On Wolfe's Algorithm for Quadratic Programming

Ernesto de Castro Icogo (Author)
Institute of Mathematical Sciences and Physics
University of the Philippines at Los Banos
College, Laguna, Philippines
Email: ernesto.icogo@yahoo.com

Abstract. *The Kuhn-Tucker Theorem is applied to quadratic programming problem leading to Wolfe's algorithm. A computer program implementation of the Wolfe's algorithm and its application to specific practical problem are given.*

AMS subject classifications:

Key Words: Kuhn-Tucker Theorem, Wolfe's Algorithm, quadratic programming problems

I. INTRODUCTION

*A. **Background of the Study***

The general quadratic programming problem can be written:

Maximize

$$Z = x'Dx + c'x \qquad (****)$$

subject to:

$$\begin{array}{llll} g_i(x) & \leq & b_i & (i = 1, \dots, s) \\ g_i(x) & \geq & b_i & (i = s+1, \dots, t) \\ g_i(x) & = & b_i & (i = t+1, \dots, m) \\ x_i & \geq & 0 & \end{array}$$

The matrix D is in the order (nxn) and is assumed, without loss of generality, to be symmetric; A is an (mxn) matrix, b an m-component column vector and c' an n-component row vector. Another assumption is that the quadratic form x'Dx is either negative definite or negative semidefinite.

Applications of quadratic programming problem usually arise in such diverse problems as:

Regression analysis with non-negativity constraints and/or constraints of upper or lower bounds on the regression parameters;

To find the minimum of a general convex function to linear inequalities, where the convex function is twice differentiable and may be locally approximated by a positive definite quadratic form;

Production planning by maximizing profit when the marginal cost is linearly varying, and also the production function is linearly varying with the problem variable;

In a linear model, for given ranges of expected profits, to find the solution which minimize the variance (risk level) of such profit.

Also, quadratic programming problem has also been applied in areas as optimal use of milk in Netherlands, or optimal alignment of the vertical profile of highways.

For the past years, different algorithms were developed to solve such problems. However, none of these algorithms were implemented using computers.

One of the methods known to solve quadratic programming problems is Wolfe's algorithm. This method is the simplest and the most easy to implement in a computer program. It can also be used to solve large class of quadratic programming problems

.

II. STATEMENT OF THE PROBLEM

There is a need that Wolfe's algorithm should be implemented using computers to do away with complicated and strenuous manual computations and faster solutions to its applications mentioned above.

This research problem is then devoted to studying Wolfe's algorithm. In particular, it deals with the mathematical foundations and underlying theory of the algorithm, its implementation as a computer code, its application to such diverse quadratic programming problems mentioned, and its efficiency analysis.

III. SIGNIFICANCE OF THE STUDY

The development and implementation of a computer program using Wolfe's algorithm will present an efficient and compact solution to quadratic programming problems.

In generating the computer implementation, complicated and strenuous manual computations will be eliminated and faster solutions will be achieved.

Also, large and complicated class of QP problems can be easily solved without the natural computational (manual) errors and it is more convenient.

And because of these needs, this research study will "bridge the gap" between the method, itself, and its technical importance to different applications mentioned earlier.

IV. OBJECTIVES

The primary objectives of this study are:

1. To study the background necessary to understand Wolfe's algorithm
2. To study how Wolfe's algorithm can be implemented as computer code
3. To devise a computer program for Wolfe's algorithm to solve quadratic programming problems
4. To implement the study on a practical quadratic programming problem
5. To make an analysis on the efficiency of the computer implementation

V. Review of Literature

The development and success of Dantzig's Simplex Method in Linear Programming encouraged other Operations Research specialists to venture to other fields of optimization. One of the areas which get considerable attention was Quadratic Programming (QP), which falls under Nonlinear Programming.

Although, at first, they encountered difficulties because of nonlinearity constraints, they somehow managed to get away with these and developed different methods.

Beale (1958) has developed a method of great intuitive appeal which is particularly suited to hand computation. The method is an extension of the simplex method for linear programming. (5)

Hildreth and D'Esopo (1958) has developed an asymptotic method for the solution of a quadratic program which makes direct use of the duality. The individual steps in the iteration are of the greatest simplicity, which may compensate for relatively poor convergence. The applicability of the method is somewhat limited by the fact that the objective function must be strictly convex. (5)

Barankin and Dorfman (1958) first pointed out that, if the linear Lagrangian conditions of optimality were combined with those of the original system, the optimum solution was a basic solution in the enlarged system with the property that only one of the certain pairs of variables were in the basic set. (1)

Markowitz (1956-59), on the other hand, showed that it was possible to modify the enlarged system and then parametrically generate a class of basic solutions which converged to the optimum in a finite number of iterations. (1)

Finally, Wolfe (1959) proved that by slightly modifying the simplex algorithm so as not to all a variable to enter the basic set. Thus, by modifying a few instructions in a simplex code for linear programs, it was possible to solve a convex quadratic program! (7)

In his paper, Wolfe (1959) listed four ways where Quadratic programming can arise: (7)

Regression: To find the best least-square fit to given data, where a certain parameters are known a priori to satisfy linear inequalities constraints

Efficient Production: Maximization of profit, assuming linear production functions and linearly varying marginal cost (Dorfman, 1959)

Minimum Variance: To find the solution of a linear program with variable cost coefficients which could have given expected costs and minimum variance (Markowitz, 1956)

Convex Programming: To find the minimum of a general convex function under linear constraints and quadratic approximations. (White, Johnson and Dantzig)

J. Boot (1963) discussed the problem on how the solution vector of a quadratic programming problem changes in case of infinitesimal changes in the data of the problem. He applied these results to a numerical example that arose in optimal use of milk in Netherlands.

