No. 722

An Algorithm for Strictly Convex Quadratic Programming with Box Constraints

by

Xiaojun Liu, Yong-Chang Jiao, and Satoru Fujishige

April 1997

An Algorithm for Strictly Convex Quadratic Programming with Box Constraints*

Xiaojun Liu, Yong-Chang Jiao† and Satoru Fujishige‡

Institute of Policy and Planning Sciences
University of Tsukuba
Tsukuba, Ibaraki 305, Japan

April 1997

Abstract

We propose an efficient algorithm for solving strictly convex quadratic programs with box constraints (i.e., lower and upper bounds on each variable). Our algorithm is based on the active set and the Newton method. We repeatedly compute relevant inverse matrices efficiently and, starting from an initial feasible point, we find an optimal solution of the problem in finitely many steps. Our experimental results show that the new algorithm is practically efficient.

Key words: Quadratic programming, algorithm, active set method, Newton method, box constraints.

*This is a revised and full version of our paper [7] presented at the Second International Symposium on Operations Research and Its Applications held in Guilin, China on December 11-14, 1996.

†Visiting Research Fellow supported by Japan Society for the Promotion of Science. His permanent address: Institute of Antennas and EM Scattering, Xidian University, Xi'an, Shaanxi 710071, P. R. China.

‡Present address: Division of Systems Science, Department of Systems and Human Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560, Japan.

1. Introduction

We consider a strictly convex (i.e., positive definite) quadratic programming problem subject to box constraints:

$$\begin{array}{lll} \textbf{(QP)} & \text{Minimize} & f(x) = \frac{1}{2}x^\top Ax + b^\top x \\ & \text{subject to} & c \le x \le d. \end{array} \tag{1.1}$$

where $A = [a_{ij}]$ is an $n \times n$ symmetric positive definite matrix, and b, c and d are n-vectors. Let $g(x)$ be the gradient, $Ax + b$, of $f(x)$ at x. Without loss of generality we assume $c_i < d_i$ for each i with $1 \le i \le n$.

Applications of the above-mentioned Problem **(QP)** with box constraints include large linear least squares problem with bounded variables, linear complementarity problems, and dual problems arising in a sequential quadratic programming algorithm. This last application [9] motivated the present work. Other applications can be found in [4].

Yang and Tolle [10] presented conjugate gradient-type algorithms for **(QP)**. Our algorithm is based on the idea of the active set method and we keep the active (and nonactive) set and compute related matrices efficiently.

In Section 2 we give some definitions and notations to be used later and describe the optimality condition for **(QP)**, based on which an algorithm will be constructed. We propose an algorithm for solving **(QP)** in Section 3. The proof of the validity of the proposed algorithm is provided in Section 4. Repeated updating of inverse matrices is required in the algorithm. We give an efficient procedure of updating relevant matrices in Section 5. Finally, in Section 6, we present numerical results and compare our algorithm with other ones. We also give discussions on the behavior of our algorithm. Conclusions are also given in Section 7.

2. Definitions and the Optimality Condition

We denote by K the set $\{1, 2, \cdots, n\}$ of the subscript indices of the variable x appearing in (1.1).

Definition: Denote by $x^{(l)}$ the current solution obtained at the lth iteration in the algorithm to be given in Section 3. For the current $x^{(l)}$ with $c \le x^{(l)} \le d$

$(l = 0, 1, \cdots)$ we define the *nonactive set*

$$\mathcal{NA}^{(l)} = \{i \mid c_i < x_i^{(l)} < d_i, 1 \leq i \leq n\} \tag{2.1}$$

and $n_l = |\mathcal{NA}^{(l)}|$.

For the *nonactive set* $\mathcal{NA}^{(l)}$, also define the $n \times n_l$ matrix

$$P_l = [p_{ij}]_{n \times n_l} \tag{2.2}$$

with the row index set K and the column index set $\mathcal{NA}^{(l)}$ such that $p_{ii} = 1$ if $i \in \mathcal{NA}^{(l)}$ and $p_{ij} = 0$ otherwise. Two related matrices are defined as follows:

$$H_l = P_l^{\top} A P_l, \quad B_l = H_l^{-1}. \tag{2.3}$$

Note that the row and column index sets of H_l and B_l are both $\mathcal{NA}^{(l)}$.

The Karush-Kuhn-Tucker optimality condition for **(QP)** is given as follows.

Theorem 2.1: *A feasible point x^* is an optimal solution of Problem* **(QP)** *if and only if for $j = 1, 2, \cdots, n$*

$$g_j(x^*) \geq 0, \quad when \ x_j^* = c_j \tag{2.4}$$

$$g_j(x^*) \leq 0, \quad when \ x_j^* = d_j \tag{2.5}$$

$$g_j(x^*) = 0, \quad when \ c_j < x_j^* < d_j. \tag{2.6}$$

□

We derive an algorithm for solving **(QP)**, based on Theorem 2.1.

3. An Algorithm

We give an algorithm for solving Problem **(QP)** as follows.

Step 1: Choose an initial feasible point $x^{(0)}$ with $c_i \leq x_i^{(0)} \leq d_i$ $(i \in K)$ such that for some $i_0 \in K$ we have $c_{i_0} < x_{i_0}^{(0)} < d_{i_0}$. Compute $\mathcal{NA}^{(0)}, P_0, H_0, n_0, B_0$, and set $l = 0$.

Step 2: Compute

$$g^{(l)} = Ax^{(l)} + b \tag{3.1}$$

$$\bar{g}^{(l)} = P_l^{\top} g^{(l)} \tag{3.2}$$

$$s^{(l)} = -B_l \bar{g}^{(l)} \tag{3.3}$$

$$\alpha^{(l)} = \min\{1, \min_{j \in \mathcal{NA}^{(l)},\ s_j^{(l)} < 0} \frac{c_j - x_j^{(l)}}{s_j^{(l)}}, \min_{j \in \mathcal{NA}^{(l)},\ s_j^{(l)} > 0} \frac{d_j - x_j^{(l)}}{s_j^{(l)}}\} \tag{3.4}$$

$$\tilde{x}^{(l+1)} = x^{(l)} + \alpha^{(l)} P_l s^{(l)}. \tag{3.5}$$

(Here, note that the index set of vector $s^{(l)}$ should be regarded as $\mathcal{NA}^{(l)}$.)

Step 3: If $\alpha^{(l)} < 1$ and $n_l > 1$, choose some $\gamma \in \mathcal{NA}^{(l)}$ such that

$$\alpha^{(l)} = \frac{c_\gamma - x_\gamma^{(l)}}{s_\gamma^{(l)}}, \qquad s_\gamma^{(l)} < 0, \tag{3.6}$$

or

$$\alpha^{(l)} = \frac{d_\gamma - x_\gamma^{(l)}}{s_\gamma^{(l)}}, \qquad s_\gamma^{(l)} > 0, \tag{3.7}$$

set

$$\mathcal{NA}^{(l+1)} = \mathcal{NA}^{(l)} \setminus \{\gamma\}, \quad n_{l+1} = n_l - 1, \tag{3.8}$$

$$x^{(l+1)} = \tilde{x}^{(l+1)}, \tag{3.9}$$

and go to Step 5; otherwise go to Step 4.

Step 4: Compute

$$g^{(l+1)} = A\tilde{x}^{(l+1)} + b, \tag{3.10}$$

$$\Phi_{l1} = \{i \mid \tilde{x}_i^{(l+1)} = c_i, g_i^{(l+1)} < 0\}, \tag{3.11}$$

$$\Phi_{l2} = \{i \mid \tilde{x}_i^{(l+1)} = d_i, g_i^{(l+1)} > 0\}, \tag{3.12}$$

$$\Phi_l = \Phi_{l1} \cup \Phi_{l2}. \tag{3.13}$$

If $\Phi_l \neq \emptyset$, compute for each $i \in \Phi_l$

$$\lambda_i = \frac{g_i^{(l+1)}}{(d_i - c_i)a_{ii}}, \tag{3.14}$$

$$\bar{\gamma} = arg\max\{ \begin{array}{l} \dfrac{(g_i^{(l+1)})^2}{2a_{ii}} \ \text{with}\, i \in \Phi_{l1} \ \text{and} \ -\lambda_i < 1; \\ -\dfrac{1}{2}(d_i - c_i)^2 a_{ii} - g_i^{(l+1)}(d_i - c_i) \ \text{with}\, i \in \Phi_{l1} \ \text{and} \ -\lambda_i \geq 1; \end{array}$$

$$\frac{(g_i^{(l+1)})^2}{2a_{ii}} \quad \text{with } i \in \Phi_{l2} \quad \text{and } \lambda_i < 1;$$
$$-\frac{1}{2}(d_i - c_i)^2 a_{ii} + g_i^{(l+1)}(d_i - c_i) \quad \text{with } i \in \Phi_{l2} \quad \text{and } \lambda_i \geq 1\}. \tag{3.15}$$

There are four cases (a)$\sim$(d) with respect to $\bar{\gamma}$ as follows:

(a) If $\bar{\gamma} \in \Phi_{l1}$ and $-\lambda_{\bar{\gamma}} < 1$, set

$$x^{(l+1)} = \begin{cases} \tilde{x}_i^{(l+1)}, & i \neq \bar{\gamma}, \\ c_i - \frac{g_i^{(l+1)}}{a_{ii}}, & i = \bar{\gamma}, \end{cases} \tag{3.16}$$

$$\mathcal{NA}^{(l+1)} = \mathcal{NA}^{(l)} \cup \{\bar{\gamma}\}, \tag{3.17}$$

$$n_{l+1} = n_l + 1, \tag{3.18}$$

and go to Step 6.

(b) If $\bar{\gamma} \in \Phi_{l1}$ and $-\lambda_{\bar{\gamma}} \geq 1$, set

$$\tilde{x}^{(l+1)} = \begin{cases} \tilde{x}_i^{(l+1)}, & i \neq \bar{\gamma}, \\ d_i, & i = \bar{\gamma}, \end{cases} \tag{3.19}$$

and go to the beginning of Step 4.

(c) If $\bar{\gamma} \in \Phi_{l2}$ and $\lambda_{\bar{\gamma}} < 1$, set

$$x^{(l+1)} = \begin{cases} \tilde{x}_i^{(l+1)}, & i \neq \bar{\gamma}, \\ d_i - \frac{g_i^{(l+1)}}{a_{ii}}, & i = \bar{\gamma}, \end{cases} \tag{3.20}$$

$$\mathcal{NA}^{(l+1)} = \mathcal{NA}^{(l)} \cup \{\bar{\gamma}\}, \tag{3.21}$$

$$n_{l+1} = n_l + 1, \tag{3.22}$$

and go to Step 6.

(d) If $\bar{\gamma} \in \Phi_{l2}$, $\lambda_{\bar{\gamma}} \geq 1$, set

$$\tilde{x}^{(l+1)} = \begin{cases} \tilde{x}_i^{(l+1)}, & i \neq \bar{\gamma}, \\ c_i, & i = \bar{\gamma}, \end{cases} \tag{3.23}$$

and go to the beginning of Step 4.
If $\Phi_l = \emptyset$ and $\max_{i \in \mathcal{NA}^{(l+1)}} |g_i^{(l+1)}| = 0$, **STOP** (the current $\tilde{x}^{(l+1)}$ is an optimal solution), and if $\Phi_l = \emptyset$ and $\max_{i \in \mathcal{NA}^{(l+1)}} |g_i^{(l+1)}| \neq 0$, go to Step 2. (Here, we judge $\alpha \equiv \max_{i \in \mathcal{NA}^{(l+1)}} |g_i^{(l+1)}| = 0$ if $\alpha < \epsilon (= 10^{-10})$, and $\alpha \neq 0$ otherwise.)

