Journal of Computational Mathematics, Vol.19, No.4, 2001, 337–346.

# GLOBAL CONVERGENCE AND IMPLEMENTATION OF NGTN METHOD FOR SOLVING LARGE-SCALE SPARSE NONLINEAR PROGRAMMING PROBLEMS[*1)]

Qin Ni
(*Department of Mathematics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*)

**Abstract**

An NGTN method was proposed for solving large-scale sparse nonlinear programming (NLP) problems. This is a hybrid method of a truncated Newton direction and a modified negative gradient direction, which is suitable for handling sparse data structure and possesses Q-quadratic convergence rate. The global convergence of this new method is proved, the convergence rate is further analysed, and the detailed implementation is discussed in this paper. Some numerical tests for solving truss optimization and large sparse problems are reported. The theoretical and numerical results show that the new method is efficient for solving large-scale sparse NLP problems.

*Key words*: Nonlinear programming, Large-scale problem, Sparse.

## 1. Introduction

Consider the following NLP problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_j(x) \geq 0, \quad j \in J = \{1, \ldots, m\}. \end{aligned} \tag{1.1}$$

where the function $f : R^n \to R^1$ and $g_j : R^n \to R^1,\ j \in J$ are twice continuously differentiable. In particular, we discuss the case, where the number of variables and the number of constraints in (1.1) are large and second derivatives in (1.1) are sparse.

There are some methods which can solve large-scale problems, e.g. Lancelot in [2] and TDSQPLM in [9]. But they can not take advantage of sparse structure of the problem. A new efficient method which is called NGTN method is studied in [11] for solving large-scale sparse NLP problems. In this method, a new nonlinear system which is equivalent to Kuhn-Tucker conditions of the problem is developed. NCP function is used in the nonlinear system such that the nonnegativity of some variables is avoided. The truncated Newton method is used to solve the nonlinear system. In order to guarantee the global convergence, a robust loss function is chosen as a merit function and a modified negative gradient direction is used to descrease the merit function. This NGTN method is easy to carry out, possesses Q-quadratic convergence rate, and is suitable for solving large-scale sparse NLP problems.

In this paper, the global convergence of NGTN method is proved, Q-quadratical convergence rate is further analysed, and the detailed implementation is discussed. In addition, NGTN algorithm is used for solving the sparse truss problems, the dimensions of which range from 183 to 827, and large problem with 70000 variables and 40000 constraints. The theoretical and numerical results show that NGTN method is efficient for solving large-scale sparse NLP

* Received April 9, 1998; Final revised December 2, 1998.
[1)] This research was supported by Nationa Natural Science Foundation of China, LSEC of CAS in Beijing and Natural Science Foundation.

problems. This paper is organized as follows. In Section 2 we give the construction of a NGTN algorithm for solving the large-scale problem (1.1). We discuss the global convergence of NGTN in Section 3. The detailed implementation and numerical results of NGTN are given in Section 4.

## 2. NGTN Algorithm

### 2.1. Notations

In order to describe the NGTN algorithm, let

$$L(x,u) = f(x) - \sum_{j=1}^{m} u_j g_j(x) \tag{2.1}$$

be the Lagrangian function of problem (1.1), let $\bigtriangledown_x L(x,u)$ and $\bigtriangledown^2_{xx} L(x,u)$ denote the gradient and Hessian of $L(x,u)$ at $x$, respectively. With this notation, a pair $(x^*,u^*)$ is called a Kuhn-Tucker pair of (1.1) if $(x^*,u^*)$ satisfies the following Kuhn-Tucker conditions:

$$\bigtriangledown_x L(x^*,u^*) = 0,\ g_j(x^*) - t_j^* = 0,\ j \in J,$$
$$u_j^* t_j^* = 0,\ u_j^* \geq 0,\ t_j^* \geq 0,\ j \in J.$$

The Kuhn-Tucker conditions are equivalent to the system

$$0 = \quad q(z) = \begin{bmatrix} \bigtriangledown_x L(x,u) \\ - - - - - - - - \\ t_j - g_j(x) \\ - - - - - - - - \\ \phi_j(u,t) \end{bmatrix}_{j \in J} \tag{2.2}$$

where $q : R^{n+2m} \to R^{n+2m}$, $z = (x,u,t)$ and

$$\phi_j(u,t) = \sqrt{u_j^2 + t_j^2} - (u_j + t_j) \tag{2.3}$$

is a NCP-function (see [3]).