Guder and Buongiorno successfully made an economic model of the North American newsprint industry forecasts over long time periods and by regions, the amount of newsprint produced, consumed, its price, and the quantities transported between regions. The model forecasts the same data for the main raw materials, pulpwood and wastepaper. The core of the model is a recursive quadratic programming system that simulates the behavior of a competitive industry. The model has been use to make a 10-year forecasts of key variables describing the industry, based on specific scenarios on economic and demographic growth in the various regions of North America. (8,9)

This vast importance of the problem, however, remains in methods with no computer implementations. This research study will bridge the gap between methods and its technical importance.

VI. METHODOLOGY

Activities:

1. Study of Kuhn-Tucker Theorem and its applications to Quadratic Programming

Here the mathematical foundations of the problems were tackled and analyzed

2. Study the use of the Linear Programming to obtain solution to Kuhn-Tucker conditions

Here the theories behind Linear Programming were applied to the problem on how to get a distinct solution to the Kuhn-Tucker conditions. This is actually the study o Wolfe's algorithm, itself.

3. Computer implementation of Wolfe's algorithm

The computer program was devised and in good running condition

4. Testing and validation of the computer implementation

The computer program is then applied to a practical quadratic problem and numerical results were obtained.

5. Efficiency studies of the computer implementation

Here analysis on the numerical examples of the problem applications was undertaken

6. Writing of Manuscript

VII. MATHEMATICAL PRELIMINARIES

In this chapter, the reader is assumed to have previous knowledge in linear algebra and mathematical programming. Nevertheless, in order to insure clarity, basic concepts and definitions will be further discussed. However, a number of proofs were omitted and for further details, the interested reader may consult the given references.

A. The Simplex Method

The simplex method for solving linear programming problems was developed by George Dantzig in connection with his work on planning problems for the federal government.

The algorithm is given by the following steps:

1. *Initialization step:*

Start with a basic feasible canonical form.

2. *Optimality test:*

Are C_j's ≥ 0? Where Cj's are the coefficients of the objective functions.

If yes: STOP → OPTIMAL!
If no: CONTINUE

3. *Entrance rule:*

   Choose the pivot column by identifying the most negative coefficient in the X-row. (Objective function row)

4. *Exit rule:*

   Scan the pivot column for positive coefficients

   Let c – be the index of the pivot column

   Compute the ratios:

$$\theta_i = \frac{b_i}{a_{ic}}, \quad \text{if } a_{ic} > 0$$

$$‘ \quad \text{if } a_{ic} \leq 0$$

$$\theta = \min_{\text{for every } i} \{\theta_i\} \rightarrow r^{th} \text{ row is pivot row}$$

$\rightarrow$ basic variable along rth (X_r) leaves the basis.

If there is no leaving variable $\rightarrow$ unbounded

5. Pivot on the r^{th}-*element*.
   Get the new canonical form and GO BACK TO STEP 2.

THE TWO-PHASE Simplex:

From the regular simplex method emerged the Two-Phase simplex which can be used when the basic feasible canonical form is not evident. The first phase of the method was later used in this study.

The algorithm is given below:

PHASE 1:

Step 1:

Add artificial variables $r_i \geq 0$ that will correspond to the missing unit columns

Step 2:

Ignore the original objective function. Concentrate on minimizing the sum of the artificial variables r's

$$\text{Min } r_0 = \sum r_i$$

Step 3:

Set up the 1st Phase I tableau
But the r_i's are supposed to be basic variables, hence, they should have zero coefficients in the r_0-row

Step 4: Do simplex

a) Modified entrance rule: Pivot along column with the most negative coefficients in the objective function
b) If an artificial variable leaves the basis, the column for that variable should be deleted in the next tableau

Step 5:

There are two possible cases at the end of Phase I

Case1: At the optimal tableau, VOF could be
$r_0 > 0 \rightarrow$ The original L.P. is infeasible.

Case2: At the optimal tableau, VOF could be
$r_0 = 0 \rightarrow$L.P. is feasible

PHASE I has generated a feasible canonical form in terms of the original variables x'.

Go to transition step.

TRANSITION STEP:

If the variable involved in the original objective function are in the basis of the optimal PHASE I tableau, substitute these out of the objective function row.

PHASE II:

1. Set up the first PHASE II tableau
2. Pivot as usual

THEOREM 2.1: If the linear programming problem has an optimal solution, then it has a basic feasible solution that is optimal.

On Quadratic Forms

In this section, we consider some properties that certain quadratic forms may possess. These properties are of essential use in the understanding of the Quadratic programming problem.

Definition 1: The quadratic form $q(x) = x'Dx$ is:

i) *positive definite* if $q(x) \geq 0$ for all $x=0$ in E
ii) *positive semi-definite* if $q(x) \geq 0$ for all x in E but q(x) is not positive definite.

Negative definite and semi-definite forms are analogously defined.

THEOREM 2.2: Let the symmetric n-by-n matrix D be positive (negative) definite,

a) the inverse D exists
b) D^{-1} is a positive (negative) definite and
c) ADA' is positive (negative) semi-definite for any m-by-n matrix A

CONVEX AND CONCAVE FUNCTIONS

In this section, we shall be particularly interested in the maximization and minimization of these two special classes of mathematical functions and also their properties.

The basic concepts in this theme are of great importance in the development of the Kuhn-Tucker Theory, which is considered the backbone of this study.