Step 5: Form P_{l+1} by deleting column γ from P_l, $\bar{B}_l$ by deleting row γ and column γ from B_l and a vector h by deleting component γ from column γ of B_l. Compute

$$\begin{aligned} H_{l+1} &= P_{l+1}^{\top} A P_{l+1}, \\ t &= [B_l]_{\gamma\gamma}, \\ B_{l+1} &= \bar{B}_l - h h^{\top}/t, \end{aligned} \tag{3.24}$$

set $l = l + 1$, and go to Step 2.

Step 6: Compute P_{l+1} and H_{l+1} corresponding to $\mathcal{NA}^{(l+1)}$. Form a vector h by deleting component $\bar{\gamma}$ from column $\bar{\gamma}$ in H_{l+1}, set $t = [H_{l+1}]_{\bar{\gamma}\bar{\gamma}}$, compute

$$\begin{aligned} \alpha &= \frac{1}{t - h^{\top} B_l h}, \\ \hat{h} &= -\alpha B_l h, \\ B_{l+1} &= \begin{bmatrix} B_l + \hat{h}\hat{h}^{\top}/\alpha & \hat{h} \\ \hat{h}^{\top} & \alpha \end{bmatrix}, \end{aligned} \tag{3.25}$$

set $l = l + 1$, and go to Step 2.

4. Validity of the Algorithm

We can show the following.

Lemma 4.1: *For all* $l \geq 0$, $0 < \alpha^{(l)} \leq 1$.

(Proof) For all $j \in \mathcal{NA}^{(l)}$ we have $c_j < x_j^{(l)} < d_j$ by definition.
Hence,

$$\min_{j \in \mathcal{NA}^{(l)},\ s_j^{(l)} < 0} \frac{c_j - x_j^{(l)}}{s_j^{(l)}} > 0, \tag{4.1}$$

and

$$\min_{j \in \mathcal{NA}^{(l)},\ s_j^{(l)} > 0} \frac{d_j - x_j^{(l)}}{s_j^{(l)}} > 0. \tag{4.2}$$

□

Lemma 4.2: *In Step 4 we have*

$$f(x^{(l+1)}) < f(\tilde{x}^{(l+1)}) \quad or \quad f(\tilde{x}^{(l+1)}_{new}) < f(\tilde{x}^{(l+1)}), \tag{4.3}$$

where $\tilde{x}^{(l+1)}_{new}$ *is the new point obtained from* $\tilde{x}^{(l+1)}$ *in Cases* (b) *and* (d) *in Step 4.*
(Proof) We only consider Cases (a) and (b) here. Cases (c) and (d) can be proved in the same way.
In Case (a), since $g^{(l+1)}_{\bar{\gamma}} < 0$, we have

$$0 < -\lambda_{\bar{\gamma}} = \frac{-g^{(l+1)}_{\bar{\gamma}}}{(d_{\bar{\gamma}} - c_{\bar{\gamma}})a_{\bar{\gamma}\bar{\gamma}}} < 1, \tag{4.4}$$

that is,

$$c_{\bar{\gamma}} < c_{\bar{\gamma}} - \frac{-g^{(l+1)}_{\bar{\gamma}}}{a_{\bar{\gamma}\bar{\gamma}}} < d_{\bar{\gamma}}. \tag{4.5}$$

Therefore,

$$c_{\bar{\gamma}} < x^{(l+1)}_{\bar{\gamma}} < d_{\bar{\gamma}}. \tag{4.6}$$

We thus have

$$\begin{aligned} f(\tilde{x}^{(l+1)}) - f(x^{(l+1)}) &= \frac{1}{2}(\tilde{x}^{(l+1)})^{\top} A\tilde{x}^{(l+1)} + b^{\top}\tilde{x}^{(l+1)} - \frac{1}{2}(x^{(l+1)})^{\top} Ax^{(l+1)} - b^{\top}x^{(l+1)} \\ &= \frac{1}{2}(\tilde{x}^{(l+1)} - x^{(l+1)})^{\top} A(\tilde{x}^{(l+1)} + x^{(l+1)}) + b^{\top}(\tilde{x}^{(l+1)} - x^{(l+1)}) \\ &= -\frac{1}{2}(\tilde{x}^{(l+1)} - x^{(l+1)})^{\top} A(\tilde{x}^{(l+1)} - x^{(l+1)}) \\ &\quad +(A\tilde{x}^{(l+1)} + b)^{\top}(\tilde{x}^{(l+1)} - x^{(l+1)}) \\ &= -\frac{1}{2}\frac{(g^{(l+1)}_{\bar{\gamma}})^2}{a_{\bar{\gamma}\bar{\gamma}}} + \frac{(g^{(l+1)}_{\bar{\gamma}})^2}{a_{\bar{\gamma}\bar{\gamma}}} \\ &= \frac{1}{2}\frac{(g^{(l+1)}_{\bar{\gamma}})^2}{a_{\bar{\gamma}\bar{\gamma}}} > 0. \end{aligned} \tag{4.7}$$

In Case (b) we also have

$$\begin{aligned} f(\tilde{x}^{(l+1)}) - f(\tilde{x}^{(l+1)}_{new}) &= -\frac{1}{2}(\tilde{x}^{(l+1)} - \tilde{x}^{(l+1)}_{new})^{\top} A(\tilde{x}^{(l+1)} - \tilde{x}^{(l+1)}_{new}) \\ &\quad +(A\tilde{x}^{(l+1)} + b)^{\top}(\tilde{x}^{(l+1)} - \tilde{x}^{(l+1)}_{new}) \\ &= -\frac{1}{2}(d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}} - g^{(l+1)}_{\bar{\gamma}}(d_{\bar{\gamma}} - c_{\bar{\gamma}}) \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}(d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}} - g_{\bar{\gamma}}^{(l+1)}(d_{\bar{\gamma}} - c_{\bar{\gamma}}) - (d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}} \\
&= \frac{1}{2}(d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}} + (d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}}(-\lambda_{\bar{\gamma}} - 1) \\
&\geq \frac{1}{2}(d_{\bar{\gamma}} - c_{\bar{\gamma}})^2 a_{\bar{\gamma}\bar{\gamma}} > 0. \qquad (4.8)
\end{aligned}
$$

□

Theorem 4.3: *The proposed algorithm strictly decreases the value of the objective function in each iteration.*

(Proof) Since we have in Step 2,

$$
\begin{aligned}
\tilde{x}^{(l+1)} &= x^{(l)} + \alpha^{(l)} P_l s^{(l)} \\
&= x^{(l)} - \alpha^{(l)} P_l B_l \bar{g}^{(l)}, \qquad (4.9)
\end{aligned}
$$

then

$$
\begin{aligned}
f(x^{(l)}) - f(\tilde{x}^{(l+1)}) &= \frac{1}{2}(x^{(l)})^{\mathsf{T}} A x^{(l)} + b^{\mathsf{T}} x^{(l)} - \frac{1}{2}(\tilde{x}^{(l+1)})^{\mathsf{T}} A \tilde{x}^{(l+1)} - b^{\mathsf{T}} \tilde{x}^{(l+1)} \\
&= \frac{1}{2}(x^{(l)})^{\mathsf{T}} A x^{(l)} + b^{\mathsf{T}}(x^{(l)} - \tilde{x}^{(l+1)}) \\
&\quad -\frac{1}{2}(x^{(l)} - \alpha^{(l)} P_l B_l \bar{g}^{(l)})^{\mathsf{T}} A (x^{(l)} - \alpha^{(l)} P_l B_l \bar{g}^{(l)}) \\
&= \alpha^{(l)}(x^{(l)})^{\mathsf{T}} A P_l B_l \bar{g}^{(l)} - \frac{1}{2}(\alpha^{(l)})^2 (P_l B_l \bar{g}^{(l)})^{\mathsf{T}} A (P_l B_l \bar{g}^{(l)}) \\
&\quad +\alpha^{(l)} b^{\mathsf{T}} P_l B_l \bar{g}^{(l)} \\
&= \alpha^{(l)}(Ax^{(l)} + b)^{\mathsf{T}} P_l B_l \bar{g}^{(l)} - \frac{1}{2}(\alpha^{(l)})^2 (\bar{g}^{(l)})^{\mathsf{T}} B_l \bar{g}^{(l)} \\
&= \alpha^{(l)}[P_l{}^{\mathsf{T}}(Ax^{(l)} + b)]^{\mathsf{T}} B_l \bar{g}^{(l)} - \frac{1}{2}(\alpha^{(l)})^2 (\bar{g}^{(l)})^{\mathsf{T}} B_l \bar{g}^{(l)} \\
&= \alpha^{(l)}(\bar{g}^{(l)})^{\mathsf{T}} B_l \bar{g}^{(l)} - \frac{1}{2}(\alpha^{(l)})^2 (\bar{g}^{(l)})^{\mathsf{T}} B_l \bar{g}^{(l)} \\
&= \alpha^{(l)}[1 - \frac{1}{2}\alpha^{(l)}](\bar{g}^{(l)})^{\mathsf{T}} B_l \bar{g}^{(l)}.
\end{aligned}
$$

We consider the following two cases :

Case 1: $\bar{g}^{(l)} = 0$. We have $s^{(l)} = 0$ and hence $\tilde{x}^{(l+1)} = x^{(l)}$. Since $\alpha^{(l)} = 1$, we obtain a new $x^{(l+1)}$ in Step 4 where we go when $\Phi_l \neq \emptyset$. From Lemma 4.2, we have

$$
f(x^{(l+1)}) < f(\tilde{x}^{(l+1)}) = f(x^{(l)}). \qquad (4.10)
$$

Case 2: $\bar{g}^{(l)} \neq 0$. We have

$$f(x^{(l)}) - f(\tilde{x}^{(l+1)}) = \alpha^{(l)}[1 - \frac{1}{2}\alpha^{(l)}](\bar{g}^{(l)})^{\top} B_l \bar{g}^{(l)}, \tag{4.11}$$

