Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 4, No. 2, pp. 335 – 351, May 2008

A SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM WITH NON-MONOTONE LINE SEARCH*

Yu-Hong Dai and Klaus Schittkowski

Abstract: Today, many practical smooth 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. 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 reduce the number of false terminations, a non-monotone line search is proposed which allows the acceptance of a step length even with an increased merit function value. Thus, the subsequent step may become larger than in case of a monotone line search and the whole iteration process is stabilized. Convergence of the new SQP algorithm is proved assuming exact arithmetic, and numerical results are included. As expected, no significant improvements are observed if function values are computed within machine accuracy. To model more realistic and more difficult situations, we add randomly generated errors to function values and show that a drastic improvement of the performance is achieved compared to monotone line search. This situation is very typical for complex simulation programs producing inaccurate function values and where, even worse, derivatives are nevertheless computed by forward differences.

Key words: *SQP, sequential quadratic programming, nonlinear programming, non-monotone line search, merit function, convergence, numerical results*

Mathematics Subject Classification: *65K05, 90C26*

1 Introduction

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

$$\begin{array}{ll} \text{minimize} & f(x) \\ x \in I\!R^n: & g_j(x) = 0 \ , \ j = 1, \ldots, m_e, \\ & g_j(x) \geq 0 \ , \ j = m_e + 1, \ldots, m, \end{array} \tag{1.1}$$

where x is an n-dimensional parameter vector. It is assumed that all problem functions $f(x)$ and $g_j(x)$, $j = 1, \ldots, m$, are continuously differentiable on the $I\!R^n$. Without loss of generality, bound constraints of the form $x_l \leq x \leq x_u$ are dropped to simplify the notation.

Sequential quadratic programming became a highly popular general purpose method to solve smooth nonlinear optimization problems during the last 25 years, at least if the nonlinear program does not possess any special mathematical structure, for example a least squares objective function, large number of variables with sparsity patterns in derivatives, etc.

*This work was supported by the Alexander von Humboldt foundation under grant CHN/1112740 STP.

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.

Our proposal makes use of non-monotone line search. The basic idea is to replace the reference value $\phi_k(0)$ of a line search termination criterion

$$\phi_k(\alpha_k) \le \phi_k(0) + \mu\alpha_k\phi_k'(0) \ ,$$

where $\phi_k(\alpha)$ is a suitable merit function with $\phi_k'(0) < 0$ at the k-th iterate and $\mu > 0$ a tolerance, by $\max\{\phi_j(0) : j = \max(0, k-L), \ldots, k\}$. Thus, we accept larger stepsizes and are able to overcome situations where the quadratic programming subproblem yields insufficient search directions because of inaccurate gradients. If, however, the queue length L is set to 0, we get back the original SQP method with monotone line search.

The proposal is not new and for example described in Dai [5], where a general convergence proof for the unconstrained case is presented. The general idea goes back to Grippo, Lampariello, and Lucidi [11], and was extended to constrained optimization and trust region methods in a series of subsequent papers, see Bonnans et al. [2], Deng et al. [6], Grippo et al. [12, 13], Ke and Han [16], Ke et al. [17], Panier and Tits [19], Raydan [23] and Toint [33]. But there is a basic difference in the methodology: Our numerical results indicate that it is preferable to allow monotone line searches as long as they terminate successfully, and to apply a non-monotone one only in an error situation.

To illustrate the usefulness of the non-monotone line search in an error situation, we give a one-dimensional example here

$$f(x) = \frac{1}{2}x^2, \quad x \in R^1.$$

In the exact arithmetic situation, Newton's method will reach the unique minimizer $x^* = 0$ in one iteration. Now we assume that the calculations of the second derivative is exact and there is a relative noise in the caculations of each gradient component. More exactly, we assume that $f''(x_k) \equiv 1$ and $f'(x_k) = x_k(1+\varepsilon_k)$, where ε_k has the independent normal distribution $N(0, \sigma^2)$ with $\sigma < 1$. In this case, Newton's method (with the unique step length) defines the iterates $\{x_k\}$ by

$$x_{k+1} = x_k - x_k(1+\varepsilon_k) = -\varepsilon_k x_k = \cdots = (-1)^k x_1 \prod_{i=1}^{k} \varepsilon_i.$$

Since the distribution of $\prod_{i=1}^{k} \varepsilon_i$ is $N(0, \sigma^{2k})$, we know by this and $\sigma < 1$ that x_k tends to the solution $x^* = 0$ with probability one. If a non-montone line search is used, the new iterate x_{k+1} will be accepted at a large probability which depends on the line search parameter L. On the contrary, since Newton's direction is not a descent direction providing that $\varepsilon_k \le -1$, a satisfactory point cannot be found by a monotone line search.

It is also important to note that there exists an alternative technique to stabilize an SQP-based nonlinear programming algorithm and to establish global convergence results, the trust region method. The basic idea is to compute a new iterate x_{k+1} by a second order model or any close approximation, where the step size is restricted by a trust region radius. Subsequently, the ratio of the actual and the predicted improvement subject to a merit function is computed. The trust region radius is either enlarged or decreased depending on the deviation from the ideal value one. A comprehensive review on trust region methods is given by Conn, Gould, and Toint [4]. Fletcher [9] introduced a second order correction, for which superlinear convergence can be shown, see also Yuan [34]. Numerical comparisons of Exler and Schittkowski [8] show that the efficiency in terms of number of function and gradient evaluations is comparable to an SQP method with line search.

It is not assumed that information about statistical properties of possible noise is available. Thus, we proceed from the standard version of an SQP algorithm and consider only the question, what happens if we apply this one to inaccurate function and gradient evaluations. On the other hand, there are proposals to exploit existing information and to modify an SQP method accordingly, see e.g. Hintermüller [15].