The Jacobian matrix of $q(z)$ is

$$Q(z) = \begin{pmatrix} \bigtriangledown^2_{xx} L(x,u) & -A^T(x) & 0 \\ -A(x) & 0 & I_m \\ 0 & \Phi_1(u,t) & \Phi_2(u,t) \end{pmatrix}, \tag{2.4}$$

where

$$\begin{aligned} A(x) &= (\bigtriangledown g_1(x), \ldots, \bigtriangledown g_m x))^T \in R^{m \times n}, \\ \Phi_1(u,t) &= \mathrm{diag}(\frac{\partial \phi_1(u,t)}{\partial u_1}, \ldots, \frac{\partial \phi_m(u,t)}{\partial u_m}), \\ \Phi_2(u,t) &= \mathrm{diag}(\frac{\partial \phi_1(u,t)}{\partial t_1}, \ldots, \frac{\partial \phi_m(u,t)}{\partial t_m}). \end{aligned}$$

$(\Delta x_k, \Delta \tilde{u}_k)$ is called a truncated solution, if the following equation

$$\begin{pmatrix} H_k & -\tilde{A}_k^T \\ -\tilde{A}_k & -D_k \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \tilde{u} \end{pmatrix} = \begin{pmatrix} b_{k1} \\ b_{k2} \end{pmatrix} \tag{2.5}$$

is solved such that the inequality

$$\| \begin{pmatrix} H_k & -\tilde{A}_k^T \\ -\tilde{A}_k & -D_k \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta \tilde{u} \end{pmatrix} - \begin{pmatrix} b_{k1} \\ b_{k2} \end{pmatrix} \| \leq \eta_k \tag{2.6}$$

holds for some $\eta_k > 0$. Here

$$\tilde{A}_k^T = (\bigtriangledown g_j(x_k))_{j \in J_k} \in R^{n \times m_k}, J_k = \{j \in J :\ |\eta_j^{(k)}| \geq \epsilon_1 \ \text{ or } \ |\xi_j^{(k)}| \leq \epsilon_2\} \tag{2.7}$$

for the constants $\epsilon_1 > 0,\ \epsilon_2 > 0$, $m_k$ is the cardinal number of the index $J_k$,

$$\xi_j^{(k)} = \frac{\partial \phi_j(u_k, t_k)}{\partial u_j^{(k)}},\quad \eta_j^{(k)} = \frac{\partial \phi_j(u_k, t_k)}{\partial t_j^{(k)}}.$$

$$H_k = \bigtriangledown_{xx}^2 L(x_k, u_k) + \sum_{j \in J/J_k} \eta_j^{(k)} (\xi_j^{(k)})^{-1} \bigtriangledown g_j(x_k) \bigtriangledown g_j(x_k)^T \tag{2.8}$$

$$b_{k1} = -\bigtriangledown_x L(x_k, u_k) - \sum_{j \in J/J_k} [\phi_j(u_k, t_k) + \eta_j^{(k)} (g_j(x_k) - t_j^{(k)})] \bigtriangledown g_j(x_k)/\xi_j^{(k)}, \tag{2.9}$$

$$b_{k2} = (g_j(x_k) - t_j^{(k)} + \phi_j(u_k, t_k)/\eta_j^{(k)})_{j \in J_k} \in R^{m_k}, \tag{2.10}$$

$$D_k = \mathrm{diag}\left(\xi_j^{(k)} (\eta_j^{(k)})^{-1}\right)_{j \in J_k}. \tag{2.11}$$

In order to guarantee global convergence, consider the following merit function

$$F_h(z) = \sum_{j=1}^{n+2m} \rho_{h_j}(q_j(z)) \tag{2.12}$$

where

$$\rho_{h_j}(v) = \begin{cases} v^2/2 & |v| \le h_j \\ h_j|v| - h_j^2/2 & |v| > h_j \end{cases}$$

for the constants $h_j \ge 1,\ j = 1, \ldots, n+2m, z = (x, u, t)$, $q_j(z)$ is the $j$-th component of $q(z)$ in (2.2).

The gradient of $F_h(z)$ is

$$\bigtriangledown F_h(z) = Q^T(z) q_h(z) \tag{2.13}$$

where

$$\begin{aligned} q_h(z) &= (\sum_{j \in J_h} q_j e_j + \sum_{j \in K_h} sign(q_j) h_j e_j) \\ J_h &= \{j :\ 1 \le j \le n+2m,\ |q_j(z)| \le h_j\}, \\ K_h &= \{1, \ldots, n+2m\}/J_h \end{aligned}$$

$e_j \in R^{n+2m}$ is the $j$-th column vector of unit matrix in $R^{(n+2m)\times(n+2m)}$ .

It is noted that

$$F_h(z) = \frac{1}{2} \sum_{j=1}^{n+2m} q_j^2(z),\ \text{when } |q_j(z)| \le h_j,\ j = 1, \ldots, n+2m. \tag{2.14}$$

### 2.2. NGTN Algorithm

In the following we describe the NGTN algorithm, which is a hybrid method of modified negative gradient direction and truncated Newton direction.

