Journal of the Operations Reseach
Society of Japan
Vol. 19, No. 2, June, 1976

ALGORITHMS FOR QUADRATIC FRACTIONAL PROGRAMMING PROBLEMS

TOSHIHIDE IBARAKI
HIROAKI ISHII
JIRO IWASE
TOSHIHARU HASEGAWA and
HISASHI MINE, *Kyoto University*

(Received August 7, 1975; Revised January 28, 1976)

Abstract

Consider the nonlinear fractional programming problem $max\{f(x)/g(x) | x\epsilon S\}$, where $g(x)>0$ for all $x\epsilon S$. Jagannathan and Dinkelbach have shown that the maximum of this problem is equal to ξ^0 if and only if $max\{f(x)-\xi g(x) | x\epsilon S\}$ is 0 for $\xi=\xi^0$.

Based on this result, we treat here a special case: $f(x)=\frac{1}{2} x^tCx+r^tx+s$, $g(x)=\frac{1}{2} x^tDx+p^tx+q$ and S is a polyhedron, where C is negative definite and D is positive semidefinite. Two algorithms are proposed; one is a straightforward application of the parametric programming technique of quadratic programming, and the other is a modification of the Dinkelbach's method. It is proved that both are finite algorithms. In the computational experiment performed for the case of D=0, the followings are observed: (i) The parametric programming approach is slightly faster than the Dinkelbach's, but there is no significant difference, and (ii) the quadratic fractional programming problems as above can usually be solved in computation time only slightly greater (about 10-20%) than that required by the ordinary (concave) quadratic programming problems.

1. Introduction

Consider the following nonlinear fractional programming problem:

(1.1) $$\underset{x\in S}{\text{maximize}}\ f(x)/g(x),$$

where $g(x) > 0$ for all $x \in S$. The next problem is associated with (1.1).

(1.2) $$\underset{x\in S}{\text{maximize}}\ z(\xi) = f(x) - \xi g(x).$$

The maximum value of (1.2) is denoted $z^0(\xi)$.

Jagannathan [3] and Dinkelbach [1] have shown that the maximum of $f(x)/g(x)$ of (1.1) is equal to ξ^0 satisfying $z^0(\xi^0) = 0$.

A special case of (1.1) is discussed in this paper, in which $f(x)$ and $g(x)$ are quadratic functions, and S is a polyhedron defined by a set of linear inequalities:

(1.3) $$QF: \quad \text{maximize } (\tfrac{1}{2} x^tCx + r^tx + s)/(\tfrac{1}{2} x^tDx + p^tx + q)$$
$$\text{subject to } Ax \le b,$$

where C is an $n\times n$ negative definite matrix, D is an $n\times n$ positive semidefinite matrix, A is an $m\times n$ matrix, r, p are n-vectors, b is an m-vector, and x is an n-vector of variables. t denotes the transpose. All coefficients in matrices and vectors are reals. It is also assumed that

(1.4) $$\tfrac{1}{2} x^tCx + r^tx + s \ge 0 \text{ for some } x \in S$$
$$\tfrac{1}{2} x^tDx + p^tx + q > 0 \text{ for all } x \in S,$$

where $S = \{x \mid Ax \le b\}$.

Corresponding to problem QF, (1.2) is written as follows.

(1.5) $$Q(\xi): \quad \text{maximize } z(\xi) = \tfrac{1}{2} x^tK(\xi)x + c(\xi)^tx + d(\xi)$$
$$\text{subject to } Ax \le b,$$

where

(1.6) $$K(\xi) = C - \xi D$$
$$c(\xi) = r - \xi p$$
$$d(\xi) = s - \xi q.$$

Note that $K(\xi)$ is negative definite for $\xi \ge 0$, and finite algorithms for solving quadratic programming problem $Q(\xi)$ are known (e.g., [2,5]). In other words, $z^0(\xi)$ for a given ξ can be calculated in finite steps. *QF* is then solved by finding $\xi=\xi^0$ such that $z^0(\xi^0)=0$. The search for ξ^0 would be facilitated by resorting to the parametric programming technique developed for quadratic programming.

Based on the above observation, this paper proposes two algorithms which are both proved to be finite. Some computational results are given in Section 6.

2. Algorithm based on parametric programming

It is known [1,3] that $z^0(\xi)$ defined above is a continuous decreasing convex function of ξ, which also satisfies $z^0(0)\geq 0$ (by (1.4)) and $z^0(\infty)=-\infty$. (See Fig.1.) Thus $\xi=\xi^0$ may be found by starting from $\xi=0$ (or other appropriate point) and continuously increasing ξ until $z^0(\xi)=0$ is attained.