Numerical results are included to test different line search variants. However, there are nearly no differences of the overall performance in case of providing function and especially gradient values within machine accuracy. The main reason is that the step length one satisfies the termination criterion of a line search algorithm in most steps, especially when approaching a solution, see Schittkowski [26] for a theoretical justification.

Thus, the purpose of the theoretical and numerical investigations of this paper is to show that non-monotone line search is more robust under side conditions which are often satisfied in practical situations. If function values are inaccurate and if in addition derivatives are approximated by a difference formula, standard monotone line search leads to an irregular termination in many situations, where a non-monotone one terminates successfully because of accepting larger steps.

In Section 2, we outline an SQP algorithm, especially the quadratic programming subproblem, and the used merit function. The non-monotone line search and the new SQP algorithm are discussed and some convergence results are given following the analysis of Schittkowski [26] for the monotone case. Section 3 contains some numerical results obtained for a set of 306 standard test problems of the collections published in Hock and Schittkowski [14] and in Schittkowski [27]. They show the stability of the new algorithm with respect to the influence of noise in function evaluations. Conclusions and some discussions about monotone and non-monotone are made at the last section.

2 Analysis of an SQP Algorithm with Non-Monotone Line Search

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 (1.1). The theoretical background is described e.g. in Stoer [32] in form of a review or in Spellucci [31] 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 [20] and Edgar, Himmelblau [7]. Their excellent numerical performance was tested and compared with other methods in Schittkowski [25], 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

$$L(x,u) \doteq f(x) - u^T g(x) \tag{2.1}$$

quadratically, where $x \in \mathbb{R}^n$ is the primal variable, $u \in \mathbb{R}^m$ the dual variable, i.e., the multiplier vector, and where $g(x) = (g_1(x), \ldots, g_m(x))^T$. Assume that $x_k \in \mathbb{R}^n$ is an actual approximation of the solution, $v_k \in \mathbb{R}^m$ 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

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}d^T B_k d + \nabla f(x_k)^T d \\ d \in I\!R^n : & \nabla g_j(x_k)^T d + g_j(x_k) = 0 \ , \quad j \in E, \\ & \nabla g_j(x_k)^T d + g_j(x_k) \geq 0 \ , \quad j \in I \end{array} \tag{2.2}$$

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

$$z_k \doteq \begin{pmatrix} x_k \\ v_k \end{pmatrix} \ , \quad p_k \doteq \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \tag{2.3}$$

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

$$z_{k+1} \doteq z_k + \alpha_k p_k \ , \tag{2.4}$$

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

However, the linear constraints in (2.2) can become inconsistent even if the original problem (1.1) is solvable. As in Powell [21], we add an additional variable δ to (2.2) and solve an $(n + 1)$-dimensional subproblem with consistent constraints.

Another numerical drawback of (2.2) 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 gradients of inactive nonlinear constraints is redundant. Given a constant $\varepsilon > 0$, we define the sets

$$\overline{I}_1^{(k)} = \{j \in I : \ g_j(x_k) \leq \varepsilon \text{ or } v_k^{(j)} > 0\} \ , \quad \overline{I}_2^{(k)} = I \setminus \overline{I}_1^{(k)} \ , \tag{2.5}$$

$v_k = (v_k^{(1)}, \ldots, v_k^{(m)})^T$, and solve the following subproblem in each step,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}d^T B_k d + \nabla f(x_k)^T d + \frac{1}{2}\varrho_k \delta^2 \\ d \in I\!R^n, \delta \in [0, 1] : & \nabla g_j(x_k)^T d + (1-\delta) g_j(x_k) = 0 \ , \quad j \in E, \\ & \nabla g_j(x_k)^T d + (1-\delta) g_j(x_k) \geq 0 \ , \quad j \in \overline{I}_1^{(k)}, \\ & \nabla g_j(x_{\kappa(k,j)})^T d + g_j(x_k) \geq 0 \ , \ j \in \overline{I}_2^{(k)}. \end{array} \tag{2.6}$$

The indices $\kappa(k, j) \leq k$ 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 (2.6), 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 linear independency 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 together with a stabilization to guarantee positive definite matrices, see Powell [21]. The penalty parameter ϱ_k is required to reduce the perturbation of the search direction by the additional variable δ as much as possible. A suitable choice is given in [26].

To enforce global convergence of the SQP method, we have to select a suitable step length α_k, see (2.4), subject to a merit function $\phi_k(\alpha)$. We use the differentiable augmented Lagrange function of Rockafellar [24],

$$\Phi_r(x, v) \doteq f(x) - \sum_{j \in E \cup I_1} (v_j g_j(x) - \frac{1}{2} r_j g_j(x)^2) - \frac{1}{2} \sum_{j \in I_2} v_j^2 / r_j \ , \tag{2.7}$$

with $v = (v_1, \ldots, v_m)^T$, $r = (r_1, \ldots, r_m)^T$, $I_1(x, v, r) \doteq \{j \in I : g_j(x) \leq v_j / r_j\}$ and $I_2(x, v, r) \doteq I \setminus I_1(x, v, r)$, cf. Schittkowski [26]. If there is no confusion, we will just denote $I_1(x, v, r)$ and $I_2(x, v, r)$ by I_1 and I_2, respectively. In addition, to simplify the notation, we mean v_j and r_j by the j-th coefficients of the vectors v and r, respectively, as one can easily tell. The merit function is then defined by

$$\phi_k(\alpha) \doteq \Phi_{r_{k+1}}(z_k + \alpha p_k) \ , \tag{2.8}$$

see also (2.3). To ensure that p_k is a descent direction of $\Phi_{r_{k+1}}(z_k)$, i.e., that