$0 < \alpha^{(l)} \leq 1$ from Lemma 4.1 and B_l is a positive definite matrix. Hence, we have $f(x^{(l)}) - f(\tilde{x}^{(l+1)}) > 0$, that is,

$$f(\tilde{x}^{(l+1)}) < f(x^{(l)}). \tag{4.12}$$

If $\alpha^{(l)} < 1$ and $n_l > 1$, then $x^{(l+1)} = \tilde{x}^{(l+1)}$ and we have

$$f(x^{(l+1)}) = f(\tilde{x}^{(l+1)}) < f(x^{(l)}). \tag{4.13}$$

If $\alpha^{(l)} = 1$ or $n_l = 1$, we have from Lemma 4.2

$$f(x^{(l+1)}) < f(\tilde{x}^{(l+1)}) < f(x^{(l)}). \tag{4.14}$$

□

Lemma 4.4: *For two iteration numbers* l_1 *and* l_2 $(l_1 < l_2)$ *with* $\alpha^{(l_1)} = 1$ *or* $n_{l_1} = 1$, *and* $\alpha^{(l_2)} = 1$ *or* $n_{l_2} = 1$, *respectively, if we have*

$$\mathcal{NA}^{(l_1)} = \mathcal{NA}^{(l_2)}, \tag{4.15}$$

then there exists at least one $j_0 \in K \setminus \mathcal{NA}^{(l_1)}$ *such that*

$$x_{j_0}{}^{(l_1)} \neq x_{j_0}{}^{(l_2)}. \tag{4.16}$$

(Proof) If we have $x_j{}^{(l_1)} = x_j{}^{(l_2)}$ for all $j \in K \setminus \mathcal{NA}^{(l_1)} (= K \setminus \mathcal{NA}^{(l_2)})$, then

$$x^{(l_1)} - P_{l_1} P_{l_1}{}^{\top} x^{(l_1)} = x^{(l_2)} - P_{l_2} P_{l_2}{}^{\top} x^{(l_2)}, \tag{4.17}$$

since $\mathcal{NA}^{(l_1)} = \mathcal{NA}^{(l_2)}$, we have $n_{l_1} = n_{l_2}$. We consider the following two cases:
Case 1: $\alpha^{(l_1)} = \alpha^{(l_2)} = 1$. Since $\mathcal{NA}^{(l_1)} = \mathcal{NA}^{(l_2)}$, we have

$$P_{l_1} = P_{l_2}, \quad H_{l_1} = H_{l_2}, \quad B_{l_1} = B_{l_2}. \tag{4.18}$$

In Step 2,

$$\begin{aligned}
\tilde{x}^{(l_1+1)} &= x^{(l_1)} + \alpha^{(l_1)} P_{l_1} s^{(l_1)} \\
&= x^{(l_1)} + P_{l_1}[-B_{l_1} \bar{g}^{(l_1)}] \\
&= x^{(l_1)} - P_{l_1} B_{l_1}[H_{l_1} P_{l_1}{}^{\top} x^{(l_1)} + P_{l_1}{}^{\top} b + P_{l_1}{}^{\top} A(I - P_{l_1} P_{l_1}{}^{\top}) x^{(l_1)}] \\
&= x^{(l_1)} - P_{l_1} P_{l_1}{}^{\top} x^{(l_1)} - P_{l_1}{}^{\top} B_{l_1} P_{l_1}{}^{\top} b - P_{l_1} B_{l_1} P_{l_1}{}^{\top} A[x^{(l_1)} - P_{l_1} P_{l_1}{}^{\top} x^{(l_1)}] \\
&= [I - P_{l_1} B_{l_1} P_{l_1}{}^{\top} A][x^{(l_1)} - P_{l_1} P_{l_1}{}^{\top} x^{(l_1)}] - P_{l_1} B_{l_1} P_{l_1}{}^{\top} b.
\end{aligned} \tag{4.19}$$

In the same way, we have

$$\tilde{x}^{(l_2+1)} = [I - P_{l_2} B_{l_2} P_{l_2}{}^{\top} A][x^{(l_2)} - P_{l_2} P_{l_2}{}^{\top} x^{(l_2)}] - P_{l_2} B_{l_2} P_{l_2}{}^{\top} b. \tag{4.20}$$

¿From (4.17)∼(4.20), we have

$$\tilde{x}^{(l_1+1)} = \tilde{x}^{(l_2+1)}. \tag{4.21}$$

Case 2: $n_{l_1} = n_{l_2} = 1$, $\alpha^{(l_1)} < 1$ or $\alpha^{(l_2)} < 1$. In this case, there is only one element in $\mathcal{NA}^{(l_1)}$ and $\mathcal{NA}^{(l_2)}$ respectively. Suppose that $\mathcal{NA}^{(l_1)} = \mathcal{NA}^{(l_2)} = \{k\}$ for some k with $1 \leq k \leq n$. From (4.17), $x_i^{(l_1)} = x_i{}^{(l_2)}$ $(i \neq k,\ 1 \leq i \leq n)$ and from the definition of P_l we have

$$P_{l_1} = P_{l_2} = (0, \cdots, 0, 1, 0, \cdots, 0)^{\top}, \tag{4.22}$$

where the kth component is one. Then $H_{l_1} = H_{l_2} = [a_{kk}]$, $B_{l_1} = B_{l_2} = [a_{kk}^{-1}]$ and we also have $\tilde{x}_k^{(l_1+1)} = \tilde{x}_k^{(l_2+1)}$. In fact,

$$\begin{aligned}
s^{(l_1)} &= -B_{l_1} \bar{g}^{(l_1)} \\
&= -B_{l_1}[H_{l_1} P_{l_1}{}^{\top} x^{(l_1)} + P_{l_1}{}^{\top} b + P_{l_1}{}^{\top} A(I - P_{l_1} P_{l_1}{}^{\top}) x^{(l_1)}] \\
&= -x_k^{(l_1)} - a_{kk}^{-1} b_k - a_{kk}^{-1}[a_{k1} x_1^{(l_1)} + \cdots + a_{kk-1} x_{k-1}^{(l_1)} + a_{kk+1} x_{k+1}^{(l_1)} + \cdots + a_{kn} x_n^{(l_1)}] \\
&= -x_k^{(l_1)} - a_{kk}^{-1} b_k - a_{kk}^{-1} \sum_{i=1, i\neq k}^{n} a_{ki} x_i^{(l_1)}.
\end{aligned} \tag{4.23}$$

Since $\alpha^{(l_1)} < 1$, we have

$$s^{(l_1)} < 0 \quad \text{and} \quad \frac{c_k - x_k^{(l_1)}}{s^{(l_1)}} < 1 \tag{4.24}$$

or

$$s^{(l_1)} > 0 \quad \text{and} \quad \frac{d_k - x_k^{(l_1)}}{s^{(l_1)}} < 1. \tag{4.25}$$

When $s^{(l_1)} < 0$ and $\frac{c_k - x_k^{(l_1)}}{s^{(l_1)}} < 1$, i.e.,

$$x_k^{(l_1)} - c_k < -s^{(l_1)}, \tag{4.26}$$

we have from (4.23)

$$\begin{aligned}
x_k^{(l_1)} - c_k &< x_k^{(l_1)} + a_{kk}^{-1} b_k + a_{kk}^{-1} \sum_{i=1, i\neq k}^{n} a_{ki} x_i^{(l_1)}, \\
-c_k &< a_{kk}^{-1} b_k + a_{kk}^{-1} \sum_{i=1, i\neq k}^{n} a_{ki} x_i^{(l_1)}.
\end{aligned} \tag{4.27}$$

In the same way, we have

$$s^{(l_2)} = -x_k^{(l_2)} - a_{kk}^{-1} b_k - a_{kk}^{-1} \sum_{i=1, i\neq k}^{n} a_{ki} x_i^{(l_2)}. \tag{4.28}$$

Since $x_i^{(l_1)} = x_i^{(l_2)}$ $(i \neq k,\ 1 \leq i \leq n)$, we have from (4.27)

$$s^{(l_2)} = -x_k^{(l_2)} - a_{kk}^{-1} b_k - a_{kk}^{-1} \sum_{i=1, i\neq k}^{n} a_{ki} x_i^{(l_1)} < c_k - x_k^{(l_2)} < 0, \tag{4.29}$$

$$\frac{c_k - x_k^{(l_2)}}{s^{(l_2)}} < 1. \tag{4.30}$$

Hence, $\tilde{x}_k^{(l_2+1)} = \tilde{x}_k^{(l_1+1)} = c_k$.

When $s^{(l_1)} > 0$ and $\frac{d_k - x_k^{(l_1)}}{s^{(l_1)}} < 1$, we can also prove that

$$\tilde{x}_k^{(l_2+1)} = \tilde{x}_k^{(l_1+1)} = d_k. \tag{4.31}$$

Therefore, for Case 2 we also have

$$\tilde{x}^{(l_1+1)} = \tilde{x}^{(l_2+1)}. \tag{4.32}$$

Since $l_1 < l_2$, this is a contradiction to Theorem 4.3. □

Theorem 4.5: *The proposed algorithm solves Problem* (**QP**) *in finitely many steps.*
(Proof) Since the number of possible nonactive sets (and active sets) is finite, the finiteness of the proposed algorithm follows from Lemma 4.4. □

5. Computing Relevant Matrices

In the algorithm, the nonactive set $\mathcal{NA}^{(l)}$ is repeatedly changed by adding or removing a nonactive or active constraint and the relevant matrices are updated accordingly. In this section we describe an efficient way of updating relevant matrices. Before we get into the detail, we first describe how to initialize the data for the algorithm. We choose an initial point given as follows. Choose an index $i_0 \in K$ and t with $c_{i_0} < t < d_{i_0}$, and put

$$x_i^{(0)} = \begin{cases} t, & i = i_0, \\ c_i \text{ or } d_i, & i \neq i_0. \end{cases}$$

Then we can start the algorithm with very simple parameters:

$$\mathcal{NA}^{(0)} = \{i_0\}, \tag{5.1}$$

$$P_0 = (0, \cdots, 0, 1, 0, \cdots, 0)^{\mathsf{T}}, \tag{5.2}$$

$$H_0 = P_0^{\mathsf{T}} A P_0 = [a_{i_0 i_0}], \quad B_0 = H_0^{-1} = [1/a_{i_0 i_0}]. \tag{5.3}$$

Notice that 1 appearing in P_0 is the i_0th component.

5.1. Removing a nonactive constraint

In this case, $|\mathcal{NA}^{(l)}| = |\mathcal{NA}^{(l-1)}| - 1$, then the number of columns in P_l is one less than that in P_{l-1}. Without loss of generality we assume that $P_{l-1} = [P_l, p_1]$. Then H_{l-1} and H_l have the following relationship:

$$H_{l-1} = \begin{bmatrix} H_l & P_l^{\mathsf{T}} A p_1 \\ p_1^{\mathsf{T}} A P_l & p_1^{\mathsf{T}} A p_1 \end{bmatrix}.$$

Also, we have

$$B_{l-1} = \begin{bmatrix} \bar{B}_{l-1} & h_1 \\ h_1^{\mathsf{T}} & t_1 \end{bmatrix},$$

and

$$B_l = H_l^{-1} = \bar{B}_{l-1} - h_1 h_1^{\mathsf{T}} / t_1.$$

5.2. Adding a nonactive constraint

In this case, $|\mathcal{NA}^{(l)}| = |\mathcal{NA}^{(l-1)}| + 1$, then the number of columns in P_l is one more than that in P_{l-1}. We also assume that $P_l = [P_{l-1}, p_2]$. Then,

$$H_l = \begin{bmatrix} H_{l-1} & P_{l-1}^{\mathsf{T}} A p_2 \\ p_2^{\mathsf{T}} A P_{l-1} & p_2^{\mathsf{T}} A p_2 \end{bmatrix}. \tag{5.4}$$

Letting

$$h_2 = P_{l-1}^{\mathsf{T}} A p_2, \quad t_2 = p_2^{\mathsf{T}} A p_2, \quad \alpha = 1/(t_2 - h_2^{\mathsf{T}} B_{l-1} h_2), \quad \hat{h} = -\alpha B_{l-1} h_2,$$

we have

$$B_l = H_l^{-1} = \begin{bmatrix} B_{l-1} + \hat{h}\hat{h}^\top/\alpha & \hat{h} \\ \hat{h}^\top & \alpha \end{bmatrix}. \tag{5.5}$$

In this way we can efficiently compute H_l^{-1}.