Definition 2: The scalar function f(x) is a convex function defined over a convex set X in E^n if for any two points x_1 and x_2 in X,

$$F(\lambda x_1 + (1-\lambda)\, x_2) \quad <= \quad \lambda f(x_1) + (1-\lambda)\, f(x_2)$$

for all λ such that $0 \leq \lambda \leq 1$. Similarly, f(x) is a concave function if for any x_1 and x_2 in X and any λ satisfying $0 \leq \lambda \leq 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda)\, f(x_2)$$

THEOREM 2.3: If f(x) is convex, then – f(x) is concave and vice-versa.

This theorem is consistent with the notion that:

Max $z = f(x)$ is equivalent to min $-z = -f(x)$

THEOREM 2.4: The sum of two or more convex (concave) functions is convex (concave).

THEOREM 2.5: The feasible region of the mathematical program

Max $f(x)$ $\qquad x = (x_1, x_2, \dots , x_n)$

subject to:

$$g_i(x) \quad \begin{matrix} \leq \\ \geq \\ = \end{matrix} \quad b_i \qquad i = 1,2, \dots , n$$

is a convex set if the following three sufficient conditions are met:

a) For all constraints $g_i(x) \leq b_i$, the function $g_i(x)$ is convex.
b) For all constraints $g_i(x) \geq b_i$, the function $g_i(x)$ is concave.
c) For all constraints $g_i(x) = b_i$, the function $g_i(x)$ is linear.

THE KUHN-TUCKER THEOREM

In 1951, Kuhn and Tucker produced some results which gave new insight to the nature of the optimal solutions of general nonlinear programming problem (NLP). They obtained necessary conditions for a local optimum and in certain important special cases, necessary and sufficient conditions for a global optimum.

In many cases of practical interest, it turns out that the application of Kuhn-Tucker Theory to Quadratic Programming Problem (QPP) leads to a very efficient algorithm for its solution called the Wolfe's algorithm.

Application of Kuhn-Tucker to general Nonlinear Programming Problem (NLP)

Consider the general nonlinear programming problem:

Maximize $z = f(x)$

subject to

$$\begin{array}{llll} g_i(x) & \leq & b_i & (I = 1, \dots , s) \\ g_i(x) & \geq & b_i & (I = s+1, \dots , t) \\ g_i(x) & = & b_i & (I = t+1, \dots , m) \\ x_i & \geq & 0 & \forall i \end{array}$$

And define the index set:

$$\begin{array}{lll} I_a & = & \{i: g_i(x^*) = b_i\} \\ I_p & = & \{i: g_i(x^*) \neq b_i\} \\ J_a & = & \{j: x_j^* = 0\} \\ J_p & = & \{j: x^* \neq 0\} \end{array} \qquad (**)$$

Where $x = x^*$ is any feasible solution satisfying the constrained in (*). The subscript a and p on I and J denote active and passive constraints and non-negativity restrictions respectively. Also, define the extended Lagrangean function

$$Fe\ (x,\lambda) = f(x) + \sum \lambda_i\ (b_i - g_i\ (x)) - \sum \lambda_{m+1}\ x_j \qquad (***)$$

Where $\lambda_1, \lambda_2, \dots , \lambda_{m-1}, \lambda_m, \dots , \lambda_{m+j}$ are Lagrange multipliers or Kuhn-Tucker multipliers associated with these (m+n) constraints.

The following theorem directly leads to the Kuhn-Tucker conditions.

THEOREM 3.1

For the problem (*) the m Lagrange multipliers in the extended Lagrangian function satisfy the following conditions:

$$\frac{\partial f(x^*)}{\partial x_j} - \sum \lambda i \frac{g_i(x)}{\partial x_j} \quad \begin{array}{ll} \leq 0 & (j \in J_a) \\ = 0 & (j \in J_p) \end{array} \qquad (1.1)$$

$$\lambda_i^* \quad \begin{array}{lll} \geq & 0 & (i = 1, \ldots, s) \\ \leq & 0 & (i = s+1, \ldots, t) \\ \text{Unrestricted} & & (i = t+1, \ldots, m) \end{array}$$

For the proof of this, see the book of Walsh (2).

The four Kuhn-Tucker conditions can be derived directly from Theorem 1. For the derivations, refer to Walsh (2).

THEOREM 3.2: KUHN-TUCKER (K-T) CONDITIONS

If f(x) takes on a constrained local maximum at the point x = x*, it is necessary that a vector exists such that:

KUHN-TUCKER CONDITION I (K-T- I)

$$\nabla_x F(x^*,\lambda^*) = f(x^*) - \sum \lambda_i^* \Delta g_i (x^*) \leq 0 \qquad (2.1)$$

KUHN-TUCKER CONDITION II (K-T- II)

$$[\nabla_x F(x^*,\lambda^*)]\, x^* = \sum \left\{ \frac{\partial f}{\partial x_j}(x^*) - \sum \frac{\partial g_i}{\partial x_j}(x^*) \right\} x^* = 0 \qquad (2.2)$$

KUHN-TUCKER CONDITION III (K-T- III)

$$F(x^*,\lambda^*) = [\, b_1 - g_1 (x^*), \ldots, b_m - g_m(x^*) \,] \leq 0 \qquad (2.3)$$

KUHN-TUCKER CONDITION IV (K-T- IV)

$$[\nabla_\lambda F(x^*,\lambda^*)]\, \lambda^* = \sum [[\, b_1 - g_1 (x^*) \,]\, \lambda_1^* = 0 \qquad (2.4)$$