$$\phi_k'(0) = \nabla \Phi_{r_{k+1}}(z_k)^T p_k < 0 \ , \tag{2.9}$$

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

$$r_{k+1}^{(j)} \doteq \max \left(\sigma_k^{(j)} r_k^{(j)}, \ \frac{2m(u_k^{(j)} - v_k^{(j)})}{(1 - \delta_k) d_k^T B_k d_k} \right) \tag{2.10}$$

with $u_k = (u_k^{(1)}, \ldots, u_k^{(m)})^T$, $v_k = (v_k^{(1)}, \ldots, v_k^{(m)})^T$ and $j = 1, \ldots, m$. The sequence $\{\sigma_k^{(j)}\}$ is introduced to allow decreasing penalty parameters at least in the beginning of the algorithm by assuming that $\sigma_k^{(j)} \leq 1$. A sufficient condition to guarantee convergence of $\{r_k^{(j)}\}$ is that there exists a positive constant ζ with

$$\sum_{k=0}^{\infty} \left[1 - (\sigma_k^{(j)})^{\zeta} \right] < \infty \tag{2.11}$$

for $j = 1, \ldots, m$. The above condition is somewhat weaker than the one in [26] obtained for $\zeta = 1$.

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

$$\phi_k(\alpha_k) \leq \phi_k(0) + \mu \alpha_k \phi_k'(0) \ , \tag{2.12}$$

see for example Ortega and Rheinboldt [18], 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 chosen between 0 and 0.5.

However, practical experience shows that monotonicity requirement (2.12) 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 (2.12), we accept a stepsize α_k as soon as the inequality

$$\phi_k(\alpha_k) \leq \max_{k - l(k) \leq j \leq k} \phi_j(0) + \mu \alpha_k \phi_k'(0) \tag{2.13}$$

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 (2.12).

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.

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

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

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

1) If
$$\phi_k(\alpha_{k,i}) \leq \max_{k-l(k) \leq j \leq k} \phi_j(0) + \mu \, \alpha_{k,i} \, \phi_k'(0) \ , \tag{2.14}$$
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 [26]. $\bar{\alpha}_{k,i}$ is the minimizer of the quadratic interpolation and we use a relaxed Armijo-type descent property for checking termination. Step 3) is required to prevent too small step sizes, see above. 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.

To prove the global convergence of the SQP algorithm, i.e., the approximation of a Karush-Kuhn-Tucker (KKT) point of (1.1) starting from an arbitrary $x_0 \in \mathbb{R}^n$, we closely follow the analysis of Schittkowski [26] for a monotone line search. We assume throughout this section that the linear independency constraint qualification is satisfied at all Karush-Kuhn-Tucker points of the nonlinear program (1.1), and at all iterates of the SQP algorithm. This is a standard assumption for proving global and local convergence theorems and serves to guarantee that the multipliers of the quadratic subproblems are unique and remain bounded.

An important general assumption is that the feasible domain of the nonlinear program (1.1) is bounded. However, we dropped additional bounds of the form

$$x_l \leq x \leq x_u$$

from (1.1) only to simplify the notation. They can be added to all practical optimization problems without loss of generality, and are also included in corresponding implementations of SQP methods. Since bounds are transformed directly to bounds of the quadratic programming subproblem (2.6), subsequent iterates will also satisfy them and, moreover, the subproblem is always uniquely solvable.

The subsequent theorem will be fundamental for the convergence analysis, which is taken from [26] and which does not depend on the new line search procedure. It shows that the search direction computed from (2.6) is a descent direction of the merit function $\phi_k(\alpha)$, i.e. that $\phi_k'(0) < 0$, and that therefore the line search is well-defined. Moreover, it is possible to show that there is a sufficiently large decrease of the merit function from which the global convergence can be derived.

Theorem 2.2. *Let x_k, v_k, d_k, δ_k, u_k, B_k, r_k, ϱ_k, and $\bar{I}_1^{(k)}$ be given iterates of the SQP algorithm under consideration, $k \geq 0$, and assume that there are positive constants γ and $\bar{\delta}$ with*

(i) $d_k^T B_k d_k \geq \gamma \|d_k\|^2$ *for some $\gamma \in (0, 1]$ and all k,*

(ii) $\delta_k \leq \bar{\delta} < 1$ *for all k,*

(iii) $\varrho_k \geq \dfrac{1}{\gamma(1-\bar{\delta})^2} \| \sum_{j \in \bar{I}_1^{(k)}} v_k^{(j)} \nabla g_j(x_k) \|^2$ *for all k.*

Then

$$\phi_k'(0) = \nabla \Phi_{r_{k+1}}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \leq -\frac{1}{4} \gamma \|d_k\|^2 \quad . \tag{2.15}$$

Any properties of a quasi-Newton update formula for B_k are not exploited to get the sufficient decrease property. The only requirement for the choice of B_k is that the eigenvalues of this positive definite matrix remain bounded away from zero. In the extreme case, $B_k = I$ and $\gamma = 1$ satisfy *(i)*.

It is shown in Lemma 4.4 of [26] that

$$\alpha_{k,i+1} \leq \max\left(\beta, \frac{1}{2(1-\mu)}\right) \alpha_{k,i} \tag{2.16}$$

for $\phi_k'(0) < 0$ and an iteration sequence $\{\alpha_{k,i}\}$ of the line search algorithm, whenever (2.14) is not valid for an $i \geq 0$. Since $\alpha_{k,i} \to 0$ for $i \to \infty$ and $\phi_k'(0) < 0$ is impossible without violating (2.14), we get also the finite termination of the line search procedure 2.1.

Next, the convergence of the penalty parameters r_k is shown, see also [26].

