NLPQLF: A Fortran Implementation of a Feasible SQP Method for Solving Nonlinear Constrained Optimization Problems

- User's Guide -

Address: Prof. K. Schittkowski
Siedlerstr. 3
D - 95488 Eckersdorf
Germany

Phone: (+49) 921 32887

E-mail: klaus@schittkowski.de

Web: http://www.klaus-schittkowski.de

Date: December, 2013

Abstract

The Fortran subroutine NLPQLF solves smooth nonlinear programming problems and is an extension of the code NLPQLP. It is assumed that objective function or constraints can be evaluated only at argument values within a convex set described by some other feasibility constraints. The numerical method is a two-stage process. Starting from a feasible point, a new search direction is computed by solving a quadratic program expanded by the nonlinear feasibility constraints. Thus, the new iterate is feasible subject to these constraints and objective function as well as the remaining constraint function values can be evaluated. The usage of the code is documented and illustrated by an example. Also some numerical test results are presented.

Keywords: SQP, sequential quadratic programming, nonlinear programming, feasibility restoration, numerical algorithm, Fortran codes

1 Introduction

We consider the optimization problem to minimize an objective function f under nonlinear equality and inequality constraints,

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

where x is an n-dimensional parameter vector. It is assumed that so-called feasibility functions $e_i(x)$, $i = 1$, ..., m, are concave and continuously differentiable on the whole $I\!R^n$. Moreover, we suppose that the remaining scalar functions $f(x)$ and $g_j(x)$, $j = 1$, ..., m_f are continuously differentiable and that they can be evaluated only on the convex subset

$$
F := \{x \in I\!R^n : e_i(x) \geq 0, i = 1, \ldots, m\} \ . \tag{2}
$$

We implicitly assume that the computation of feasibility constraints $e_i(x)$, $i = 1$, ..., m, is much less expensive than an evaluation of then objective function $f(x)$ or the remaining constraints $g_j(x)$, $j = 1$, ..., m_f.

A typical situation which motivates the implementation of NLPQLF, is nonlinear semi-definite programming in topology optimization, where the variables are semi-definite matrices and where objective function and constraints cannot be evaluated if any of these matrices is indefinite. This new branch of topology optimization is called free material optimization (FMO), where the coefficients of elementary material matrices are optimization variables. All these matrices of dimension 3 or 6, respectively, must be positive definite in order to get a positive definite global stiffness matrix, for which structural responses like displacement, stress, or dynamical constraints can be computed. For more details, see Lehmann et al. [7], where a feasible sequential convex programming algorithm is introduced.

Feasible direction methods compute a feasible direction $d^{(k)}$ which ensures the existence of a descent step inside the feasible domain. To improve the performance and to get a higher convergence order, a quadratic programming subproblem similar to SQP methods is formulated. The resulting search direction may not be feasible, since active constraints can lead to a search direction tangential to the feasible region, see, e.g., Panier and Tits [8]. Thus, a correction is determined by tilting the original direction towards the feasible region. To ensure fast convergence near a solution an additional search direction is computed by bending. An extended line search is performed along the search arc consisting of all three directions, such that feasibility and a sufficient descent in the objective function is guaranteed.

Moreover, feasible direction interior point algorithms (FDIP) are developed, see, e.g., Herskovitz [4]. Interior point methods (IPM) compute in each iteration a Newton descent direction by solving a linear system of equations. The resulting search direction might not be a feasible direction. Thus, a second linear system is formulated where the right hand side is perturbed ensuring a feasible direction. Some of the FDIP methods solve a third linear system to ensure superlinear convergence near a stationary point. Analogously, to feasible direction SQP methods, a line search along the search arc is performed to ensure both feasibility and a descent in the objective function.

Basically, NLPQLF is a sequential quadratic programming method which is frequently used to solve smooth nonlinear optimization problems, and we proceed from the version developed by Schittkowski [11, 12, 13, 14]. The resulting Fortran code is called NLPQLP, see [17].

To guarantee feasibility of constraints $e_1(x)$, $\ldots$, $e_m(x)$, we assume that the starting point $x_0 \in I\!R^n$ is feasible subject to these constraints. SQP methods formulate quadratic programming subproblems by a quadratic approximation of the Lagrangian and by linearizing constraints. Whereas constraints $g_1(x)$, $\ldots$, $g_{m_f}(x)$ remain as linear functions in the QP, the linearized feasibility constraints are replaced by the original nonlinear constraints $e_i(x) \geq 0$, $i = 1, \ldots, m$. Thus, we obtain a nonlinear subproblem to be solved in each step of the modified SQP algorithm, where the objective function is a quadratic one and where the constraints consist of a mixture of linear and nonlinear ones. This subproblem is then solved by an SQP algorithm, in our case by the code NLPQLP, see Schittkowski [17]. A particular advantage is that a subsequent line search will not violate feasibility because of the convexity of the convex set F, see (2).

In Section 2 we outline the general mathematical structure of an SQP algorithm and the modifications to guarantee feasibility. Section 4 contains some numerical test results, and the usage of the Fortran subroutine is documented in Section 4.

2 A New Feasible Sequential Quadratic Programming Method

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 (1). The theoretical background is described, e.g., in Stoer [20] in form of a review, or in Spellucci [19] in form of an extensive text book. Their excellent numerical performance is tested and compared with other methods in Schittkowski [10], and since many years they belong to the most frequently used algorithms to solve practical optimization problems.

