Journal of the Operations Research
Society of Japan
2005, Vol. 48, No. 1, 41-56

GLOBAL CONVERGENCE OF A TRUST REGION SEQUENTIAL QUADRATIC PROGRAMMING METHOD

Hiroshi Yamashita Hiroshige Dan
Mathematical Systems, Inc.

(Received February 2, 2004; Revised June 25, 2004)

Abstract In this paper we propose a trust region sequential quadratic programming (SQP) method to solve large-scale nonlinear optimization problems. The main shortcoming of the ordinary trust region SQP method is that the QP subproblem with the trust region constraint may not be feasible when a radius of the trust region is small. The trust region SQP methods which have been proposed so far are so complicated to resolve this shortcoming. It is not desirable in view of implementation and computational time. Moreover, many of the previous trust region SQP methods have another difficulty to solve the QP subproblem which is not necessarily convex. In this paper, we propose a new trust region SQP method which eliminates these two shortcomings. In our method, we solve two types of subproblem that one is a convex QP problem and the other is a system of linear equations.

Keywords: Nonlinear programming, sequential quadratic programming method, trust region, large-scale problem

1. Introduction

The purpose of this paper is to propose a method to solve the following nonlinear optimization problem with equality and inequality constraints:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_j(x) = 0 \ (j \in J_E), \\ & g_j(x) \geq 0 \ (j \in J_I), \end{array} \tag{1.1}$$

where $f : \Re^n \to \Re^1$, $g_j : \Re^n \to \Re^1$ $(j \in J_E \cup J_I)$ and $J_E \cap J_I = \emptyset$. Throughout this paper, we assume that f and g_j $(j \in J_E \cup J_I)$ are twice continuously differentiable.

In this paper, we use the sequential quadratic programming (SQP) method [1] as a basic framework to solve (1.1). Moreover, we construct an algorithm which is applicable to large-scale problems. The SQP method with quasi-Newton methods has been well known as one of effective methods for nonlinear optimization problems. However, it is not easy to apply quasi-Newton methods to large-scale optimization problems because the Hessian of the Lagrangian, which is sparse in many cases, is approximated by a dense matrix in quasi-Newton methods usually. On the contrary, the trust region SQP method is considered to be applicable to large-scale optimization problems because it uses the Hessian of the Lagrangian itself, so the sparsity of the Hessian is preserved.

In the ordinary trust region SQP method, the QP subproblem at a current point x_k is

basically defined by

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta x^T G_k \Delta x + \nabla f(x_k)^T \Delta x \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0 \ (j \in J_E), \\ & g_j(x_k) + \nabla g_j(x_k)^T \Delta x \geq 0 \ (j \in J_I), \\ & \|\Delta x\| \leq \delta_k, \end{array} \tag{1.2}$$

where G_k is the Hessian of the Lagrangian of (1.1) at x_k and δ_k is a radius of the trust region. The ordinary trust region SQP method has some difficulties in execution. One of such difficulties is that it is possible that (1.2) has no feasible region when δ_k is too small. In the algorithms which have been proposed so far, some parameters are introduced to make subproblems feasible [3, 4, 11, 13], or subproblems are solved by two steps [9, 10]. These modifications of algorithms are so complicated that it is not favorable for implementation and calculation time. Moreover, another difficulty is to solve QP subproblems which are not necessarily convex because G_k is not necessarily positive semidefinite.

In this paper, we propose a new trust region SQP method which eliminates these two difficulties. In our method, we solve two types of subproblem. One is a convex QP problem and the other is a system of linear equations. Our method is applicable to large-scale optimization problems. In addition, we note that our approach in this paper resembles an approach of [15], which deals with the primal-dual interior point trust region method.

This paper is organized as follows: In Section 2, we introduce some basic concepts. In Section 3, we propose the new trust region SQP method. In Section 4, we show the global convergence of the proposed method. In Section 5, we show some local property of the proposed method. In Section 6, we report numerical results by the proposed method. In Section 7, we make some concluding remarks and discuss future research topics.

2. Preliminaries

In this section, we introduce some basic concepts which are necessary in the following sections.

First, we prepare some notations. The Lagrangian of (1.1) is

$$L(x, y) = f(x) - \sum_{j \in J_E} y_j g_j(x) - \sum_{j \in J_I} y_j g_j(x),$$

where $y_j (j \in J_E \cup J_I)$ are Lagrange multipliers for $g_j(x) = 0 \ (j \in J_E)$ and $g_j(x) \geq 0 \ (j \in J_I)$. Together with this, the KKT conditions for (1.1) are

$$\begin{cases} \nabla_x L(x, y) = \nabla f(x) - \sum_{j \in J_E} y_j \nabla g_j(x) - \sum_{j \in J_I} y_j \nabla g_j(x) = 0, \\ g_j(x) = 0 \ (j \in J_E), \\ y_j g_j(x) = 0, \ y_j \geq 0, \ g_j(x) \geq 0 \ (j \in J_I). \end{cases} \tag{2.1}$$

The penalty function for (1.1) is defined by

$$F(x) = f(x) + \sum_{j \in J_E} \rho_j |g_j(x)| + \sum_{j \in J_I} \rho_j |\min\{0, g_j(x)\}|,$$

where $\rho_j \ (j \in J_E \cup J_I)$ is the penalty parameter which is sufficiently large. It is known that the local minimum of (1.1) is also that of $F(x)$ under appropriate conditions [5] .

The first-order approximation of F at x in the direction $d \in \Re^n$ is defined by

$$F_l(x; d) = f(x) + \nabla f(x)^T d + \sum_{j \in J_E} \rho_j |g_j(x) + \nabla g_j(x)^T d| + \sum_{j \in J_I} \rho_j |\min\{0, g_j(x) + \nabla g_j(x)^T d\}|.$$

In addition, we define the second-order approximation of F at x in the direction d by

$$F_q(x;d) = F_l(x;d) + \frac{1}{2}d^T G d,$$

where $G = \nabla_x^2 L(x,y)$. The difference of F, F_l and F_q in the direction d are

$$\begin{aligned} \Delta F(x;d) &= F(x+d) - F(x), \\ \Delta F_l(x;d) &= F_l(x;d) - F(x), \\ \Delta F_q(x;d) &= F_q(x;d) - F(x), \end{aligned}$$

respectively.

Next, we explain two types of QP subproblem which we solve at each iteration in our algorithm. In what follows, k denotes an iteration number of our algorithm.

One of the QP subproblems is

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta x^T D_k \Delta x + \nabla f(x_k)^T \Delta x \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0 \ (j \in J_E), \\ & g_j(x_k) + \nabla g_j(x_k)^T \Delta x \geq 0 \ (j \in J_I), \end{array} \tag{2.2}$$

where D_k is a diagonal matrix whose diagonal elements are positive. Δx_k^{SD} and $y_{k+1,j}^{SD}$ $(j \in J_E \cup J_I)$ denote a solution and Lagrange multipliers of (2.2), respectively. As we will show in Lemma 4.1, Δx_k^{SD} is a descent direction of the penalty function F, because D_k is a positive definite matrix. This fact contributes greatly to the global convergence of our algorithm. Moreover, (2.2) is a convex QP problem, so we can use various methods to solve it. By the way, we now assume that (2.2) is feasible. We consider the infeasible case in Remark 4.1.