The Kuhn-Tucker theorem (e.g., [2]) shows that an optimal solution of

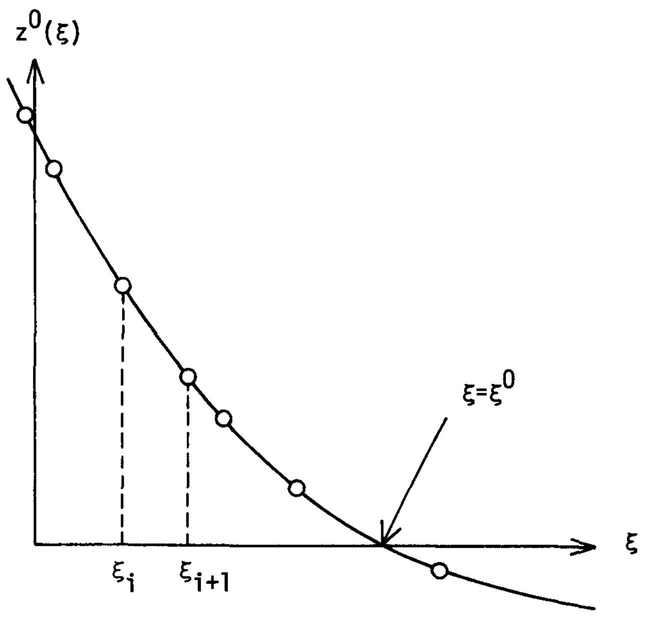

Fig. 1. $z^0(\xi)$ vs. ξ.

$Q(\xi)$ is obtained by solving the following problem.

$$Q'(\xi): \quad (2.1) \qquad n+m \left\{ \overbrace{\begin{bmatrix} K(\xi) & A^t & 0 \\ A & 0 & I_m \end{bmatrix}}^{n+2m} \right. \begin{pmatrix} x \\ u \\ y \end{pmatrix} = \begin{pmatrix} -c(\xi) \\ b \end{pmatrix}$$

$$(2.2) \qquad \begin{array}{l} y^t u = 0 \\ y \geq 0, \quad u \leq 0, \end{array}$$

where I_m is the unit matrix of order m, and y, u are m-vectors of variables.

Now let B be a basis of (2.1) (i.e., $(n+m)\times(n+m)$ nonsingular submatrix of the matrix in (2.1)) and let x_B, u_B, y_B be the basic variables corresponding to columns of B. Then

$$(2.3) \qquad \begin{pmatrix} x_B \\ u_B \\ y_B \end{pmatrix} = B^{-1} \begin{pmatrix} -c(\xi) \\ b \end{pmatrix}$$
$$\text{nonbasic variables} = 0,$$

is the basic solution corresponding to B. The solution (or B) is said feasible if (2.2) is also satisfied. It is known that x_B of any feasible solution (2.3) is an optimal solution of $Q(\xi)$. For a feasible B for $\xi=\xi_i$, the interval $[\xi_i, \xi_{i+1}]$ exists such that ξ_{i+1} is the maximum to keep B feasible for all $\xi\in[\xi_i, \xi_{i+1}]$. Thus the curve $z^0(\xi)$ as illustrated in Fig.1 is partitioned into subintervals, each of which corresponds to one basis B. (In Fig.1, a small circle indicates a point of basis change.)

Now we have the following algorithm for QF.

Algorithm A

A1 : $i\leftarrow 0$ and $\xi_i\leftarrow 0$ (or an appropriate value such that $z^0(\xi_0)\geq 0$).

A2 : Solve $Q'(\xi_i)$ (this implies to solve $Q(\xi_i)$). If $Q'(\xi_i)$ is infeasible, so is QF; halt. Otherwise obtain the maximum ξ_{i+1} such that $\xi_{i+1}\geq\xi_i$ and $Q'(\xi)$ has the same feasible basis for all $\xi\in[\xi_i, \xi_{i+1}]$.

A3 : If $z^0(\xi_{i+1})\leq 0$ or $\xi_{i+1}=\infty$, go to A4 ; otherwise let $i\leftarrow i+1$ and return to A2.

A4 : Calculate $\xi^0\in[\xi_i, \xi_{i+1}]$ such that $z^0(\xi^0)=0$, and halt. ξ^0 is the optimal value of QF.

Remark 2.1. $Q'(\xi_i)$ is solved in finite steps for example by the Wolfe's method [5,2]. (The Wolfe's method is used in the experiment of Section 6.) Given an optimal tableau for $Q'(\xi_i)$, an optimal tableau for $Q'(\xi_{i+1})$ is easily obtained by using the parametric programming technique developed by Ritter [4] and Wolfe [5] (Wolfe treats the case of $D=0$) in one pivot operation (provided that the nondegeneracy assumption holds). It is not necessary to solve $Q'(\xi_{i+1})$ from the initial tableau.

