot 10^{-10}$
8	$0.2251719 \cdot 10^{-9}$	-3.0000000	-3.0000000	0.0	$0.12 \cdot 10^{-18}$

Table 1: SQP Iterates

5 Numerical Results

Our numerical tests use the 306 academic and real-life test problems published in Hock and Schittkowski [25] and in Schittkowski [45]. Part of them are also available in the CUTE library, see Bongartz et. al [7], and their usage is described in Schittkowski [49].

Since analytical derivatives are not available for all problems, we approximate them numerically. The test examples are provided with exact solutions, either known from analytical solutions or from the

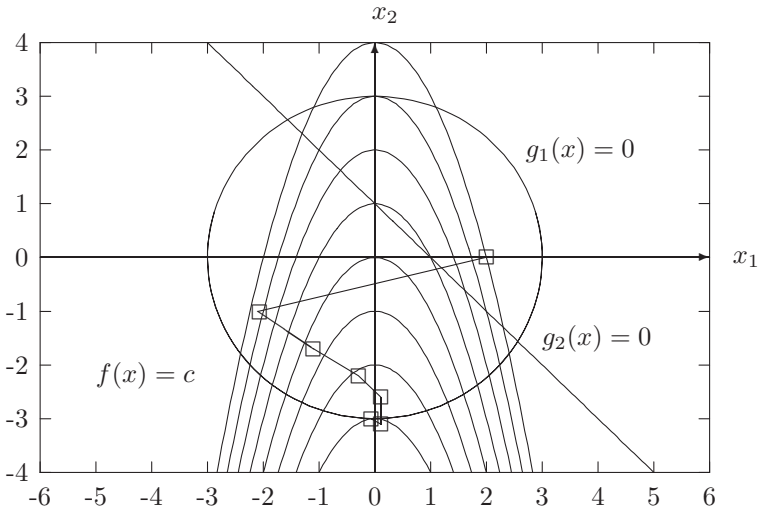

Figure 4: SQP Iterates

best numerical data found so far. The Fortran codes are compiled by the Intel Visual Fortran Compiler, Version 8.0, under Windows XP, and executed on a Pentium IV processor with 2.8 GHz. Total calculation time for solving the test problems with forward differences for the gradient approximation is about 1 sec.

First we need a criterion to decide, whether the result of a test run is considered as a successful return or not. Let $\varepsilon > 0$ be a tolerance for defining the relative accuracy, x_k the final iterate of a test run, and $x^\star$ the supposed exact solution known from the two test problem collections. Then we call the output a successful return, if the relative error in the objective function is less than ε and if the maximum constraint violation is less than ε^2, i.e. if

$$f(x_k) - f(x^\star) < \varepsilon |f(x^\star)| \ , \text{ if } f(x^\star) <> 0$$

or

$$f(x_k) < \varepsilon \ , \text{ if } f(x^\star) = 0$$

and

$$r(x_k) \doteq \max(\|g(x_k)^+\|_\infty) < \varepsilon^2 \ \ ,$$

where $\|\ldots\|_\infty$ denotes the maximum norm and $g_j(x_k)^+ \doteq -\min(0, g_j(x_k))$ for $j > m_e$ and $g_j(x_k)^+ \doteq g_j(x_k)$ otherwise.

We take into account that a code returns a solution with a better function value than the known one within the error tolerance of the allowed constraint violation. However, there is still the possibility that an algorithm terminates at a local solution different from the given one. Thus, we call a test run a successful one, if the internal termination conditions are satisfied subject to a reasonably small termination tolerance, and if in addition to the above decision,

$$f(x_k) - f(x^\star) \geq \varepsilon |f(x^\star)| \ , \text{ if } f(x^\star) <> 0$$

or

$$f(x_k) \geq \varepsilon \ , \text{ if } f(x^\star) = 0$$

and

$$r(x_k) < \varepsilon^2 \ \ .$$

For our numerical tests, we use $\varepsilon = 0.01$, i.e., we require a final accuracy of one per cent. Gradients are approximated by the forward difference formula:

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{\eta_i} \Big(f(x + \eta_i e_i) - f(x) \Big) \tag{24}$$

Here $\eta_i = \eta \max(10^{-5}, |x_i|)$ and e_i is the i-th unit vector, $i = 1, \ldots, n$. The tolerance η depends on the difference formula and is set to $\eta = {\eta_m}^{1/2}$, where η_m is a guess for the accuracy by which function values

	non-monotone line search				monotone line search			
ε_{err}	n_{succ}	n_{func}	n_{grad}	n_{equ}	n_{succ}	n_{func}	n_{grad}	n_{equ}
0	303	33	21	331	301	33	21	331
10^{-12}	297	46	22	357	299	38	22	348
10^{-10}	300	111	25	472	295	46	22	370
10^{-8}	299	94	29	508	282	58	23	402
10^{-6}	295	121	31	749	251	82	27	523
10^{-4}	279	159	30	538	196	107	29	502
10^{-2}	235	204	29	504	103	188	35	538

Table 2: Test Results for Forward Differences (24)

are computed, i.e., either machine accuracy in case of analytical formulae or an estimate of the noise level in function computations. In a similar way, derivatives of constraints are computed.

The Fortran implementation of the SQP method introduced in the previous section, is called NLPQLP, see Schittkowski [54]. Functions and gradients must be provided by reverse communication and the quadratic programming subproblems are solved by the primal-dual method of Goldfarb and Idnani [16] based on numerically stable orthogonal decompositions, see also Schittkowski [52]. NLPQLP is executed with termination accuracy ACC=10^{-7} and the maximum number of iterations is $MAXIT$=*500*.

In the subsequent tables, we use the notation

number of successful test runs (according to above definition)	:	n_{succ}
number of runs with error messages of NLPQLP (IFAIL>0)	:	n_{err}
average number of function evaluations	:	n_{func}
average number of gradient evaluations or iterations	:	n_{grad}
average number of equivalent function calls (function calls counted also for gradient approximations)	:	n_{equ}

To get n_{func}, we count each single function call, also in the case of several simulated processors, $P > 1$. However, additional function evaluations needed for gradient approximations, are not counted. Their total average number is $n_{grad}n_{func}$ for the forward difference formula. One gradient computation corresponds to one iteration of the SQP method.

To test the stability of the SQP method, we add some randomly generated noise to each function value. Non-monotone and monotone line search is applied with a queue size of $L = 30$, and the serial line search calculation by Algorithm 4.1 is required.