**NGTN Algorithm.**

**Step 0.** Choose $\epsilon_1, \epsilon_2 > 0, \epsilon_3, \epsilon_4 \in (0, 0.5), \beta \in (0, 0.1], h_j \ge 1, j = 1, \ldots, n+2m; x_0 \in R^n, u_0, t_0 \in R^m$. Compute $q(z_0)$, where $z_0 = (x_0, u_0, t_0)$. Set $k = 0$.

**Step 1.** If the termination conditions are satisfied, then stop.

1.1) If $z_k$ satisfies $|q_j(z_k)| \le h_j,\ j = 1, \ldots, n+2m$, then go to Step 2.

1.2) Set

$$\Delta z_k = \begin{pmatrix} \Delta x_k \\ \Delta u_k \\ \Delta t_k \end{pmatrix} = -\Lambda_k \bigtriangledown F_h(z_k),$$

where $\Lambda_k = \mathrm{diag}[\lambda_1, \ldots, \lambda_{n+2m}], \lambda_j = (\max\{1, \|Q(z_k)e_j\|_2^2\})^{-1}$. s=1, go to Step 3.

**Step 2.** Determine $(\Delta x_k, \Delta u_k, \Delta t_k)$ by using truncated Newton method.

2.1) Compute $J_k, \tilde{A}_k, H_k, D_k, b_{k1}$ and $b_{k2}$ according to (2.7)-(2.11);

2.2) Determine $(\Delta x_k, \Delta \tilde{u}_k)$ which satisfies

$$\| \begin{pmatrix} H_k & -\tilde{A}_k^T \\ -\tilde{A}_k & -D_k \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta \tilde{u}_k \end{pmatrix} - \begin{pmatrix} b_{k1} \\ b_{k2} \end{pmatrix} \| < \frac{1}{4}\|q_h(z_k)\|. \tag{2.15}$$

Set s=2, and

$$\Delta t_k = A(x_k)\Delta x_k + g(x_k) - t_k,$$

$$\Delta u_j^{(k)} = -(\phi_j(u_k, t_k) + \frac{\partial \phi_j(u_k, t_k)}{\partial t_j^{(k)}} \Delta t_j^{(k)}) / \frac{\partial \phi_j(u_k, t_k)}{\partial u_j^{(k)}},\ j \in J/J_k.$$

**Step 3.** Determine a steplength $\alpha_k$ .

If $s = 1$, then find an $\alpha_k$ such that

$$-\alpha_k(1-\epsilon_3)\| \bigtriangledown F_h(z_k)\|^2 \le F_h(z_k + \alpha_k \Delta z_k) - F_h(z_k) \le -\epsilon_4 \alpha_k \| \bigtriangledown F_h(z_k)\|^2. \tag{2.16}$$

If $s = 2$, then find an $\alpha_k$ such that

$$F_h(z_k + \alpha_k \Delta z_k) \le (1 - \beta\alpha_k) F_h(z_k). \tag{2.17}$$

Set $z_{k+1} = z_k + \alpha_k \Delta z_k,\ k = k+1$. If $\alpha_k \le 0.01$, then go to 1.2), otherwise go to Step 1.

**Remark.** The truncated solution in Step 2.2 is obtained by using conjugate gradient method for solving the problem (2.5), and the detailed description refers to [11]. If in Step 1.2 $\bigtriangledown F_h(z_k) = 0$ and $F_h(z_k) \neq 0$, then this is a degenerate case and can be skiped by some perturbation of the components of $z_k$.

## 3. Convergence Analysis

In order to discuss the convergence of NGTN Algorithm, the following assumptions are considered.

**Assumption 1 (A1)**

The point $(x^*, u^*, t^*)$ satisfies the following conditions,

(i) $u_j^* t_j^* = 0,\ |u_j^*| + |t_j^*| \neq 0,\ j = 1, \ldots, m.$

(ii)The gradients $\bigtriangledown g_j(x^*),\ j \in J(u^*)$ are linearly independent.

(iii) $x^T \bigtriangledown_{xx}^2 L(x^*, u^*)x > 0$ for all $x \neq 0$ with

$\bigtriangledown g_j(x^*)^T x = 0,\ j \in J(u^*)$, where

$$J(u^*) = \{j :\ u_j^* \neq 0\}. \tag{3.1}$$

It is noted that if $(x^*, u^*)$ is a Kuhn-Tucker pair of (1.1), $t^* = g(x^*)$ , then the conditions in A1 are Jacobian uniqueness condition. In addition, A1 resembles some conditions to non-Kuhn-Tucker vectors in [12].