Remark 2.2. ξ_{i+1} in Step A2 is obtained from (2.3) by calculating the maximum of $\xi'(\geq\xi_i)$ such that the basic feasible solution (2.3) is feasible for all ξ satisfying $\xi_i\leq\xi\leq\xi'$. (See Example 3.1 of Section 3, in which this process is carried out.) If $D=0$, this computation becomes particularly simple (as studied by Wolfe [5]) since B^{-1} is independent of ξ. ($D=0$ is assumed in the computational experiment in Section 6.)

Remark 2.3. ξ^0 in Step A4 is the solution of equation

$$(2.4) \qquad \frac{1}{2}x^tK(\xi)x + c(\xi)^tx + d(\xi) = 0$$

which is the smallest not smaller than ξ_i, where x is given by (2.3). If $D=0$, (2.4) is a quadratic equation, and the smaller of two solutions is ξ^0.

Remark 2.4. An optimal solution of QF is easily obtained from (2.3) by setting $\xi=\xi^0$.

3. Finiteness of Algorithm A

Provided that a finite algorithm is used to solve $Q'(\xi_i)$ in Step A2 (see Remark 2.1), Algorithm A is proved to be finite if the number of intervals of $z^0(\xi)$ each of which corresponds to a basis (see Section 2) is finite. Since the number of possible bases of (2.1) is finite, it then suffices to show that the same basis B appears only finitely many times corresponding to different intervals.

Note that the finiteness is not tirvial since there is a case in which the same basis corresponds to more than one interval, as given in the next example. It also helps to visualize the idea used in the proof for the finiteness in the latter half of this section.

Example 3.1.

QF : maximize

$$\frac{\frac{1}{2}(x_1, x_2)\begin{bmatrix}-1 & -1\\ 1 & -2\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + (1, -1)\begin{pmatrix}x_1\\ x_2\end{pmatrix} + 180}{\frac{1}{2}(x_1, x_2)\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + (-1,-7)\begin{pmatrix}x_1\\ x_2\end{pmatrix} + 1}$$

subject to

$$\begin{bmatrix}1 & 1\\ -1 & 0\\ 0 & -1\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} \le \begin{pmatrix}7\\ 0\\ 0\end{pmatrix}.$$

The objective function of $Q(\xi)$ is then given by

$$z(\xi) = (x_1, x_2)\begin{bmatrix}-1-\xi & -1\\ 1 & -2-\xi\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + (1+\xi, -1+7\xi)\begin{pmatrix}x_1\\ x_2\end{pmatrix} + 180-\xi.$$

Constraint (2.1) of the Kuhn-Tucker theorem is

$$\begin{bmatrix}-1-\xi & -1 & 1 & -1 & 0 & \\ 1 & -2-\xi & 1 & 0 & -1 & 0\\ 1 & 1 & & & & \\ -1 & 0 & & 0 & & I_3\\ 0 & -1 & & & & \end{bmatrix}\begin{pmatrix}x_1\\ x_2\\ u_1\\ \vdots\\ u_3\\ y_1\\ \vdots\\ y_3\end{pmatrix} = \begin{pmatrix}-1-\xi\\ 1-7\xi\\ 7\\ 0\\ 0\end{pmatrix}.$$

Now take the following basis :

$$B = \begin{bmatrix}-1-\xi & -1 & \\ 1 & -2-\xi & 0\\ 1 & 1 & \\ -1 & 0 & I_3\\ 0 & -1 & \end{bmatrix}. \tag{3.1}$$

B^{-1} is then given by

$$B^{-1} = \begin{bmatrix} (-2-\xi)/\Delta & 1\ /\Delta & 0 \\ -1\ /\Delta & (-1-\xi)/\Delta & \\ (3+\xi)/\Delta & \xi\ /\Delta & \\ (-2-\xi)/\Delta & 1\ /\Delta & I_3 \\ -1\ /\Delta & (-1-\xi)/\Delta & \end{bmatrix}$$

where

$$\Delta = \det B = 3 + 3\xi + \xi^2.$$

Note that $x_B = (x_1, x_2)^t$, $u_B = \phi$ and $y_B = (y_1, y_2, y_3)^t$, and

$$\begin{pmatrix} x_B \\ \\ y_B \end{pmatrix} = \begin{pmatrix} (3-4\xi+\xi^2)/\Delta \\ (7\xi+7\xi^2)/\Delta \\ (18+18\xi-\xi^2)/\Delta \\ (3-4\xi+\xi^2)/\Delta \\ (7\xi+7\xi^2)/\Delta \end{pmatrix}.$$

Since $\Delta>0$ for any ξ, and

$$\begin{aligned} y_1 &\geq 0 \quad \text{for} \quad -0.95 \leq \xi \leq 18.95 \\ y_2 &\geq 0 \quad \text{for} \quad \xi \leq 1 \quad \text{or} \quad \xi \geq 3 \\ y_3 &\geq 0 \quad \text{for} \quad \xi \leq 1 \quad \text{or} \quad \xi \geq 0, \end{aligned}$$

(2.2) is satisfied in two intervals :

$$[-0.95,\ 1] \quad \text{and} \quad [3,\ 18.95].$$

When Algorithm A starts from $\xi_0=0$, therefore, basis B of (3.1) appears twice corresponding to intervals

$$[0,\ 1] \quad \text{and} \quad [3,\ 18.95].$$

(Note that $z^0(\xi_i)>0$ still holds for $\xi_i=3$.)