A. Application of Kuhn-Tucker to Quadratic Programming Problem (QPP)

Consider the Quadratic Programming Problem (QPP)

maximize $x'Dx + c'x$

subject to

$$\begin{array}{lll} g_i(x) & \leq b_i & (I = 1, \dots , s) \\ g_i(x) & \geq b_i & (I = s+1, \dots ,t) \qquad (****) \\ g_i(x) & = b_i & (I = t+1, \dots , m) \end{array}$$

From (***)

$$F(x,\lambda) = f(x) + \sum \lambda_i (b_i - g_i(x)) - \sum \lambda_{m+j} x_j$$

Define:

i) $x'Dx = \sum\sum x_i d_{ij} x_j$
$= x_1d_{11}x_1 + x_1d_{12}x_2 + \dots + x_nd_{nn}x_n$ (3.1)

ii) $c'x = \sum c_jx_j$
$= c_1x_1 + c_2x_2 + c_3x_3 + \dots + c_nx_n$ (3.2)

iii) $\sum\lambda_i(b_i - \sum a_{ij} x_j) = \lambda_1 (b_1 - (a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n)) + \lambda_1 (b_2 - (a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n)) + \quad . \quad . \quad + \quad .$
$+ \quad . \quad .$
$+ \quad . \quad . \quad . \quad .$
$+ \lambda_m (b_m - (a_{m1}x_1 + a_{m1}x_2 + \dots + a_{mn}x_n))$ (3.3)

Since D is a symmetric matrix, from (3.1)

$$\frac{\partial (x'Dx)}{\partial x_1} = 2 (d_{11}x_1 + d_{12}x_2 + \dots + d_{1n}x_n)$$

$$\frac{\partial (x'Dx)}{\partial x_2} = 2 (d_{21}x_1 + d_{22}x_2 + \dots + d_{2n}x_n)$$

$$\begin{array}{cccc} . & . & . & . \\ . & . & . & . \\ . & . & . & . \end{array}$$

$$\frac{\partial (x'Dx)}{\partial x_n} = 2 (d_{n1}x_1 + d_{n2}x_2 + \dots + d_{nn}x_n)$$

$$= 2 \begin{pmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$= 2Dx$$

Also, from (3.2)

$$\frac{\partial (c'x)}{\partial x_1} = c_1 , \qquad \frac{\partial (c'x)}{\partial x_2} = c_2, \dots ,$$

$$\dots \qquad \frac{\partial (c'x)}{\partial x_n} = c_n$$

In matrix form,

$$\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = c$$

Similarly, from (3.3)

$$\frac{\partial}{\partial x_i} \left(\lambda_i (bi - \sum aijxj \right) =$$

$$\begin{array}{cccc} -a_{11}\lambda_1 - a_{21}\lambda_2 - \dots - a_{m1}\lambda_m \\ -a_{12}\lambda_1 - a_{22}\lambda_2 - \dots - a_{m2}\lambda_m \\ . \quad . \quad . \quad . \\ . \quad . \quad . \quad . \\ -a_{1n}\lambda_1 - a_{2n}\lambda_2 - \dots - a_{mn}\lambda_m \end{array}$$

In matrix form,

$$\begin{pmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{nm} \end{pmatrix} = - A'\lambda$$

So that,

$$\frac{\partial}{\partial x_i} F(x,\lambda) = 2Dx + c - A'\lambda$$

Also,

$$\frac{\partial}{\partial x_i} F(x,\lambda) = b_i - \sum a_{ij} x_j$$

$$= b - Ax$$

Moreover,

$$2Dx + c - A'\lambda \leq 0$$

So that,

$$2Dx + c - A'\lambda + v = 0$$

Where: v is a slack variable

$$v = -2Dx - c + A'\lambda$$
$$v = -F(x,\lambda)$$

Hence,

$-F(x^*, \lambda^*)\ x^* = 0$ *From K-T II*

or

$vx = 0$

Differentiating or contrasting λ for each case we have from the above derivations the following results:

Case 1.

Max $x'Dx + c'x$

s.t.

$Ax \leq b$
$x \geq 0$

From K-T I

$$2Dx + c - A'\lambda \leq 0$$

From K-T II

$v'x = 0 \quad v'x = 0 \ \forall j$

From K-T III

$$b - Ax \geq 0$$

From K-T IV

$\lambda'w = 0$
$\lambda'_i w_i = 0 \quad i = 1, \ldots, s$
≥ 0

where λ_i is represented by ρ_i

Case 2

max $x'Dx + c'x$

s.t.

$Ax \geq b$

From K-T I

$$2Dx + c - A'\lambda \leq 0$$

From K-T II

$v'x = 0$ $\quad$ $v'x = 0 \quad \forall j$

From K-T III

$b - Ax \leq 0$

K-T IV

$\lambda' u = 0 \quad \lambda_i u = 0 \quad i = s+1, \ldots, t$

$- \lambda_i \leq 0$

where λ_i is represented by μ_i

Case 3:

max $\quad x'Dx + c'x$

s.t.