The KKT conditions for (2.2) are

$$D_k \Delta x_k^{SD} + \nabla f(x_k) - \sum_{j \in J_E \cup J_I} y_{k+1,j}^{SD} \nabla g_j(x_k) = 0, \tag{2.3}$$

$$g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD} = 0 \ (j \in J_E), \tag{2.4}$$

$$\begin{cases} y_{k+1,j}^{SD}(g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD}) = 0, \\ y_{k+1,j}^{SD} \geq 0, \ g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD} \geq 0 \ (j \in J_I). \end{cases} \tag{2.5}$$

Now, let J_{A_k} be the index set of active constraints of (2.2) at the solution Δx_k^{SD}, that is,

$$J_{A_k} := \{ j \in J_E \cup J_I \mid g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD} = 0 \}.$$

As we will show in Lemma 5.1, when x_k is sufficiently close to a solution, J_{A_k} coincides with the index set of active constraints of (1.1) under appropriate assumptions.

The other QP subproblem is defined by

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta x^T G_k \Delta x + \nabla f(x_k)^T \Delta x \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0 \ (j \in J_{A_k}), \end{array} \tag{2.6}$$

where $G_k = \nabla_x^2 L(x_k, y_k)$, Δx_k^N and $y_{k+1,j}^N$ $(j \in J_{A_k})$ denote a solution and Lagrange multipliers of (2.6), respectively. We note that (2.6) is feasible when (2.2) is feasible. The KKT conditions of (2.6) are

$$G_k \Delta x_k^N + \nabla f(x_k) - \sum_{j \in J_{A_k}} y_{k+1,j}^N \nabla g_j(x_k) = 0, \tag{2.7}$$

$$g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^N = 0 \ (j \in J_{A_k}). \tag{2.8}$$

As we mentioned, when x_k is sufficiently close to a solution, J_{A_k} coincides with the index set of active constraints at a solution of (1.1). Accordingly, in a neighborhood of a solution of (1.1), the conditions (2.7) and (2.8) are good approximation of the KKT conditions of (1.1), except for the condition of inactive constraints whose Lagrange multipliers' value are equal to 0. Thus we set $y^N_{k+1,j} = 0$ $(j \notin J_{A_k})$. So $(\Delta x^N_k, y^N_{k+1})$ is a favorable direction for fast convergence in a neighborhood of a solution.

We can rewrite (2.7) and (2.8) as

$$\begin{pmatrix} G_k & -\nabla g_J(x_k) \\ \nabla g_J(x_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta x^N_k \\ y^N_{k+1,J} \end{pmatrix} = \begin{pmatrix} -\nabla f(x_k) \\ -g_J(x_k) \end{pmatrix}, \tag{2.9}$$

where $g_J(x_k)$ and $y^N_{k+1,J}$ are vectors which are composed of $g_j(x_k)$ and $y^N_{k+1,j}$ $(j \in J_{A_k})$, respectively. We note that we can get a solution of (2.6) with less computational time than general QP problems, because the KKT conditions of (2.6) are equivalent to a system of linear equations (2.9). However, (2.9) may be ill-posed when, for example, G_k is singular and Δx^N_k is not bounded. We will consider this case in Remark 4.2.

3. Algorithm

The algorithm which we propose in this paper is as follows.

Algorithm TRSQP

Step 0. Set an initial point $x_0 \in \Re^n$, a positive definite diagonal matrix $D_0 \in \Re^{n\times n}$, a symmetric matrix $G_0 \in \Re^{n\times n}$, and parameters $\delta_0 > 0$ and $M > 1$. Set $k := 0$.

Step 1. Compute $(\Delta x^{SD}_k, y^{SD}_{k+1})$ and $(\Delta x^N_k, y^N_{k+1})$ by solving (2.2) and (2.6), respectively. If

$$\|\Delta x^N_k\| \le M\|\Delta x^{SD}_k\| \tag{3.1}$$

is not satisfied, modify G_k to satisfy (3.1).

Step 2. If (x_k, y_{k+1}) satisfies (2.1), where

$$y_{k+1} = \begin{cases} y^N_{k+1}, & \text{if } y^N_{k+1,j} \ge 0 \ (\forall j \in J_{A_k} \cap J_I), \\ y^{SD}_{k+1}, & \text{otherwise,} \end{cases}$$

then stop.

Step 3. Find $s_k \in \Re^n$ which satisfies

$$\|s_k\| \le \delta_k, \tag{3.2}$$

$$\|s_k\| \le M\|\Delta x^{SD}_k\|, \tag{3.3}$$

$$\Delta F_q(x_k; s_k) \le \frac{1}{2}\Delta F_q(x_k; \alpha^*_k \Delta x^{SD}_k), \tag{3.4}$$

where

$$\alpha^*_k = \min\left\{1, \frac{\delta_k}{\|\Delta x^{SD}_k\|}, -\frac{\Delta F_l(x_k; \Delta x^{SD}_k)}{\max\{0, \Delta x^{SD\,T}_k G_k \Delta x^{SD}_k\}}\right\} \tag{3.5}$$

and the last term in the braces in the right hand side is assumed to give the value $+\infty$ if the value of the denominator is 0.

Step 4. Set δ_{k+1} as follows:

$$\begin{array}{rcl} \Delta F(x_k; s_k) > \frac{1}{4}\Delta F_q(x_k; s_k) & \Rightarrow & \delta_{k+1} := \frac{1}{2}\delta_k, \\ \Delta F(x_k; s_k) \le \frac{3}{4}\Delta F_q(x_k; s_k) & \Rightarrow & \delta_{k+1} := 2\delta_k, \\ \text{otherwise} & \Rightarrow & \delta_{k+1} := \delta_k. \end{array}$$

Step 5. If $\Delta F(x_k; s_k) \leq 0$, set $x_{k+1} := x_k + s_k$, otherwise set $x_{k+1} := x_k$. Set $D_{k+1}, G_{k+1} := \nabla_x^2 L(x_{k+1}, y_{k+1})$ and $k := k + 1$. Go to Step 1. □

In Remark 4.2, we will explain how G_k is modified when (3.1) is not satisfied in Step 1 of Algorithm TRSQP.

At this moment, we make a remark on a difference between the ordinary trust region SQP method and Algorithm TRSQP. In the ordinary trust region SQP method, we find a solution of a subproblem with a trust region constraint. Thus a solution satisfies a trust region constraint and a solution itself can be a candidate of the next iteration. On the other hand, in Algorithm TRSQP, we solve (2.2) and (2.6) as subproblems and a solution Δx_k^{SD} and Δx_k^N, respectively, does not necessarily satisfy a trust region constraint. Thus, we have to make a candidate of the next iteration which satisfies a trust region constraint. We will consider a concrete method for it in Remark 4.3.

4. Global Convergence of Our Algorithm

In this section, we show that Algorithm TRSQP has the global convergence property.

First, we assume the following.