It was first proved by Ritter [4] in a more general setting that a basis

B appears only finitely many times as feasible basis when ξ is continuously increased.

In the following, the same result is proved by deriving an explicit upper bound (not given in [4]) on how many times a basis B appears in Step A2, by refining the argument used in Sections 3 and 4 of [4].

It is known in the theory of quadratic programming (e.g., [4]) that condition $y^t u=0$ of (2.2) permits us to consider only a basic solution with basis of the form

$$B = \begin{array}{cccc} \overbrace{}^{n} & \overbrace{}^{k} & \overbrace{}^{m-k} & \\ K(\xi) & A_1^t & 0 & \} n \\ A_1 & 0 & 0 & \} k \\ A_2 & 0 & I_{m-k} & \} m-k, \end{array} \tag{3.2}$$

where $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, and $(k\times n)$ matrix A_1 has full rank. Thus we consider in the subsequent discussion only bases in this form. Since $K(\xi)$ is assumed to be negative definite,

$$(-1)^n \det K(\xi) > 0 \tag{3.3}$$

holds for any ξ.

Lemma 3.1. B of (3.2) satisfies

$$(-1)^n \det B > 0$$

for any ξ. (Thus det B does not change its sign when ξ is continuously increased.)

Proof.

$$(-1)^n \det B = (-1)^n \det \begin{bmatrix} K(\xi) & A_1^t \\ A_1 & 0 \end{bmatrix} \det I_{m-k} \quad \text{(by (3.2))}$$

$$=(-1)^n \det \begin{bmatrix} I_n & 0 \\ A_1 K(\xi)^{-1} & I_k \end{bmatrix} \begin{bmatrix} K(\xi) & 0 \\ 0 & -A_1 K(\xi)^{-1} A_1^t \end{bmatrix} \begin{bmatrix} I_n & K(\xi)^{-1} A_1^t \\ 0 & I_k \end{bmatrix}$$

$$=(-1)^n \det K(\xi) \det [-A_1 K(\xi)^{-1} A_1^t] > 0.$$

The last relation follows from (3.3) and the property that $-A_1K(\xi)^{-1}A_1^{t}$ is positive definite (hence $\det[-A_1K(\xi)^{-1}A_1^{t}] > 0$). Q.E.D.

Lemma 3.2. Let u_i and y_j be elements of u_B and y_B defined in (2.3) for B of (3.2). Then the number of intervals in which u_i assumes nonpositive values when ξ is changed from 0 to ∞ is at most $\left\lceil \frac{n-k+2}{2} \right\rceil$, where $\lceil x \rceil$ denotes the smallest integer not smaller than x. Similarly, the number of intervals in which y_j assumes nonnegative values is at most $\left\lceil \frac{n-k+1}{2} \right\rceil$.

Proof. From (2.3), we have

$$\begin{pmatrix} x_B \\ u_B \\ y_B \end{pmatrix} = B^{-1} \begin{pmatrix} -c(\xi) \\ b \end{pmatrix}$$

$$= \begin{bmatrix} \Delta_{11}/\Delta & \Delta_{21}/\Delta \;\ldots & \Delta_{n+m1}/\Delta \\ \vdots & & \vdots \\ \Delta_{1n+m}/\Delta & \cdots\cdots & \Delta_{n+mn+m}/\Delta \end{bmatrix} \begin{pmatrix} -c(\xi) \\ b \end{pmatrix}, \tag{3.4}$$

where Δ_{ij} is the (i, j)-th cofactor of B and $\Delta = \det B$. Since each element of $K(\xi)$ is linear in ξ and other elements of B do not depend on ξ, the degree (in ξ) of the numerator of each element of B^{-1} is easily calculated ; it is given below.

$$\begin{array}{c|c|c|l} \overbrace{}^{n} & \overbrace{}^{k} & \overbrace{}^{m-k} & \\ (n-k-1) & (n-k) & (0) & \} \, n \\ \hline (n-k) & (n-k+1) & (0) & \} \, k \\ \hline (n-k-1) & (n-k) & (0) & \} \, m-k \end{array} \tag{3.5}$$

Note that only the numerator is important from the view point of the sign, since the denominator Δ does not change its sign by Lemma 3.1. From (3.5), the degrees of the numerators of x_B, u_B, y_B are obtained, using that $c(\xi)$ is linear in ξ :

$$\begin{pmatrix} x_B \\ u_B \\ y_B \end{pmatrix} : \begin{pmatrix} (n-k) \\ \hline (n-k+1) \\ \hline (n-k) \end{pmatrix} \begin{array}{l} \} \, n \\ \} \, k \\ \} \, m-k. \end{array} \tag{3.6}$$

Now note that the number of intervals in which a polynomial of degree r assumes nonnegative (nonpositive) values is at most $\left\lceil \frac{r+1}{2} \right\rceil$. By (3.6), this proves the lemma statement. Q.E.D.