Table 2 shows the corresponding results for increasing random perturbations (ε_{err}). The tolerance for approximating gradients, η_m, is set to the machine accuracy in case of $\varepsilon_{err} = 0$, and to the random noise level otherwise.

The results are quite surprising and depend heavily on the new non-monotone line search strategy. We are able to solve about 75 % of the test examples in case of extremely noisy function values with at most two correct digits. In the monotone case, we can only solve about 30 % of all problems successfully

To investigate the situation in more detail, we list now the number of successful returns, n_{succ}, and the number of test runs where NLPQLP terminated because of an error message IFAIL>0, n_{err}. The results are listed in Table 3. They clearly indicate the advantage of non-monotone line searches over the monotone ones. Robustness and stability of the SQP method are significantly increased especially in case of large noise in the function evaluations.

A further series of test runs concerns the situation that a user fixes the tolerance η for gradient approximations, e.g., to $\eta = 10^{-7}$. This is a unlikely worst-case scenario and should only happen in a a situation, where a *black-box* derivative calculation is used and where a user is not aware of the accuracy by which derivatives are approximated. Whereas nearly all test runs break down with error messages for the monotone line search and large random perturbations, the non-monotone line search is still able to find to terminate in at least 30 % NLPQLP calls, see Table 4.

Now we investigate the question, how parallel line searches influence the overall performance. Table 5 shows the number of successful test runs, the average number of function calls, and the average number of iterations or gradient evaluations, for an increasing number of simulated parallel calls of model functions denoted by P.

$P = 1$ corresponds to the sequential case, when Algorithm 4.1 is applied for the line search, consisting

	non-monotone		monotone	
ε_{err}	n_{succ}	n_{err}	n_{succ}	n_{err}
0	303	3	301	10
10^{-12}	297	8	299	27
10^{-10}	300	8	295	58
10^{-8}	299	8	282	106
10^{-6}	295	24	251	159
10^{-4}	279	44	196	210
10^{-2}	235	78	103	251

Table 3: Successful and Non-Successful Returns

	non-monotone		monotone	
ε_{err}	n_{succ}	n_{err}	n_{succ}	n_{err}
0	303	4	301	12
10^{-12}	300	4	299	35
10^{-10}	295	14	278	83
10^{-8}	265	43	196	145
10^{-6}	183	124	49	265
10^{-4}	114	194	13	298
10^{-2}	108	199	23	288

Table 4: Successful and Non-Successful Returns, $\eta = 10^{-7}$

P	n_{succ}	n_{func}	n_{grad}
1	303	33	21
3	237	434	136
4	265	442	108
5	291	318	63
6	296	235	38
7	298	220	31
8	296	223	27
9	299	231	25
10	296	241	24
15	299	342	23
20	298	466	23
50	299	1,146	23

Table 5: Performance Results for Parallel Line Search

of a quadratic interpolation combined with an Armijo-type bisection strategy. Since we need at least one function evaluation for the subsequent iterate, we observe that the average number of additional function evaluations needed for the line search, is about two.

In all other cases, $P > 1$ simultaneous function evaluations are made according to Algorithm 4.2. Thus, the total number of function calls n_{func} is quite big in Table 5. If, however, the number of parallel machines is sufficiently large in a practical situation, we need only one simultaneous function evaluation in each step of the SQP algorithm. To get a reliable and robust line search, we need at least 5 parallel processors. No significant improvements are observed, if we have more than 10 parallel function evaluations.

The most promising possibility to exploit a parallel system architecture occurs, when gradients cannot be calculated analytically, but have to be approximated numerically, for example by forward differences, two-sided differences, or even higher order methods. Then we need at least n additional function calls, where n is the number of optimization variables, or a suitable multiple of n.

6 Numerical Pitfalls

A particular advantage od SQP methods is their numerical stability and robustness. Only very few parameters must be set to call the corresponding code, typically only the maximum number of iterations and a termination tolerance. Nevertheless numerous pitfalls and traps exist which can occur during the modelling process and which can prevent a successful and efficient solution. We show some remedies and outline how to avoid them that at least in certain situations.

6.1 Inappropriate Termination Accuracy

Termination of an SQP algorithm depends on certain conditions, in our case summarized by some checks (22), and a tolerance $\varepsilon > 0$. The choice of ε is crucial for the overall success of an optimization run. A too small value could prevent successful termination completely, since internal round-off errors can prevent satisfaction of (22). On the other hand, a too big value stops the algorithm at an intermediate iterate and we do not know by sure, whether at least a local solution is approximated or not.

On the other hand, the calculation of a search direction mainly depends on the accuracy by which gradients are computed. Objective function values, for example, are not part of the quadratic programming subproblem (8) at all. Only the line search requires function values, but the strategy is to reach the stopping criterion (20) as early as possible.

Thus, a reasonable recommendation is to choose an $\varepsilon > 0$ which is in the order or even a bit higher than the accuracy by which gradients are computed.

6.2 Violation of Constraint Qualification

A certain regularity condition called *constraint qualification* is required for being able to formulate the necessary optimality conditions, see Definition 2.3. However, also the local convergence theorems and most solvers for the underlying quadric program depend on the validity of this assumption at least in the neighborhood of an optimal solution. In general, it is not possible to validate this condition a priory before starting an optimization algorithm. On the other hand, the evaluation of active constraints and the rank determination of the Jacobian of the active constraints depends on a predetermined tolerance, which is be difficult to setup a priory.

If an optimization problem (2) does not satisfy this assumption when approaching a local minimum, the corresponding code typically slows down convergence speed and superlinear convergence is prevented. In other cases, the SQP algorithm could stop because of numerical instabilities when trying to solve the quadratic programming problem (8).

At least in some simple situations, it is possible to prevent violation of the constraint qualification. Consider, for example, a problem with an equality constraint $g(x) = 0$, but the available implementation is unable to handle equality constraints directly as is the case for many structural design optimization codes. The straightforward transformation of each equality constraint to two inequality constraints of the form $g(x) \geq 0$ and $-g(x) \geq 0$, however, should be prevented. In this case, the constraint qualification is always violated.