Lemma 2.3. *Assume that $\{r_k^{(j)}\}_{k \in N}$ is bounded and $\sigma_k^{(j)} \leq 1$ for all k. If (2.11) holds for a $\zeta > 0$, there is a $r_*^{(j)} \geq 0$, $j = 1, \ldots, m$, with*

$$\lim_{k \to \infty} r_k^{(j)} = r_*^{(j)} \quad .$$

Proof. Let R be an upper bound of the penalty parameters. Assume that there are two different accumulation points $r_*^{(j)}$ and $r_{**}^{(j)}$ of $\{r_k^{(j)}\}$ with $r_*^{(j)} < r_{**}^{(j)}$. Then for $\varepsilon = \frac{1}{3}[(r_{**}^{(j)})^\zeta - (r_*^{(j)})^\zeta] > 0$ there exist infinitely many indices k and $k + q_k$ with

$$|(r_{k+q_k}^{(j)})^\zeta - (r_*^{(j)})^\zeta| \leq \varepsilon, \quad |(r_k^{(j)})^\zeta - (r_{**}^{(j)})^\zeta| \leq \varepsilon \quad .$$

It follows that

$$0 < \varepsilon = (r_{**}^{(j)})^\zeta - (r_*^{(j)})^\zeta - 2\varepsilon \leq -[(r_{k+q_k}^{(j)})^\zeta - (r_k^{(j)})^\zeta] \leq R \sum_{i=0}^{q_k - 1} [1 - (\sigma_{k+i}^{(j)})^\zeta] \quad .$$

Since the above inequality is valid for infinitely many k and the right-hand side tends to zero, we get a contradiction. $\square$

The subsequent lemma shows a certain continuity property of the merit function subject to the penalty parameters.

Lemma 2.4. *Assume that $\Omega \in I\!R^{m+n}$ is a compact subset and $r_j \geq c$ for a positive constant c and $j \in E \cup I$. For any $\varepsilon > 0$, there exists a $\xi > 0$ such that if $|r_j - \tilde{r}_j| < \xi$ for $j \in E \cup I$, then*

$$|\Phi_r(x,v) - \Phi_{\tilde{r}}(x,v)| \leq \varepsilon \quad \textit{for all } (x,\, v) \in \Omega. \tag{2.17}$$

Proof. Let $m > m_e$ without loss of generality. Since Ω is a compact subset and all $g_j(x)$ is continuous differentiable, there exists $M > 0$ such that

$$|g_j(x)| \leq M, \quad |v_j| \leq M, \quad \text{for all } j \in E \cup I \text{ and } (x,v) \in \Omega. \tag{2.18}$$

Denote $I_1 = I_1(x,v,r)$, $I_2 = I_2(x,v,r)$, $\tilde{I}_1 = I_1(x,v,\tilde{r})$ and $\tilde{I}_2 = I_2(x,v,\tilde{r})$. For any ε, we know from (2.18) and the assumption that there exists $\xi > 0$ such that if $|r_j - \tilde{r}_j| < \xi$ for $j \in E \cup I$. Then

$$\Delta_1 \doteq \left| \Phi_r(x,v) - \left[f(x) - \sum_{j \in E \cup I_1} (v_j g_j(x) - \frac{1}{2} \tilde{r}_j g_j(x)^2) - \frac{1}{2} \sum_{j \in I_2} v_j^2 / \tilde{r}_j \right] \right| \leq \frac{1}{2} \varepsilon \tag{2.19}$$

and

$$\left| \frac{1}{r_j} - \frac{\tilde{r}_j}{2\, r_j^2} - \frac{1}{2\, \tilde{r}_j} \right| \leq \frac{\varepsilon}{2(m-m_e)R^2} \quad . \tag{2.20}$$

By noting that $I_1 \backslash \tilde{I}_1 = \tilde{I}_2 \backslash I_2$ and $\tilde{I}_1 \backslash I_1 = I_2 \backslash \tilde{I}_2$, we get

$$\begin{aligned}
\Delta_2 \quad &\doteq \quad \left| \Phi_{\tilde{r}}(x,v) - \left[f(x) - \sum_{j \in E \cup I_1} (v_j g_j(x) - \frac{1}{2} \tilde{r}_j g_j(x)^2) - \frac{1}{2} \sum_{j \in I_2} v_j^2 / \tilde{r}_j \right] \right| \\
&= \quad \left[\sum_{j \in I_1 \backslash \tilde{I}_1} + \sum_{j \in \tilde{I}_1 \backslash I_1} \right] \left| v_j g_j(x) - \frac{1}{2} \tilde{r}_j g_j(x)^2 - \frac{1}{2} \frac{v_j^2}{\tilde{r}_j} \right| \\
&\leq \quad \left[\sum_{j \in I_1 \backslash \tilde{I}_1} + \sum_{j \in \tilde{I}_1 \backslash I_1} \right] v_j^2 \left| \frac{1}{r_j} - \frac{\tilde{r}_j}{2\, r_j^2} - \frac{1}{2\, \tilde{r}_j} \right| \\
&\leq \quad (m-m_e) R^2 \left| \frac{1}{r_j} - \frac{\tilde{r}_j}{2\, r_j^2} - \frac{1}{2\, \tilde{r}_j} \right| \\
&\leq \quad \frac{1}{2} \varepsilon \quad .
\end{aligned} \tag{2.21}$$

The first inequality (2.21) uses the fact that $\left| v_j g_j(x) - \frac{1}{2} \tilde{r}_j g_j(x)^2 - \frac{1}{2} \frac{v_j^2}{\tilde{r}_j} \right|$ achieves its maximum at $g_j(x) = v_j / r_j$ for any $j \in (I_1 \backslash \tilde{I}_1) \cup (\tilde{I}_1 \backslash I_1)$. Combining (2.19) and (2.21), we obtain

$$|\Phi_r(x,v) - \Phi_{\tilde{r}}(x,v)| \leq \Delta_1 + \Delta_2 \leq \varepsilon \quad , \tag{2.22}$$

which completes our proof. $\square$

The non-monotone line search makes use of a bounded length of the queue of known merit function values. Basically, the situation can be illustrated by the subsequent lemma from which a contradiction is later derived.