Theorem 3.3. Let B be given by (3.2). When ξ is increased from 0 to ∞, the number of intervals of ξ in which u_B and y_B of basic solution (2.3) satisfy $u_B \leq 0$ and $y_B \geq 0$ (i.e., feasible) is at most

$$(3.7) \qquad N(k) = k\left\lceil \frac{n-k+2}{2} \right\rceil + (m-k)\left\lceil \frac{n-k+1}{2} \right\rceil - (m-1).$$

Proof. Let $p_i(\xi)$ be a polynomial of degree r_i, $i=1,2,\dots,m$. Then it is easy to show that the number of intervals of ξ in which $p_1(\xi),\dots, p_k(\xi)$ are nonpositive and $p_{k+1}(\xi),\dots, p_m(\xi)$ are nonnegative is at most

$$(3.8) \qquad \sum_{i=1}^{m} \left\lceil \frac{r_i+1}{2} \right\rceil - (m-1).$$

(3.7) follows from (3.8) by substituting

$$r_i = \begin{cases} n-k+1 & , \quad i = 1,2,\dots, k \\ n-k & , \quad i = k+1,\dots, m \end{cases}$$

obtained in Lemma 3.2. Q.E.D.

Corollary 3.4. $N(n) = 1$.

A basis B with $k=n$ represents an extreme point of polyhedron $Ax \leq b$. This corollary tells that such basis appears at most once as a feasible basis in Algorithm A.

Corollary 3.5. $N(0) = m\left\lceil \frac{n+1}{2} \right\rceil + 1-m$.

Note that a basis B with $k=0$ corresponds to an interior point of polyhedron $Ax \leq b$.

Theorem 3.6. Algorithm A of Section 2 gives an optimal solution of QF

or indicates its infeasibility, after executing Step A2 finite times, provided that $Q'(\xi_i)$ is solved by a finite algorithm.

Proof. The finiteness follows from the argument given in the beginning of this section and Theorem 3.3. If Algorithm A halts in Step A2 indicating the infeasibility of $Q'(\xi_i)$, then $Ax \leq b$ is vacuous (i.e., QF is infeasible) since it is known that a quadratic programming problem with a negative definite objective function (such as $Q(\xi_i)$) always has an optimal solution if $Ax \leq b$ is not vacuous. On the other hand, if Algorithm A halts in Step A4, ξ^0 is the maximum value of QF as proved in [1,3]. Q.E.D.

4. Two special cases

In this section, two special cases $D=0$ and $C=0$ are discussed. In either case, it is shown that each basis B appears at most once in Algorithm A.

Theorem 4.1. Assume $D=0$ in QF. Then a basis B of (3.2) appears at most once as a feasible basis of $Q'(\xi)$ in Step A2 of Algorithm A.

Proof. In this case, no element of B of (3.2) contains ξ since $K(\xi)=C-\xi D=C$; hence no element of B^{-1} contains ξ. Thus x_B, u_B, y_B given by (2.3) is linear in variable ξ, since $-c(\xi)$ is linear in ξ. Letting $r_i=1$ in (3.8) gives $N(k)=1$ in the proof of Theorem 3.3. Q.E.D.

As mentioned in remarks given to Algorithm A, the case of $D=0$ has also other computational advantages. The computational experiment in Section 6 is therefore done for this simple case only.

The case of $C=0$ is similar. However, it is necessary to assume that

(4.1) D is positive definite

(4.2) $r^t x+s > 0$ for some $x \in S$

in addition to (1.4), since C is not negative definite in this case. Algorithm A should be started from $\xi_0=\delta$, where δ is an appropriate positive number satisfying $z^0(\delta) \geq 0$.

Theorem 4.2. Assume $C=0$ in QF and let (4.1) (4.2) be satisfied. Then basis B of (3.2) appears at most once as a feasible basis in Step A2 of Algorithm A, provided that ξ_0 is set to the above δ in Step A1.