6.3 Non-Differentiable Model Functions

In most practical and complex applications, it not guaranteed that the model functions by which $f(x)$ and $g_j(x)$, $j = 1, \ldots$, are computed, are continuously differentiable as required. A typical simple standard situation is found if the nonlinear program consists of minimizing the maximum or the sum of absolute values of differentiable functions $f_1(x)$, ..., $f_p(x)$,

$$\begin{aligned} & \min \ \max\{f_i(x) : i = 1, \ldots, p\} \\ x \in I\!R^n : \ & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \end{aligned} \tag{25}$$

or

$$\begin{aligned} & \min \textstyle\sum_{i=1}^{p} |f_i(x)| \\ x \in I\!R^n : \ & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ . \end{aligned} \tag{26}$$

In the first case, we add one additional variable and p additional constraints, and get

$$\begin{aligned} & \min \ x_{n+1} \\ x \in I\!R^{n+1} : \ & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \\ & f_i(x) \leq x_{n+1} \ , \quad i = 1, \ldots, p \ , \end{aligned} \tag{27}$$

in the second case, we add p additional variables and $2p$ additional constraints,

$$\begin{aligned} & \min \ \textstyle\sum_{i=1}^{p} x_{n+i} \\ & g_j(x) = 0 \ , \quad j = 1, \ldots, m_e \ , \\ x \in I\!R^{n+p} : \ & g_j(x) \geq 0 \ , \quad j = m_e + 1, \ldots, m \ , \\ & f_i(x) \leq x_{n+i} \ , \quad i = 1, \ldots, p \ , \\ & f_i(x) \geq -x_{n+i} \ , \quad i = 1, \ldots, p \ . \end{aligned} \tag{28}$$

We obtain equivalent nonlinear programs of the form (2) which can be solved by any algorithm for smooth optimization.

However, the general situation is much more complex and cannot be resolved in the same simple way. Sometimes, a careful gradient calculation based on smooth approximations are applicable. In the simulation code by which objective function and constraint function values are computed, one could try to prevent typical situations leading to instabilities *by hand*, e.g., by smoothing techniques. For example, the Kreiselmeier-Steinhauser function is often used to smoothen the maximum function by

$$\Psi(x) = \frac{1}{\rho} \log \left(\sum_{i=1}^{p} \exp(\rho f_i(x) \right) \ .$$

6.4 Empty Feasible Domains

There is no mathematical criterion nor any applicable numerical method that allows us to detect whether the feasible domain P of (2) is empty or not. Infeasible domains occur for example in case of modelling errors or programming bugs, but also in a systematical way. The subsequent case study of Section 8, the design of surface acoustic wave filters, is a typical example. If the design demands of a customer are too stringent, the feasible domain of (37) is empty.

A simple remedy is to perform a certain regularization of (1) by introducing an additional variable x_{n+1} of the form

$$\begin{aligned} & \min f(x) + \rho x_{n+1} \\ x \in I\!R^{n+1} : \ & g_j(x) \geq -x_{n+1} \ , \quad j = m_e + 1, \ldots, m \ , \\ & x_l \leq x \leq x_u \ , \quad x_{n+1} \geq 0 \ . \end{aligned} \tag{29}$$

For simplicity, we omit the equality constraints. A penalty parameter ρ is to reduce the additional perturbation of the solution by x_{n+1} as much as possible, and must be chosen carefully.

6.5 Undefined Model Function Values

SQP methods have a particular practical advantage. Linear constraints and bounds of variables remain satisfied in all iterations, if the starting point x_0 satisfies them. Thus, these constraints can be used to prevent undefined function calls. For example, if the simulation code for computing $f(x)$ or any of the constraints $g_j(x)$, $j = 1, \ldots, m$, contains an expression of the form $\log(a^T x)$, it is recommended to add a constraint of the form $g_{m+1}(x) \doteq a^T x - \varepsilon$, $\varepsilon > 0$.

In general, however, it is not possible to prevent a function call from the infeasible domain. If there is an internal error code, one could try to return a function value of the form $\delta \|x_k - \overline{x}\|$ with $\overline{x} \in P$. But if the corresponding constraint becomes active at a solution, we will run into an instability and an SQP code will break done with an error message. In such a situation, it is recommended to shrink the feasible set by replacing $g_j(x)$ by $g_j(x) - \delta$, $\delta > 0$, and to adapt δ step-by-step.

6.6 Wrong Function and Variable Scaling

There is no way to find a general scaling routine which automatically predetermines perfect scaling parameters for optimization variables and constraints. The goal is to find positive numbers by which all functions and all individual variables are multiplied, so that, roughly speaking, a change of the scaled variables in the order of one leads to an alteration of the scaled function values in the order of one. However, these coefficients can be determined only at the very beginning of an optimization run based on some information available at the starting point x_0.

But depending on the application, it is sometimes possible to scale the problem in a suitable way. If some constraints have a physical background to prevent them from exceeding an upper bound, for example stress constraints in a mechanical structural optimization model, it is possible to normalize them accordingly. A constraint of the form $\sigma(x) \leq \sigma_0$ with an upper stress limit σ_0 could be replaced by $\frac{\sigma(x)}{\sigma_0} \leq 1$, for example. Also non-negative variables for which reasonable upper bounds are available, i.e., bounds which a likely to be attained, can be scaled to remain between zero and one.

7 Case Study: Horn Radiators for Satellite Communication

Corrugated horns are frequently used as reflector feed sources for large space antennae, for example for INTELSAT satellites. The goal is to achieve a given spatial energy distribution of the radio frequency (RF) waves, called the radiation or directional characteristic. The transmission quality of the information carried by the RF signals is strongly determined by the directional characteristics of the feeding horn as determined by its geometric structure.

The electromagnetic field theory is based on Maxwell's equations relating the electrical field E, the magnetic field H, the electrical displacement, and the magnetic induction to electrical charge density and current density, see Collin [9] or Silver [56]. Under some basic assumptions, particularly homogeneous and isotropic media, Maxwell's equations can be transformed into an equivalent system of two coupled equations. They have the form of a wave equation,

$$\nabla^2 \Psi - c^2 \frac{\partial^2}{\partial t^2} \Psi + f = 0$$

with displacement f enforcing the wave, and wave velocity c. Ψ is to be replaced either by E or H, respectively.

For circular horns with rotational symmetry, the usage of cylindrical coordinates (ρ, ϕ, z) is advantageous, especially since only waves propagating in z direction occur. Thus, a scalar wave equation in cylindrical coordinates can be derived from which general solution is obtained, see for example Collin [9] for more details.