6. Computational Experiments

6.1. The test problems

We generated problem instances as follows.

Given the size n of a matrix A, we generate a lower triangular matrix $L = [l_{ij}]_{n\times n}$ with $l_{ii} = 1$ and l_{ij} $(i > j)$ $(i, j = 1, \ldots, n)$ chosen at random from $[-20, 20]$. We then set $A = LDL^\top$, where D is a diagonal matrix with positive diagonal elements taken at random from $[5, 20]$. We also choose lower and upper bounds c_i $(i = 1, \ldots, n)$ from $[-10, 10]$ and d_i $(i = 1, \ldots, n)$ from $[-5, 15]$ at random, respectively (we swap c_i for d_i if $c_i > d_i$). b_i $(i = 1, \ldots, n)$ are chosen from $[-10, 10]$ at random. We call the execution of the algorithm starting from Step 2 till the next Step 2 a *cycle*.

6.2. Discussions

We carried out computational experiments by using numbers in double precision in C. The numerical results are shown in Figures 6.1∼6.13. Figures 6.1∼6.7 show basic characteristics of our algorithm. The numerical comparisons with [8], [2] and [1] are shown in Figures 6.8∼6.13. All the experiments are made on SUN S-4/10 model 41.

Figures 6.1 ∼ 6.5 show sample behaviors of our algorithm for problems generated as above for n=10, 20, 30, 40 and 50, respectively.

Figure 6.6 shows a ten-sample average behavior of the number of cycle versus the dimension. Figure 6.7 shows a ten-sample average behavior of the running time (seconds) versus the dimension.

From Figures 6.1 ∼ 6.5 we can see that the objective function $f(x)$ decreases very rapidly and an optimal solution can be found in only several steps for each dimension n=10 ∼ 50. As the dimension increases, the number of cycles increases moderately (in Figure 6.6) and the required running time is nearly proportional to the dimension (in Figure 6.7).

We used S. G. Nash's code [8] to compare its CPU time to our algorithm in Figure 6.8 and Figure 6.9, where our algorithm is denoted by J.L.F.. In lower dimensions, the required running times are nearly same, but our algorithm is getting faster than S. G. Nash's as the dimension increases.

We also compared our algorithm with codes L-BFGS-B [11] and LANCELOT [3] denoted by B.L.N.Z. and B.C.G.T., respectively in the figures. From Figures 6.10 and 6.11 we can see that, when the dimension n is low ($n \leq 70$), the difference of the running time between our algorithm and code L-BFGS-B is not so large, but the running time of code L-BFGS-B increases more quickly than that of our's. The same fact is also demonstrated in Figures 6.12 and 6.13 that CPU time of our algorithm is less than that of code LANCELOT in all dimensions shown here.

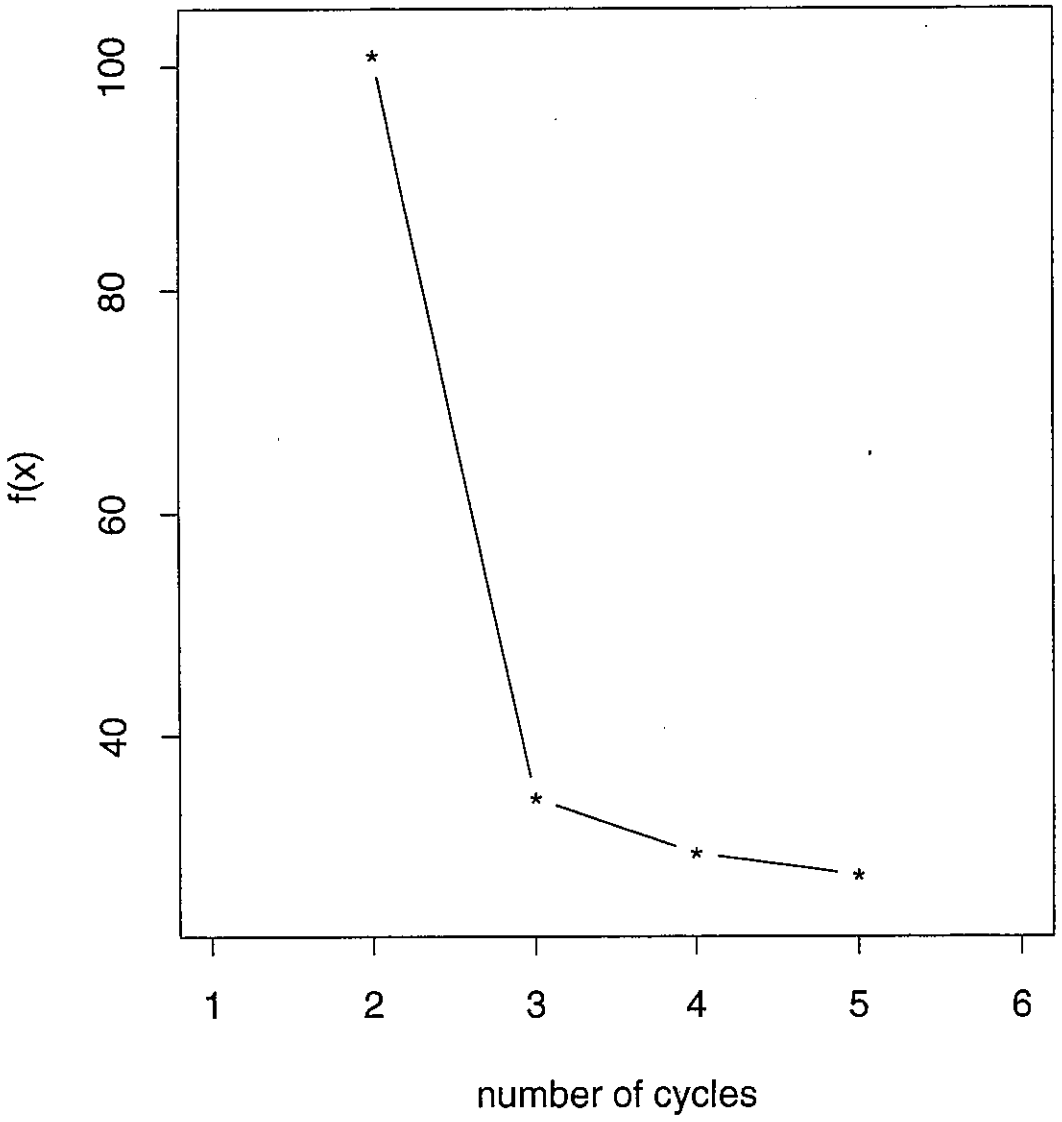

Figure 6.1: a sample behavior (n=10).

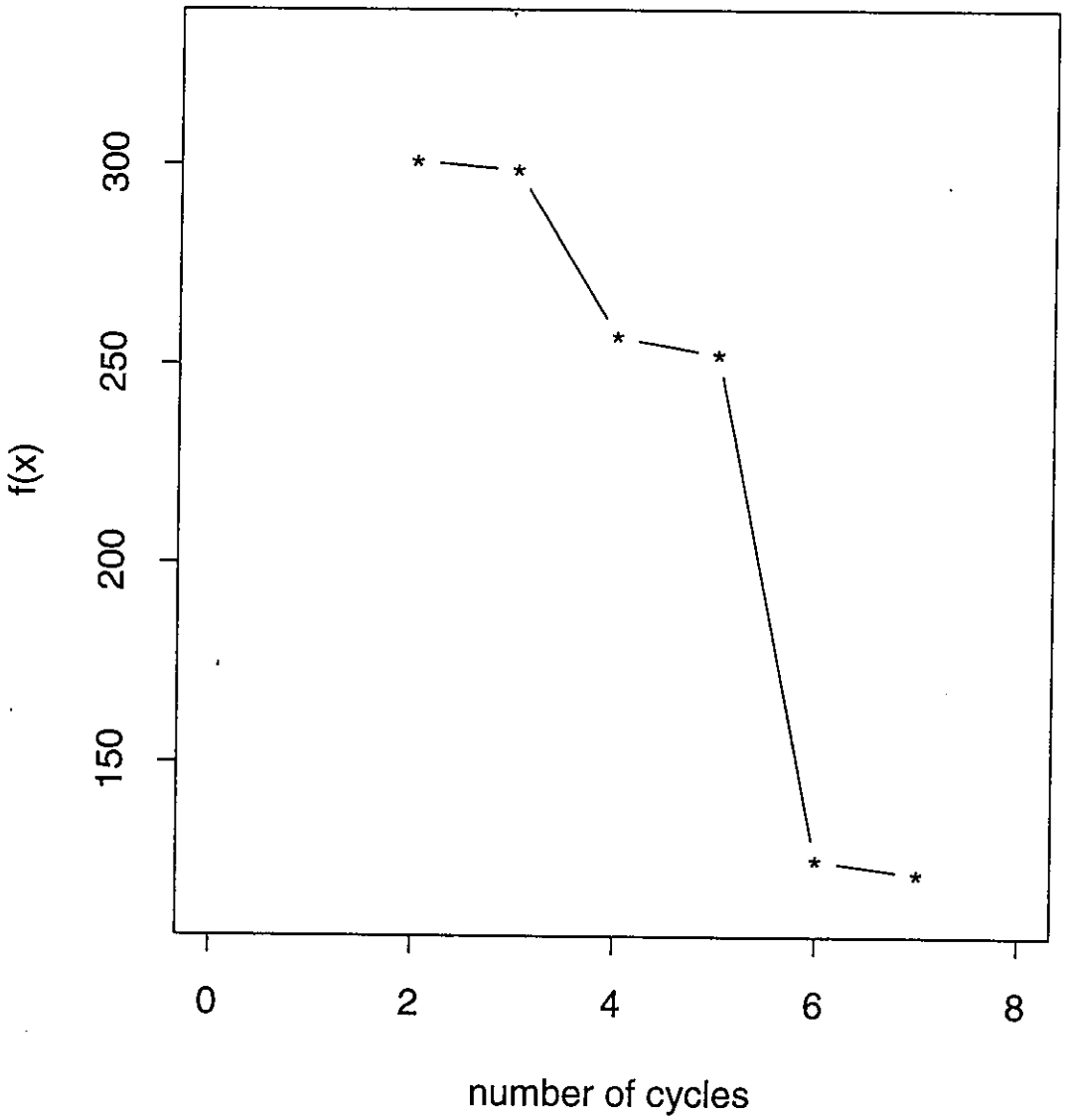

Figure 6.2: a sample behavior (n=20).