Assumption 4.1 **(1)** *The vectors* $\{y_{k+1,j}^{SD}\}$ *and* $\{y_{k+1,j}^{N}\}$ $(j \in J_E \cup J_I)$ *are bounded. In addition,* $\rho_j > |y_{k+1,j}^{SD}|$ $(j \in J_E \cup J_I)$ *holds for all* k.

(2) *The penalty function* $F(x)$ *is bounded below and its level set at the initial point* x_0*, that is,* $\{x \in \Re^n \mid F(x) \leq F(x_0)\}$*, is compact.*

(3) *The matrix* D_k *is uniformly positive definite and uniformly bounded. The matrix* G_k *is uniformly bounded.*

(4) *There exists a positive constant* $M > 0$ *which satisfies (3.1) for all* k.

Now we show the next lemma.

Lemma 4.1 *Suppose that Assumption 4.1 holds. Then* $\Delta F_l(x_k; \Delta x_k^{SD}) \leq 0$ *holds. If* $\Delta x_k^{SD} \neq 0$, $\Delta F_l(x_k; \Delta x_k^{SD}) < 0$ *holds.*

Proof: By the definition of ΔF_l, (2.3), (2.4) and (2.5), we have

$$\begin{aligned}
\Delta F_l(x_k; \Delta x_k^{SD}) &= \nabla f(x_k)^T \Delta x_k^{SD} + \sum_{j \in J_E} \rho_j^k (|g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD}| - |g_j(x_k)|) \\
&\quad + \sum_{j \in J_I} \rho_j^k (|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD}\}| - |\min\{0, g_j(x_k)\}|) \\
&= -\Delta x_k^{SD^T} D_k \Delta x_k^{SD} + \sum_{j \in J_E} y_{k+1,j}^{SD} \nabla g_j(x_k)^T \Delta x_k^{SD} + \sum_{j \in J_I} y_{k+1,j}^{SD} \nabla g_j(x_k)^T \Delta x_k^{SD} \\
&\quad - \Big[\sum_{j \in J_E} \rho_j^k |g_j(x_k)| + \sum_{j \in J_I} \rho_j^k |\min\{0, g_j(x_k)\}| \Big]. \qquad (4.1)
\end{aligned}$$

Moreover, from (2.4) and (2.5), we have

$$\begin{aligned}
\sum_{j \in J_E} y_{k+1,j}^{SD} \nabla g_j(x_k)^T \Delta x_k^{SD} &\leq \sum_{j \in J_E} \left| y_{k+1,j}^{SD} \right| \left| \nabla g_j(x_k)^T \Delta x_k^{SD} \right| = \sum_{j \in J_E} \left| y_{k+1,j}^{SD} \right| |g_j(x_k)|, \qquad (4.2) \\
\sum_{j \in J_I} y_{k+1,j}^{SD} \nabla g_j(x_k)^T \Delta x_k^{SD} &= \sum_{j \in J_I} y_{k+1,j}^{SD} (-g_j(x_k)) \leq \sum_{j \in J_I} y_{k+1,j}^{SD} |\min\{0, g_j(x_k)\}|. \qquad (4.3)
\end{aligned}$$

It then follows from (4.1), (4.2) and (4.3) that

$$\Delta F_l(x_k; \Delta x_k^{SD}) \leq -\Delta x_k^{SD^T} D_k \Delta x_k^{SD} + \sum_{j \in J_E} (|y_{k+1,j}^{SD}| - \rho_j^k) |g_j(x_k)|$$

$$
\begin{aligned}
& + \sum_{j \in J_I} (y^{SD}_{k+1,j} - \rho^k_j) |\min\{0, g_j(x_k)\}| \\
\leq \ & -\Delta x^{SD\,T}_k D_k \Delta x^{SD}_k,
\end{aligned}
$$

where the last inequality follows from Assumption 4.1 (1). Therefore we obtain $\Delta F_l(x_k; \Delta x^{SD}_k) \leq 0$ from Assumption 4.1 (3). In particular, $\Delta F_l(x_k; \Delta x^{SD}_k) < 0$ holds when $\Delta x^{SD}_k \neq 0$. □

Lemma 4.1 shows that Δx^{SD}_k is a descent direction of the penalty function F. This property contributes greatly to the global convergence of Algorithm TRSQP.

Lemma 4.2 *Suppose that Assumption 4.1 holds. Then,*

$$\Delta F_l(x; \alpha \Delta x^{SD}_k) \leq \alpha \Delta F_l(x; \Delta x^{SD}_k), \quad \forall \alpha \in [0, 1]$$

holds.

Proof: First, we consider the index $j \in J_E$. From (2.4), we have

$$
\begin{aligned}
& |g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k| - |g_j(x_k)| \\
= \ & |(1-\alpha) g_j(x_k) + \alpha (g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k)| - |g_j(x_k)| \\
= \ & -\alpha |g_j(x_k)| \\
= \ & \alpha \left(|g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k| - |g_j(x_k)| \right) \quad (j \in J_E).
\end{aligned} \tag{4.4}
$$

Next, we consider the index $j \in J_I$ in two cases: (i) $g_j(x_k) \geq 0$ and (ii) $g_j(x_k) < 0$.

Case (i): In this case, $g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k \geq 0$ holds from (2.5) and $\alpha \in [0, 1]$. Thus we have

$$
\begin{aligned}
& |\min\{0, g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \\
= \ & 0 \\
= \ & \alpha \left(|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \right).
\end{aligned}
$$

Case (ii): In this case, we consider two cases: (a) $g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k \geq 0$ and (b) $g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k < 0$.

Case (a): By (2.5) and $\alpha \in [0, 1]$, we have

$$
\begin{aligned}
& |\min\{0, g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \\
= \ & 0 - |\min\{0, g_j(x_k)\}| \\
\leq \ & -\alpha |\min\{0, g_j(x_k)\}| \\
= \ & \alpha \left(|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \right).
\end{aligned}
$$

Case (b): In this case, $\nabla g_j(x_k)^T \Delta x^{SD}_k \geq -g_j(x_k) > 0$ holds from (2.5). Thus we have

$$
\begin{aligned}
& |\min\{0, g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \\
= \ & -(g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x^{SD}_k) + g_j(x_k) \\
= \ & -\alpha \nabla g_j(x_k)^T \Delta x^{SD}_k \\
\leq \ & \alpha g_j(x_k) \\
= \ & -\alpha |\min\{0, g_j(x_k)\}| \\
= \ & \alpha \left(|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k\}| - |\min\{0, g_j(x_k)\}| \right).
\end{aligned}
$$

Now we summarize the results mentioned above. For $j \in J_I$, we obtain

$$\begin{aligned} & |\min\{0, g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x_k^{SD}\}| - |\min\{0, g_j(x_k)\}| \\ \leq\ & \alpha \left(|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD}\}| - |\min\{0, g_j(x_k)\}| \right) \quad (j \in J_I). \end{aligned} \tag{4.5}$$