Lemma 2.5. *For any constant $\varepsilon > 0$ and a positive integer L, consider a sequence $\{s_k : k = 0, 1, 2, \ldots\}$ of real numbers satisfying*

$$s_{k+1} \leq \max_{k-L \leq i \leq k} s_i - \varepsilon \ , \quad \textit{for all } k \geq L. \tag{2.23}$$

Then s_k tends to $-\infty$ as $k \to \infty$.

Proof. We show by induction that

$$s_{k+j} \leq \max_{k-L \leq i \leq k} s_i - \varepsilon \tag{2.24}$$

for all $k \geq L$ holds, $j \geq 0$. (2.23) implies that (2.24) holds with $j = 0$. Assume that (2.24) is true for $j = 1, \ldots, j_0$. Then by (2.23) with k replaced by $k + j_0$ and the induction assumption, we get

$$\begin{aligned} s_{k+j_0+1} &\leq \max_{k+j_0-L \leq i \leq k+j_0} s_i - \varepsilon \\ &\leq \max \left\{ \max_{k \leq i \leq k+j_0} s_i, \max_{k-L \leq i \leq k} s_i \right\} - \varepsilon \\ &\leq \max_{k-L \leq i \leq k} s_i - \varepsilon \ . \end{aligned} \tag{2.25}$$

Thus, (2.24) is true with $j = j_0 + 1$. By induction, we know that (2.24) holds for all $j \geq 0$. Let $\psi(j) \doteq \max\{s_i : jL \leq i \leq (j+1)L\}$ for all $j \geq 0$. By (2.24) and the definition of $\psi(j)$, we conclude that

$$\psi(j+1) \leq \psi(j) - \varepsilon \tag{2.26}$$

holds and that

$$\psi(j) \leq \psi(0) - j\,\varepsilon \quad \text{for all } j \geq 1, \tag{2.27}$$

which implies that $\psi(j)$ and hence s_k tend to $-\infty$. $\square$

Now we prove the following main convergence result, a generalization of Theorem 4.6 in [26].

Theorem 2.6. *Let x_k, v_k, d_k, δ_k, u_k, B_k, r_k, ϱ_k, and $\bar{I}_1^{(k)}$ be given iterates of the SQP algorithm, $k \geq 0$. Assume that there are positive constants γ and $\bar{\delta}$ with*

(i) $d_k^T B_k d_k \geq \gamma \|d_k\|^2$ *for all k,*

(ii) $\delta_k \leq \bar{\delta} < 1$ *for all k,*

(iii) $\varrho_k \geq \dfrac{1}{\gamma(1-\bar{\delta})^2} \| \sum_{j \in \bar{I}_1^{(k)}} v_k^{(j)} \nabla g_j(x_k) \|^2$ *for all k,*

(iv) $\{x_k\}$, $\{d_k\}$, $\{u_k\}$, *and $\{B_k\}$ are bounded.*

Then for any small $\varepsilon > 0$ there exists a $k \geq 0$ with

a) $\|d_k\| \leq \varepsilon$,

b) $\|R_{k+1}^{-1/2}(u_k - v_k)\| \leq \varepsilon$.

The assumptions are not restrictive at all. Since upper and lower bounds can be added without loss of generality, all iterates x_k remain bounded, thus also all d_k, and, because of the constraint qualification, also all multipliers u_k. This condition also guarantees that the additional variables δ_k introduced to avoid inconsistent linearized constraints, remain bounded away from one. Assumption *(iii)* is satisfied by choosing a sufficiently large penalty factor ϱ_k.

Proof. First note that the boundedness of $\{u_k\}$ implies the boundedness of $\{v_k\}$, since $\alpha_k \le 1$ for all k. To show a), let us assume that there is an $\varepsilon > 0$ with

$$\|d_k\| \ge \varepsilon \tag{2.28}$$

for all k. From the definition of r_{k+1}, $k > 0$, we obtain either $r_{k+1}^{(j)} \le \sigma_0^{(j)} r_0^{(j)}$ or

$$r_{k+1}^{(j)} \le \frac{2m(u_{k^*}^{(j)} - v_{k^*}^{(j)})^2}{(1-\delta_{k^*}) d_{k^*}^T B_{k^*} d_{k^*}} \le \frac{2m(u_{k^*}^{(j)} - v_{k^*}^{(j)})^2}{(1-\delta)\,\gamma\,\varepsilon^2}$$

for some $k^* \le k$, $j = 1, ..., m$. Since u_k and therefore also v_k are bounded, we conclude that $\{r_k\}$ remains bounded and Lemma 2.3 implies that there is some $r > 0$ with

$$\lim_{k\to\infty} r_k = r \quad . \tag{2.29}$$

Now consider an iteration index k and introduce again the compound vectors z_k for the iterates and p_k for the search direction, see (2.3). Then by Theorem 2.2,

$$\begin{aligned}
\Phi_{r_{k+1}}(z_{k+1}) &\le \max_{k-l(k)\le i\le k} \Phi_{r_{i+1}}(z_i) + \mu\alpha_k \nabla\Phi_{r_{k+1}}(z_k)^T p_k \\
&\le \max_{k-l(k)\le i\le k} \Phi_{r_{i+1}}(z_i) - \frac{1}{4}\mu\alpha_k\gamma\|d_k\|^2 \\
&< \max_{k-l(k)\le i\le k} \Phi_{r_{i+1}}(z_i) - \frac{1}{4}\mu\gamma\varepsilon^2\alpha_k \quad .
\end{aligned} \tag{2.30}$$