**Assumption 2 (A2)**

The functions $\bigtriangledown^2 f$ and $\bigtriangledown^2 g_j$ $(j \in J)$ are Lipschitz-continuous in a neighbourhood of $x^*$, i.e. there are positive numbers $R$ and $L$ such that

$$\max\{\| \bigtriangledown^2 f(x) - \bigtriangledown^2 f(\bar{x})\|,\ \| \bigtriangledown^2 g_j(x) - \bigtriangledown^2 g_j(\bar{x})\|,\ j \in J\} \le L\|x - \bar{x}\|$$

for all $x,\ \bar{x} \in B(x^*, R)$. The set $B(x^*, R) = \{x :\ \|x - x^*\| \le R\}$ denotes the closed ball with the center $x^*$ and the radius $R$.

In order to ensure the descent property of NGTN algorithm, the truncated solution $(\Delta x_k, \Delta \tilde{u}_k)$ in Step 2 should satisfy a descent condition. The following lemma shows that the condition is easily satisfied.

**Lemma 3.1.** *Let $\Delta z_k = (\Delta x_k, \Delta u_k, \Delta t_k)$ be generated by Step 2 in NGTN Algorithm. Assume that $(\Delta x_k, \Delta \tilde{u}_k)$ satisfies the condition (2.15) and the used vector norm is Euclidean norm.*

*Then*

$$(\bigtriangledown F_h(z_k))^T \Delta z_k \leq -\frac{1}{2} F_h(z_k). \tag{3.2}$$

*Proof.* Let $(\Delta x_k, \Delta \tilde{u}_k)$ satisfy

$$M_k \begin{pmatrix} \Delta x \\ \Delta \tilde{u} \end{pmatrix} - \begin{pmatrix} b_{k1} \\ b_{k2} \end{pmatrix} + r_k \;=\; 0,$$

where

$$M_k = \begin{pmatrix} H_k & -\tilde{A}_k^T \\ -\tilde{A}_k & -D_k \end{pmatrix}.$$

From Lemma 3.2 in [11], we have

$$Q(z_k)\Delta z_k + q(z_k) + \theta_k = 0$$

and $\|\theta_k\| = \|r_k\|$. Hence,

$$(\bigtriangledown F_h(z_k))^T \Delta z_k = q_h(z_k)^T Q(z_k) \Delta z_k = q_h(z_k)^T (-q(z_k) - \theta_k).$$

where $q_h(z_k)$ refers to (2.13). Because

$$q_h(z_k)^T q(z_k) = \sum_{j \in J_h} q_j^2(z_k) + \sum_{j \in K_h} |q_j(z_k)| h_j \geq F_h(z_k),$$

and

$$F_h(z_k) \geq \frac{1}{2} \sum_{j \in J_h} q_j^2(z_k) + \frac{1}{2} \sum_{j \in K_h} h_j^2 = \frac{1}{2} \|q_h(z_k)\|^2,$$

we obtain

$$\begin{aligned} (\bigtriangledown F_h(z_k))^T \Delta z_k &\leq -q_h(z_k)^T q(z_k) + \|q_h(z_k)\| \|\theta_k\| \\ &\leq -F_h(z_k) + \frac{1}{4} \|q_h(z_k)\|^2 \leq -\frac{1}{2} F_h(z_k). \end{aligned}$$

This theorem is proved.

If $M_k$ is nonsingular, the condition (2.15) is easily satisfied. If $M_k$ is singular, some strategies are added such that (2.15) and hence (3.2) are also satisfied. The detailed description will be given in the following section. In addition, if $\Delta z_k = (\Delta x_k, \Delta u_k, \Delta t_k)$ is generated by Step 1.2 in NGTN Algorithm, then this is a descent direction when $\bigtriangledown F_h(z_k) \neq 0$.

The following lemma shows that the Jacobian matrix $Q(z)$ is nonsingular if $z$ satisfies A1.

**Lemma 3.2.** *Let $z^* = (x^*, u^*, t^*)$ be an accumulation point of a sequence $\{z_k\}$ produced by NGTN algorithm. Suppose that the assumption A1 holds for $z^*$. Then the Jacobian matrix $Q(z^*)$ is nonsingular.*

*Proof.* According to the assumption A1, if $j \in J(u^*)$, then $t_j^* = 0,\ u_j^* \neq 0$, which implies

$$\frac{\partial \phi_j(u^*, t^*)}{\partial u_j} = 0,\ \frac{\partial \phi_j(u^*, t^*)}{\partial t_j} = -1, \text{ for } j \in J(u^*). \tag{3.3}$$

If $j \in J/J(u^*)$, then we have $t_j^* \neq 0,\ u_j^* = 0$,

$$\frac{\partial \phi_j(u^*, t^*)}{\partial u_j} = -1,\ \frac{\partial \phi_j(u^*, t^*)}{\partial t_j} = 0, \text{ for } j \in J/J(u^*). \tag{3.4}$$