To facilitate the notation of this section, we assume that upper and lower bounds x_u

and x_l are not handled separately, i.e., we consider the somewhat simpler formulation

$$x \in I\!R^n : \begin{array}{ll} \min f(x) & \\ g_j(x) = 0 \ , & j = 1, \ldots, m_e \ , \\ g_j(x) \geq 0 \ , & j = m_e + 1, \ldots, m_f \ , \\ e_i(x) \geq 0 \ , & i = 1, \ldots, m \ . \end{array} \tag{3}$$

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

$$L(x, u) := f(x) - \sum_{j=1}^{m_f} u_j^g g_j(x) - \sum_{i=1}^{m} u_i^e e_i(x) \tag{4}$$

quadratically, where $x \in I\!R^n$ is the primal variable and $u^g = (u_1^g, \ldots, u_{m_f}^g)^T \in I\!R^{m_f}$ and $u^e = (u_1^e, \ldots, u_m^e)^T \in I\!R^m$ the two multiplier vectors, which we stack to get

$$u := \begin{pmatrix} u^g \\ u^e \end{pmatrix} \in I\!R^{m_f + m} \ .$$

To formulate the quadratic programming subproblem, we proceed from given iterates $x_k \in I\!R^n$, an approximation of the solution, $v_k \in I\!R^{m_f+m}$, an approximation of the multipliers, and $B_k \in I\!R^{n \times n}$, an approximation of the Hessian of the Lagrangian function. We consider now the extended subproblem

$$y \in I\!R^n : \begin{array}{ll} \min \frac{1}{2}(y - x_k)^T B_k (y - x_k) + \nabla f(x_k)^T (y - x_k) & \\ \nabla g_j(x_k)^T (y - x_k) + g_j(x_k) = 0 \ , & j = 1, \ldots, m_e \ , \\ \nabla g_j(x_k)^T (y - x_k) + g_j(x_k) \geq 0 \ , & j = m_e + 1, \ldots, m_f \ , \\ e_i(y) \geq 0 \ , & i = 1, \ldots, m \ . \end{array} \tag{5}$$

Let y_k be the optimal solution and u_k the corresponding multiplier of this subproblem. A new iterate is obtained by

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} := \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} y_k - x_k \\ u_k - v_k \end{pmatrix} \tag{6}$$

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

Although we are able to guarantee that the matrix B_k is positive definite, it is possible that (5) is not solvable due to inconsistent constraints. One possible remedy is to introduce an additional variable $\delta \in I\!R$, leading to a modified quadratic programming problem, see Schittkowski [15] for details.

The steplength parameter α_k is required in (6) to enforce global convergence of the SQP method, i.e., the approximation of a point satisfying the necessary Karush-Kuhn-Tucker optimality conditions when starting from arbitrary initial values, typically a user-provided $x_0 \in I\!R^n$ and $v_0 = 0$, $B_0 = I$. α_k should satisfy at least a sufficient decrease condition of a merit function $\phi_r(\alpha)$ given by

$$\phi_r(\alpha) := \psi_r\left(\left(\begin{array}{c} x_k \\ v_k \end{array}\right) + \alpha\left(\begin{array}{c} y_k - x_k \\ u_k - v_k \end{array}\right)\right) \tag{7}$$

with a suitable penalty function $\psi_r(x, v)$, e.g., an augmented Lagrangian function as used in [13]. The objective function is *penalized* as soon as an iterate leaves the feasible domain. The corresponding penalty parameters r_j, $j = 1, \ldots, m_f + m$ which control the degree of constraint violation, must carefully be chosen to guarantee a sufficient descent direction of the merit function, see Schittkowski [13]. Also in our special situation, it is possible to show that

$$\phi'_{r_k}(0) = \bigtriangledown\psi_{r_k}(x_k, v_k)^T\left(\begin{array}{c} y_k - x_k \\ u_k - v_k \end{array}\right) < 0 \enspace , \tag{8}$$

a fundamental property of all line search based optimization methods.

Finally one has to approximate the Hessian matrix of the Lagrangian function in a suitable way. To avoid calculation of second derivatives and to obtain a final superlinear convergence rate, the standard approach is to update B_k by the BFGS quasi-Newton formula, cf. Powell [9] or Stoer [20].

3 Performance Evaluation

3.1 The Test Environment

Our numerical tests use the 306 academic and real-life test problems published in Hock and Schittkowski [5] and in Schittkowski [16]. Part of them are also available in the Cute library, see Bongartz et. al [2], and their usage is described in Schittkowski [18].

Since analytical derivatives are not available for all problems, we approximate them numerically by forward differences. The test examples are provided with exact solutions, either known from analytical precalculations *by hand* or from the best numerical data found so far.

First we need a criterion to decide whether the result of a test run is considered as a successful return or not. Let $\epsilon > 0$ 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 test problem collection. 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

$$f(x_k) - f(x^\star) < \epsilon|f(x^\star)| \enspace , \text{ if } f(x^\star) \neq 0$$

or

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

and

$$r(x_k) = \|g(x_k)^-\|_\infty < \epsilon^2 \ \ ,$$

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

We take into account that a code returns a solution with a better function value than the known one, subject to 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 known one. Thus, we call a test run a successful one, if in addition to the above decision the internal termination conditions are satisfied subject to a reasonably small tolerance (*IFAIL=0*), and if

$$f(x_k) - f(x^\star) \geq \epsilon |f(x^\star)| \ , \ \text{if } f(x^\star) \neq 0$$

or

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

and

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

For our numerical tests, we use $\epsilon = 0.01$ to determine a successful return, i.e., we require a final accuracy of one per cent. Note that in all cases, NLPQLF is called with a termination tolerance of 10^{-7}, which is also used for solving the internal quadratic program (5) extended by the feasibility constraint.

The Fortran implementation of the feasible SQP method introduced in the previous section, is called NLPQLF. 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 [3] based on numerically stable orthogonal decompositions.

However, the model functions of all test examples can be evaluated everywhere, and we have to simulate infeasible domains. We consider only test problems with at least one inequality constraint. This gives a subset of 94 problems, where we consider all inequality constraints as feasible ones, i.e., we suppose that objective function and equality constraints can be evaluated only at argument values x satisfying all inequality constraints.