Next we have to prove that α_k cannot tend to zero. Since all functions defining Φ_r are continuously differentiable, r_{k+1} is bounded, and z_k, p_k remain in a compact subset of $\mathbb{R}^{n+m}$, we can find an $\bar\alpha > 0$ with

$$\begin{aligned}
|\nabla\Phi_{r_{k+1}}(z_k+\alpha p_k)^T p_k - \nabla\Phi_{r_{k+1}}(z_k)^T p_k| &\le \|\nabla\Phi_{r_{k+1}}(z_k+\alpha p_k) - \nabla\Phi_{r_{k+1}}(z_k)\|\|p_k\| \\
&\le \tfrac{1}{4}\,(1-\mu)\,\gamma\,\varepsilon^2
\end{aligned} \tag{2.31}$$

for all $\alpha \le \bar\alpha$ and for all k. Using the mean value theorem, (2.31), Theorem 2.2 and (2.28), we obtain for all $\alpha \le \bar\alpha$ and $k \ge 0$

$$\begin{aligned}
&\Phi_{r_{k+1}}(z_k+\alpha p_k) - \max\nolimits_{k-l(k)\le i\le k} \Phi_{r_{i+1}}(z_i) - \mu\alpha\nabla\Phi_{r_{k+1}}(z_k)^T p_k \\
&\quad \le \Phi_{r_{k+1}}(z_k+\alpha p_k) - \Phi_{r_{k+1}}(z_k) - \mu\alpha\nabla\Phi_{r_{k+1}}(z_k)^T p_k \\
&\quad = \alpha\nabla\Phi_{r_{k+1}}(z_k+\xi_k\alpha p_k)^T p_k - \mu\alpha\nabla\Phi_{r_{k+1}}(z_k)^T p_k \\
&\quad \le \alpha\nabla\Phi_{r_{k+1}}(z_k)^T p_k + \tfrac{1}{4}\,\alpha\,(1-\mu)\,\gamma\,\varepsilon^2 - \mu\alpha\nabla\Phi_{r_{k+1}}(z_k)^T p_k \\
&\quad \le -\tfrac{1}{4}\,\alpha(1-\mu)\,\gamma\,\|d_k\|^2 + \tfrac{1}{4}\,\alpha\,(1-\mu)\,\gamma\,\varepsilon^2 \\
&\quad \le 0 \quad .
\end{aligned} \tag{2.32}$$

In the above equation, $\xi_k \in [0,1)$ depends on k. The line search algorithm guarantees that

$$\alpha_{k,i_k-1} \geq \bar{\alpha} \ ,$$

since otherwise α_{k,i_k-1} would have satisfied the stopping condition (2.13). Furthermore

$$\alpha_k = \alpha_{k,i_k} \geq \beta\alpha_{k,i_k-1} > \beta\bar{\alpha} \ .$$

It follows from (2.30) that

$$\Phi_{r_{k+1}}(z_{k+1}) \leq \max_{k-l(k)\leq i\leq k} \Phi_{r_{i+1}}(z_i) - 2\,\bar{\varepsilon} \tag{2.33}$$

for $\varepsilon \doteq \frac{1}{8}\mu\gamma\varepsilon^2\beta\bar{\alpha}$. Now we consider the difference $\Phi_{r_{k+2}}(z_{k+1}) - \Phi_{r_{k+1}}(z_{k+1})$. Since $r_{k+1} \to r^* > 0$ as $k \to \infty$, $g_j(x_k)$ and v_k are bounded, we know by Lemma 2.4 that there exists some integer k_0 such that

$$\Phi_{r_{k+2}}(z_{k+1}) - \Phi_{r_{k+1}}(z_{k+1}) \leq \bar{\varepsilon} \quad \text{for all } k \geq k_0. \tag{2.34}$$

By (2.33), (2.34) and $l(k) \leq L$, we obtain

$$\Phi_{r_{k+2}}(z_{k+1}) \leq \max_{k-L\leq i\leq k} \Phi_{r_{i+1}}(z_i) - \bar{\varepsilon} \tag{2.35}$$

for all sufficiently large k. Thus, we conclude from Lemma 2.5 that $\Phi_{r_{k+1}}(z_k)$ tends to $-\infty$. This is a contradiction to the fact that $\{\Phi_{r_{k+1}}(z_k)\}$ is bounded below and proves statement a). Statement b) follows from the corresponding proof for the monotone case, see [26]. □

A direct conclusion is under the assumptions of Theorem 2.6 there exists an accumulation point $(x^*,\, u^*)$ of $\{(x_k,\, u_k)\}$ satisfying the Karush-Kuhn-Tucker conditions for problem (1.1).

3 Numerical Results

We add now lower and upper bounds to the nonlinear program (1.1) as was implicitly assumed in the previous section for getting bounded iterates,

$$\begin{array}{ll} \text{minimize} & f(x) \\ x \in \mathbb{R}^n: & g_j(x) = 0 \ , \ j = 1,\ldots,m_e, \\ & g_j(x) \geq 0 \ , \ j = m_e+1,\ldots,m, \\ & x_l \leq x \leq x_u \ . \end{array} \tag{3.1}$$

Our numerical tests use the 306 academic and real-life test problems published in Hock and Schittkowski [14] and in Schittkowski [27]. Part of them are also available in the CUTE library, see Bongartz et al [3], and their usage is described in Schittkowski [28]. The test problems represent many possible difficulties observed in practice, ill-conditioning, badly scaled variables and functions, violated constraint qualification, numerical noise, non-smooth functions, or multiple local minima. All examples are provided with exact solutions, either known from analytical solutions or from the best numerical data found so far. However, since most problems are non-convex, we only know of the existence of one local minimizer. Thus, the SQP code might terminate at a better local minimizer without knowing whether this is a global one or not.