Then, from (4.4) and (4.5), we have

$$\begin{aligned} \Delta F_l(x_k; \alpha \Delta x_k) &= \alpha \nabla f(x_k)^T \Delta x_k + \sum_{j \in J_E} \rho_j^k (|g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x_k| - |g_j(x_k)|) \\ &\quad + \sum_{j \in J_I} \rho_j^k (|\min\{0, g_j(x_k) + \alpha \nabla g_j(x_k)^T \Delta x_k\}| - |\min\{0, g_j(x_k)\}|) \\ &\leq \alpha \Big[\nabla f(x_k)^T \Delta x_k + \sum_{j \in J_E} \rho_j^k (|g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k| - |g_j(x_k)|) \\ &\quad + \sum_{j \in J_I} \rho_j^k (|\min\{0, g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k\}| - |\min\{0, g_j(x_k)\}|) \Big] \\ &= \alpha \Delta F_l(x_k; \Delta x_k). \end{aligned}$$

This completes the proof. □

Lemma 4.3 *Suppose that Assumption 4.1 holds. Then,*

$$\Delta F_q(x_k; \alpha_k^* \Delta x_k^{SD}) \leq \frac{1}{2} \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}).$$

Proof: From Lemma 4.2, we obtain

$$\begin{aligned} \Delta F_q(x_k; \alpha_k^* \Delta x_k^{SD}) &= \Delta F_l(x_k; \alpha_k^* \Delta x_k^{SD}) + \frac{1}{2} (\alpha_k^*)^2 \Delta x_k^{SD^T} G_k \Delta x_k^{SD} \\ &\leq \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}) + \frac{1}{2} (\alpha_k^*)^2 \Delta x_k^{SD^T} G_k \Delta x_k^{SD}. \end{aligned} \tag{4.6}$$

Moreover, we have from Lemma 4.1 that $\Delta F_l(x_k; \Delta x_k^{SD}) \leq 0$.

To show this lemma, we consider two cases: (i) $\Delta x_k^{SD^T} G_k \Delta x_k^{SD} > 0$ and (ii) $\Delta x_k^{SD^T} G_k \Delta x_k^{SD} \leq 0$.

Case (i): It follows from (3.5) that $0 \leq \alpha_k^* \leq -\frac{\Delta F_l(x_k; \Delta x_k^{SD})}{\Delta x_k^{SD^T} G_k \Delta x_k^{SD}}$. Thus we have from (4.6) that

$$\Delta F_q(x_k; \alpha_k^* \Delta x_k^{SD}) \leq \frac{1}{2} \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}).$$

Case (ii): From (4.6), we obtain

$$\begin{aligned} \Delta F_q(x_k; \alpha_k^* \Delta x_k^{SD}) &\leq \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}) \\ &\leq \frac{1}{2} \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}). \end{aligned}$$

This completes the proof. □

Lemma 4.4 *Suppose that Assumption 4.1 holds. Then,* $\liminf_{k \to \infty} \|\Delta x_k^{SD}\| = 0$ *holds.*

Proof: First, from Step 3 of Algorithm TRSQP and Lemmas 4.1 and 4.3, we obtain

$$\begin{aligned} \Delta F_q(x_k; s_k) &\leq \frac{1}{2} \Delta F_q(x_k; \alpha_k^* \Delta x_k^{SD}) \\ &\leq \frac{1}{4} \alpha_k^* \Delta F_l(x_k; \Delta x_k^{SD}) \\ &= \frac{1}{4} \min \left\{ 1, \frac{\delta_k}{\|\Delta x_k^{SD}\|}, -\frac{\Delta F_l(x_k; \Delta x_k^{SD})}{\max\{0, \Delta x_k^{SD^T} G_k \Delta x_k^{SD}\}} \right\} \Delta F_l(x_k; \Delta x_k^{SD}) \\ &\leq 0. \end{aligned} \tag{4.7}$$

Next, we define the sets K_1 and K_2 by

$$\begin{aligned} K_1 &:= \{k \in \{0,1,...\} \mid \Delta F(x_k; s_k) > \frac{1}{4}\Delta F_q(x_k; s_k)\}, \\ K_2 &:= \{k \in \{0,1,...\} \mid \Delta F(x_k; s_k) \leq \frac{1}{4}\Delta F_q(x_k; s_k)\}. \end{aligned} \tag{4.8}$$

Now we assume that $\liminf_{k\to\infty} \|\Delta x_k^{SD}\| > 0$. Then, from Lemma 4.1 and Assumption 4.1 (3), there exists a positive constant ϵ_1 such that

$$\liminf_{k\to\infty} |\Delta F_l(x_k; \Delta x_k^{SD})| > \epsilon_1. \tag{4.9}$$

Moreover, from Assumption 4.1 (3), there exists a positive constant R_1 for all k such that

$$\|\Delta x_k^{SD}\| < R_1. \tag{4.10}$$

In addition, from Assumption 4.1 (3) and (4.10), there exists a positive constant R_2 for all k such that

$$|\Delta x_k^{SD^T} G_k \Delta x_k^{SD}| < R_2. \tag{4.11}$$

Now we consider two cases: (i) $\limsup_{k\to\infty} \delta_k = 0$ and (ii) $\limsup_{k\to\infty} \delta_k > 0$.

Case (i): This case is equivalent to $\lim_{k\to\infty} \delta_k = 0$. Thus we have from Step 3 of Algorithm TRSQP that $\lim_{k\to\infty} \|s_k\| = 0$. Moreover, from Step 4 of Algorithm TRSQP, K_1 has an infinite number of elements. Now we have from the definition of ΔF and (4.8) that

$$\begin{aligned} \Delta F(x_k; s_k) &= \Delta F_l(x_k; s_k) + O(\|s_k\|^2) \\ &= \Delta F_q(x_k; s_k) + O(\|s_k\|^2) > \frac{1}{4}\Delta F_q(x_k; s_k) \quad (k \in K_1) \end{aligned}$$

when k is large enough, where $M = O(A)$ means that there exists a positive constant β such that $\|M\| \leq \beta A$. From this fact and $\Delta F_q(x_k; s_k) < 0$, we have

$$|\Delta F_q(x_k; s_k)| = O(\|s_k\|^2) \quad (k \in K_1) \tag{4.12}$$

when k is large enough. On the other hand, we obtain from (4.9) and (4.11) that

$$\liminf_{k\to\infty} \left(-\frac{\Delta F_l(x_k; \Delta x_k^{SD})}{\max\{0, \Delta x_k^{SD^T} G_k \Delta x_k^{SD}\}} \right) > \frac{\epsilon_1}{R_2} > 0.$$

From this fact, we have that $\alpha_k^* = \delta_k / \|\Delta x_k^{SD}\|$ $(k \in K_1)$ when k is large enough. Therefore we obtain from (4.7) and (4.10) that

$$\begin{aligned} |\Delta F_q(x_k; s_k)| &\geq \frac{1}{4}\alpha_k^* |\Delta F_l(x_k; \Delta x_k^{SD})| \\ &\geq \frac{1}{4}\frac{\delta_k}{\|\Delta x_k^{SD}\|}\epsilon_1 \\ &\geq \frac{\epsilon_1}{4R_1}\|s_k\|. \end{aligned} \tag{4.13}$$

Thus we obtain from (4.10) that

$$|\Delta F_q(x_k; s_k)| = \Omega(\|s_k\|), \tag{4.14}$$

where $M = \Omega(A)$ means that there exists a positive constant $\beta > 0$ such that $\|M\| \geq \beta A$. However, (4.14) contradicts (4.12).