In the subsequent tables, we use the notation

code	n_{succ}	n_{func}	n_{grad}	n_{func}^f	n_{grad}^f	*time*
NLPQLF	92	9	8	13	6	0.3
NLPQLP	94	26	15	-	-	0.1

Table 1: Performance Results for Feasible and Non-Feasible SQP Codes

n_{succ}	-	number of successful test runs according to above definition
n_{func}	-	average number of function evaluations in the outer cycle
n_{grad}	-	average number of gradient evaluations or iterations, respectively, in the outer cycle
n_{func}^f	-	average number of function evaluations to satisfy feasibility constraints in (5)
n_{grad}^f	-	average number of gradient evaluations to satisfy feasibility constraints in (5)
$f(x)$	-	final objective function value
$r(x)$	-	final maximum constraint violation
time	-	total execution time for all test runs in seconds

To get n_{func} or n_{grad}, we count each evaluation of a whole set of function or gradient values, respectively, for a given iterate x_k. However, additional function evaluations needed for gradient approximations, are not counted by n_{func}. Their average number is n_{func} for forward differences used. The Fortran codes are compiled by the Intel Visual Fortran Compiler, Version 11.0, 64 bit, under Windows 7 and Intel(R) Core(TM) i7-2720 CPU, 2.2 GHz, with 8 GB RAM.

3.2 An Ill-Post Test Case

A free material optimization (FMO) problem can be formulated by a nonlinear semidefinite programming (NSDP) problem based on a finite element discretization with m elements. The common FMO formulation is to minimize the maximal compliance

$$f_j^T K^{-1}(E) f_j$$

for loads f_j, $j = 1, \ldots, l$, where l is the number of load cases and $K(E)$ the global stiffness matrix. A mathematically rigorous description is found, e.g., in Kocvara and Zowe [6]. As a measure of the material stiffness, we use the traces of the elasticity matrices E_i. They fulfill the basic requirements of linear elasticity, i.e., they are symmetric and positive semidefinite. Moreover, volume and box constraints preventing singularities are included.

The design variable E is a block diagonal matrix consisting of symmetric matrices E_i, $i = 1, \ldots, m$, representing material properties of each finite element. The matrices E_i, $i = 1, \ldots, m$, must be symmetric and positive semidefinite to satisfy the basic requirements of linear elasticity, see Bendsøe et al. [1]. Moreover, the variables might become zero in some regions. This situation is known as vanishing material and E_i is interpreted as void.

For two-dimensional designs, we get elementary stiffness matrices of the form

$$E_i \quad := \quad \begin{pmatrix} e_{i1} & e_{i2} & e_{i3} \\ e_{i2} & e_{i4} & e_{i5} \\ e_{i3} & e_{i5} & e_{i6} \end{pmatrix} \succeq 0, \quad i = 1, \ldots, m \tag{9}$$

with six variables identifying the symmetric matrix E_i.

The stiffness of the structure with respect to loads f_j, $j = 1, \ldots, l$, is then given by

$$f_j^T u_j, \quad j = 1, \ldots, l, \tag{10}$$

where $u_j(E) \in I\!R^3$ is the displacement vector determined by the equilibrium equations

$$K(E)\, u_j(E) = f_j, \quad j = 1, \ldots, l, \tag{11}$$

which are derived from Hooke's law subject to internal forces and acting loads.

We consider now the simplest possible case with one element ($m = 1$) and one load case ($l = 1$), to illustrate how NLPQLF treats the feasibility constraints which are, in this case,

$$e_1(E) \quad := \quad e_3 e_1 - e_2^2 \tag{12}$$
$$e_2(E) \quad := \quad e_6\left(e_3 e_1 - e_2^2\right) - e_4^2 e_3 - e_5^2 e_1 + 2 e_2 e_4 e_5 \tag{13}$$

where $e_1 > 0$ is enforced by a lower bound. Moreover, we have to ensure that these constraints remain positive throughout the algorithm also in case of round-off errors, which can be prevented by an additional lower bound $\epsilon > 0$. Together with a volume constraint

$$\text{trace} E \leq V$$

with a suitable available volume constant $V > 0$, we get our simplified test example

$$e \in I\!R^6, u \in I\!R^3 : \quad \begin{array}{ll} \min f^T u & \\ Eu = f \ , & , \\ e_3 e_1 - e_2^2 \geq \epsilon \ , & \\ e_6(e_3 e_1 - e_2^2) - e_4^2 e_3 - e_5^2 e_1 + 2 e_2 e_4 e_5 \geq \epsilon & \\ e_1 \geq \epsilon & , \end{array}$$

Note that if E is positive definite when evaluating the equality constraints, u is uniquely determined. The optimal elasticity tensor is

$$E^\star \quad := \quad \begin{pmatrix} 2 & 4 & 2 \\ 4 & 8 & 0.04 \\ 2 & 0.04 & 0.0012 \end{pmatrix} \tag{14}$$