$Ax = b$

From K-T I

$2Dx + c - A'\lambda = 0$

From K-T II

$v'x = 0 \quad v'x = 0 \quad \forall j$

From K-T III

$b - Ax = 0$

From K-T IV

$\lambda'\theta = 0$

$\lambda_i\theta_i = 0 \quad i = t+1, \ldots, m$

λ_i is unrestricted

where λ is represented by $\theta_i^{(+)}$, $\theta_i^{(-)}$

Summarizing the application of Kuhn-Tucker Theorem to Quadratic programming problem, we have the following theorem:

THEOREM 3.3 (Necessary Condition for Quadratic Programming Problem)

Suppose x* is the optimal solution for (****) then x* satisfies the following conditions:

I. $\nabla_x F(x^*, \lambda^*) = 2Dx^* + c - A'\lambda^* \leq 0$

II. $(\nabla_x F(x^*, \lambda^*))\, x = 0$

III. $\nabla_{\lambda} F(x^*, \lambda^*) =$

$b - Ax^* \geq 0$	if $Ax^* \leq b$
$b - Ax^* \leq 0$	if $Ax^* \geq b$
$b - Ax^* = 0$	if $Ax^* = b$

IV. $(\nabla_x F(x^*, \lambda^*))\ \lambda^* \quad =$

$\lambda_i^* w_i$	=	0	if $\lambda^* \geq 0$	$(i = 1, \dots, s)$
$\lambda_i^* u$	=	0	if $\lambda^* \leq 0$	$(i = s+1, \dots, t)$
$\lambda_i^* \theta$	=	0	if $\lambda^* = 0$	$(i = t+1, \dots, m)$

Illustration: (NLP)

Maximize $\quad 4x_1 + 6x_2 - x_1^3 - 2x_2^2$

s.t.
$$x_1 + 3x_2 \leq 8$$
$$5x_1 + 2x_2 \leq 14$$
$$x_1 \geq 0, \quad x_2 \geq 0$$

$F(x,\lambda) = 4x_1 + 6x_2 - x_1^3 - 2x_2^2 + \lambda_1 (8 - (x_1 + 3x_2)) + \lambda_2 (14 - (5x_1 + 2x_2)$

From K-T I,

$$\nabla F_{x1} (x,\lambda) = 4 - 3x_1^2 - \lambda_1 - 5\lambda_2 \leq 0$$
$$\nabla F_{x2} (x,\lambda) = 6 - 4x_2 - 3\lambda_1 - 2\lambda_2 \leq 0$$

From K-T II,

$(x_1^*, x_2^*)\ (F^*x_1 (x,\lambda)) \quad (4 - 3x_1^* - \lambda_1 - 5)\ x_1^* +$
$(F^*x_2 (x,\lambda)) = \quad (6 - 4x_2 - 3\lambda_1 - 2\lambda 2)\ x_2^* \quad = 0$

From K-T III,

a) $(x_1 + 3x_2 - 8) \leq 0$
b) $(5x_1 + 2x_2 - 14) \leq 0$

From K-T IV,

a) $\lambda_1 (x_1 + 3x_2 - 8) = 0$
b) $\lambda_2 (5x_1 + 2x_2 - 14) = 0$

Illustration: (QPP)

maximize $\quad z = -2x_1 2 - x_2 2 + 4x_1 + 6x_2$

s.t.
$$x_1 + 3x_2 \leq 3$$
$$x_1 \geq 0, \quad x_2 \geq 0$$

$F(x,\lambda) = -2x_1^2 - x_2^2 + 4x_1 + 6x_2 + \lambda_i (3 - (x + 3x_2)) \quad (\lambda_2 x_1 + \lambda_3 x_2)$

From K-T I,

$$\nabla\ F_{x1}(x,\lambda) = -4x_1^* + 4 - \lambda_1 - \lambda_2 \quad \leq \quad 0$$
$$\nabla\ F_{x2}(x,\lambda) = -2x_2^* + 6 -3\lambda_1 - \lambda_2 \quad \leq \quad 0$$

From K-T II,

$$(x_1^*, x_2^*)\ F_{x1}(x,\lambda)\ (-4x_1^* + 4 - \lambda_1 - \lambda_2)\ x_1^* +$$
$$F_{x2}(x,\lambda)\ \ (-2x_1^* + 6 - 3\lambda_1 - \lambda_3)\ x_2^* \quad = \ 0$$

From K-T III,

$$3 \quad - x_1 - 3x_2 \qquad \geq \qquad 0$$
$$-x_1 \qquad \leq \qquad 0$$
$$- x_2 \qquad \leq \qquad 0$$

From K-T IV,

$$\lambda_1^*\ (\ 3 - x_1 - 3x_2) \qquad = \qquad 0$$
$$\lambda_2^*\ (-x_1) \qquad = \qquad 0$$
$$\lambda_3^*(-x_2) \qquad = \qquad 0$$

THE WOLFE'S ALGORITHM

One of the earliest and simplest methods for solving quadratic programs was proposed by Philip Wolfe (7) in 1959; despite its rather venerable history, it is still quite commonly used today. In theory, Wolfe's algorithm can be applied directly to any quadratic programming problem as in the form (****).

The basic approach of the algorithm is to generate, by means of a modified simplex pivoting procedure, a sequence of feasible points that terminates at a solution point x* where the Kuhn-Tucker conditions are satisfied. Also, the Wolfe-Simplex method differs from the ordinary simplex method of linear programming in that, in the former, the variable v_j cannot enter the basis if x is already a basic variable (unless x_j will be held at zero or driven to zero by the ensuing pivot and vice versa. Thus the complementary slackness conditions $x_jv_j = 0$, $j = 1, \ldots, n$, are enforced at every pivot).

This special pivoting rule, whereby certain otherwise eligible variables are prohibited from being brought into the basis, is what is known as *restricted basis entry*.

The Wolfe-simplex method terminates when no variable having a negative reduced cost – can be pivoted into the basis without violating the complementary slackness conditions.