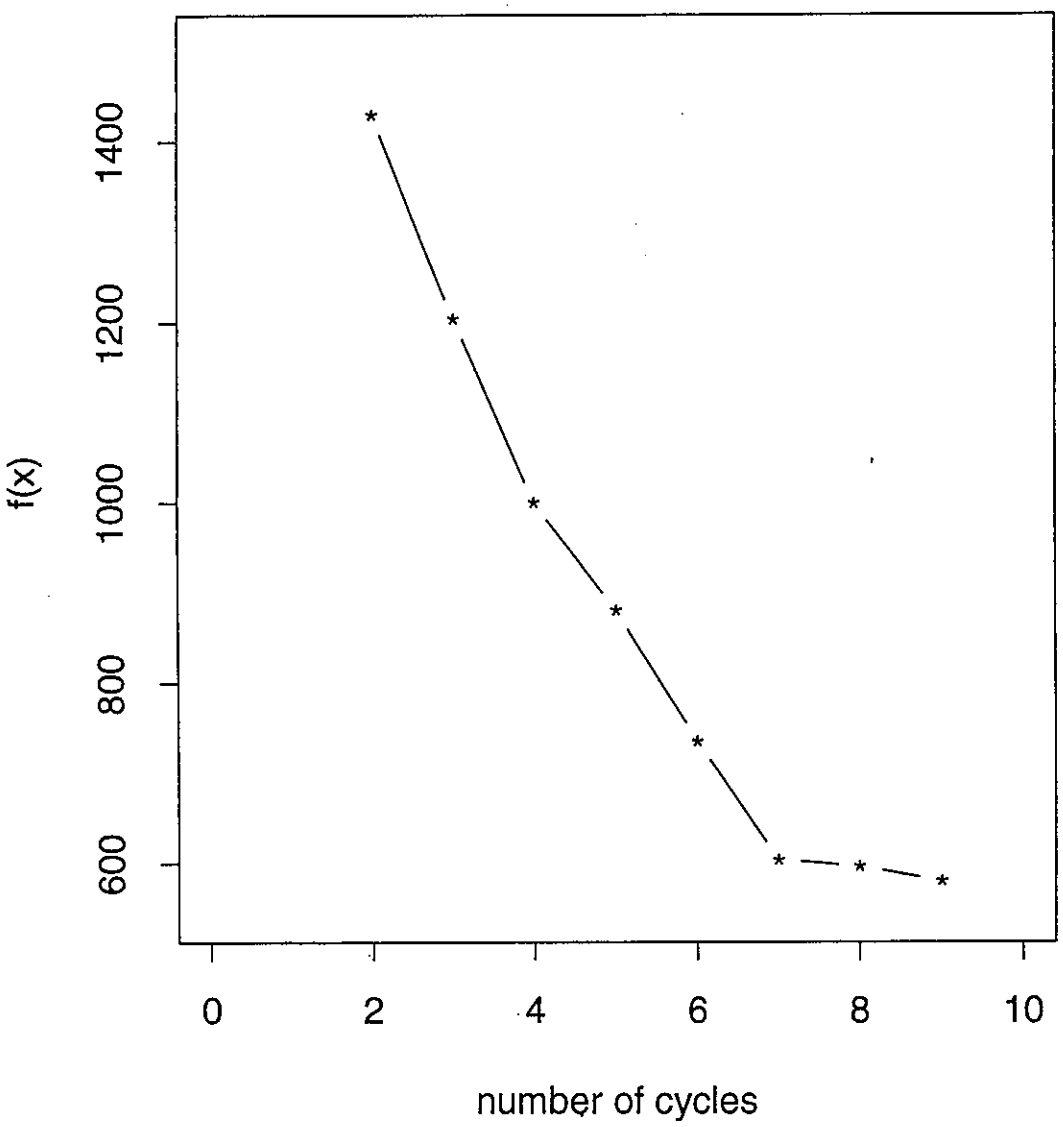

Figure 6.3: a sample behavior (n=30).

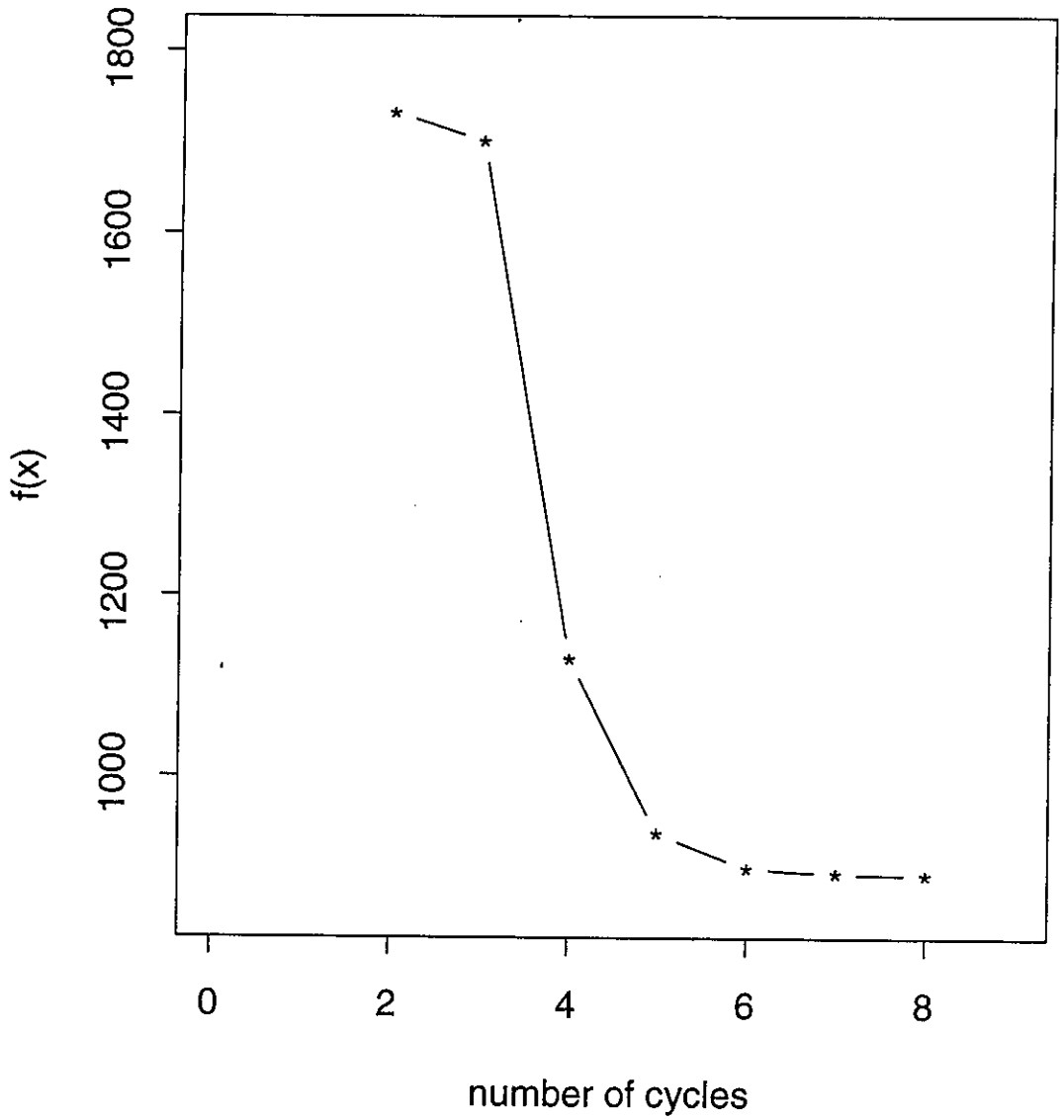

Figure 6.4: a sample behavior (n=40).

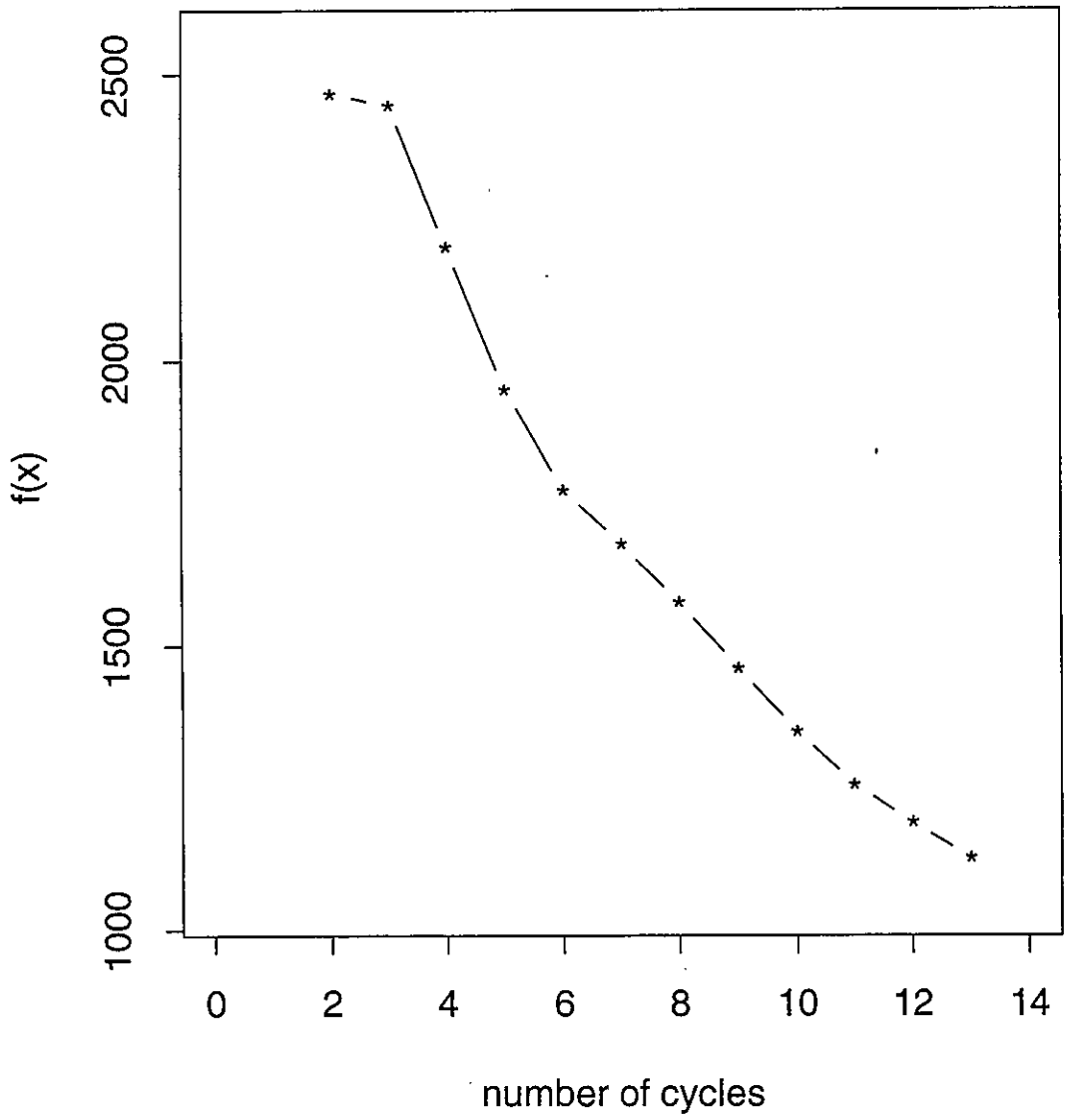

Figure 6.5: a sample behavior (n=50).