Case (ii): In this case, from Step 4 in Algorithm TRSQP, K_2 has an infinite number of elements. From (4.7), there exists α_k^* such that

$$\Delta F(x_k; s_k) \leq \frac{1}{4}\Delta F_q(x_k; s_k) \leq \frac{\alpha_k^*}{16}\Delta F_l(x_k; \Delta x_k^{SD}) < 0 \quad (k \in K_2).$$

Moreover, $\{F(x_k)\}$ is bounded below and nonincreasing because of Assumption 4.1 (2) and Step 5 of Algorithm TRSQP. Thus we have

$$F(x_{k+1}) - F(x_k) = \Delta F(x_k; s_k) \to 0 \quad (k \to \infty, k \in K_2).$$

Hence, from (4.9), we obtain that $\alpha_k^* \to 0$ $(k \in K_2)$. Now we consider two cases: (a) K_1 has finite elements and (b) K_1 has infinite elements.

Case (a): From Step 4 of Algorithm TRSQP, $\liminf_{k\to\infty, k\in K_2} \delta_k > 0$ holds. Therefore, from (4.11) and (3.5), we have $\Delta F_l(x_k; \Delta x_k^{SD}) \to 0$ $(k \in K_2)$. However, this contradicts (4.9).

Case (b): Now we assume that $\limsup_{k\to\infty, k\in K_2} \delta_k = 0$. From this assumption and Step 4 of Algorithm TRSQP, we have

$$0 \leq \limsup_{k\to\infty, k\in K_1} \delta_k \leq 2 \cdot \limsup_{k\to\infty, k\in K_2} \delta_k = 0,$$

so we obtain that $\limsup_{k\to\infty, k\in K_1} \delta_k = 0$. Therefore we have $\limsup_{k\to\infty} \delta_k = 0$. However, this contradicts the assumption of Case (ii). Hence, we have $\limsup_{k\to\infty, k\in K_2} \delta_k > 0$. Therefore, from (4.11) and (3.5), we have $\liminf_{k\to\infty, k\in K_2} \Delta F_l(x_k; \Delta x_k^{SD}) = 0$. However, this contradicts (4.9).

Therefore, we conclude $\liminf_{k\to\infty} \|\Delta x_k^{SD}\| = 0$. □

Theorem 4.1 *Suppose that Assumption 4.1 holds. Let $\{x_k\}$ be a sequence generated by Algorithm TRSQP. Then, $\{x_k\}$ has an accumulation point. Moreover, there exists an accumulation point x^* of $\{x_k\}$, which has an appropriate Lagrange multiplier y^* such that (x^*, y^*) satisfies the KKT conditions for (1.1).*

Proof: First, from Step 5 of Algorithm TRSQP, $\{F(x_k)\}$ is nonincreasing. Then, from Assumption 4.1 (2), $\{x_k\}$ is in a compact set, thus $\{x_k\}$ has an accumulation point x^*. Now, without loss of generality, we assume that a subsequence $\{x_k\}_{k\in K_3}$ converges to x^* and $\lim_{k\to\infty, k\in K_3} \|\Delta x_k^{SD}\| = 0$ from Lemma 4.4. Moreover, since the number of elements of $J_E \cup J_I$ is finite, then possible combinations of elements which are included in J_{A_k} are finite. Therefore there exists an infinite set $K_4 \subseteq K_3$ such that $J_{A_k} = J_{\bar{A}}$ $(k \in K_4)$ for a particular index set $J_{\bar{A}}$. Thus we have $J_{A_k} = J_{\bar{A}}$ $(k \in K_4), \lim_{k\to\infty, k\in K_4} x_k = x^*$ and $\lim_{k\to\infty, k\in K_4} \|\Delta x_k^{SD}\| = 0$. Moreover, from (3.1), we have $\lim_{k\to\infty, k\in K_4} \|\Delta x_k^{N}\| = 0$.

To show this theorem, we consider two cases: (i) there exists k_0 such that $y_{k+1,j}^N \geq 0$ $(j \in J_{A_k} \cap J_I, k \geq k_0, k \in K_4)$ and (ii) the case except (i).

Case (i): We note that $y_{k+1,j}^N = 0$ $(j \notin J_{A_k}, k \geq k_0, k \in K_4)$. For $k \geq k_0$, we have from (2.5), (2.7), (2.8) and the definition of J_{A_k} that

$$\begin{array}{l} G_k\Delta x_k^N + \nabla f(x_k) - \sum_{j\in J_E} y_{k+1,j}^N \nabla g_j(x_k) - \sum_{j\in J_I} y_{k+1,j}^N \nabla g_j(x_k) = 0, \\ g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^N = 0 \ (j \in J_E), \\ y_{k+1,j}^N (g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^N) = 0 \ (j \in J_{\bar{A}} \cap J_I), \\ y_{k+1,j}^N \geq 0, \ g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^N = 0 \ (j \in J_{\bar{A}} \cap J_I), \\ y_{k+1,j}^N (g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD}) = 0 \ (j \notin J_{\bar{A}}), \\ y_{k+1,j}^N = 0, \ g_j(x_k) + \nabla g_j(x_k)^T \Delta x_k^{SD} > 0 \ (j \notin J_{\bar{A}}). \end{array} \tag{4.15}$$

Now, from Assumption 4.1 (1), $\{y^N_{k+1}\}_{k\in K_4}$ has an accumulation point y^{N^*}. Thus, without loss of generality, we assume that $\{y^N_{k+1}\} \to y^{N^*}$ $(k \to \infty, k \in K_4)$. Therefore, from $\lim_{k\to\infty, k\in K_4} \|\Delta x^{SD}_k\| = \lim_{k\to\infty, k\in K_4} \|\Delta x^N_k\| = 0$, we obtain that (x^*, y^{N^*}) satisfies (2.1) when $k \to \infty$ $(k \in K_4)$ in (4.15).

Case (ii): In this case, there exists an infinite number of k such that $y_{k+1} = y^{SD}_{k+1}$. Thus, without loss of generality, $y_{k+1} = y^{SD}_{k+1}$ holds when k is sufficiently large. So we have (2.3), (2.4) and (2.5) when k is sufficiently large. Moreover, from Assumption 4.1 (1), $\{y^{SD}_{k+1}\}_{k\in K_4}$ has an accumulation point y^{SD^*}. Now we assume without loss of generality that $\{y^{SD}_{k+1}\} \to y^{SD^*}$ $(k \to \infty)$. Therefore, from $\lim_{k\to\infty, k\in K_4} \|\Delta x^{SD}_k\| = 0$, we obtain that (x^*, y^{SD^*}) satisfies (2.1) when $k \to \infty$ $(k \in K_4)$ in (2.3), (2.4) and (2.5).