By assuming that the surface of the wave guide has ideal conductivity, and that homogeneous Dirichlet boundary conditions $\Psi = 0$ for $\Psi = E$ and Neumann boundary conditions $\partial\Psi/\partial n = 0$ for $\Psi = H$ at the surface are applied, we get the eigenmodes or eigenwaves for the circular wave guide. Since they form a complete orthogonal system, electromagnetic field distribution in a circular wave guide can be expanded into an infinite series of eigenfunctions, and is completely described by the amplitudes of the modes. For the discussed problem, only the transversal eigenfunctions of the wave guides need to be considered and the eigenfunctions of the circular wave guide can be expressed analytically by trigonometric and Bessel functions.

In principle, the radiated far field pattern of a horn is determined by the field distribution of the waves emitted from the aperture. On the other hand, the aperture field distribution itself is uniquely determined by the excitation in the feeding wave guide and by the interior geometry of the horn. Therefore, assuming a given excitation, the far field is mainly influenced by the design of the interior geometry of the horn. Usually, the horn is excited by the TE_{11} mode, which is the fundamental, i.e., the first solution of the wave equation in cylindrical coordinates. In order to obtain a rotational symmetric distribution of the energy density of the field in the horn aperture, a quasi-periodical corrugated wall structure according to Figure 5 is assumed, see Johnson and Jasik [27].

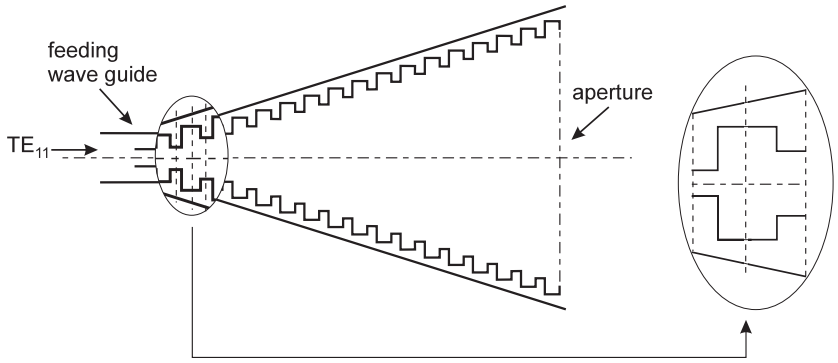

Figure 5: Cross Sectional View of a Circular Corrugated Horn

To reduce the number of optimization parameters, the horn geometry is described by two envelope functions from which the actual geometric data for ridges and slots can be derived. Typically, a horn is subdivided into three sections, see Figure 6, consisting of an input section, a conical section, and an aperture section. For the input and the aperture section, the interior and outer shape of slots and ridges is approximated by a second-order polynomial, while a linear function is used to describe the conical section. It is assumed that the envelope functions of ridges and slots are parallel in conical and aperture section. By this simple analytical approach, it is possible to approximate any reasonable geometry with sufficient accuracy by the design parameters.

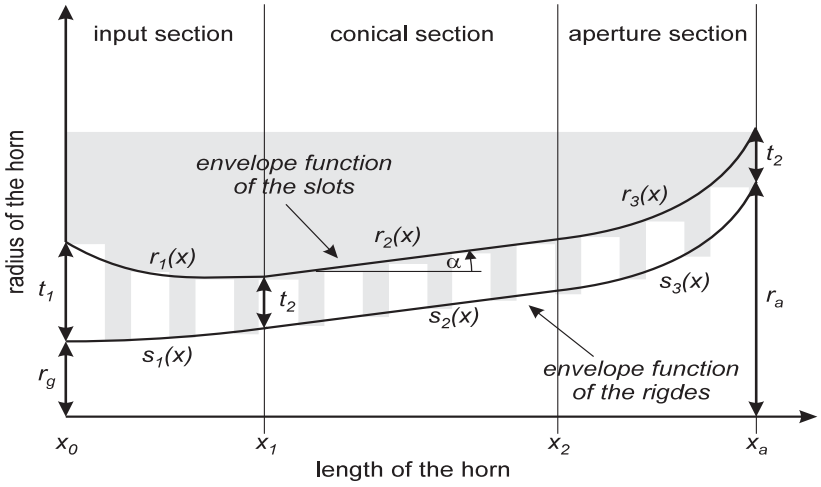

Figure 6: Envelope Functions of a Circular Corrugated Horn

A circular corrugated horn has a modular structure, where each module consists of a step transition between two circular wave guides with different diameters, see Figure 7. The amplitudes of waves, travelling towards and away from the break point, are coupled by a so-called scattering matrix. By combining all modules of the horn step by step, the corresponding scattering matrix describing the total transition of amplitudes from the entry point to the aperture can be computed by successive matrix

operations, see Hartwanger et al. [22] or Mittra [30].

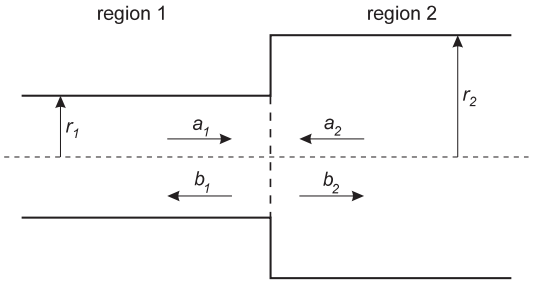

Figure 7: Cross Sectional View of One Module

From Maxwell's equations, it follows that the tangential electrical and magnetic field components must be continuous at the interface between two wave guides. This continuity condition is exploited to compute a relation between the mode amplitudes of the excident $b^k_{E,j}$, $b^k_{H,j}$ and incident $a^k_{E,j}$, $a^k_{H,j}$ waves in each wave guide of a module, see Figure 7, $k = 1, 2$. Then voltage and current coefficients $U^k_{H,j}$, $U^k_{E,j}$, $I^k_{H,j}$, and $I^k_{E,j}$ are defined by the amplitudes.