The Jacobian matrix $Q(z^*)$ is

$$Q(z^*) = \begin{pmatrix} \bigtriangledown_{xx}^2 L(x^*, u^*) & -A^T(x^*) & 0 \\ -A(x^*) & 0 & I_m \\ 0 & \Phi_1(u^*, t^*) & \Phi_2(u^*, t^*) \end{pmatrix}.$$

Let $Q(z^*)q = 0, q = (q^{(1)}, q^{(2)}, q^{(3)}) \in R^{(n+2m)}$ a suitably partitioned vector. Then

$$\begin{aligned} \nabla^2_{xx} L(x^*, u^*)q^{(1)} - A^T(x^*)q^{(2)} &= 0 \quad (3.5) \\ -A(x^*)q^{(1)} + q^{(3)} &= 0 \quad (3.6) \\ \Phi_1(u^*, t^*)q^{(2)} + \Phi_2(u^*, t^*)q^{(3)} &= 0 \quad (3.7) \end{aligned}$$

Using (3.3)-(3.7), we obtain that

$$\begin{cases} q_j^{(2)} = 0 & \text{for } j \in J/J(u^*), \\ q_j^{(3)} = 0, \ \nabla g_j(x^*)^T q^{(1)} = 0, & \text{for } j \in J(u^*), \end{cases} \quad (3.8)$$

which implies

$$q^{(2)T} A(x^*) q^{(1)} = 0.$$

Multiplication of (3.5) with $q^{(1)T}$ yields

$$q^{(1)T} \nabla^2_{xx} L(x^*, u^*) q^{(1)} = q^{(1)T} A^T(x^*) q^{(2)} = 0.$$

This equality, together with (3.8) and (iii) in A1, implies

$$q^{(1)} = 0.$$

From (3.5) and (3.6), it follows that

$$q^{(3)} = 0, \ A^T(x^*) q^{(2)} = 0.$$

Becauce of (3.8) and (ii) in A1, we get

$$q^{(2)} = 0.$$

Hence, $q = 0$, and this proves the theorem.

In order to prove the global convergence of NGTN Algorithm, we define the index set

$$S = \{k \in N : \ x_{k+1} \text{ is computed by Step 1.2 and Step 3 in NGTN Algorithm}\}$$

where $N$ is the set of natural numbers.

**Theorem 3.3.** *Let $z_k = (x_k, u_k, t_k)$ be iterate generated by NGTN Algorithm. Assume that any accumulation point $(x^*, u^*, t^*)$ of $\{z_k\}$ satisfies A1. Then either NGTN Algorithm terminates after a finite number of iterations with $z_k = z^*$ or $\{z_k\}$ convergens to $z^*$, where $z^* = (x^*, u^*, t^*)$, where $(x^*, u^*)$ is a Kuhn-Tucker pair of (1.1).*

*Proof.* From (2.12) we have that if $F_h(z) = 0$, then $q(z) = q(x, u, t) = 0$, which implies that $(x, u)$ is a Kuhn-Tucker pair of (1.1). Hence we only prove that $F_h(z_k) = 0$ or $F_h(z_k) \to 0$. Now suppose $F_h(z_k) \neq 0$ for $k = 1, \ 2, \ \ldots$, and consider two cases.

(i) The index set $S$ is infinite. In the k-th iteration step, $k \in S$, the modified steepest descent method is carried out. In the other k-th iteration step, $k \notin S$, $F_h(z_k)$ is not increased. Hence, from Theorem 10.1 in [1], every accumulation point $z^*$ of $\{z_k\}$ satisfies $\nabla F_h(z^*) = 0$. From A1 and Lemma 2, it follows that $Q(z^*)$ is nonsingular. This, together with (2.13), implies $F_h(z^*) = 0$.

(ii) The index set $S$ is finite. Then there exists a $k_1$ such that

$$\alpha_k \geq 0.01, \quad F_h(z_{k+1}) \leq (1 - \beta\alpha_k) F_h(z_k), \quad \text{for all } \ k \geq k_1,$$

which implies $F_h(z_k) \to 0$. Thus this theorem is right.

In order to prove that $\alpha_k = 1$ is admissible for $k \geq k_0$, the following assumption is added.

**Assumption 3 (A3)**

Suppose that $z^*$ is a solution of (2.2). The function $F_h : R^{n+2m} \to R$ satisfies

$$F_h(z + w) - F_h(z) = w^T \nabla F_h(z) + o(|F_h(z)|),$$

for all $z \in B(z^*, r_1), \ w \in B(0, r_2)$, where $\lim_{t \to 0} o(t)/t = 0$.