This completes the proof. □

Remark 4.1 *Problem (2.2) may be infeasible even if the original problem (1.1) is feasible. In such a case, we solve the following problem instead of (2.2):*

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta x^T D_k \Delta x + \nabla f(x_k)^T \Delta x + \sum_{j\in J_E} \rho_j(\xi^+_j + \xi^-_j) + \sum_{j\in J_I} \rho_j \eta_j \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x + \xi^+_j - \xi^-_j = 0 \ (j \in J_E), \\ & g_j(x_k) + \nabla g_j(x_k)^T \Delta x + \eta_j \geq 0 \ (j \in J_I), \\ & \xi^+_j \geq 0, \ \xi^-_j \geq 0 \ (j \in J_E), \\ & \eta_j \geq 0, \ (j \in J_I), \end{array} \tag{4.16}$$

where ξ^+_j, ξ^-_j $(j \in J_E)$ and η_j $(j \in J_I)$ are elastic variables. Moreover, active constraints and active elastic variables in (4.16) are treated as counterparts of constraints composing J_{A_k}, and we construct a subproblem which corresponds to (2.6). To show the validity of considering (4.16), we confirm the equivalence of following problems at first:

$$\begin{array}{lll} \text{minimize} \ |h(x)| & \Leftrightarrow & \begin{array}{ll} \text{minimize} & \xi^+ + \xi^- \\ \text{subject to} & h(x) + \xi^+ - \xi^- = 0, \\ & \xi^+ \geq 0, \ \xi^- \geq 0, \end{array} \\ \\ \text{minimize} \ |\min\{0, h(x)\}| & \Leftrightarrow & \begin{array}{ll} \text{minimize} & \eta \\ \text{subject to} & h(x) + \eta = 0, \\ & \eta \geq 0, \end{array} \end{array}$$

where $h : \Re^n \to \Re^1$. From these properties, we can easily reformulate (4.16) as

$$\text{minimize} \quad \bar{F}_l(x_k; \Delta x) = F_l(x_k; \Delta x) + \tfrac{1}{2}\Delta x^T D_k \Delta x. \tag{4.17}$$

In what follows, $\Delta \bar{x}^{SD}_k$ denotes a solution of (4.17). It follows from $\bar{F}_l(x_k; \Delta \bar{x}^{SD}_k) \leq \bar{F}_l(x_k; 0)$ that

$$F_l(x_k; \Delta \bar{x}^{SD}_k) + \frac{1}{2}\Delta \bar{x}^{SD^T}_k D_k \Delta \bar{x}^{SD}_k \leq F_l(x_k; 0) = F(x_k),$$

then we have

$$\Delta F_l(x_k; \Delta \bar{x}^{SD}_k) \leq -\frac{1}{2}\Delta \bar{x}^{SD^T}_k D_k \Delta \bar{x}^{SD}_k \leq 0.$$

This result is a counterpart of Lemma 4.1. Moreover, in the same way of this section, we can show the same property as Theorem 4.1 when we solve (4.16) instead of (2.2). However, a complete proof will be complicated a little bit, so it is eliminated here. □

Remark 4.2 *When G_k is singular, we may fail to find a solution of (2.9). Moreover, when (2.9) is ill-posed, $\|\Delta x_k^N\|$ would be so large that (3.1) may be violated. In such cases, G_k is replaced by $\bar{G}_k$ which is defined by*

$$\bar{G}_k := G_k + \mu_k I,$$

where μ_k is a positive parameter. When μ_k is sufficiently large, $\bar{G}_k$ is nonsingular and (2.9) is well-posed, so (3.1) is satisfied. Practically, we choose a tiny positive value on μ_k at first. If (3.1) is not satisfied, we increase the value of μ_k gradually. □

Remark 4.3 *In Step 3 of Algorithm TRSQP, we can choose any s_k which satisfies (3.2), (3.3) and (3.4). Now we consider an algorithm to construct s_k which is valid.*

For a convex combination of Δx_k^{SD} and Δx_k^N, that is,

$$\Delta \bar{x}_k(\nu_k) := \nu_k \Delta x_k^{SD} + (1-\nu_k)\Delta x_k^N, \quad \nu_k \in [0,1],$$

$\bar{\alpha}_k^*(\nu_k)$ is defined by

$$\bar{\alpha}_k^*(\nu_k) := \min\left\{1, \frac{\delta_k}{\|\Delta \bar{x}_k(\nu_k)\|}, -\frac{\Delta F_l(x_k; \Delta \bar{x}_k(\nu_k))}{\max\{0, \Delta \bar{x}_k(\nu_k)^T G_k \Delta \bar{x}_k(\nu_k)\}}\right\},$$

where the last term in the braces is assumed to give the value $+\infty$ if the value of the denominator is 0. Now we consider $\bar{s}_k(\nu_k) := \bar{\alpha}_k^*(\nu_k)\Delta \bar{x}_k(\nu_k)$. It is easy to show that $\bar{s}_k(1) = \bar{\alpha}_k^*(1)\Delta \bar{x}_k(1)$ satisfies (3.2), (3.3) and (3.4). Moreover, as we will see in Section 5, $\Delta \bar{x}_k(0)$ is equivalent to the search direction of the ordinary SQP method under some assumptions, so $\Delta \bar{x}_k(0)$ is a good direction for fast convergence practically. Therefore, in view of efficient implementation, we set 0 on ν_k at first, and repeatedly increase the value of ν_k by 0.1 until $\bar{s}_k(\nu_k)$ satisfies (3.2), (3.3) and (3.4). □

Remark 4.4 *In this paper, we assume that the penalty parameter ρ_j $(j \in J_E \cup J_I)$ satisfies Assumption 4.1 (1). This assumption is tactically prepared for the proofs in this section, so we should update the value of ρ_j $(j \in J_E \cup J_I)$ at each iteration with an appropriate rule when we implement Algorithm TRSQP. For example, we can adopt the following update rule:*

- *When $k = 0$, we initialize $\rho_{k,j} := \max\{L\,|y_{k,j}^{SD}|,\ \rho^{min}\}$ $(j \in J_E \cup J_I)$,*
- *When $k > 0$, we set $\rho_{k,j} := \max\{L\,|y_{k,j}^{SD}|,\ \rho_{k-1,j}\}$ $(j \in J_E \cup J_I)$,*

where $\rho_{k,j}$ $(j \in J_E \cup J_I)$ denotes the penalty parameter at the iteration k, L and ρ^{min} are constants such that $L > 1$ and $\rho^{min} > 0$. The adoption of this update rule does not affect the proofs in this section. □

5. Local Property of Our Algorithm

In this section, we consider some property of Algorithm TRSQP in a neighborhood of a solution.

First, we assume the following.

Assumption 5.1 (1) *The sequence $\{(x_k, y_{k+1})\}$ which is generated by Algorithm TRSQP converges to (x^*, y^*), and (x^*, y^*) satisfies (2.1). Moreover, $\lim_{k\to\infty}\|\Delta x_k^{SD}\| = 0$ holds and $\{y_{k+1}^{SD}\}$ converges to y^{SD^*}.*

(2) *The linear independence constraint qualification holds at (x^*, y^*). Moreover, the strict complementarity in (2.1), that is,*

$$y_j g_j(x) = 0,\ y_j \geq 0,\ g_j(x) \geq 0,\ y_j + g_j(x) > 0 \quad (j \in J_I)$$

is satisfied at (x^*, y^*).