I. Description

Step 1: Construct the modified simplex tableau

Refer to Chapter IV-B

Step 2: Input the matrix A, the symmetric matrix D, the vectors b and c.

Step 3: Use the 1st phase of the Two-phase simplex method to obtain the feasible solution.

Use the same regular pivot rule for entering
and leaving variables except for the following
modification.

Check for the complement of the entering variable in the basis.

IF PRESENT: Choose the next most negative
IF NOT : Pivot as usual

Step 4: If a basic feasible solution exists, maximize the difference of the artificial variables r with the first-phase of the Two-phase
simplex method.

Result: An optimal solution of the problem
The value of the objective function can be determined by substituting the optimal values x* for x in the objective function z.

Note: It is important to remember that no two variables xj and vj are basic variables with current values greater than zero.

THE MODIFIED TABLEAU

In this algorithm, a special simplex tableau was devised to suit the modification of the regular simplex method.

The general form of the tableau is given below:

B.V.	x_j	λ				v_j	w_i	u_i	$r_i^{(1)}$	$r_i^{(2)}$	$r_i^{(3)}$	RHS
		ρ_i	μ_i	$\Theta_i^{(+)}$	$\Theta_i^{(-)}$							
w	A	0	0	0	0	0	I	0	0	0	0	
$r^{(1)}$		0	0	0	0	0	0	-I	I	0	0	
$r^{(2)}$		0	0	0	0	0	0	0	0	I	0	b
$r^{(3)}$	2D	$-A^T$				I	0	0	0	0	I	
$r^{(3)}$	-2D	A^T				-I	0	0	0	0	-I	-c
	<---- Z_p-row ---->											VOF

where the complementary slackness:

$X_j v_j = 0$ Ɏj
$P_i w_i = 0$ Ɏi
$\Theta_i^{(+)}, \theta_i^{(-)}, r_i^{(2)} = 0$ Ɏi
$\mu_i u_i = 0$ Ɏi

holds.

Also, the variables used here are defined as follows:

x_j =	the variable used in the objective function to be maximized
v_j =	the slack variable of the inequality constraints obtained by applying the K-T Theorem to QPP
ρ_i =	the Lagrangean multiplier used for "≤" type constraints
μ_i =	the Lagrangean multiplier used for "≥" type constraints
θ_i^+, θ_i^- =	the Lagrangean multiplier used for "=" type constraints
w_i =	the slack variable used for "≤" type constraints
u_i =	the slack variable used for "≥" type constraints
$r_i^{(1)}$ =	the artificial variable used in "≥" type constraints
$r_I^{(2)}$ =	the artificial variable used in "=" type constraints
$r_i^{(3)}$ =	the artificial variable used in the inequality constraints obtained by applying the K-T Theorem to QPP.
b_i =	the right-hand-side values of the original constraints
c =	the right-hand-side values of the obtained inequality constraints

VIII. A Practical Application of the QP Problem

Optimal Use of Milk in Holland

The Dutch Government buys all milk produced by the farmers against a fixed rate. It then tries to sell this milk and its derived products, butter and cheese varieties, on the Dutch market. In this process, the Government can be considered a price adapting monopolist. The question is, given the available quantity of milk, and linear demand functions for milk, butter, fat cheese, and 40+ cheese, how should the Government fix prices so as to maximize revenue?

Write x1, x2, x3 and x4 for the quantities (in 1000 tons) produced of milk, butter, fat cheese and 40+ cheese respectively. And p1, p2, p3 and p4 for the corresponding prices, in 1000 guilders per 1000 tons. Then the demand functions were specified as follows:

$$\begin{array}{lll}
(x_1 - \ddot{x}_1) / \ddot{x}_1 = - (p_1 - \dot{p}_1) / \dot{p}_1 & \text{(milk for consumption)} & (1) \\
(x_2 - \ddot{x}_2) / \ddot{x}_2 = - (p_2 - \dot{p}_2) / \dot{p}_2 & \text{(butter)} & (2) \\
(x_3 - \ddot{x}_3) / \ddot{x}_3 = - (p_3 - \dot{p}_3) / \dot{p}_3 & \text{(fat cheese)} & (3) \\
(x_4 - \ddot{x}_4) / \ddot{x}_4 = - (p_4 - \dot{p}_4) / \dot{p}_4 & \text{(40+ cheese)} & (4)
\end{array}$$

Where x and p stand for their 1960 values:

$$\begin{array}{lcrll}
\ddot{x}_1 & = & 2055 & \dot{p}_1 = 400 & \\
\ddot{x}_2 & = & 54 & \dot{p}_2 = 4000 & \\
\ddot{x}_3 & = & 63 & \dot{p}_3 = 3250 & (5) \\
\ddot{x}_4 & = & 17 & \dot{p}_4 = 2500 &
\end{array}$$

The elasticities are rather uncertain – see below. They were specified as follows:

$$\begin{array}{lcrll}
e_1 & = & 0.3 & e_4 = 0.4 & \\
e_2 & = & 1.5 & e_{34} = 0.06 & (6) \\
e_3 & = & 0.7 & e_{43} = 0.3 &
\end{array}$$

Substituting the values of (5) and (6) into (1) and (4) gives the demand functions numerically:

$$\begin{array}{ll}
x_1 & = -1.5413p + 2671 \\
x_2 & = -0.0203p + 135 \\
x_3 & = -0.0136p + 0.0015p + 103 \\
x_4 & = 0.0016p \; - \; 0.0027p + 19
\end{array}$$