The code NLPQLF returns the subsequent output:

```
 --------------------------------------------------------------------
  START OF THE SQP FEASIBILITY RESTORATION ALGORITHM
 --------------------------------------------------------------------

  Parameters:
     N      =        9
     MF     =        4
     MEF    =        3
     M      =        2
     ACC    =   0.1000D-11
     ACCF   =   0.1000D-11
     ACCQP  =   0.1000D-13
     MAXFUN =       20
     MAXNM  =       20
     MAXIT  =     1000
     IPRINT =        2

  Output in the following order:
     IT    - iteration number
     F     - objective function value
     SCV   - sum of constraint violations
     NA    - number of active constraints
     I     - number of line search iterations
     ALPHA - steplength parameter
     DELTA - additional variable to prevent inconsistency
     KKT   - Karush-Kuhn-Tucker optimality criterion


   IT         F           SCV       NA  I    ALPHA     DELTA     KKT
 --------------------------------------------------------------------
    1  0.00000000D+00  0.30D+01    6  0  0.00D+00  0.00D+00  0.10D+02
    2  0.50001000D+01  0.00D+00    3  1  0.10D+01  0.00D+00  0.97D+00
    3  0.40309784D+01  0.11D+00    3  1  0.10D+01  0.00D+00  0.25D+01
                                 .............
   17  0.59678624D+00  0.26D+00    6  2  0.44D+00  0.00D+00  0.14D+00
   18  0.51824810D+00  0.82D-01    6  1  0.10D+01  0.00D+00  0.14D-01
   19  0.52789050D+00  0.52D-03    6  1  0.10D+01  0.00D+00  0.23D-01
   20  0.50454185D+00  0.61D-02    6  1  0.10D+01  0.00D+00  0.49D-02
   21  0.50001636D+00  0.34D-03    6  1  0.10D+01  0.00D+00  0.49D-04
   22  0.50006505D+00  0.94D-08    6  1  0.10D+01  0.00D+00  0.51D-14

   --- Final Convergence Analysis (NLPQLF) at Best Iterate ---

     Best result at iteration:      ITER  =      22
     Objective function value:      F(X)  =  0.50006505D+00
     Solution values:               X     =
        0.19998267D+01  0.39994450D+01  0.19997590D-01  0.79989732D+01
        0.40000214D-01  0.12000658D-02  0.83230626D-01  0.20842296D+00
       -0.11494539D-02
     Constraint values:             G(X)  =
        0.20751578D-09  0.15685549D-09 -0.44194164D-08  0.00000000D+00
       -0.46272847D-08 -0.28232193D-10
     Multipliers for constraints:   U     =
        0.19568907D-05  0.32168807D-05  0.72772721D-07  0.83927742D-06
        0.83919703D-06  0.14401838D-06
     Number of function calls:      NFUNC =      36
     Number of gradient calls:      NGRAD =      22
     Average number of feasibility
        iterations:                 NFEAS =       3
```

The extremely fast final convergence step is remarkable. On the other hand, NLPQLP returns the subsequent lines, where an additional message is displayed whenever a feasibility constraint is violated:

```
    IT          F           SCV      NA  I    ALPHA     DELTA     KKT
  --------------------------------------------------------------------
     1  0.00000000D+00  0.30D+01    6  0  0.00D+00  0.00D+00  0.15D+02
     2  0.50001000D+01  0.00D+00    3  1  0.10D+01  0.00D+00  0.31D+00
     3  0.46944595D+01  0.11D-01    3  1  0.10D+01  0.00D+00  0.13D+01
                                            .............
    13  0.13543634D+01  0.22D+00    3  2  0.10D+00  0.00D+00  0.12D+01
 *** G(6) <0: -0.364653442476918
    14  0.12412657D+01  0.22D+00    3  2  0.10D+00  0.00D+00  0.11D+01
 *** G(6) <0:  -1.20368856330328
    15  0.11376287D+01  0.22D+00    3  2  0.10D+00  0.00D+00  0.10D+01
                                            .............
```

```
*** G(5) <0: -4.412994731370517E-004
    28  0.50005501D+00  0.44D-03    6  1  0.10D+01  0.00D+00  0.84D-07
*** G(5) <0: -2.017533619177438E-004
    29  0.50005503D+00  0.20D-03    6  1  0.10D+01  0.00D+00  0.16D-05
*** G(5) <0: -5.733124207759175E-003
*** G(5) <0: -2.389092677688828E-004
*** G(5) <0: -2.003091407198023E-004
    30  0.50005503D+00  0.20D-03    6  3  0.10D-01  0.00D+00  0.20D-04
*** G(5) <0: -1.256826789441750E-005
    31  0.50006484D+00  0.13D-04    6  1  0.10D+01  0.00D+00  0.88D-06
*** G(5) <0: -8.820606257881015E-006
    32  0.50006459D+00  0.88D-05    6  1  0.10D+01  0.00D+00  0.88D-06
    33  0.50006503D+00  0.20D-10    6  1  0.10D+01  0.00D+00  0.67D-10
    34  0.50006503D+00  0.11D-11    6  1  0.10D+01  0.00D+00  0.33D-09
                                         .............
    43  0.50006501D+00  0.60D-08    6  1  0.10D+01  0.00D+00  0.11D-07
    44  0.50006500D+00  0.34D-07    6  1  0.10D+01  0.00D+00  0.11D-07
    45  0.50006501D+00  0.16D-11    6  1  0.10D+01  0.00D+00  0.16D-12

      Objective function value:      F(X)  =  0.50006501D+00
      Solution values:               X     =
         0.19998200D+01  0.39994400D+01  0.19995400D-01  0.79989800D+01
         0.39990800D-01  0.11999340D-02  0.10001110D+00  0.20002195D+00
         0.10001045D-02
      Constraint values:             G(X)  =
         0.22204460D-14 -0.91042489D-15 -0.20244223D-14  0.00000000D+00
        -0.15739283D-11 -0.55424415D-15
      Multipliers for constraints:   U     =
         0.10001100D+00  0.20002200D+00  0.10000196D-02  0.50012502D-01
         0.50011502D-01  0.50019006D-02
      Number of function calls:      NFUNC =     65
      Number of gradient calls:      NGRAD =     45
      Number of calls of QP solver:  NQL   =     45
```

3.3 Some Test Problems with Hard Feasibility Constraints

We generate a series of test examples with square roots of nonlinear expressions depending on optimization variables. Proceeding from $l = 4$, $n = 3l$, $m_e = 0$, $m_f = 2l + 1$, $m = 2l$, some data

$$\begin{array}{rcl} \text{a} & = & (1500,\ 1400,\ 1000,\ 900)^T \\ \text{b} & = & (\text{-}500,\ \text{-}400,\ \text{-}100,\ \text{-}100)^T \\ \text{c} & = & (2500,\ 2700,\ 2000,\ 2400)^T \\ \text{d} & = & (3000,\ 3000,\ 3000,\ 3000)^T \\ \text{e} & = & (3500,\ 3400,\ 3100,\ 3100)^T \end{array}$$

and $\nu = 250$, $\mu = 0.02$, and $\gamma_k = 100k + 200$ for $k = 1, \ldots, 16$, we define

$$f(x) := -\sum_{i=1}^{l} c_i \left(\frac{f_{i1}(x)}{\sqrt{d_i}} + 1 - \frac{f_{i1}(x)^2}{d_i} \right) \sqrt{f_{i2}(x)} \tag{15}$$