Proof. In this case, B of (3.2) is given by

$$B = \begin{bmatrix} -\xi D & A_1^t & 0 \\ A_1 & 0 & 0 \\ A_2 & 0 & I_{m-k} \end{bmatrix} .$$

In a manner similar to the proof of Lemma 3.1, we have

$$\begin{aligned} \Delta(=\det B) &= \det K(\xi) \det [-A_1 \; K(\xi)^{-1} A_1^t] \\ &= (-\xi)^{n-k} \det D \det[-A_1 D^{-1} A_1^t]. \end{aligned}$$

Thus B^{-1} in this case has the following form.

$$B^{-1} : \begin{array}{c} \begin{array}{ccc} \overbrace{\quad}^{n} & \overbrace{\quad}^{k} & \overbrace{\quad}^{m-k} \end{array} \\ \left[\begin{array}{c|c|c} \frac{1}{\xi}\alpha & \alpha & \alpha \\ \hline \alpha & \xi\alpha & \alpha \\ \hline \frac{1}{\xi}\alpha & \alpha & \alpha \end{array}\right] \end{array} \begin{array}{l} \\ \} \; n \\ \} \; k \\ \} \; m-k \end{array} ,$$

where α stands for a real number independent of ξ (each α may represents a different number). Since it can be written that

$$\begin{pmatrix} -c(\xi) \\ b \end{pmatrix} : \left(\begin{array}{c} \xi\alpha+\beta \\ \hline \alpha \end{array}\right) \begin{array}{l} \} \; n \\ \} \; m \end{array}$$

by using the same notation as above, we have

$$\begin{pmatrix} x_B \\ u_B \\ y_B \end{pmatrix} = B^{-1} \begin{pmatrix} -c(\xi) \\ \\ b \end{pmatrix} : \left(\begin{array}{c} \frac{1}{\xi}(\xi\alpha+\beta) \\ \hline \xi\alpha+\beta \\ \hline \frac{1}{\xi}(\xi\alpha+\beta) \end{array}\right) \begin{array}{l} \} \; n \\ \} \; k \\ \} \; m-k \end{array} .$$

This proves that the numerator of each element of x_B, u_B, y_B is linear in ξ ;

the denominator does not change its sign by Lemma 3.1 and $\xi>0$. Thus we can let $r_i=1$ in (3.8), obtaining $N(k)=1$ in the proof of Theorem 3.3. Q.E.D.

5. Dinkelbach's method and its modification

In order to solve QF, the Dinkelbach's method [1] may also be directly applied. It obtains ξ^* (≥ 0) and the corresponding feasible solution x^* of QF such that

$$0 \leq z^0(\xi^*) \leq \varepsilon,$$

where ε is a given nonnegative constant.

Dinkelbach's algorithm

D1 : $i \leftarrow 0$, $\xi_i \leftarrow 0$ (or an appropriate value such that $z^0(\xi_0) \geq 0$).

D2 : Solve $Q'(\xi_i)$. If $Q'(\xi_i)$ is infeasible, so is QF ; halt. Otherwise let x-part of a feasible solution of $Q'(\xi_i)$ be $x(i)$ (i.e., $x(i)$ is an optimal solution of $Q(\xi_i)$).

D3 : If $z^0(\xi_i) \leq \varepsilon$, let $\xi^* \leftarrow \xi_i$, $x^* \leftarrow x(i)$ and halt ; otherwise

$$\xi_{i+1} \leftarrow (\tfrac{1}{2}\, x(i)^t Cx(i) + r^t x(i) + s) / (\tfrac{1}{2}\, x(i)^t Dx(i) + p^t x(i) + q)$$

$$i \leftarrow i+1$$

and return to D2.

If $\varepsilon>0$, this algorithm halts after executing Steps D2 and D3 finitely many times. If $\varepsilon=0$, i.e., an exact optimal solution is sought, however, this algorithm usually requires an infinite number of iterations of Steps D2 and D3. This difficulty can be easily removed by modifying it as follows, by making use of the property that interval $[\xi_i, \xi_i']$ maintaining the same basis is easily calculated as discussed in Section 2. (This idea is also implicitly used in the numerical example in Appendix of [1].)

Algorithm B (Modified Dinkelbach's method)

B1 : $i \leftarrow 0$, $\xi_i \leftarrow 0$ (or an appropriate value such that $z^0(\xi_0) \geq 0$).

B2 : Solve $Q'(\xi_i)$. If $Q'(\xi_i)$ is infeasible, so is QF ; halt. Otherwise obtain the maximum ξ_i' such that $\xi_i' \geq \xi_i$ and $Q'(\xi)$ has the same feasible basis for all $\xi \in [\xi_i, \xi_i']$. Let x-part of a feasible solution of $Q'(\xi_i')$ be $x'(i)$

(i.e., $x'(i)$ is an optimal solution of $Q'(\xi_i')$).

B3 : If $z^0(\xi_i')\leq 0$ or $\xi_i'=\infty$, go to B4 ; otherwise

$$\xi_{i+1} \leftarrow (\tfrac{1}{2}\, x'(i)^t Cx'(i)+r^t x'(i)+s)/(\tfrac{1}{2}\, x'(i)^t Dx'(i)+p^t x'(i)+q)$$

$i \leftarrow i+1$

and return to B2.