As mentioned before, the tangential fields must be continuous at the transition between two wave guides. Moreover, boundary conditions must be satisfied, $E_2 = 0$ for $r_1 \le r \le r_2$. Now only n_1 eigenwaves in region 1 and n_2 eigenwaves in region 2 are considered. The electric field in area 1 is expanded subject to the eigenfunctions in area 2 and the magnetic field in area 2 subject to the eigenfunctions in area 1. After some manipulations, in particular interchanging integrals and finite sums, the following relationship between voltage coefficients in region 1 and 2 can be formulated in matrix notation,

$$\begin{pmatrix} U^2_E \\ U^2_H \end{pmatrix} = \begin{pmatrix} X_{EE} & X_{HE} \\ X_{EH} & X_{HH} \end{pmatrix} \begin{pmatrix} U^1_E \\ U^1_H \end{pmatrix} \quad . \tag{30}$$

Here U^k_E and U^k_H are vectors consisting of the coefficients $U^k_{E,j}$ and $U^k_{H,j}$ for $j = 1, \ldots, n_k$, respectively, $k = 1, 2$. The elements of the matrix X_{EE} are given by

$$X^{ij}_{EE} = \int_0^{r_2} \int_0^{2\pi} e^2_{E,i}(\rho, z, \phi)^T \; e^1_{E,j}(\rho, z, \phi) \; \rho \, d\phi \, d\rho \tag{31}$$

with tangential field vectors $e^k_{E,i}(\rho, z, \phi)$ for both regions $k = 1$ and $k = 2$. In the same way X_{HE}, X_{EH}, and X_{EE} are defined. Moreover, matrix equations for the current coefficients are available.

Next, the relationship between the mode amplitude vectors b^k_E and b^k_H of the excident waves $b^k_{E,j}$, $b^k_{H,j}$, and a^k_E and a^k_H of the incident waves $a^k_{E,j}$, $a^k_{H,j}$, $j = 1, \ldots, n_k$, $k = 1, 2$, are evaluated through a so-called scattering matrix. By combining all scattering matrices of successive modules, we compute the total scattering matrix relating the amplitudes at the feed input with those at the aperture,

$$\begin{pmatrix} b_1(x) \\ b_2(x) \end{pmatrix} = \begin{pmatrix} S^\star_{11}(x) & S^\star_{12}(x) \\ S^\star_{21}(x) & S^\star_{22}(x) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad . \tag{32}$$

The vector a_1 describes the amplitudes of the modes exciting the horn, the TE_{11} mode in our case. Thus, a_1 is the $2n_1$-dimensional unity vector. The vector a_2 contains the amplitudes of the reflected modes at the horn aperture, known from the evaluation of the far field. Only a simple matrix $\times$ vector computation is performed to get the modes of reflected waves $b_1(x)$ and $b_2(x)$, once the scattering matrix is known.

The main goal of the optimization procedure is to find an interior geometry x of the horn so that the distances of $b_2(x)^j$ from given amplitudes $\overline{b}^j_2$ for $j = 1, \ldots, 2n_2$ become as small as possible. The first component of the vector $b_1(x)$ is a physically significant parameter, the so-called return loss, representing the power reflected at the throat of the horn. Obviously, this return loss should be minimized as well. The phase of the return loss and further components of $b_1(x)$ are not of interest.

From these considerations, the least squares optimization problem

$$\begin{aligned} x \in I\!R^n : \quad & \min \textstyle\sum_{j=1}^{2n_2} (b^j_2(x) - \overline{b}^j_2)^2 + \mu \, b^1_1(x)^2 \\ & x_l \le x \le x_u \end{aligned} \tag{33}$$

name	x_0^i	x_{opt}^i	*comment*
x_1	50.0	111.85	length of input section
x_{con}	50.0	0.00	length of conical section
x_o	50.0	47.00	length of output section
α	28.0	29.00	semi flare angle of conical section
q	0.25	0.20	quotient of slot and ridge width
t_1	12.5	11.97	depth of first slot in input section
t_2	7.2	7.82	depth of slots in conical section

Table 6: Initial and Optimal Parameter Values

is obtained. The upper index j denotes the j-th coefficient of the corresponding vector, μ a suitable weight, and x_l, x_u lower and upper bounds for the parameters to be optimized. Note also that complex numbers are evaluated throughout this section, leading to a separate evaluation of the regression function of (33) for the real and imaginary parts of $b_2^j(x)$.

The least squares problem is solved by a special variant of NLPQL called DFNLP, see Schittkowski [46, 53], which retains main features of a Gauss-Newton method after a certain transformation. For a typical test run under realistic assumptions, the radius of the feeding wave guide, and the radius of the aperture are kept constant, $r_g = 11.28\,mm$ and $r_a = 90.73\,mm$, where 37 ridges and slots are assumed. Parameter names, initial values x_0, and optimal solution values x_{opt} are listed in Table 7. The number of modes, needed to calculate the scattering matrix, is 70. Forward differences are used to evaluate numerical derivatives subject to a tolerance of 10^{-7}, and $\mu = 1$ was set for weighting the return loss. NLPQL needed 51 iterations to satisfy the stopping tolerance 10^{-7}.

8 Case Study: Design of Surface Acoustic Wave Filters

Computer-aided design optimization of electronic components is a powerful tool to reduce development costs on one hand and to improve the performance of the components on the other. A bandpass filter selects a band of frequencies out of the electro-magnetic spectrum. In this section, we consider surface-acoustic-wave (SAW) filters consisting of a piezo-electric substrate, where the surface is covered by metal structures. The incoming electrical signal is converted to a mechanical signal by this setup. The SAW filter acts as a transducer of electrical energy to mechanical energy and vice versa. The efficiency of the conversion depends strongly on the frequencies of the incoming signals and the geometry parameters of the metal structures, for example length, height, etc. On this basis, the characteristic properties of a filter are achieved.

Due to small physical sizes and unique electrical properties, SAW-bandpass filters raised tremendous interest in mobile phone applications. The large demand of the mobile phone industry is covered by large-scale, industrial mass-production of SAW-filters. For industrial applications, bandpass filters are designed in order to satisfy pre-defined electrical specifications. The *art of filter design* consists of defining the internal structure, or the geometry parameters, respectively, of a filter such that the specifications are satisfied. The electrical properties of the filters are simulated based on physical models. The simulation of a bandpass filter consists of the acoustic tracks, i.e., the areas on the piezo-electrical substrate on which the electrical energy is converted to mechanical vibrations and vice versa, and the electrical combinations of the different acoustic tracks. Typically, only the properties of the acoustic tracks are varied during the design process, and are defined by several physical parameters. As soon as the filter properties fit to the demands, the mass production of the filter is started.