$$g_i(x) := \frac{x_{l+i} + \mu f(x)}{\nu} - 1 \tag{16}$$

$$g_{l+1}(x) := 1 - \frac{1}{\gamma_k} \sum_{i=1}^{l} x_i x_{2l+i} \tag{17}$$

$$g_{l+1+i}(x) := -\frac{x_{l+i}}{e_i}(x_{2l+i} - a_i f_{i1}(x)) \tag{18}$$

$$\tag{19}$$

where

$$f_{i1}(x) := \sqrt{x_i x_{2l+i} - b_i} \tag{20}$$

$$f_{i2}(x) := 1 - \frac{\sqrt{e_i}}{a_i f_{i1}(x)} x_{l+i} \tag{21}$$

for $i = 1, \ldots, l$. Thus, feasibility constraints become

$$e_{i1}(x) := x_i x_{2l+i} - b_i - \epsilon \tag{22}$$

$$e_{i2}(x) := a_i^2 x_i x_{2l+i} - a_i^2 b_i - e_i x_{l+i}^2 - \epsilon \tag{23}$$

where $\epsilon > 0$ is introduced for numerical reasons, since inequality constraints can be satisfied only subject to a certain tolerance. Note that all variables are non-negative.

Numerical results are presented in Tables 2 and 3 for $l = 4$. For test problems not mentioned in Table 3, NLPQLP produced a severe error message (*uphill search direction*) at start. In the other cases, NLPQLP stopped with a severe error when trying to solve the first quadratic programming problem.

k	i_{fail}	n_{func}	n_{grad}	$f(x)$	$r(x)$
1	0	5	5	-55.587	0.27D-08
2	0	6	5	-57.768	0.00D+00
3	0	7	6	-61.718	0.11D-15
4	0	8	7	-62.469	0.72D-09
5	0	11	7	-61.756	0.00D+00
6	0	14	8	-60.999	0.00D+00
7	0	13	7	-62.539	0.00D+00
8	0	13	7	-63.556	0.00D+00
9	0	13	7	-63.055	0.19D-07
10	0	12	7	-65.962	0.48D-07
11	0	14	8	-64.532	0.00D+00
12	0	14	8	-65.973	0.00D+00
13	0	12	7	-64.136	0.00D+00
14	0	14	8	-65.994	0.00D+00
15	0	14	8	-66.822	0.67D-13
16	0	13	8	-69.726	0.16D-13

Table 2: Performance Results for NLPQLF

k	i_{fail}	n_{func}	n_{grad}	$f(x)$	$r(x)$
4	136	1	1	187149.650	0.77D+04
5	122	1	1	218767.143	0.90D+04
8	129	1	1	313760.457	0.13D+05
9	135	1	1	345459.642	0.14D+05
10	122	1	1	377172.218	0.15D+05
12	123	1	1	440631.279	0.18D+05
13	129	1	1	472375.332	0.19D+05
14	123	1	1	504127.785	0.21D+05
15	123	1	1	535887.860	0.22D+05
16	136	1	1	567654.894	0.23D+05

Table 3: Performance Results for NLPQLP

4 Program Documentation

NLPQLF is implemented in form of a Fortran subroutine following the original implementation, i.e., the code NLPQLP, as closely as possible. The nonlinear programming subproblem (5) is solved by NLPQLP. together with the quadratic programming solver QL, an implementation of the primal-dual method of Goldfarb and Idnani [3].

Model functions and gradients are called by reverse communication. The user has to provide functions and gradients in the same program which executes NLPQLF, according to the following rules:

1. Choose starting values for the variables to be optimized, and store them in X.
2. Compute objective and all constraint function values values, and store them in F and G, respectively.
3. Compute gradients of objective and all constraint functions, and store them in DF and DG.
4. Set IFAIL=0 and execute NLPQLF.
5. If NLPQLF returns IFAIL=-1, compute objective and all constraint function values $g_1(x)$, …, $g_{m_f}(x)$, $e_1(x)$, …, $e_m(x)$ and store them in G in the same order. Then call NLPQLF again, but do not change IFAIL.
6. If NLPQLF terminates with IFAIL=-2, compute gradient values subject to variables stored in X for the objective function $f(x)$ and all constraints and store them in DG. Then call NLPQLF again without changing IFAIL.
7. If NLPQLF returns IFAIL=-3 and IFAILF=-1, compute constraint function values of the feasibility constraints $e_1(x)$, …, $e_m(x)$ and store them in G at positions MF+1,...,MF+M. Then call NLPQLF again.
8. If NLPQLF returns IFAIL=-3 and IFAILF=-2, compute gradient values for the constraints $e_1(x)$, …, $e_m(x)$ and store them in DG at row positions MF+1,...,MF+M. Then call NLPQLF again.
9. If NLPQLF terminates with IFAIL=0, the internal stopping criteria are satisfied. The variable values found in X are considered as a local solution of the nonlinear program.
10. In case of IFAIL>0, an error occurred.

Usage:

```
      CALL   NLPQLF(    MF,     MEF,      M,     MMAX,     N,
  /                   NMAX,    MNN6,      X,        F,     G,
  /                     DF,      DG,      U,       XL,    XU,
  /                    ACC,    ACCF,  ACCQP,  MAXFUN, MAXIT,
  /                   MAXF,    RHOB, IPRINT,    IPRTF,  IOUT,
  /                  IFAIL,  IFAILF,     WA,      LWA,   KWA,
  /                   LKWA,   LOGWA, LLOGWA                  )
```

Definition of the parameters:

MF :	Number of constraints for which feasibility must be restored.
MEF :	Number of equality constraints of these constraints.
M :	Number of remaining feasibility constraints.
MMAX :	Row dimension of GRAD and dimension of G. MMAX must be at least one and not smaller than MF + M.
N :	Number of variables.
NMAX :	Dimensioning parameter, at least two and greater than N + 1.
MNN6 :	Dimensioning parameter, must be set to MF + M + 2*N + 6 or higher when calling NLPQLF.
X(NMAX) :	On input, X has to contain an initial guess for the solution. On return, X is replaced by the last computed iterate.
F	Objective function value $f(x)$ passed to NLPQLF by reverse communication.
G(MMAX) :	Constraint function values passed to NLPQLF by reverse communication in the order $g_1(x)$, …, $g_{m_f}(x)$, $e_1(x)$, …, $e_m(x)$.
DF(NMAX) :	Gradient of objective function.
DG(MMAX,NMAX) :	DG is used to store gradients of the constraints at a current iterate X in the order given above. In the driving program, the row dimension of DG must be equal to MMAX.
U(MNN6) :	On return, U contains the multipliers with respect to the last computed iterate. The first MF + M locations contain the multipliers of the MF + M nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the following N locations the multipliers of the upper bounds. At an optimal solution, all multipliers with respect to inequality constraints must be nonnegative.