B4 : Calculate $\xi^0\in[\xi_i,\ \xi_i']$ such that $z^0(\xi^0)=0$ and halt ; ξ^0 is the maximum value of QF.

The computational process of these algorithms are illustrated in Fig.2, in which solid arrows correspond to Algorithm B and broken arrows to the Dinkelbach's method. ξ_i generated in the Dinkelbach's method is shown with bar $\bar{\xi}_i$ to distingush it from ξ_i of Algorithm B. It is noted that ξ_{i+1} obtained in Step D3 (resp. Step B3) is given as the intersection point of ξ-axis and the line tangent to $z^0(\xi)$ at $\bar{\xi}_i$ (resp. ξ_i'). From Fig.2, it may be seen that Algorithm B requires less number of iterations than the original Dinkelbach's method.

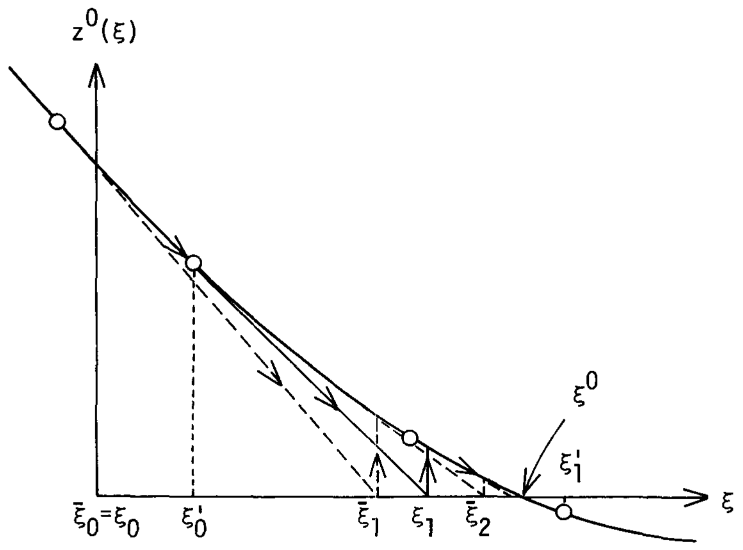

Fig. 2. Illustration of computational processes of the Dinkelbach's method and Algorithm B. (Solid arrows indicate Algorithm B and broken arrows indicate the Dinkelbach's method.)

Remark 5.1. $Q'(\xi_i)$ $(i>0)$ in Steps D2 or B2 may be solved starting from the final tableau of $Q'(\xi_{i-1})$. This is a great computational saving, compared with solving $Q'(\xi_i)$ from scratch. However, it may still require a considerable number of pivot operations if ξ_i is far from ξ_{i-1}. This should be compared with the fact that only one pivot operation is required in Algorithm A to solve $Q'(\xi_i)$ (Remark 2.1). Although Step B2 of Algorithm B is usually carried out far fewer times than Step A2 of Algorithm A, it is not clear which of Algorithm A and Algorithm B is more efficient.

Theorem 5.1. Algorithm B gives an optimal solution of *QF* or indicates its infeasibility, after executing Step B2 finite times, provided that $Q'(\xi_i)$ is solved by a finite algorithm.

Proof. The validity of Algorithm B follows from the validity of the Dinkelbach's method [1]. The finiteness can be proved in a manner similar to Algorithm A (Theorem 3.6). Q.E.D.

6. Computational results

Algorithm A and Algorithm B discussed in the previous sections are implemented in *FORTRAN* and run on *FACOM* 230/60 (roughly equivalent to *IBM* 360/65 or *UNIVAC* 1108) and *FACOM* 230/75 (roughly equivalent to *IBM* 370/165) of Kyoto University. Problems *QF* with $D=0$ (see (1.3)) are exclusively treated in the experiment.

For various sizes n and m of *QF*, coefficients are randomly generated by the following rule :

C : A negative definite symmetric matrix C of size $n\times n$ is obtained by

$$C=-PP^t$$

for a nonsingular matrix P. P is generated by (i) randomly specifying nonzero elements of P with probability *NZC* (program parameter), (ii) assigning a nonnegative (two digit) number randomly taken from the uniform distribution with interval [0.0, 9.9] to each nonzero element, and (iii) randomly inverting the sign of nonzero elements with probability *NC* (program parameter). Note that C has a considerably higher nonzero density than *NZC* ; for example, C of two typical problems in Table 1 have nonzero densities 89.5% for *NZC*=0.4 and 56% for *NZC*=0.25.