**Lemma 3.4.** *Let $z_k = (x_k, u_k, t_k)$ be iterate generated by NGTN algorithm. Assume*
(i) *$z_k \to z^*$, for $k \to \infty$, where $z^*$ solves (2.2).*
(ii) *$F_h(z)$ satisfies A3 for some positive constants $r_1$ and $r_2$.*
(iii) *$\epsilon_1, \epsilon_2$ in $J_k$ are less than 0.5.*
(iv) *The assumptions A1 and A2 hold for $z^*$.*
*Then there exists a $k_0$ such that in Step 3 of NGTN algorithm $\alpha_k = 1$ is admissible for $k \geq k_0$.*

*Proof.* From Lemma 3.1 in [11], we obtain

$$M^* = \begin{pmatrix} \nabla^2_{xx} L(x^*, u^*) & -\tilde{A}^T(x^*) \\ -\tilde{A}(x^*) & 0 \end{pmatrix}$$

is nonsingular, where

$$\tilde{A}(x^*) = (\nabla g_j(x^*))^T_{j \in J(x^*,0)},$$

and $J(x^*, 0) = \{j,\ g_j(x^*) = 0\}$.

From the condition (i) and A1, it follows that $t^* = g(x^*)$ and $J(u^*) = J(x^*, 0)$. According to similar deduction in Lemma 3.2, we can obtain

$$\frac{\partial \phi_j(u^*, t^*)}{\partial u_j} = 0,\ \frac{\partial \phi_j(u^*, t^*)}{\partial t_j} = -1, \text{ for } j \in J(x^*, 0).$$

and

$$\frac{\partial \phi_j(u^*, t^*)}{\partial u_j} = -1,\ \frac{\partial \phi_j(u^*, t^*)}{\partial t_j} = 0, \text{ for } j \in J/J(x^*, 0).$$

Hence, from the definition of $J_k$ in (2.7) and the condition (iii), there exists a $k_1$ such that

$$J_k = J(x^*, 0),\quad J/J_k = J/J(x^*, 0),\quad \text{for } k \geq k_1.$$

Therefore, from the definition of $H_k$ and $D_k$, we have

$$M_k = \begin{pmatrix} H_k & -A_k^T \\ -A_k & -D_k \end{pmatrix} \to M^*,$$

which means that there exists a $k_2 (\geq k_1)$ such that $M_k$ is nonsingular for $k \geq k_2$.Thus, when $q_h(z_k) \neq 0$, (2.15) and (3.2) are always satisfied for $k \geq k_2$.

The condition (i) means that $\Delta z_k \to 0, q(z_k) \to 0$. Then there exists a $k_3$ such that $z_k \in B(x^*, r_1), \Delta z_k \in B(0, r_2), |q_j(z_k)| < h_j$, for all $k \geq k_3 (\geq k_2)$.

Hence, from Lemma 3.1, we obtain

$$F_h(z_k + \Delta z_k) - F_h(z_k) = \Delta z_k^T \bigtriangledown F_h(z_k) + o(|F_h(z_k)|) < -\frac{1}{4} F_h(z_k),$$

which means

$$F_h((z_k + \Delta z_k) \leq (1 - \beta) F_h(z_k)$$

for $k \geq k_3$. Thus, $\alpha_k = 1$, and Steps 2-3 are carried out continuously for $k \geq k_3$.

From Theorem 3.3 in [11], Theorem 3.3 and Lemma 3.4 in this paper, we obtain the following theorem.

**Theorem 3.5.** *Let $z_k$ be iterate generated by NGTN algorithm. Assume that the conditions of Lemma 3.4 hold. Then $\{z_k\}$ generated by NGTN Algorithm converges Q-quadratically to $z^*$, where $z^* = (x^*, u^*, t^*)$ and $(x^*, u^*)$ is a Kuhn-Tucker pair of (1.1).*

## 4. Implementation and Numerical Tests

In this section, we discuss some detailed implementation of NGTN algorithm and give some numerical results for medium-sized truss optimization and large sparse problems.

### 4.1. Implementation of NGTN Algorithm

In order to guarantee the numerical stability, we determine the truncated solution $(\Delta x_k, \Delta\tilde{u}_k)$ by solving the problem

$$(\frac{M_k^T M_k}{\|M_k\|^2} + \omega I)\begin{pmatrix} \Delta x \\ \Delta\tilde{u} \end{pmatrix} = \frac{M_k^T}{\|M_k\|^2}\begin{pmatrix} b_{k1} \\ b_{k2} \end{pmatrix} \tag{4.1}$$

instead of the problem (2.5), where $\omega$ is a damping factor. If $M_k$ is nonsingular and $\omega = 0$, then the problem (4.1) is equivalent to the problem (2.5). At first, (4.1) is solved with $\omega = 0$. If the truncated solution does not generate a good descent direction, then (4.1) is solved with $\omega = \frac{1}{(n+m_k)^2}$ in the next iteration.