From the investigation in Section 4, Assumption 5.1 (1) is not unrealistic. Moreover, Assumption 5.1 (2) is not so restrictive.

Let J_{A_*} be the index set of active constraints at x^*, that is,

$$J_{A_*} := \{j \in J_E \cup J_I \mid g_j(x^*) = 0\}.$$

We show that the next lemma holds under Assumptions 4.1 and 5.1.

Lemma 5.1 *Suppose that Assumptions 4.1 and 5.1 hold. Then, $J_{A_k} = J_{A_*}$ holds for sufficiently large k.*

Proof: From Assumption 5.1 (2), Lagrange multiplier which satisfies (2.1) at x^* is unique, thus $y^{SD^*} = y^*$ holds from Assumption 5.1 (1). If we assume that $J_{A_k} = J_{A_*}$ does not hold for sufficiently large k, one of the following cases holds:

(i) $y^{SD}_{k+1,j} = 0$ for any j which satisfies $y^{SD^*}_j > 0$,

(ii) $y^{SD}_{k+1,j} > 0$ for any j which satisfies $y^{SD^*}_j = 0$.

First, the case (i) does not occur because of $\lim_{k\to\infty} y^{SD}_{k+1,j} \neq y^{SD^*}_j$. Now we consider the case (ii). We have $g_j(x_k) + \nabla g_j(x_k)^T \Delta x^{SD}_k = 0$ for sufficiently large k. Thus $g_j(x^*) = 0$ holds from Assumption 5.1 (1). However, this contradicts Assumption 5.1 (2) because of $y^{SD^*}_j = 0$ in this case. This completes the proof. □

Let us consider the ordinary SQP method (without trust region constraints). The subproblem of this method is as follows:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\Delta x^T G_k \Delta x + \nabla f(x_k)^T \Delta x,\ \Delta x \in \Re^n \\ \text{subject to} & g_j(x_k) + \nabla g_j(x_k)^T \Delta x = 0\ (j \in J_E), \\ & g_j(x_k) + \nabla g_j(x_k)^T \Delta x \geq 0\ (j \in J_I). \end{array} \tag{5.1}$$

We can easily show from Lemma 5.1 that the KKT conditions of (5.1) coincide with (2.7) and (2.8), which are the KKT conditions of (2.6) for sufficiently large k. Thus we obtain $\Delta x^N_k = \Delta x^{SQP}_k$, where Δx^{SQP}_k is a solution of (5.1). It is known that the SQP method has fast local convergence under certain conditions. Consequently, we can expect that Δx^N_k is a good direction for fast convergence in a neighborhood of a solution.

6. Numerical Results

In this section, we present numerical results of Algorithm TRSQP.

6.1. Implementation and parameters

We coded Algorithm TRSQP in C++ and tested this implementation on various problems from Hock and Schittkowski's book [8] and CUTE/CUTEr problem archive [2, 7]. Algorithm TRSQP run on a 1.5 Ghz Pentium 4 processor with 1 Gbyte RAM.

In this experiment, we employ the methods which are explained in Remarks 4.1, 4.2, 4.3 and 4.4. Especially, in Remark 4.2, we set $\mu_k = 10^{-10}$ at first and replace μ_k with $\mu_k := 2\mu_k$ until (3.1) is satisfied, and in Remark 4.4, we set $L = 1.2$ and $\rho^{min} = 10^{-6}$. Moreover, we employ the Gill–Murray method [6] and the supernodal right-looking method [12] to solve (2.2) and (2.9) in Step 1 of Algorithm TRSQP, respectively.

All experiments were executed with following parameters and initial settings: $G_0 := \nabla^2 f(x)$, $(D_k)_{ii} := \max\{|(G_k)_{ii}|,\ 10^{-3}\}$ $(i = 1, 2, \ldots, n)$, $\delta_0 = \max\{\|\Delta x^{SD}_0\|, \|\Delta x^N_0\|\} \times 10^2$ and $M = 10^5$.

Table 1: Result for Hock & Schittkowski's problems

Number of solvable problems	113
Number of failed problems	2

Moreover, we stopped Algorithm TRSQP in Step 2 successfully when the following conditions are satisfied:

$$R = \max\{R_1, R_2, R_3, R_4, R_5\} \leq \epsilon, \tag{6.1}$$

where

$$\begin{aligned}
R_1 &= \|\nabla_x L(x,y)\|_1 \ / \ \max\{1, n\|\nabla f(x)\|\}, \\
R_2 &= \sum_{j \in J_E} |g_j(x)| \ / \ |J_E|, \\
R_3 &= \sum_{j \in J_I} |y_j g_j(x)| \ / \ |J_I|, \\
R_4 &= \sum_{j \in J_I} |\min\{0, y_j\}|, \\
R_5 &= \sum_{j \in J_I} |\min\{0, g_j(x)\}|
\end{aligned}$$

and $\epsilon = \sqrt{2} \times 10^{-6}$. We can consider R as an indicator of optimality and feasibility. Moreover, when (6.1) is not satisfied until Algorithm TRSQP is iterated 150 times or 3600 seconds are passed, we stop the calculation abnormally.

6.2. Hock & Schittkowski's problems

Hock & Schittkowski's book [8] contains 115 mathematical programming problems. All problems are small (1 – 16 variables and 0 – 21 constraints), and all problems has nonlinearity. We try to solve these problems by applying Algorithm TRSQP.

Table 1 shows the result for Hock & Schittkowski's problems. Algorithm TRSQP solves all problems except Problems No.13 and No.87. With regard to Problem No.13, none of the known constraint qualifications hold at a solution, so it seems that numerical difficulties have occurred. Moreover, the objective function of Problem No.87 is not differentiable.

6.3. CUTE/CUTEr problems

CUTE/CUTEr [2, 7] is the biggest archive of mathematical programming problems, as far as we know. We try to solve 55 problems which are treated as typical problems in [2, Figures 1, 2].

Table 2 shows the result for problems whose optimal solution is calculated successfully by Algorithm TRSQP. In Table 2, "# vars.", "# cons.", "val. obj.", "# itr.", "val. R" and "time(sec.)" mean the number of variables, the number of constraints, the value of objective function at an optimal solution, the number of Algorithm TRSQP's iterations, the value of R which is defined by (6.1) and the calculation time(sec.), respectively.

We can find from Table 2 that Algorithm TRSQP works well. Especially, Algorithm TRSQP can solve problems which have some thousands variables and constraints.

Table 3 shows the problems whose optimal solution can not be calculated by Algorithm TRSQP. With regard to most failed problems, it takes a lot of time to solve the subproblem (2.2).