When observing the surface of a single-crystal, we see that any deviation of an ion from its equilibrium position provokes a restoring force and an electrical field due to the piezo-electric effect. Describing the deviations of ions at the surface in terms of a scalar potential, we conclude that the SAW is described by a scalar wave equation

$$\phi_{tt} = c^2 \Delta \phi \quad . \tag{34}$$

The boundary conditions are given by the physical conditions at the surface and are non-trivial, since the surface is partly covered by a metal layer. In addition, the piezo-electric crystal is non-isotropic, and

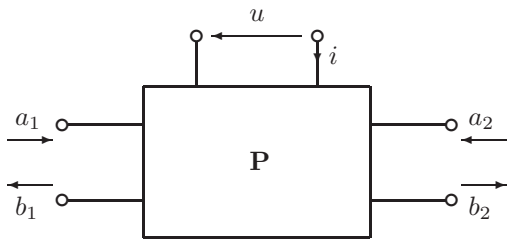

Figure 8: Base Cell of the P-Matrix Model with Two Acoustic and One Electric Port

the velocity of the wave depends on its direction. For the numerical simulation, additional effects such as polarization charges in the metal layers have to be taken into account. Consequently, the fundamental wave equation is not solvable in a closed form.

For this reason, Tobolka [59] introduced the P-matrix model as an equivalent mathematical description of the SAW. One element is a simple base cell, which consists of two acoustic ports, and an additional electric port. The acoustic ports describe the incoming and outgoing acoustic signals, the electrical ports the electric voltage at this cell, see Figure 8. The quantities a_1, a_2, b_1 and b_2 denote the intensities of the acoustic waves. In terms of a description based on the wave equation, we have $a_1 \propto \phi$, u is the electrical voltage at the base cell, and i is the electrical current.

The P-matrix model describes the interaction of the acoustic waves at the acoustic ports, with the electric port in linear form. Typically, a transformation is given in the form

$$\begin{pmatrix} b_1 \\ b_2 \\ i \end{pmatrix} = \mathbf{P} \begin{pmatrix} a_1 \\ a_2 \\ u \end{pmatrix}, \tag{35}$$

where for example

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & E \\ 1 & 0 & -E^* \\ -2E & 2E^* & 2|E|^2 + i(-\mathcal{H}\{2|E|^2\} + \omega C) \end{pmatrix} . \tag{36}$$

$\mathcal{H}$ denotes the Hilbert transformation, C the static capacity between two fingers, E the excitation given by

$$E = -i\ 0.5\ \sqrt{\omega W K} \cdot \int_{tr} \sigma_e(x)\ \exp^{-ik|x|}\, dx\ ,$$

ω the frequency, W the aperture of the IDT, K a material constant, and σ_e the electric load distribution.

In general, the elements of P are the dimensionless acoustic reflection and transmission coefficients in the case of a short-circuited electrical port. The 2×2 upper diagonal submatrix is therefore the scattering matrix of the acoustic waves and describes the interaction of the incoming and outgoing waves. Other elements characterize the relation of the acoustic waves with the electric voltage, i.e., the piezo-electric effect of the substrate, or the admittance of the base cell, i.e., the the quotient of current to voltage and the reciprocal value of the impedance.

Proceeding from the P-matrix model, we calculate the scattering matrix S. This matrix is the basic physical unit that describes the electro-acoustic properties of the acoustic tracks, and finally the filter itself. The transmission coefficient T is one element of the scattering matrix, $T = S_{21}$.

Mobile phone manufacturers provide strict specifications towards the design of a bandpass filter. Typically, the transmission has to be above certain bounds in the pass band and below certain bounds in the stop band depending on the actual frequency. These specifications have to be achieved by designing the filter in a proper way. Depending on the exact requirements upon the filter to be designed, different optimization problems can be derived.

To formulate the optimization problem, let us assume that $x \in I\!R^n$ denotes the vector of design variables. By $T(f, x)$ we denote the transmission subject to frequency f and the optimization variable vector x. Some disjoint intervals R_0, ..., R_s define the design space within the frequency interval $f_l \leq f \leq f_u$. Our goal is to maximize the minimal distance of transmission $T(f, x)$ over the interval R_0,

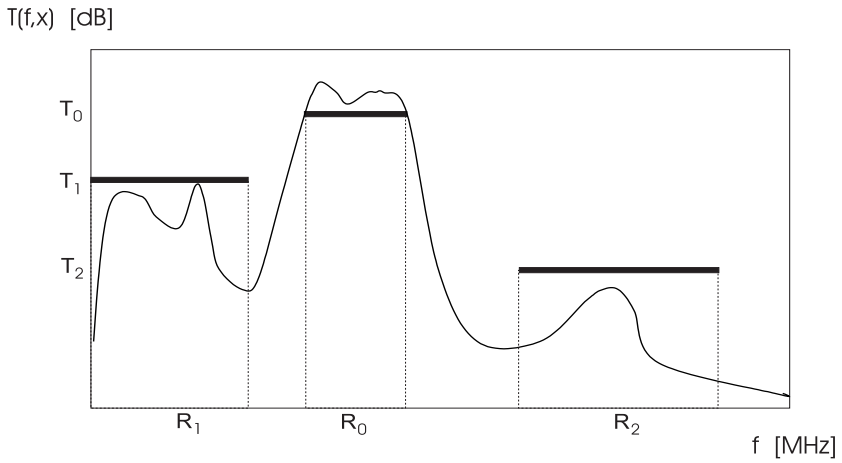

Figure 9: Design Goals of an SAW Filter

under lower bounds $T_1, \ldots, T_s$ for the transmission in the remaining intervals $R_1, \ldots, R_s$. Moreover, it is required that the transmission is always above a certain bound in R_0, i.e., that $T(f,x) \geq T_0$ for all $f \in R_0$. The optimization problem is formulated as

$$\begin{aligned} & \max \ \min \{T(f,x) : f \in R_0\} \\ x \in I\!R^n : \ & T(f,x) \leq T_i \ \text{ for } f \in R_i \ , \ i = 1, \ldots, s \ , \\ & \underline{x} \leq x \leq \overline{x} \ , \ . \end{aligned} \tag{37}$$

Here $\underline{x}$ and $\overline{x} \in I\!R^n$ are lower and upper bounds for the design variables.