The goal is to test different line search variants of the SQP algorithm under an evaluation scheme which is as close to practical situations as possible. Thus, we approximate derivatives numerically by simple forward differences, although analytical derivatives for most test problems are available. Real-life applications often lead to very noisy or inaccurate function values, which even deteriorate the accuracy by which gradients are computed. Gradients are approximated by forward differences

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{\eta_i}\Big(f(x+\eta_i e_i) - f(x) \Big) \ . \tag{3.2}$$

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 is set to $\eta = {\eta_m}^{1/2}$, where η_m is a guess for the accuracy by which function values are computed, i.e., either machine accuracy or an estimate of the noise 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 [30]. 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 [10] based on numerically stable orthogonal decompositions, see also Powell [22] and Schittkowksi [29]. NLPQLP is executed with termination accuracy 10^{-7} and the maximum number of iterations is 500. The number of line search steps is limited by 15 and the stopping condition (2.12) uses $\mu = 0.1$.

The Fortran codes are compiled by the Intel Visual Fortran Compiler, Version 9.0, under Windows XP64 and are executed on an AMD Opteron 64 bit with 4 MB memory. Total calculation time for solving all test problems 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 $x^\star$ a known local solution. 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

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

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.

In the subsequent tables, we use the notation

n_{succ}	:	number of successful test runs (according to above definition)
n_{func}	:	average number of function evaluations for successful test runs
n_{grad}	:	average number of gradient evaluations for successful test runs

One gradient computation corresponds to one iteration of the SQP method. The average numbers of function and gradient evaluations are computed only for the successful test runs. To test the stability of these formulae, we add some randomly generated noise to function values in the following way. A uniformly distributed random number $r \in (0, 1)$ and a given error level ε_{err} are used to perturb function values by the factor $1 + \varepsilon_{err}(1 - 2r)$. Non-monotone line search is applied with a queue size of $L = 30$, but two different strategies, and the line search calculations of Algorithm 2.1 are required. We test the following three situations:

(i) We let $l(k) = 0$ for all iterations, i.e. for all k. This corresponds to the standard monotone line search.

(ii) We define $l(k) = L$ for all iterations and get a non-monotone line search with fixed queue length.

(iii) We let $l(k) = 0$ for all iterations as long as the monotone line search terminates successfully. In case of an error, we apply the non-monotone line search with fixed queue length $l(k) = L$.

version		$\eta = {\eta_m}^{1/2}$			$\eta = 10^{-7}$		
	ε_{err}	n_{succ}	n_{func}	n_{grad}	n_{succ}	n_{func}	n_{grad}
(i)	0	301	31	20	301	31	20
	10^{-12}	298	37	21	300	40	22
	10^{-10}	296	41	21	281	52	22
	10^{-8}	279	47	20	205	56	20
	10^{-6}	253	52	18	45	58	9
	10^{-4}	208	54	15	15	52	6
	10^{-2}	97	52	12	18	49	5
(ii)	0	300	29	23	301	29	23
	10^{-12}	296	31	25	300	40	33
	10^{-10}	299	42	32	295	105	90
	10^{-8}	296	57	48	253	151	128
	10^{-6}	296	107	75	118	234	156
	10^{-4}	284	137	103	96	314	113
	10^{-2}	252	154	116	71	212	61
(iii)	0	303	33	20	303	69	22
	10^{-12}	301	60	23	302	53	26
	10^{-10}	300	63	24	295	94	32
	10^{-8}	300	80	26	274	136	28
	10^{-6}	293	110	28	151	222	17
	10^{-4}	280	138	27	132	324	17
	10^{-2}	237	167	23	108	360	18

Table 1: Test Results

Table 1 shows the corresponding results for the increasing random perturbations defined by 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 otherwise. The last three columns show numerical results obtained for a fixed tolerance $\eta = 10^{-7}$ for the forward difference formula (3.2).

The results are quite surprising and depend heavily on the new non-monotone line search strategy. First we observe that even in case of accurate function values, the non-monotone line search with a fixed $l(k) = L$ requires a larger number of iterations. With increasing noise, the stability is increased by cost of an higher number of iterations. On the other hand, the flexible strategy to use non-monotone line search only in case of false termination of the monotone one, is as efficient and reliable as the pure monotone line search in case of accurate function values, but much more problems can successfully be solved in case of random noise. We are able to solve 77 % of the test examples even in case of extremely noisy function values with at most two correct digits, where only one digit of the gradient values is correct.

The choice of a fixed tolerance η for gradient approximations, i.e., $\eta = 10^{-7}$, is an unlikely worst-case scenario and should only happen in 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 terminate in about 35 % of all test runs, see Table 1.

In case of an increasing number of false terminations, we observe a reduction of the average number of iterations because of the fact that only the 'simple' problems could successfully be solved. When comparing the number of function calls to the number of iterations, we see that more and more line search steps are needed.

4 Conclusions and Discussions

We present a modification of an SQP algorithm to increase its stability in case of noisy function values. Numerical tests favor a version where traditional monotone line search is applied as long as possible, and to switch to a non-monotone one only in case of false termination. Efficiency and robustness is evaluated over a set of 306 standard test problems. To represent typical practical environments, gradients are approximated by forward differences. 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 in function and one correct digit in gradient values.