XL(NMAX), XU(NMAX):	On input, the one-dimensional arrays XL and XU contain the upper and lower bounds of the variables.
ACC :	The user has to specify the desired final accuracy for the outer optimization loop (e.g. 1.0D-7). The termination accuracy should not be smaller than the accuracy by which gradients are computed.
ACCF :	The user has to specify the desired final accuracy for the SQP solver NLPQLP to achieve feasibility (e.g. 1.0D-7). The termination accuracy should not be smaller than the accuracy by which gradients are computed.
ACCQP :	The tolerance is passed to the QP solver to perform several tests, for example whether optimality conditions are satisfied or whether a number is considered as zero or not.
MAXFUN :	The integer variable defines an upper bound for the number of function calls during the line search (e.g. 20). MAXFUN must not be greater than 50. Used for both levels.
MAXIT :	Maximum number of outer iterations, where one iteration corresponds to one formulation and solution of a nonlinear programming subproblem, or, alternatively, one evaluation of gradients (e.g. 100).
MAXF :	Maximum number of feasibility iterations (e.g. 50).
RHOB :	Parameter for initializing a restart in case of IFAIL=2 by setting the BFGS-update matrix to rhob*I, where I denotes the identity matrix. The number of restarts is bounded by MAXFUN. No restart is performed if RHOB is set to zero. Must be non-negative (e.g. 100).
IPRINT :	Specification of the desired output level. 0 - No output of the program. 1 - Only final convergence analysis. 2 - One line of intermediate results for each iteration. 3 - More detailed information for each iteration. 4 - Line search data displayed in addition.
IPRTF :	Specification of the desired output level for the feasibility cycle, as above.
IOUT :	Integer indicating the desired output unit number, i.e. all write-statements start with 'WRITE(IOUT,... '. Used for both levels.

IFAIL :	The parameter shows the reason for leaving NLPQLF. Initially, IFAIL must be set to zero. On return IFAIL contains one of the following values: -3 - Compute function and gradients of $e_1, \ldots, e_m(x)$. -2 - Compute new gradient values. -1 - Compute new function values. 0 - Optimality conditions satisfied. 1 - Stop after MAXIT iterations. 2 - Uphill search direction. 3 - Underflow when computing new BFGS-update matrix. 4 - Line search exceeded MAXFUN iterations. 5 - Length of a working array too short. 6 - False dimensions, e.g., M<MW, N≥NMAX. 7 - Search direction close to zero at infeasible iterate. 8 - Starting point violates lower or upper bound. 9 - Wrong input parameter, e.g., IPRINT, IOUT. 10 - Inconsistency in QP, division by zero. 11 - Infeasible starting point. >100 - Error message of QP solver. >1000 - Error message of lower level SQP solver, as above.
IFAILF :	The parameter shows the reason for leaving NLPQLF, and especially for terminating the subproblem. Initially, IFAILF must be set to zero. On return IFAILF contains one of the following values: -2 - Compute new gradient values of $e_1, \ldots, e_m(x)$. -1 - Compute new function values of $e_1, \ldots, e_m(x)$.
WA(LWA) :	Real working array of length LWA.
LWA :	Length of the real working array WA. LWA must be at least 7*NMAX*NMAX/2 + NMAX*MMAX + 61*NMAX + 18*MMAX + 270.
KWA(LKWA) :	Integer working array of length LKWA.
LKWA :	Length of the integer working array KWA. LKWA should be at least 2*NMAX + 30. On return, KWA(1) and KWA(2) contain the number of function and derivative evaluations, respectively, and KWA(3) the number of function and derivative evaluations for feasibility restoration.

LOGWA(LLOGWA) :	Logical working array of length LLOGWA.
LLOGWA :	Length of the logical working array LOGWA. LLOGWA must be at least 4*M + 20.

Some of the termination reasons depend on the accuracy used for approximating gradients. If we assume that all functions and gradients are computed within machine precision and that the implementation is correct, there remain only the following possibilities that could cause an error message:

1. The termination parameter ACC is too small, so that the numerical algorithm plays around with round-off errors without being able to improve the solution. Especially the Hessian approximation of the Lagrangian function becomes unstable in this case. A straightforward remedy is to restart the optimization cycle again with a larger stopping tolerance.

2. The constraints are contradicting, i.e., the set of feasible solutions is empty. There is no way to find out, whether a general nonlinear and non-convex set possesses a feasible point or not. Thus, the nonlinear programming algorithms will proceed until running in any of the mentioned error situations. In this case, the correctness of the model must be very carefully checked.

3. Constraints are feasible, but some of them there are degenerate, for example if some of the constraints are redundant. One should know that SQP algorithms assume the satisfaction of the so-called constraint qualification, i.e., that gradients of active constraints are linearly independent at each iterate and in a neighborhood of an optimal solution. In this situation, it is recommended to check the formulation of the model constraints.

However, some of the error situations also occur if, because of wrong or non-accurate gradients, the quadratic programming subproblem does not yield a descent direction for the underlying merit function. In this case, one should try to improve the accuracy of function evaluations, scale the model functions in a proper way, or start the algorithm from other initial values.