There are seven constraints to be considered. First, the fat constraints:

$$0.026x_1 + 0.800x_2 + 0.306x_3 + 0.241x_4 \leq 121$$

which says that the fat content of the produced milk (2.6 per cent), butter (80 per cent), fat cheese (30.6 per cent) and 40+ cheese (24.5 per cent) cannot exceed the total fat available (121 x 1000 tons of fat). There is a similar, but as will turn out ineffective, constraints with respect to the other ingredients of the dairy product (albumen, lactose, salts), which have been combined under the name dry matter. The dry matter constraint is:

$$0.086x_1 + 0.020x_2 + 0.297x_3 + 0.371x_4 \leq 250$$

There are obviously nonnegativity constraints:

$$\begin{array}{lllllll}
x_1 & & & & \geq & 0 \\
 & x_2 & & & \geq & 0 \\
 & & x_3 & & \geq & 0 \\
 & & & x_4 & \geq & 0
\end{array}$$

Finally, a social constraint was added, stating that a weighted average of the relative price changes was not to exceed 0 percent. With weights based on the budget quotes of the different products we have:

$$6.4[((p_1 - 4000 / 4000] + 1.6 [(p_2 - 4000) / 4000] + 1.6 [(p_3 - 3250] + 0.4[p_4 - 2500) / 2500] \leq \quad 0$$

Or constraints 7

$$0.0160p_1 + 0.0004p_2 + 0.0005p_3 + 0.0002p_4 \leq 10$$

Disregarding production costs, the problem is to maximize $\sum p_i x_i$; this is view of (7), is a quadratic function in p. The maximization is subject to 7 constraints, which, also with (7), can be reformulated as constraints on the p rather than x values. Doing this, we get the quadratic programming problem of maximizing

$$a'p - \tfrac{1}{2} p'Bp$$

$$\text{subject to} \qquad C'p \quad \leq \quad d$$

where a, B, C' and d are given as:

$$a = \begin{pmatrix} 2671 \\ 135 \\ 103 \\ 19 \end{pmatrix}$$

$$B = \begin{pmatrix} 3.0625 & 0 & 0 & 0 \\ 0 & 0.0405 & 0 & 0 \\ 0 & 0 & 0.0271 & -0.0031 \\ 0 & 0 & -0.0031 & 0.0054 \end{pmatrix}$$

$$C' = \begin{pmatrix} -0.0401 & -0.0162 & -0.0039 & 0.0002 \\ -0.1326 & -0.0004 & -0.0034 & 0.0006 \\ 1.5413 & & & \\ & 0.0203 & & \\ & & 0.0136 & -0.0075 \\ & & -0.0016 & 0.0027 \\ 0.0160 & 0.0004 & 0.0005 & 0.0002 \end{pmatrix}$$

$$d = \begin{pmatrix} -92.6 \\ -29.0 \\ 2671 \\ 135 \\ 103 \\ 19 \\ 10 \end{pmatrix}$$

The above numerical values obtained were implemented using the devised computer code and an initial computation resulted in an infeasible solution. This resulted from the rounded values of the input data (as given in Boot's paper) which led to inaccuracies in the resulting binding constraints.

However, getting a hint from the author that only the first and the seventh constraints are binding or effective, the problem was reformulated and got a feasible solution although somewhat different from that obtained by Boot. (Refer to the result at the end of this chapter).

Comparing the results obtained from the two solutions, it was noticed that there was a deviation which can be considered negligible. This deviation can be attributed to numerical errors, specifically to truncation errors.

Nevertheless, it was shown that the computer implementation can work with similar practical problem.

Analysis of the Practical Application

In his paper, J.C.G. Boot using his own method to solve the given practical problem, obtained the following results:

$P_1 = 414$
$P_2 = 3970$
$P_3 = 2891$
$P_4 = 2381$

Substituting the above values to (7)

$$x_1 = -1.5413\,(414) + 2671 = 2032.90$$

$$x_2 = -0.0203\,(3970) + 135 = 54.41$$

$$x_3 = -0.0136\,(2891) + 0.0015\,(2381) + 103 = 67.25$$

$$x_4 = 0.0016\,(2891) - 0.0027\,(2381) + 19 = 7.20$$

And the value of function obtained was

$$\begin{aligned} \text{VOF} &= \sum p_i x_i \\ &= (414)\,(2032.90) + (3970)\,(54.41) + (2891)(67.25) + (2381)(7.20) \\ &= 126919.25 \end{aligned}$$

On the other hand, using the computer implementation in obtaining a solution to the same problem, we got:

$P_1 = 435.3472$
$P_2 = 3935.3142$
$P_3 = 2920.6373$
$P_4 = 0.0000$

Substituting the obtained values to (7),

$$X_1 = -1.5413\,(435.3472) + (2671) = 1999.9994$$

$$X_2 = -0.0203\,(3935.3142) + 135 = 55.1131$$

$$X_3 = -0.0136\,(2920.6367) + 0.0015\,(0) + 103 = 63.2793$$

$X_4 = .0016\ (2920.6373) - 0.0027\ (0) + 19$
$= 23.6730$

Thus, the value of function obtained was:

$\text{VOF} = \sum p_i\, x_i$

$= (435.3472)\ (1999.9994) + (3935.3142)\ (55.1131) + (2920.6373) + 0.0000$
$= 1272397.388$

Notice that there was a deviation, delta Δ, between the two methods used in solving the same given problem,

Delta Δ = / VOF (Wolfe) – VOF (Boot) /

where:

VOF (Wolfe)	= the value of function obtained using the computer implementation of the Wolfe's algorithm
VOF (Boot)	= the value of function obtained using the method of Boot

Delta Δ = / 1272397.388 – 1269191.250 /
= 3206.1376

So that;

$$\Delta = \frac{3206.1376}{1269191.250} = .0025 \text{ or } .25\%$$

Where:

$$\delta = \frac{\text{delta } \Delta}{\text{Actual value}}$$

This deviation can be attributed to truncation error since the numerical field or the decimal accuracy used in the computer implementation was different from that used by Boot in his work. However, this deviation is very small and can be considered negligible in terms of percentage with respect to the actual value obtained in the computations.