Table 2: Results for CUTE/CUTEr problems which are solved successfully by Algorithm TRSQP

Problem name	# vars.	# cons.	val. obj.	# itr.	val. R	time(sec.)
AIRCRFTA	8	5	0.000000e+00	2	1.22e-06	0.00
BDVALUE	5002	5000	0.000000e+00	1	4.16e-08	0.49
BIGBANK	2230	1113	0.000000e+00	1	2.03e-16	2.63
BRATU2D	3200	2888	0.000000e+00	4	4.33e-12	43.69
BRATU2D	5184	4900	0.000000e+00	16	1.82e-09	893.99
BRIDGEND	2734	2727	4.056176e+01	1	2.68e-15	1.43
CHANDHEQ	100	100	0.000000e+00	9	9.54e-07	0.82
CHEMRCTA	5000	5000	0.000000e+00	70	6.95e-11	119.09
CLPLATEA	5041	0	-1.259209e-02	7	1.49e-06	351.64
DALLASL	906	667	-2.026039e+05	13	2.88e-07	1.39
EXPFITC	5	502	2.330257e-02	118	5.66e-10	0.74
GAUSSELM	506	1135	-1.000000e+00	21	1.56e-17	3.66
GRIDNETA	7564	3844	4.779795e+02	2	1.81e-16	28.37
HAGER4	5001	2500	2.794102e+00	3	8.27e-07	70.45
HIMMELBI	100	12	-1.75500e+03	2	3.74e-10	0.08
HIMMELBJ	45	14	-1.903851e+03	7	6.53e-18	0.12
HIMMELBK	24	13	5.000000e-02	37	3.41e-20	0.51
HS106	8	6	7.049248e+03	8	3.43e-10	0.01
HS107	9	6	5.055012e+03	6	1.26e-07	0.00
HS114	10	11	-1.768807e+03	6	1.09e-07	0.00
HS116	13	14	9.759103e+01	6	2.33e-13	0.01
HS68	4	2	-9.204250e-01	13	5.46e-08	0.01
HS73	4	3	2.989438e+01	3	1.33e-07	0.00
HS88	2	1	1.362657e+00	16	2.92e-10	0.04
HS93	6	2	1.350760e+02	5	9.10e-07	0.00
HYDCAR20	99	99	0.000000e+00	8	3.89e-07	0.07
HYDROELL	1009	1008	-3.585547e+06	2	3.51e-15	0.58
KOWOSB	4	0	3.078009e-04	16	3.18e-07	0.01
LCH	3000	1	-3.098858e+00	2	4.34e-14	30.90
MSQRTA	1024	1024	0.000000e+00	4	2.73e-09	122.53
NLMSURF	5625	0	3.894898e+01	39	3.20e-07	1927.46
ORTHREGA	8197	4096	2.665546e+04	9	7.20e-07	406.38
PALMER1C	8	0	9.760505e-02	2	6.98e-08	0.00
PENTAGON	6	15	1.365217e-04	10	1.56e-08	0.01
PRODPL1	60	29	3.573897e+01	7	3.55e-10	0.02
ROSENBR	2	0	2.232923e-16	21	8.90e-07	0.01
SSEBNLN	194	96	1.617060e+07	1	1.33e-10	0.01
STEENBRE	540	126	0.000000e+00	1	8.22e-14	0.92
TRIGGER	7	6	0.000000e+00	17	5.23e-07	0.01
VAREIGVL	50	0	1.256894e-11	17	7.89e-07	0.09

Table 3: CUTE/CUTEr problems which can not be solved by Algorithm TRSQP

Problem name	# vars.	# cons.
BRATU1D	5003	0
BRATU3D	4913	3375
GOULDQP2	19999	10000
HS87	6	4
JNLBRNGA	10000	1
LEWISPOL	6	9
MANNE	6000	4000
MEYER3	3	0
NYSTROM5	18	20
OBSTCLBU	10000	1
RAYBENDS	2050	0
READING3	4002	2001
SEMICON2	5002	5000
SVANBERG	5000	5000
TORSION1	5476	0

7. Concluding Remarks

The trust region SQP methods which have been proposed had two difficulties: one difficulty is the QP subproblem with the trust region constraint may not be feasible in considerable cases, and the other difficulty is the QP subproblem are not convex necessarily. In this paper, we propose a new trust region SQP method which resolves these two difficulties. We can expect that our method needs less calculated amount than before.

Future work concerning this paper is to improve the method which we propose for fast convergence. It is well-known that the SQP method may fail to have fast convergence because of Maratos effect [5]. However, the SQP method with nonmonotone line search which has the superlinear convergence property has been proposed [14]. We want to apply this strategy to the method which we propose in this paper.

Acknowledgment

We would like to thank Professor Masao Fukushima of Kyoto University and two anonymous referees for their helpful comments on the earlier version of this paper. We also thank Nobuhiro Henmi and Takahito Tanabe at Mathematical Systems Inc., who implemented the Gill–Murray method and the supernodal right-looking method, respectively, which are used in the implementation of Algorithm TRSQP.

References

[1] P. T. Boggs and J. W. Tolle: Sequential Quadratic Programming. In *Acta Numerica* (Cambridge University Press, Cambridge, 1995), 1-51.

[2] I. Bongartz, A. R. Conn, N. I. M. Gould and Ph. L. Toint: CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**(1995), 123-160.

[3] R. H. Byrd, R. B. Schnabel and G. A. Schultz: A trust region algorithm for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis*, **24**(1987), 1152-1170.

[4] M. R. Celis, J. E. Dennis and R. A. Tapia: A trust region strategy for nonlinear equality constrained optimization. In P. T. Boggs, R. H. Byrd and R. B. Schnabel (eds.): *Numerical Optimization 1984* (SIAM, Philadelphia, 1985), 71-82.

[5] R. Fletcher: *Practical Methods of Optimization, Second Edition* (John Wiley & Sons, New York, 1987).

[6] P. E. Gill and W. Murray: Numerically stable methods for quadratic programming. *Mathematical Programming*, **14**(1978), 349-372.

[7] N. I. M. Gould, D. Orban and Ph. L. Toint: CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, **29**(2003), 373-394.

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

[9] M. Lalee, J. Nocedal and T. Plantenga: On the implementation of an algorithm for large-scale equality constrained optimization. *SIAM Journal on Optimization*, **8**(1998), 682-706.

[10] E. O. Omojokun: *Trust region algorithms for optimization with nonlinear equality and inequality constraints* (Ph. D. Thesis, University of Colorado, Boulder, 1989).

[11] M. J. D. Powell and Y. Yuan: A trust region algorithm for equality constrained optimization. *Mathematical Programming*, **49**(1991), 189-211.

[12] E. Rothberg and A. Gupta: Efficient sparse matrix factorization on high performance workstations — exploiting the memory hierarchy. *ACM Transactions on Mathematical Software*, **17**(1991), 313-334.

[13] A. Vardi: A trust region algorithm for equality constrained minimization: convergence properties and implementation. *SIAM Journal on Numerical Analysis*, **22**(1985), 575-591.

[14] H. Yamashita and H. Yabe: A nonmonotone SQP method with global and superlinear convergence properties. *Technical report, Mathematical Systems, Inc.*, (1996, Revised 1998).

[15] H. Yamashita, H. Yabe and T. Tanabe: A globally and superlinearly convergent primal-dual interior point trust region method for large scale constrained optimization. *Mathematical Programming*, Ser.B, **102**(2005), 111-151.

Hiroshige Dan
Mathematical Systems Inc.
10F Four Seasons Bldg.
2-4-2 Shinjuku, Shinjuku
Tokyo 160-0022, Japan
E-mail: dan@msi.co.jp