The non-monotone technique is often used to design optimization algorithms. For descent methods, the introduction of the non-monotone technique significantly improves the original monotone algorithm even for highly nonlinear functions, see e.g. Grippo, Lampariello and Lucidi [11, 12, 13] and Toint [33]. A careful implementation of the non-monotone line search is indispensable in these situations. For some optimization methods like the Barzilai-Borwein gradient method and the SQP algorithm based on the L_1 merit function, a descent direction is not guaranteed in each iteration, and usage of a non-monotone line search is mandatory, see Raydan [23] and Panier and Tits [19]).

In this paper, we found another motivation to investigate non-monotone line search, the minimization of noisy functions. If the monotone line search fails, the algorithm is often able to continue and to find an acceptable solution. However, when trying to apply the non-monotone line search from the beginning, reliability and efficiency become worse.

Our theoretical convergence results assume that there is no noise and they are deducted from existing ones based on sufficient decrease of a merit function. It is an open question whether we could get the same convergence results by taking random perturbations into account for the theoretical analysis.

Acknowledgements

The authors are very grateful to the referee for his many useful comments on this paper.

References

[1] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, *Pacific J. Math.* l (1966) 1–3.

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

[3] I. Bongartz, A.R. Conn, N. Gould and P.h. Toint, CUTE: Constrained and unconstrained testing environment, *Trans. Math. Soft.* 21 (1995) 123–160.

[4] A.R. Conn, I.M. Gould and P.L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, 2000.

[5] Y.H. Dai, On the nonmonotone line search, *J. Optim. Theory Appl.* 112 (2002) 315–330.

[6] N.Y. Deng, Y. Xiao and F.J. Zhou, Nonmonotonic trust-region algorithm, *J. Optim. Theory Appl.* 26 (1993) 259–285.

[7] T.F. Edgar and D.M. Himmelblau, *Optimization of Chemical Processes*, McGraw Hill, New York, 1988.

[8] O. Exler and K. Schittkowski, MISQP: A Fortran implementation of a trust region SQP algorithm for mixed-integer nonlinear programming - user's guide, version 2.1, Report, Department of Computer Science, University of Bayreuth, 2006.

[9] R. Fletcher, Second order correction for nondifferentiable optimization, in *Numerical analysis*, G.A. Watson (ed.), Springer Verlag, Berlin, 1982, pp. 85–114.

[10] D. Goldfarb and A. Idnani, A numerically stable method for solving strictly convex quadratic programs, *Math. Program.* 27 (1983) 1–33.

[11] L. Grippo, F. Lampariello and S. Lucidi, A nonmonotone line search technique for Newtons's method, *SIAM J. Numer. Anal.* 23 (1986) 707–716.

[12] L. Grippo, F. Lampariello and S. Lucidi, A truncated Newton method with nonmonotone line search for unconstrained optimization, *J. Optim. Theory Appl.* 60 (1989) 401–419.

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

[14] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer-Verlag, Berlin 1981.

[15] M. Hintermüller, Solving nonlienar programming problems with noisy function values and noisy gradients, *J. Optim. Theory Appl.* 114 (2002) 133–169.

[16] X. Ke and J. Han, A nonmonotone trust region algorithm for equality constrained optimization, *Science in China* 38A (1995) 683–695.

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

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

[19] E. Panier and A. Tits, Avoiding the Maratos effect by means of a nonmonotone line search, I: General constrained problems, *SIAM J. Numer. Anal.* 28 (1991) 1183–1195.

[20] P.Y. Papalambros and D.J. Wilde, *Principles of Optimal Design*, Cambridge University Press, 1988.

[21] M.J.D. Powell, A fast algorithm for nonlinearly constrained optimization calculations, in *Numerical Analysis*, G.A. Watson (ed.), Lecture Notes in Mathematics, Vol. 630, Springer, 1978, pp. 144–157.

[22] M.J.D. Powell, On the quadratic programming algorithm of Goldfarb and Idnani, Report DAMTP 1983/NA 19, University of Cambridge, Cambridge, 1983.

[23] M. Raydan, The Barzilai and Borwein gradient method for the large-scale unconstrained minimization problem, *SIAM J. Optim.* 7 (1997) 26–33.

[24] R.T. Rockafellar, Augmented Lagrange multiplier functions and duality in non-convex programming, *SIAM J. Control* 12 (1974) 268–285.

[25] K. Schittkowski, *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183, Springer-Verlag, Berlin, 1980.

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

[27] K. Schittkowski, *More Test Examples for Nonlinear Programming*, Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer-Verlag, Berlin, 1987.

[28] K. Schittkowski, Test problems for nonlinear programming - user's guide, Report, Department of Mathematics, University of Bayreuth, 2002.

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

[30] K. Schittkowski, 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.

[31] P. Spellucci, *Numerische Verfahren der nichtlinearen Optimierung*, Birkhäuser Verlag, Basel, 1993.

[32] J. Stoer, 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-Verlag, Berlin, 1985, pp. 165–208.

[33] P.L. Toint, An assessment of nonmontone line search techniques for unconstrained optimization, *SIAM J. Sci. Comp.* 17 (1996) 725–739.

[34] Y.-X. Yuan, On the superlinear convergence of a trust region algorithm for nonsmooth optimization, *Math. Program.* 31 (1985) 269–285.

Manuscript received 11 December 2007
revised 24 December 2007
accepted for publication 25 December 2007

Yu-Hong Dai
State Key Laboratory of Scientific and Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
P. O. Box 2719, Beijing 100190, P. R. China
E-mail address: `dyh@lsec.cc.ac.cn`

Klaus Schittkowski
Department of Computer Science, University of Bayreuth
D - 95440 Bayreuth, Germany
E-mail address: `klaus.schittkowski@uni-bayreuth.de`