To transform the infinite dimensional optimization problem into a finite dimensional one, we proceed from a given discretization of the frequency variable f by an equidistant grid in each interval. The corresponding index sets are called J_0, J_1, ..., J_s. Let l be the total number of all grid points. First we introduce the notation $T_j(x) = T(f_j, x)$, f_j suitable grid point, $j = 1, \ldots, l$. All indices are ordered sequentially so that $\{1, \ldots, l\} = J_0 \cup J_1 \cup \ldots \cup J_s$, i.e., $J_0 = \{1, \ldots, l_0\}$, $J_1 = \{l_0 + 1, \ldots, l_1\}$, ..., $J_s = \{l_{s-1} + 1, \ldots, l\}$. Then the discretized optimization problem is

$$\begin{aligned} & \max \ \min \{T_j(x) : j \in J_0\} \\ x \in I\!R^n : \ & T_j(x) \leq T_i \ \text{ for } j \in J_i \ , \ i = 1, \ldots, s \ , \\ & \underline{x} \leq x \leq \overline{x} \ , \ . \end{aligned} \tag{38}$$

The existence of a feasible design is easily checked by performing the test $T_j(x) \geq T_0$ for all $j \in J_0$. Problem (38) is equivalent to a smooth nonlinear program after a simple standard transformation. For more details, see Bünner et. al [8], where also integer variables are taken into account.

Lower and upper bounds for the ten design variables under consideration are shown in Table 7 together with initial values and final ones obtained by the code NLPQL. Simulation is performed with respect to 154 frequency points leading to 174 constraints. Altogether 56 iterations of NLPQL are needed.

9 Conclusions

A brief introduction is presented to illustrate optimality conditions for differentiable nonlinear programs. It is quite easy to derive the basic iteration step of an SQP method, which can be considered as a Newton step for satisfying the optimality conditions. We present a modification of an SQP algorithm designed for execution under noisy function values and/or a parallel computing environment (SPMD) and where a non-monotone line search is applied in error situations. Numerical results indicate stability and robustness for a set of more than 300 standard test problems. It is shown that not more than 6 parallel function evaluation per iterations are required for conducting the line search. Significant performance improvement is achieved by the non-monotone line search especially in case of noisy function values and

variable	*lower bound*	*initial value*	*optimal value*	*upper bound*
x_1	5.0	11.58	9.589	15.0
x_2	50.0	50.0	92.39	150.0
x_3	10.5	11.39	11.25	11.5
x_4	10.0	10.61	10.62	11.0
x_5	0.3	0.3	0.3	0.5
x_6	0.95	1.033	1.031	1.05
x_7	0.95	1.031	1.023	1.05
x_8	0.95	1.012	1.015	1.02
x_9	0.985	1.001	0.998	1.03
x_{10}	1.0	1.0	1.000	1.03

Table 7: Bounds, Initial, and Optimal Values for Design Variables

numerical differentiation. With the new non-monotone line search, we are able to solve about 80 % of the test examples in case of extremely noisy function values with at most two correct digits and forward differences for derivative calculations. Two industrial case studies from electrical engineering are outlined which show the complexity of practical simulation models and the applicability of SQP algorithms.

References

[1] Armijo L. (1966): *Minimization of functions having Lipschitz continuous first partial derivatives,* Pacific Journal of Mathematics, Vol. 16, 1–3

[2] Barzilai J., Borwein J.M. (1988): *Two-point stepsize gradient methods,* IMA Journal of Numerical Analysis, Vol. 8, 141–148

[3] Boderke P., Schittkowski K., Wolf M., Merkle H.P. (2000): *Modeling of diffusion and concurrent metabolism in cutaneous tissue,* Journal on Theoretical Biology, Vol. 204, No. 3, 393-407

[4] Bonnans J.F., Panier E., Tits A., Zhou J.L. (1992): *Avoiding the Maratos effect by means of a nonmonotone line search, II: Inequality constrained problems – feasible iterates,* SIAM Journal on Numerical Analysis, Vol. 29, 1187–1202

[5] Birk J., Liepelt M., Schittkowski K., Vogel F. (1999): *Computation of optimal feed rates and operation intervals for tubular reactors,* Journal of Process Control, Vol. 9, 325-336

[6] Blatt M., Schittkowski K. (1998): *Optimal Control of One-Dimensional Partial Differential Equations Applied to Transdermal Diffusion of Substrates*, in: Optimization Techniques and Applications, L. Caccetta, K.L. Teo, P.F. Siew, Y.H. Leung, L.S. Jennings, V. Rehbock eds., School of Mathematics and Statistics, Curtin University of Technology, Perth, Australia, Vol. 1, 81 - 93

[7] Bongartz I., Conn A.R., Gould N., Toint Ph. (1995): *CUTE: Constrained and unconstrained testing environment*, Transactions on Mathematical Software, Vol. 21, No. 1, 123–160

[8] Bünner M.J., Schittkowski K., van de Braak G. (2002): *Optimal design of surface acoustic wave filters for signal processing by mixed-integer nonlinear programming*, submitted for publication

[9] Collin R.E. (1991): *Field Theory of Guided Waves*, IEEE Press, New York

[10] Dai Y.H. (2000): *A nonmonotone conjugate gradient algorithm for unconstrained optimization,* Journal of Systems Science and Complexity, Vol. 15, No. 2, 139–145.

[11] Dai Y.H. (2002): *On the nonmonotone line search,* Journal of Optimization Theory and Applications, Vol. 112, No. 2, 315–330.

[12] Dai Y.H., Schittkowski K. (2005): *A sequential quadratic programming algorithm with nonmonotone line search*, submitted for publication.

[13] Dai Y.H., Liao L.Z. (2002): *R-Linear Convergence of the Barzilai and Borwein Gradient Method,* IMA Journal of Numerical Analysis, Vol. 22, No. 1, 1–10.