5 Example

To give an example how to organize the code, we minimize the Rosenbrock function

$$
x_1, x_2 \in I\!R : \quad \begin{array}{l} \min 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 \\ \;\; x_1 - \log(\beta - x_1^2 - x_2^2 + \epsilon) \geq 0 \;\; , \\ \;\; x_1^2 + x_2^2 \leq \beta \;\; , \\ -10 \leq x_1 \leq 10 \;\; , \\ -10 \leq x_2 \leq 10 \end{array} \tag{24}
$$

over a circle with radius $\beta = 2$ and subject to a logarithmic constraint. We have $m_f = 1$ and $m = 1$, and the feasibility constraint is active. To guarantee that the arguments of the logarithm are bounded away from zero, a small tolerance $\epsilon = 10^{-4}$ is added. Feasible starting point is $x_0 = (-0.1, -0.1)^T$.

The Fortran source code for executing NLPQLF is listed below. The function block inserted in the main program can be replaced by a subroutine call.

```
      IMPLICIT         NONE
      INTEGER          NMAX, MMAX, MNN6X, LWA, LKWA, LLOGWA
      PARAMETER       (NMAX   = 4,
     /                 MMAX   = 2,
     /                 MNN6X  = MMAX + NMAX + NMAX + 6,
     /                 LWA    = 7*NMAX*NMAX/2 + NMAX*MMAX + 61*NMAX
     /                        + 18*MMAX + 270,
     /                 LKWA   = 2*NMAX + 30,
     /                 LLOGWA = 4*MMAX + 20)
      INTEGER          KWA(LKWA), N, MF, MEF, M, MNN6, MAXIT, MAXF,
     /                 MAXFUN, IPRINT, IPRTF, IOUT, IFAIL, IFAILF, I
      DOUBLE PRECISION X(NMAX), G(MMAX), F, DF(NMAX), DG(MMAX,NMAX),
     /                 U(MNN6X), XL(NMAX), XU(NMAX), C(NMAX,NMAX),
     /                 D(NMAX), WA(LWA), ACC, ACCF, ACCQP, RHOB,
     /                 EPS, A, B, DA1, DA2
      LOGICAL          LOGWA(LLOGWA)
      EXTERNAL         QL
C
C   Set some constants and initial values
C
      EPS    = 1.0D-4
      B      = 2.0D+0
      IOUT   = 6
      ACC    = 1.0D-12
      ACCF   = 1.0D-12
      ACCQP  = 1.0D-14
      RHOB   = 0.0D0
      MAXIT  = 100
      MAXFUN = 20
      MAXF   = 50
      IPRINT = 2
      IPRTF  = 0
      N      = 2
      MF     = 1
      MEF    = 0
      M      = 1
      MNN6   = MF + M + N + N + 6
      IFAIL  = 0
      IFAILF = 0
      DO I=1,N
         X(I)  = -0.1D0
         XL(I) = -10.0D0
         XU(I) = 10.0D0
      ENDDO
C
C============================================================
C   This is the main block to compute all function and gradient
C   values.
C
    1 CONTINUE
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-1)) THEN
         F    = 100.0D0*(X(2) - X(1)**2)**2 + (X(1) - 1.0D0)**2
         A    = -X(1)**2 - X(2)**2 + B
         G(1) = X(1) - DLOG(A + EPS)
         G(2) = A
      ENDIF
C
      IF ((IFAIL.EQ.0).OR.(IFAIL.EQ.-2)) THEN
         DF(1)   = -400.0D0*X(1)*(X(2) - X(1)**2)
     /                        + 2.0D0*(X(1) - 1.0D0)
         DF(2)   = 200.0D0*(X(2) - X(1)**2)
         A       = -X(1)**2 - X(2)**2 + B
         DA1     = -2.0D0*X(1)
         DA2     = -2.0D0*X(2)
         DG(1,1) = 1.0D0 - DA1/(A + EPS)
         DG(1,2) = -DA2/(A + EPS)
         DG(2,1) = DA1
         DG(2,2) = DA2
      ENDIF
C
      IF (IFAILF.EQ.-1) THEN
         G(2) = -X(1)**2 - X(2)**2 + B
      ENDIF
C
      IF (IFAILF.EQ.-2) THEN
         DG(2,1) = -2.0D0*X(1)
         DG(2,2) = -2.0D0*X(2)
      ENDIF
C
C============================================================
C
      CALL NLPQLF (MF, MEF, M, MMAX, N, NMAX, MNN6, X, F, G, DF, DG,
     /             U, XL, XU, ACC, ACCF, ACCQP, MAXFUN, MAXIT, MAXF,
     /             RHOB, IPRINT, IPRTF, IOUT, IFAIL, IFAILF,
     /             WA, LWA, KWA, LKWA, LOGWA, LLOGWA, QL)
      IF (IFAIL.LT.0) GOTO 1
C
      STOP
      END
```