In addition, when the modified negative gradient $\Delta z_k$ is chosen as a search direction, the initial value of $\alpha$ is set to

$$\min[1, \frac{-0.2F_h(z_k)}{\bigtriangledown F_h(z_k)^T \frac{\Delta z_k}{\|\Delta z_k\|}}]\frac{1}{\|\Delta z_k\|}.$$

Numerical results show that this decreases the inner steps in line search.

In NGTN algorithm, we choose the suitable values of the constants $\epsilon_1,\ \epsilon_2,\ \epsilon_3,\ \epsilon_4, \beta$ (see Step 0 ) by

$$\epsilon_1 = 0.001,\ \epsilon_2 = 0.001, \epsilon_3 = 0.1,\ \epsilon_4 = 10^{-4}, \beta = 0.1,$$

and

$$h_j = \max\{2.5,\ 10^{-3}q_j(z_0)\}, j = 1, \ldots, n + 2m.$$

**4.2. Numerical Results**

From searching optimal topology of trusses arise some special sparse large-scale problems. In order to ensure that Assumption 2 in Section 3 is satisfied for the truss problem, we consider some perturbation of the objective function and obtain

$$\begin{aligned} \min \quad & f(x) = \lambda v - t^T y + \sum_{i=1}^{m} w_i + \tfrac{1}{2}\alpha(y^T y + \lambda^2 + \sum_{i=1}^{m} w_i^2) \\ \text{s.t.} \quad & g_{2i-1}(x) = w_i - (\tfrac{1}{2}y^T A_i y - \lambda)L_i \geq 0 \\ & g_{2i}(x) = w_i - (\tfrac{1}{2}y^T A_i y - \lambda)U_i \geq 0, i = 1, \ldots, m. \end{aligned} \tag{4.2}$$

where $x = (y, \lambda, w) \in R^{n_v}, A_i \in R^{n\times n},\ i = 1, \ldots, m$ are sparse positive semi-definite matrices, $t \in R^n$ and $\alpha \in (0, 1)$. The detailed description of the problem refers to [8]. The used termination conditions are

$$\sum_{j=1}^{2m} |\min(0, g_j(x))| \leq \tau_1 \tag{4.3}$$

$$\| \bigtriangledown_x L(x, u)\|_\infty \leq \tau_2, \tag{4.4}$$

where

$$L(x, u) = f(x) - \sum_{j=1}^{2m} u_j g_j(x).$$

Table 1. Truss Test Problems

| TRP | $n_v$ | $m_c$ | $n$ | $m$ | $v$ | $L$ | $U$ |
|---|---|---|---|---|---|---|---|
| TR1 | 183 | 280 | 42 | 140 | 10 | 0.001 | 10 |
| TR2 | 186 | 282 | 44 | 141 | 10 | 0.01 | 10 |
| TR3 | 280 | 452 | 54 | 226 | 10 | 0.001 | 9 |
| TR4 | 308 | 502 | 56 | 251 | 10 | 0.001 | 8 |
| TR5 | 391 | 652 | 64 | 326 | 10 | 0.001 | 8 |
| TR6 | 583 | 1004 | 80 | 502 | 10 | 0.01 | 10 |
| TR7 | 644 | 1118 | 84 | 559 | 10 | 0.001 | 9 |
| TR8 | 827 | 1460 | 96 | 730 | 10 | 0.001 | 10 |

Table 2. Numerical Results for Truss Test Problems

| TRP | CPU | IT | TER1 | TER2 |
|---|---|---|---|---|
| TR1 | 5.09 | 38 | 5.73D-5 | 3.40D-5 |
| TR2 | 10.72 | 31 | 3.75D-6 | 4.38D-5 |
| TR3 | 14.50 | 45 | 7.54D-6 | 8.10D-6 |
| TR4 | 19.84 | 52 | 1.46D-5 | 6.36D-4 |
| TR5 | 22.34 | 38 | 3.73D-5 | 2.65D-4 |
| TR6 | 48.76 | 41 | 1.68D-7 | 6.34D-5 |
| TR7 | 59.34 | 32 | 5.34D-6 | 5.74D-4 |
| TR8 | 147.56 | 25 | 1.78D-5 | 6.75D-5 |

Table 3. Numerical Results for Large HS-100 Problems

| N | M | CPU | IT | TER1 | TER2 |
|---|---|---|---|---|---|
| 7000 | 4000 | 75.36 | 353 | 0.728D-9 | 0.132D-10 |
| 21000 | 12000 | 422.72 | 678 | 0.213D-10 | 0.510D-10 |
| 70000 | 40000 | 1153.64 | 74 | 0.217D-4 | 0.627D-5 |

In our numerical tests, we let

$$\tau_1 = 0.0001, \quad \tau_2 = 0.001,$$

the initial point $x_0$ be not feasible.