[14] Edgar T.F., Himmelblau D.M. (1988): *Optimization of Chemical Processes,* McGraw Hill

[15] Geist A., Beguelin A., Dongarra J.J., Jiang W., Manchek R., Sunderam V. (1995): *PVM 3.0. A User's Guide and Tutorial for Networked Parallel Computing*, The MIT Press

[16] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs*, Mathematical Programming, Vol. 27, 1-33

[17] Grippo L., Lampariello F., Lucidi S. (1986): *A nonmonotone line search technique for Newtons's method,* SIAM Journal on Numerical Analysis, Vol. 23, 707–716

[18] Grippo L., Lampariello F., Lucidi S. (1989): *A truncated Newton method with nonmonotone line search for unconstrained optimization,* Journal of Optimization Theory and Applications, Vol. 60, 401–419

[19] Grippo L., Lampariello F., Lucidi S. (1991): *A class of nonmonotone stabilization methods in unconstrained optimization,* Numerische Mathematik, Vol. 59, 779–805

[20] Han S.-P. (1976): *Superlinearly convergent variable metric algorithms for general nonlinear programming problems* Mathematical Programming, Vol. 11, 263-282

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

[22] Hartwanger C., Schittkowski K., Wolf H. (2000): *Computer aided optimal design of horn radiators for satellite communication*, Engineering Optimization, Vol. 33, 221-244

[23] Frias J.M., Oliveira J.C, Schittkowski K. (2001): *Modelling of maltodextrin DE12 drying process in a convection oven*, to appear: Applied Mathematical Modelling

[24] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer

[25] Hock W., Schittkowski K. (1983): *A comparative performance evaluation of 27 nonlinear programming codes*, Computing, Vol. 30, 335-358

[26] Hough P.D., Kolda T.G., Torczon V.J. (2001): *Asynchronous parallel pattern search for nonlinear optimization*, to appear: SIAM J. Scientific Computing

[27] Johnson R.C., Jasik H. (1984): *Antenna Engineering*, McGraw Hill, New York

[28] Ke X., Liu G., Xu D. (1996): *A nonmonotone trust-region algorithm for unconstrained optimization,* Chinese Science Bulletin, Vol. 41, 197–201

[29] Kneppe G., Krammer J., Winkler E. (1987): *Structural optimization of large scale problems using MBB-LAGRANGE*, Report MBB-S-PUB-305, Messerschmitt-Bölkow-Blohm, Munich

[30] Mittra R. (1973): *Computer Techniques for Electromagnetics*, Pergamon Press, Oxford

[31] Ortega J.M., Rheinbold W.C. (1970): *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York-San Francisco-London

[32] Panier E., Tits A. (1991): *Avoiding the Maratos effect by means of a nonmonotone line search, I: General constrained problems,* SIAM Journal on Numerical Analysis, Vol. 28, 1183–1195

[33] Papalambros P.Y., Wilde D.J. (1988): *Principles of Optimal Design,* Cambridge University Press

[34] Powell M.J.D. (1978): *A fast algorithm for nonlinearly constraint optimization calculations,* in: Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer

[35] Powell M.J.D. (1978): *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, in: Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds., Academic Press

[36] Powell M.J.D. (1983): *On the quadratic programming algorithm of Goldfarb and Idnani*. Report DAMTP 1983/Na 19, University of Cambridge, Cambridge

[37] Raydan M. (1997): *The Barzilai and Borwein gradient method for the large-scale unconstrained minimization problem,* SIAM Journal on Optimization, Vol. 7, 26–33

[38] Rockafellar R.T. (1974): Augmented Lagrange multiplier functions and duality in non-convex programming, SIAM Journal on Control, Vol. 12, 268–285

[39] Schittkowski K. (1980): *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer

[40] Schittkowski K. (1981): *The nonlinear programming method of Wilson, Han and Powell. Part 1: Convergence analysis*, Numerische Mathematik, Vol. 38, 83-114

[41] Schittkowski K. (1981): *The nonlinear programming method of Wilson, Han and Powell. Part 2: An efficient implementation with linear least squares subproblems*, Numerische Mathematik, Vol. 38, 115-127

[42] Schittkowski K. (1983): *Theory, implementation and test of a nonlinear programming algorithm*, in: Optimization Methods in Structural Design, H. Eschenauer, N. Olhoff eds., Wissenschaftsverlag

[43] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction,* Optimization, Vol. 14, 197-216

[44] Schittkowski K. (1985/86): *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500

[45] Schittkowski K. (1987): *More Test Examples for Nonlinear Programming,* Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer

[46] Schittkowski K. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method,* in: Trends in Mathematical Optimization, K.-H. Hoffmann, J.-B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, 295-309

[47] Schittkowski K. (1992): *Solving nonlinear programming problems with very many constraints*, Optimization, Vol. 25, 179-196

[48] Schittkowski K. (1994): *Parameter estimation in systems of nonlinear equations*, Numerische Mathematik, Vol. 68, 129-142

[49] Schittkowski K. (2002): *Test problems for nonlinear programming - user's guide*, Report, Department of Mathematics, University of Bayreuth

[50] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems*, Kluwer Academic Publishers, Dordrecht

[51] Schittkowski K. (2002): *EASY-FIT: A software system for data fitting in dynamic systems*, Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169

[52] Schittkowski K. (2003): *QL: A Fortran code for convex quadratic programming - user's guide*, Report, Department of Mathematics, University of Bayreuth, 2003

[53] Schittkowski K. (2003): *DFNLP: A Fortran implementation of an SQP-Gauss-Newton algorithm - user's guide*, Report, Department of Mathematics, University of Bayreuth, 2003

[54] Schittkowski K. (2004): *NLPQLP20: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide*, Report, Department of Computer Science, University of Bayreuth

[55] Schittkowski K., Zillober C., Zotemantel R. (1994): *Numerical comparison of nonlinear programming algorithms for structural optimization*, Structural Optimization, Vol. 7, No. 1, 1-28

[56] Silver S. (1949): *Microwave Antenna Theory and Design*, McGraw Hill, New York

[57] Spellucci P. (1993): *Numerische Verfahren der nichtlinearen Optimierung,* Birkhäuser

[58] Stoer J. (1985): *Foundations of recursive quadratic programming methods for solving nonlinear programs,* in: Computational Mathematical Programming, K. Schittkowski, ed., NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 15, Springer

[59] Tobolka G. (1979): *Mixed matrix representation of SAW transducers*, Proceedings of the IEEE Ultrasonics Symposium, Vol. 26, 426-428

[60] Toint P.L. (1996): *An assessment of nonmontone line search techniques for unconstrained optimization,* SIAM Journal on Scientific Computing, Vol. 17, 725–739

[61] Toint P.L. (1997): *A nonmonotone trust-region algorithm for nonlinear optimization subject to convex constraints,* Mathematical Programming, Vol. 77, 69–94

[62] Wolfe P. (1969): *Convergence conditions for ascent methods,* SIAM Review, Vol. 11, 226–235