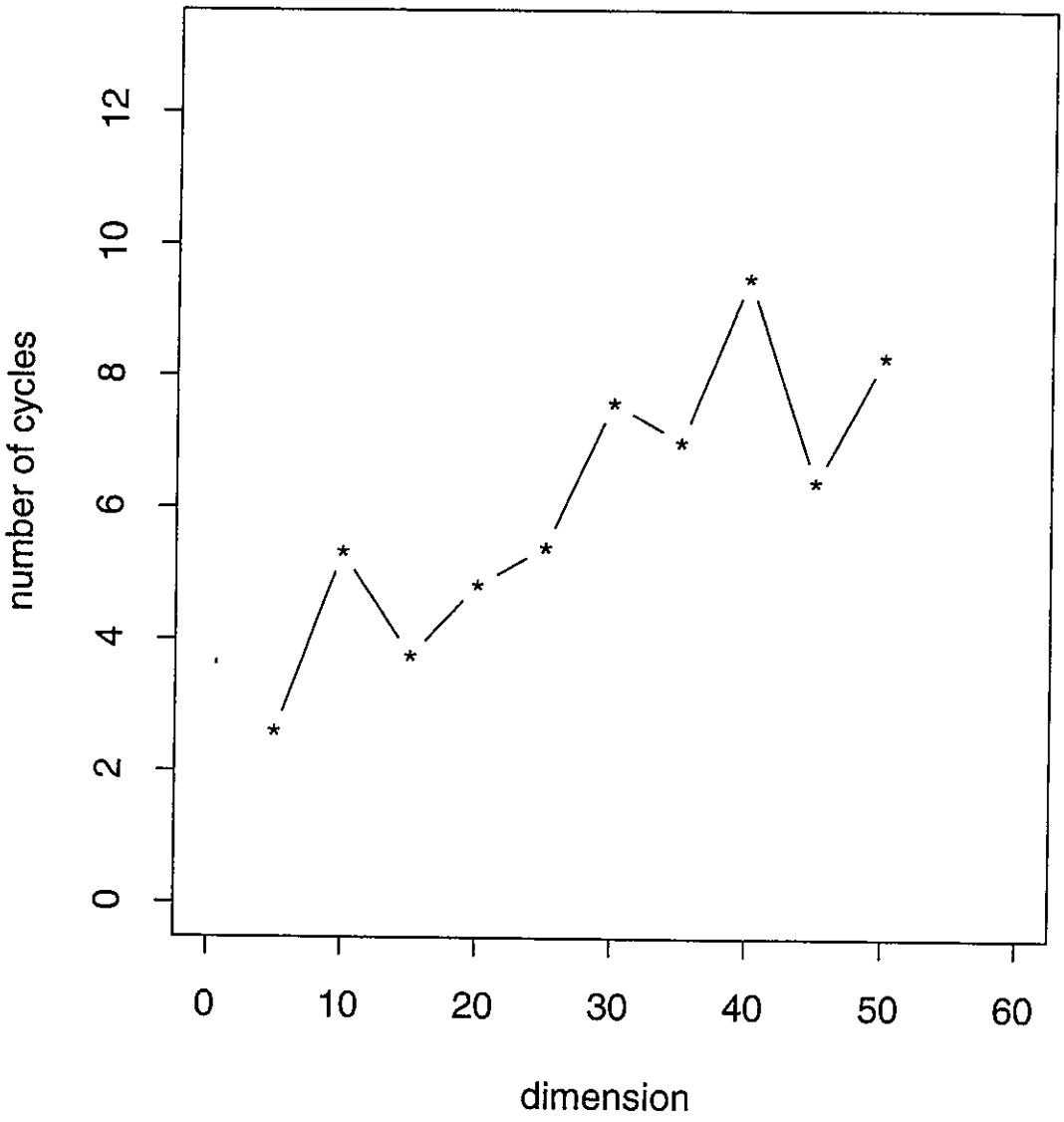

Figure 6.6: Ten-sample average behaviors

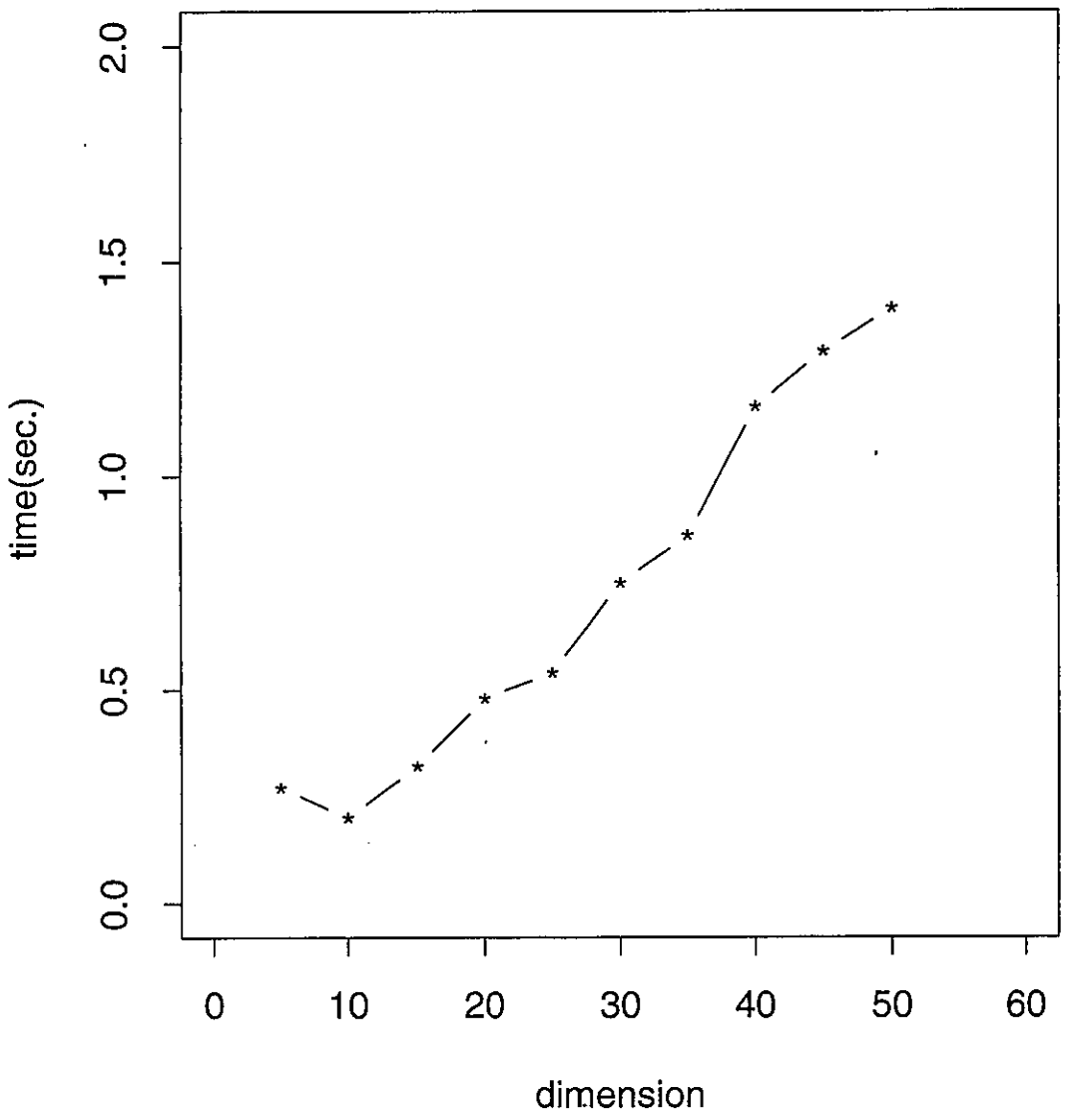

Figure 6.7: Ten-sample average behaviors

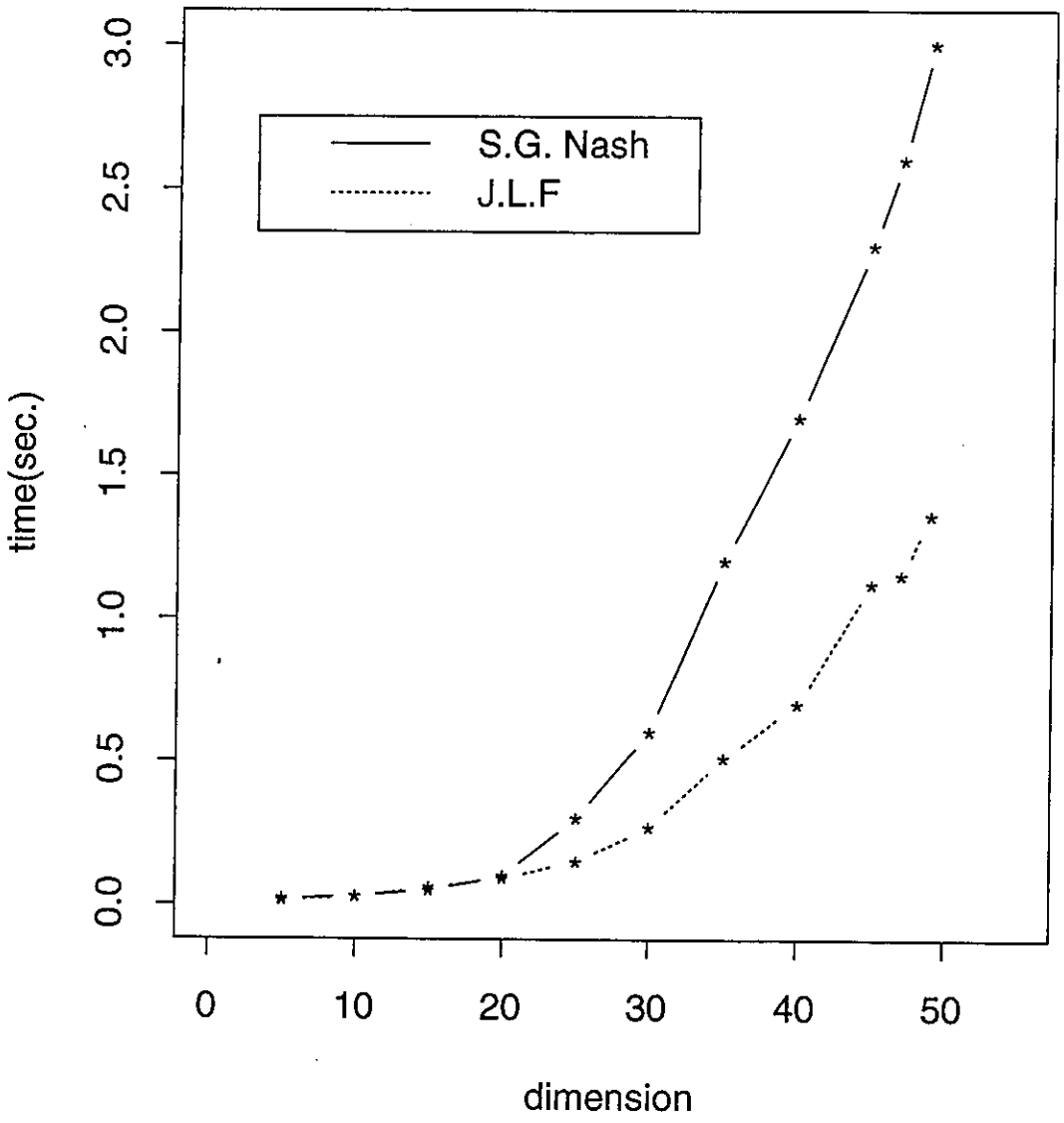

Figure 6.8: a sample behavior

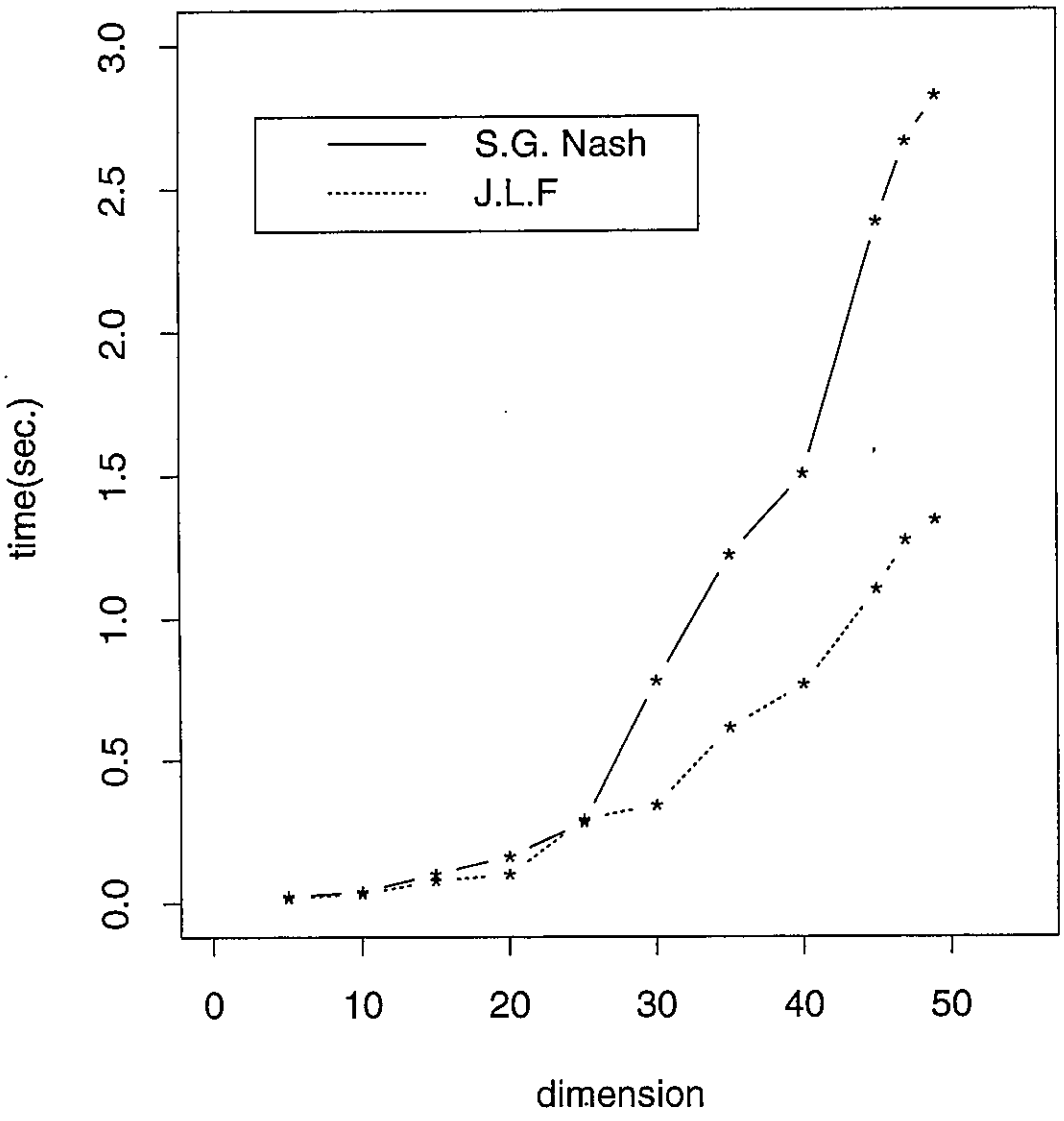

Figure 6.9: Ten-sample average behaviors

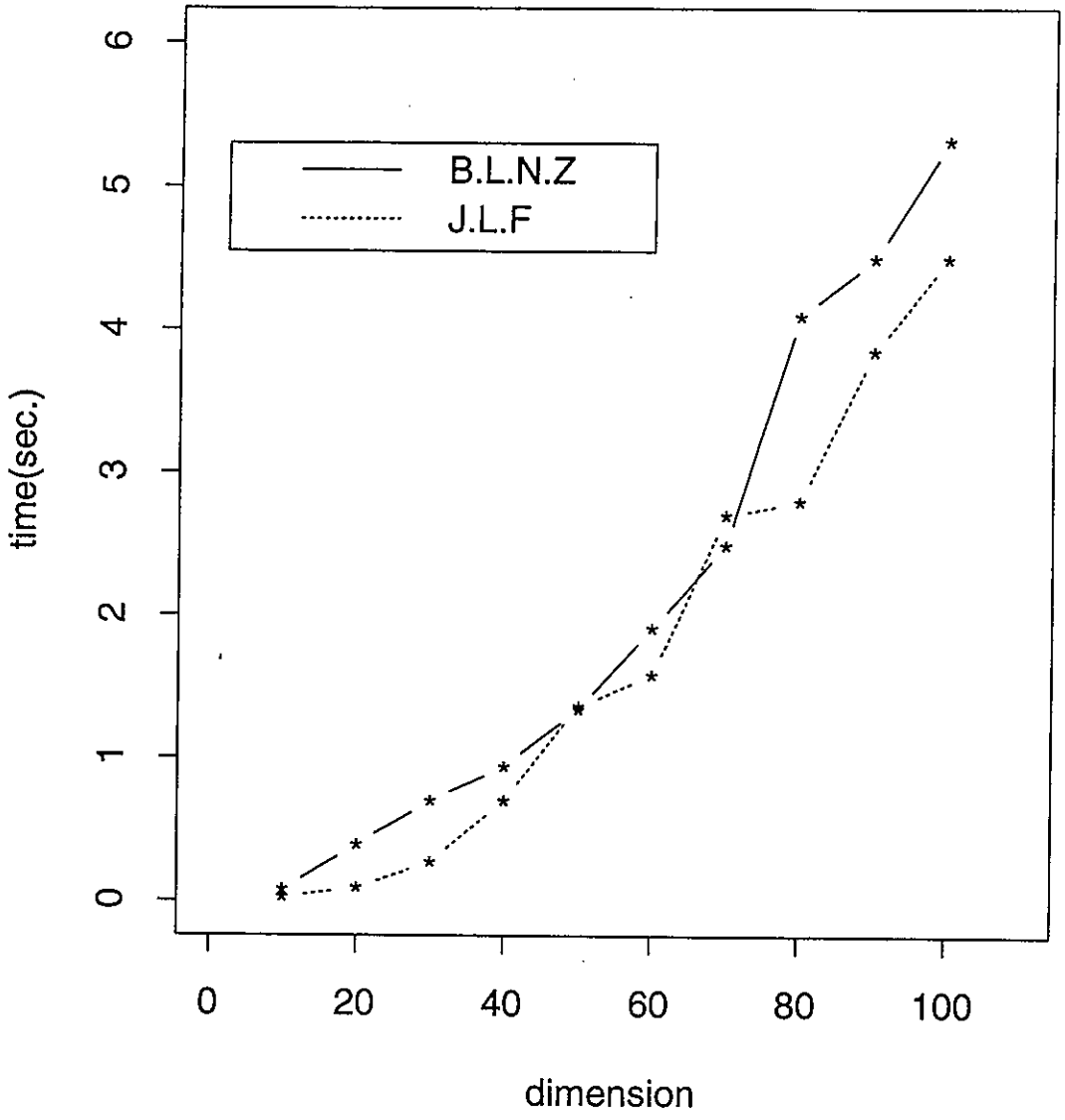

Figure 6.10: a sample behavior

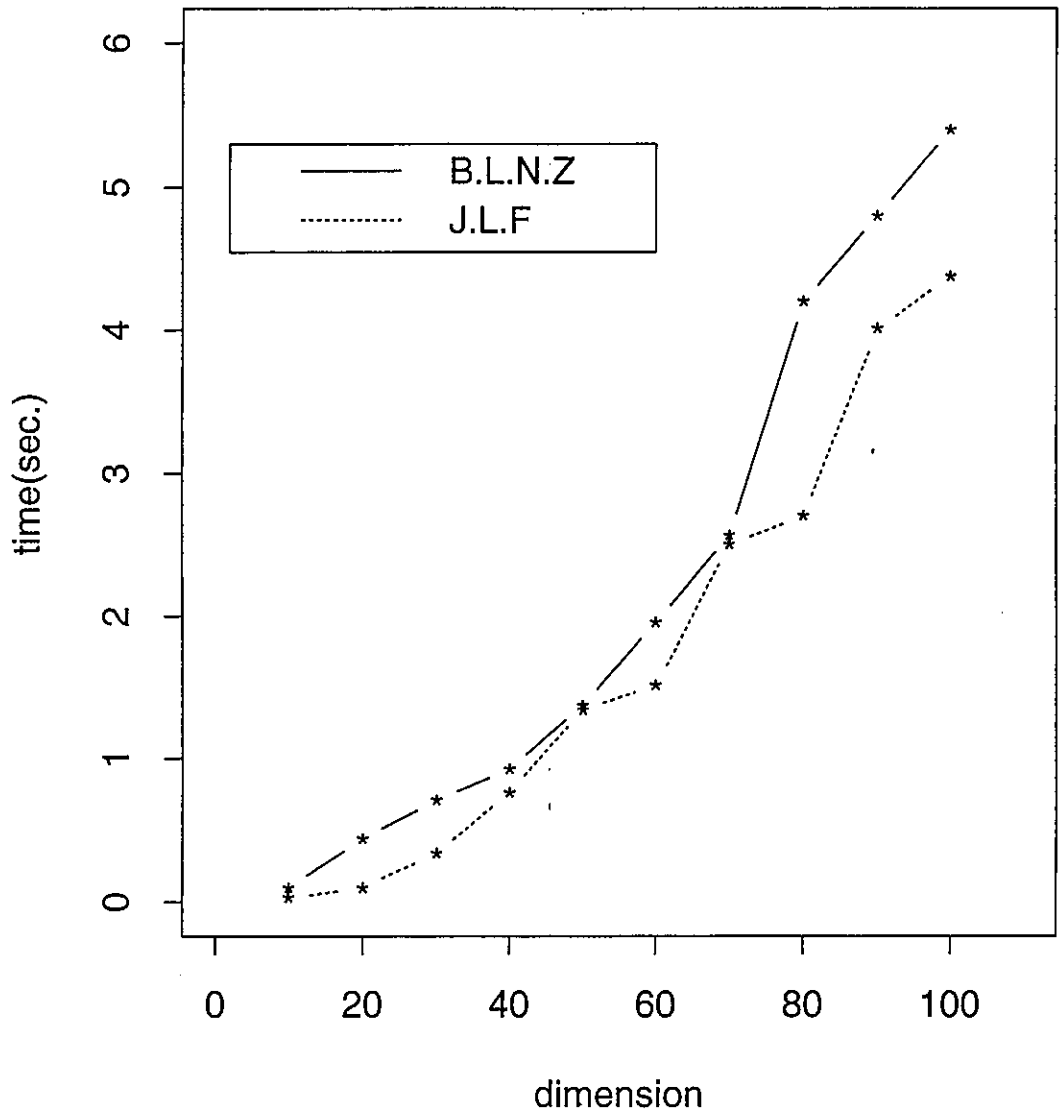

Figure 6.11: Ten-sample average behaviors

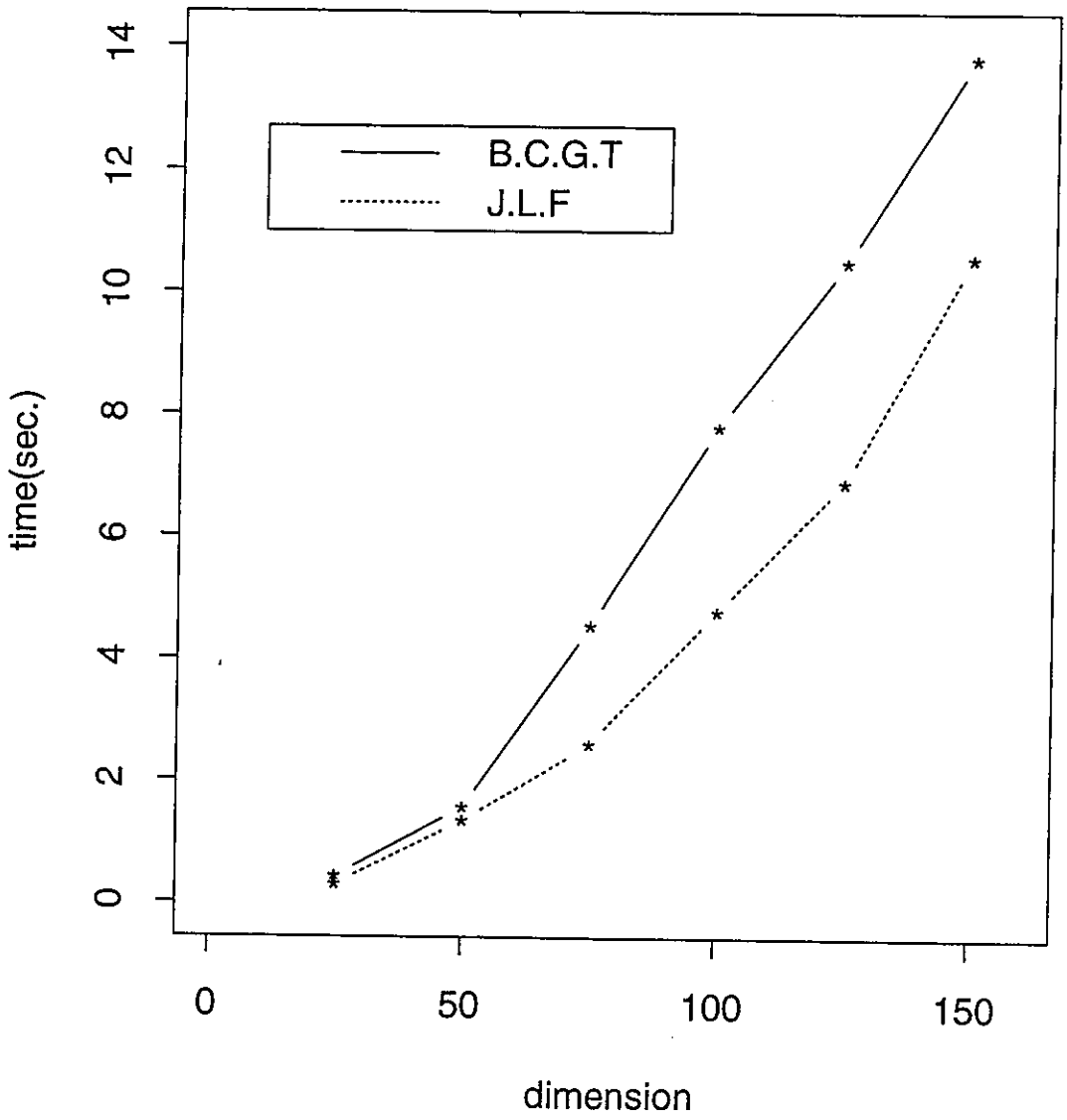

Figure 6.12: a sample behavior

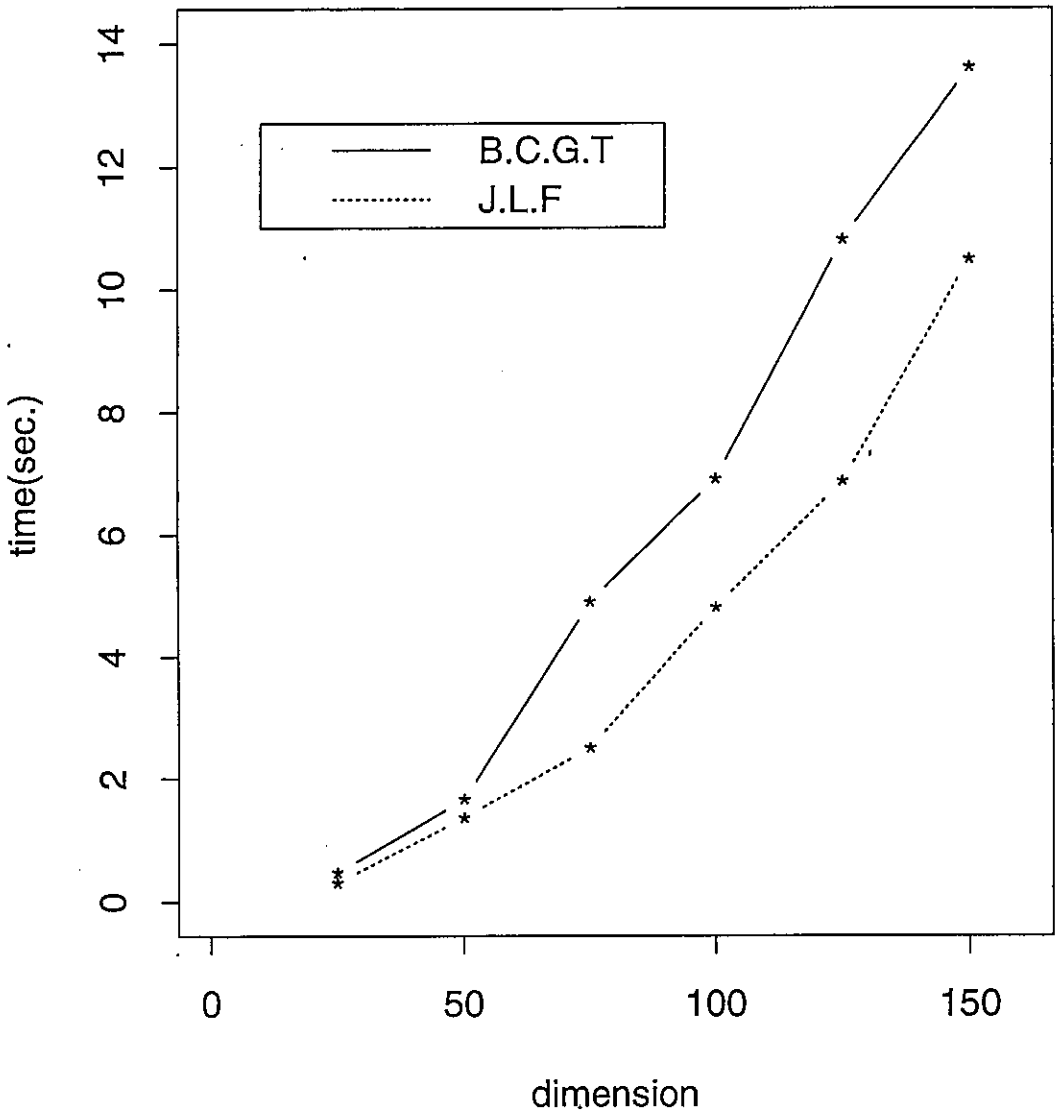

Figure 6.13: Ten-sample average behaviors

7. Conclusions

We present a new algorithm for solving convex quadratic programming problems with box constraints. Our algorithm can efficiently compute relevant inverse matrices and find an optimal solution in finitely many steps. We have carried out numerical experiments and they indicate that our algorithm is practically efficient and is faster than the existing codes of [11] and [3].

Acknowledgments