The following output will appear on screen:

```
--------------------------------------------------------------------
 START OF THE SQP FEASIBILITY RESTORATION ALGORITHM
--------------------------------------------------------------------
```

```
 Parameters:
    N      =       2
    M      =       2
    ME     =       0
    ACC    =   0.1000D-11
    ACCF   =   0.1000D-11
    ACCQP  =   0.1000D-13
    MAXFUN =      20
    MAXNM  =      40
    MAXIT  =     500
    IPRINT =       2

 Output in the following order:
    IT    - iteration number
    F     - objective function value
    SCV   - sum of constraint violations
    NA    - number of active constraints
    I     - number of line search iterations
    ALPHA - steplength parameter
    DELTA - additional variable to prevent inconsistency
    KKT   - Karush-Kuhn-Tucker optimality criterion

  IT         F          SCV      NA  I    ALPHA     DELTA     KKT
--------------------------------------------------------------------
   1  0.24200000D+01  0.78D+00    2  0  0.00D+00  0.00D+00  0.60D+02
   2  0.72443791D+01  0.11D-14    2  1  0.10D+01  0.00D+00  0.35D+01
   3  0.41623356D+01  0.22D-15    2  1  0.10D+01  0.00D+00  0.27D+01
   4  0.19040508D+01  0.00D+00    2  1  0.10D+01  0.00D+00  0.18D+01
   5  0.50640876D+00  0.14D-12    2  1  0.10D+01  0.00D+00  0.95D+00
   6  0.13288026D-02  0.00D+00    2  1  0.10D+01  0.00D+00  0.27D-02
   7  0.80093028D-06  0.64D-07    1  1  0.10D+01  0.00D+00  0.16D-05
   8  0.37580269D-10  0.00D+00    0  1  0.10D+01  0.00D+00  0.00D+00

  --- Final Convergence Analysis (NLPQLF) at Best Iterate ---

    Best result at iteration:      ITER  =       8
    Objective function value:      F(X)  =  0.37580269D-10
    Solution values:               X     =
       0.99999398D+00  0.99998807D+00
    Multiplier values:             U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
       0.00000000D+00  0.00000000D+00
    Constraint values:             G(X)  =
       0.99036505D+01  0.35891132D-04
    Distance from lower bound:     X-XL  =
       0.10999994D+02  0.10999988D+02
    Distance from upper bound:     XU-X  =
       0.90000060D+01  0.90000119D+01
    Number of function calls:      NFUNC =       8
    Number of gradient calls:      NGRAD =       8
    Average number of feasibility
       iterations:                 NFEAS =       5
```

When replacing the call of NLPQLF by a call of NLPQLP, we would get a negative argument of the log-function after the initial step and a breakdown of the code, see below.

```
--------------------------------------------------------------------
 START OF THE SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
--------------------------------------------------------------------
 Parameters:
    N      =       2
    M      =       2
    ME     =       0
    MODE   =       0
    ACC    =   0.1000D-11
    ACCQP  =   0.1000D-13
    STPMIN =   0.1000D-11
    MAXFUN =      20
    MAXNM  =      40
    MAXIT  =     500
    IPRINT =       2

 Output in the following order:
    IT    - iteration number
    F     - objective function value
    SCV   - sum of constraint violations
    NA    - number of active constraints
    I     - number of line search iterations
    ALPHA - steplength parameter
    DELTA - additional variable to prevent inconsistency
    KKT   - Karush-Kuhn-Tucker optimality criterion

  IT         F          SCV      NA  I    ALPHA     DELTA     KKT
--------------------------------------------------------------------
   1  0.24200000D+01  0.78D+00    2  0  0.00D+00  0.00D+00  0.39D+03
     *** WARNING: Starting non-monotone line search
     *** ERROR 4: Error in line search, e.g., more than MAXFUN iterations

  --- Final Convergence Analysis at Last Iterate ---

    Objective function value:      F(X)  = NaN
    Solution values:               X     =
      NaN             NaN
    Multiplier values:             U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
       0.00000000D+00  0.11900000D+02
    Constraint values:             G(X)  =
      NaN             NaN
    Distance from lower bound:     X-XL  =
      NaN             NaN
    Distance from upper bound:     XU-X  =
      NaN             NaN

  --- Final Convergence Analysis at Best Iterate ---

    Best result at iteration:      ITER  =       1
    Objective function value:      F(X)  =  0.24200000D+01
    Solution values:               X     =
      -0.10000000D+00 -0.10000000D+00
    Multiplier values:             U     =
       0.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
       0.00000000D+00  0.11900000D+02
    Constraint values:             G(X)  =
      -0.78314735D+00  0.19800000D+01
    Distance from lower bound:     X-XL  =
       0.99000000D+01  0.99000000D+01
    Distance from upper bound:     XU-X  =
       0.10100000D+02  0.10100000D+02
    Number of function calls:      NFUNC =      41
    Number of gradient calls:      NGRAD =       1
    Number of calls of QP solver:  NQL   =       1

    Average number of feasibility
```

```
        iterations:                NFEAS =       0
```

6 Conclusions

We present a modification of an SQP algorithm designed for guaranteeing feasible iterates, which always remain in a convex set defined by nonlinear inequality constraints. Thus, objective function values and also additional equality or inequality constraints are evaluated only at points which are feasible subject to these constraints. Preliminary numerical results show superlinear convergence. We have an analytical proof that the sufficient descent direction is obtained, i.e., that the standard convergence properties of SQP algorithms are not lost.

References

[1] Bendsøe M.P., Guedes J.M., Haber R.B., Pedersen P., Taylor J.E. (1994): *An analytical model to predict optimal material properties in the context of optimal structural design*, Hournal of Applied Mechanics, Vol./ 61, 930-937

[2] 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

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

[4] Herkovits J. (1998) *Feasible direction interior-point technique for nonlinear optimization*, Computational Mechanics, New Trends and Applications, Journal of Optimization Theory and Applications, Vol. 99, 121-146

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

[6] Koĉvara M., Zowe J. (2002):*Free material optimization: An overview*, in: Siddiqi A., Koĉvara M. (eds.), *Trends in Industrial and Applied Mathematics*, Kluwer Academish Publishers, Dordrecht, 181-215

[7] Lehmann S., Schittkowski K., Stingl M., Wein F., Zillober C. (2013):*A strictly feasible sequential convex programming method*, submitted for publication

[8] Panier E.R., Tits, A.L. (1993):*On combining feasibility, descent and superlinear convergence in inequality constrained optimization*, Mathematical Programming, Vol. 59, 261-276

[9] 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

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

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

[12] 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

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

[14] Schittkowski K. (1985): *On the global convergence of nonlinear programming algorithms*, ASME Journal of Mechanics, Transmissions, and Automation in Design, Vol. 107, 454-458

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

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

[17] Schittkowski K. (2006): *NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 2.2*, Report, Department of Computer Science, University of Bayreuth, 2006

[18] Schittkowski K. (2008): *An updated set of 306 test problems for nonlinear programming with validated optimal solutions - user's guide*, Report, Department of Computer Science, University of Bayreuth

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

[20] 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