A : An $m\times n$ matrix A is generated by (i) randomly specifying nonzero elements with probability NZA (program parameter), (ii) assigning a nonnegative (three digit) number randomly taken from interval [00.0, 99.9] to each nonzero element, and (iii) randomly inverting the sign of nonzero elements with probability NA (program parameter).

r, p : Each element is assigned a (two digit) nonnegative number randomly taken from [0.0, 9.9].

b : Each element is assigned a (four digit) positive number randomly taken from [100.0, 199.9].

s=20.0 and q=3.0 for all problems.

Tables 1 and 2 summarize the computational results. The results in Table 1 are the average of 10 problems with n=m=20, while Table 2 lists results for each problem of larger size.

Table 1. Computational results for problems with n=20 and m=20.

(All figures are the average of 10 problems.)

Algorithm	$Q'(\xi_0)$		$Q'(\xi_i)$, $i>0$		Iteration (e)	Total time (sec.) (c, f)	Remark
	Pivot$^{(a)}$	Time$^{(c)}$ (sec.)	Pivot$^{(b)}$	Time$^{(c)}$ (sec.)			
A	114.7	12.7	14.1	1.9	15.1	14.6	NZA=0.4-0.7 NZC=0.2-0.5
B	114.7	12.7	19.9	2.9	2.6	15.6	NA=0.2-0.4 NC=0.2-0.4

Table 2. Computational results for large problems

Problem	Algorithm	$Q'(\xi_0)$		$Q'(\xi_i)$, $i>0$		Iteration (e)	Total Time (sec.) (d, f)	Remark
		Pivot (a)	Time$^{(d)}$ (sec.)	Pivot (b)	Time$^{(d)}$ (sec.)			
n=40 m=40	A B	344 344	31.8 31.8	25 22	2.7 2.5	25 2	34.5 34.3	NZA=0.6 NZC=0.4 NA=NC=0.2
n=50 m=50	A B	586 586	85.3 85.3	25 53	4.2 9.6	26 2	89.5 94.9	NZA=0.7 NZC=0.4 NA=NC=0.2
n=50 m=50	A B	606 606	83.2 83.2	27 68	4.5 12.5	28 2	87.7 95.7	NZA=0.5 NZC=0.4 NA=NC=0.2
n=50 m=50	A B	575 575	71.1 71.1	24 39	3.9 6.7	25 2	75.0 77.8	NZA=0.35 NZC=0.4 NA=NC=0.2
n=50 m=50	A B	694 694	110.6 110.6	29 57	4.8 9.5	30 3	115.4 120.1	NZA=0.4 NZC=0.17 NA=NC=0.2

Notes

(a) The number of pivots required to solve $Q'(\xi_0)$ by the Wolfe's method.

(b) The number of pivots required to solve $Q'(\xi_i)$ for all $i>0$.

(c) Machine is *FACOM* 230/60.

(d) Machine is *FACOM* 230/75.

(e) The number of executions of A2 (or B2) including the ones for $Q'(\xi_0)$.

(f) Including the computation for A3, A4 (or B3, B4).

From these results, it may be concluded that Algorithm A is slightly faster than Algorithm B but there is no significant difference. It is also noticed that computation for $Q'(\xi_0)$ (the first one) in Step A2 or B2 is rather expensive compared with the rest (i.e., computation for $Q'(\xi_i)$, $i>0$) ; computation for $Q'(\xi_0)$ consumes roughly 80~90% of the entire computation time. Thus the use of parametric programming technique seems quite effective. This also explains a reason for the similarity (in computation time) of Algorithm A and Algorithm B mentioned above ; two algorithms differ only in the way of generating ξ_i for $i>0$.

It is also observed that program parameters specifying the ratios of non-zero and negative coefficients do not have much influence on the relative behavior of Algorithms A and B, though the higher values tend to increase the computation time. Table 1 includes problems with various parameter values.

In conclusion, it can be said that the quadratic fractional programming problem with $D=0$ is a rather easy nonlinear programming problem, which can be solved in computational effort only slightly greater than that required for the well known (concave) quadratic programming problem.

Acknowledgement

The authors are indebted to Miss T. Kanazawa for her excellent typing.

References

[1] W. Dinkelbach, On nonlinear fractional programming, Management Science, 13 (1967), pp. 492-498.

[2] G. Hadley, Nonlinear and Dynamic Programming, Addison Wesley, 1964.

[3] R. Jagannathan, On some properties of programming problems in parametric form pertaining to fractional programming, Management Science, 12 (1966), pp. 609-615.

[4] K. Ritter, Ein Verfahren zur Lösung parameterabhängiger, nichitlinearer Maximum-Probleme, Unternehmensforschung, 6 (1962), pp. 149-166. (English translation by M. Meyer, A method for solving nonlinear maximum-problems depending on parameters, Naval Research Logistics Quarterly, 14 (1967), pp. 147-162.)

[5] P. Wolfe, The simplex method for quadratic programming, Econometrica, 27 (1959), pp. 382-398.

(The authors' address : Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto Japan.)