The second author would like to express his sincere appreciation to Professor Yoshitsugu Yamamoto for providing him the opportunity to work in the Institute of Policy and Planning Sciences, University of Tsukuba for four months. Special thanks go to Japan Society for the Promotion of Science (JSPS) for the financial support to the second author. The third author's work is supported by a grant-in-aid of the Ministry of Education, Science, Sports and Culture of Japan.

References

[1] Bongartz, I., Conn, A. R., Gould, N. I. M., and Toint, Ph. L., 1993, CUTE: constrained and unconstrained testing environment. Research Report, IBM T. J. Watson Research Center, Yorktown, USA.

[2] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C., 1995, A limited memory algorithm for bound constrained optimization. *SIAM J. Scientific Computing* **16**, 1190-1208.

[3] Conn, A. R., Gould, N. I. M., and Toint, Ph. L., 1992, LANCELOT: a FORTRAN package for large-scale nonlinear optimization (Release A). No. 17 in Springer Series in Computational mathematics, Springer-Verlag, New York.

[4] Dembo, R. S., and Tulowitzski, U., 1983, On the minimization of a quadratic function subject to box constraints. Working paper No. 71, Series B, School of Organization and Management, Yale University (New Haven, CT).

[5] Goldfarb, D., and Idnani, A., 1983, A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* **27**, 1-33.

[6] Jiao, Y. C., 1990, Study on constrained optimization and its application to optimal design of antennas. Ph.D. Dissertation, Xidian University, Xi'an, May 1990.

[7] Jiao, Y. C., Liu, X. J., and Fujishige, S., 1996, An algorithm for strictly convex quadratic programming with box constraints. Proceedings of the Second International Symposium on Operations Research and Its Applications (Guilin, China, December 11-14, 1996), pp. 27-36.

[8] Nash, S. G., 1984, Newton-type minimization via the Lanczos method. *SIAM J. Numerical Analysis* **21**, 770-788.

[9] Nickel, R. H., and Tolle, J. W., 1989, A sequential quadratic programming algorithm for solving large, sparse nonlinear programs. *Journal of Optimization Theory and Its Application* **60**, 453-473.

[10] Yang, E., and Tolle, J. W., 1991, A class of methods for solving large, convex quadratic programs subject to box constraints. *Mathematical Programming* **51**, 229-245.

[11] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., 1994, L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems. Tech. Report, NAM-11, EECS Department, Northwestern University, (Latest revision June 1996).