Now NGTN algorithm is used for solving the problem (4.2). Some medium-sized truss test problems are given in Tabel 1. In this table, $n_v$ denotes the number of the variables, and $m_c$ the number of the constraints. $n$, $m$, $v$ are the same as in (4.2). $L_i$ and $U_i$ in (4.2), $i = 1, \ldots, m$, are chosen as the same values $L$ and $U$, respectively. In the eight truss test problems, the number of variables ranges from 183 to 827, while the number of constraints ranges 280 to 1460.

These test problems have been performed on an SGI INDY. All calculations, test problems and optimization codes are carried out in double precision and in Fortran program. Numerical results are given in Tabel 2, where

**TRP:** Truss test problem,

**CPU:** Execution time in seconds,

**IT:** Number of the outer iterations,

**TER1:** Termination condition (4.3),

**TER2:** Termination condition (4.4).

Another large sparse test problem is obtained by the generalization of the HS-100 problem in [5]. The number of the variables, $n$, is chosen to be 7000, 21000 and 70000. The number of constraints, $m$, is chosen to be 4000, 12000 and 40000.

This problem is also used to test NGTN algorithm on an SGI INDY. Numerical results are given in Tabel 3, where

**N:** Number of the variables,

**M:** Number of constraints,

CPU, IT, TER1 and TER2 are the same as above.

From Tables 2 and 3, we can see that all test problems were solved in proper time. These problems represent two kinds of large sparse problems. Thererfore our numerical results showed that NGTN algorithm can handle large sparse problems.

## 5. Remarks

It is noted that there are some optimality-condition based methods which also use NCP functions (see [3] and [4]). The main contribution of this paper is to develop a robust NGTN algorithm for large-scale sparse NLP problems. The global convergence and Q-quadratical convergence rate of NGTN method are also proved. In comparison with other algorithm for large-scale NLP problems, NGTN algorithm can not only handle large-scale sparse problems but also possess Q-quadratical convergence rate. In addition, NGTN algorithm does not require the feasiblity of the initial point, which is needed in hybrid methods in [6].

The Fortran code of NGTN method needs to be further developed. The extension to the method for solving problem (1.1) including equality constraints will be not difficult. The conjugte gradient direction could be used to replace the modified negative gradient direction. In addition, the trust region stradegies have strong global convergence properties and are usually very suitable for least-square calculation (see [7]). These might improve NGTN method.

**Acknowledgment.** I would like to thank referee for his helpful comments and suggestions on a previous version of this paper.

## References

[1] M. Avril, Nonlinear Programming Analysis and Methods, Prentice-Hall Co. Inc., 1976.

[2] A.R. Conn, N.I.M. Gould, P.L. Toint, LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A). Number 17 in Springer Series in Computational Mathematics, Springer Verlag, Heidelberg, Berlin, New York, 1992.

[3] A. Fischer, A special Newton-type optimization method, *Optimization,* **24** (1992), 269-284.

[4] C. Kanzow, H. Kleinmichel, A class of Newton type methods for equality and inequality constrained optimizatio, *Optimization Methods and Software,* **5** (1995), 173-198.

[5] W. Hock, K. Schittkowski, Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems 187, Springer, Berlin, Heidelberg, 1981.

[6] H. Kleinmichel, C. Richter, K. Schönefeld, On a class of hybrid methods for smooth constrained optimization, *JOTA,* **73** (1992), 465-500.

[7] L. Luksan, Hybrid methods for large sparse nonlinear least squares, *JOTA* **89**:3 (1996), 575-595.

[8] Q. Ni, General large-scale nonlinear programming using sequential quadratic programming methods, *Bayreuther Mathematischen Schriften,* **45** (1993), 133-236.

[9] Q. Ni, Truncated Dual SQP method with limited memory, *Optimization Methods and Software,* **6** (1995), 25-57.

[10] Q. Ni, A QP-free, Truncated Hybrid Method for Large-Scale Nonlinear Sparse Constrained Optimization, *J. Comput. Math.,* **17** (1997), 36-54.

[11] Q. Ni, D. Fu, A new efficient method for solving nonlinear programming problems with inequality constraints, *OR Transaction,* **3** (1999), 31-39.

[12] J.S. Pang, A B-differentiable equation-based, globally and locally quadratically convergent algorithm for nonlinear programs, complementarity and variational inequality problems, *Math. Prog.,* **51** (1991), 101-131.