SUMMARY, CONCLUSION AND RECOMMENDATIONS

This special problem and research study takes into consideration the study on the Kuhn-Tucker necessary and sufficient conditions which is considered to be the backbone for the formulation of the Wolfe's algorithm and the implementation of the algorithm as a computer code. This study is consistent with the methodological process of solving existing problems which are commonly and inevitably encountered in everyday decision making for top-level management.

Quadratic programming problems are readily solvable by this algorithm and the solution can be done manually. However, in real-life problems, to solve manually will be futile since the problems normally involved many constraints and variables. Thus, a computer code was developed that will enable managers and decision makers to make an immediate but efficient decision to these types of problems.

Although the computer implementation for such problem is limited in some extent because of the nature of the computer itself, it can safely be said that the time needed for solving the problems is lessened even though the exact execution time of the computer implementation was not taken into account.

One recommendation for future researcher interested in this research study is to produce a faster and more reliable computer code of better programming methods and languages. The implementation was done more than 20 years ago (as a pioneering study in this subject/topic) using the outmoded Pascal IV programming language and the now-obsoleted IBM PC/XT and AT.

With the advent of better and faster computer systems and languages, it is expected that the implementation of this algorithm using the same underlying principles and theories will improve the calculations and computations drastically.

Some examples of commonly encountered problems stated in this study are the regression analysis, which can be found in any practical problem involving statistical analysis, the portfolio problem which is a problem in investment analysis and production planning, an important part of a management system.

These are only few of the many applications of quadratic programming and this field is still open for further studies and developments.

One existing practical QP problem was that discussed in Chapter V entitled the "The Optimal Use of Milk in Holland". This problem was formulated by J.C.G. Boot in his paper published in an OR Journal and he solved this problem using his own method which is very far different from that developed by Wolfe. Implementing this problem using the devised computer code showed with great simplicity the importance of developing a computer solution to similar types of problems.

Although Wolfe can be said as one of the simplest and easy-to-implement algorithms, there are other algorithms suited for QP problems. The prospective researcher is, therefore, recommended to implement QP problems using the method of Beale, which was based on arguments from classical calculus rather than on the Kuhn-Tucker conditions, and Lemke's algorithm, which is rather more sophisticated and elegant than either of the two previous methods. The methods of Wolfe, Beale and Lemke are three of the best known and widely used quadratic programming problems in existence.

Also, the researcher is recommended to improve the devised computer code if he intends to continue this study and make improvement on the existing modified tableau developed by the author.

LITERATURE CITED

1. Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J. 1963
2. Walsh, G.R., *Methods of Optimization*, John Wiley & Sons Ltd., 1975
3. Bazaraa, M.S. and Shetty C.M., *Nonlinear Programming*, John Wiley & Sons Ltd., 1979
4. Boot, J.C.G. "On Sensitivity Analysis in Convex Quadratic Programming Problems", *Operations Research*, 11, pp. 771-786
5. Kunzli, H.P., et. Al., *Nonlinear Programming*, Blaisdell Publishing
6. Mitra, G., *Theory and Applications of Mathematical Programming*, Academic Press, 1976
7. Wolfe, P., "The Simplex Method for Quadratic Programming", *Econometrica*, 27, pp. 382-398, 1959
8. Guder, Faruk. "An interregional analysis of the North American newsprint industry", *School of Business Journal*, Loyola University of Chicago, 1984
9. Buongiorno, Joseph. "An interregional analysis of the North American newsprint industry", *Interfaces (U.S)*, 14{1984} 5{Sept/Oct}, pp. 85-95
10. 10. Taha, H.A., *Operations Research*, Macmillan Publishing Co. Inc., 3rd ed., 1982

IMPORTANT NOTE:

In this report, the appendix containing the computer program code and listings were not included. However, this computer code is available and can be directly requested from the above author/researcher, subject to the validity of the request, e.g. further study, academic research and presentation in events relevant to the topic of this study and request related to publication in international journals that will contribute to the expansion of knowledge in the mathematical science.

The appendix also includes the user manual, the storage and systems requirements in using the WOLFE algorithm computer program devised and developed to solve quadratic programming problems.

This research study was done in 1987 as a partial requirement to the attainment of the Bachelor of Science degree in Applied Mathematics in the University of the Philippines at Los Banos. During that period, the author is constrained with limited resources with regards to the use of computer system, programming language and other things necessary to implement Wolfe Algorithm into a working automated computational system. It is therefore recommended that prospective researchers should look at improving the computer implementation part of the research study using advanced and up-to-date methods in computational mathematics.

For the purpose of improving this existing program, it is advisable to contact the above author to request for a copy of the computer program and code subject to copyright laws and author’s right to this research study. It is also advisable and appreciated if the above author is commended and mentioned every time this study is used in other related and further studies.

The author, therefore, would like to cite the importance of this study as a pioneering works in the Philippines related to mathematical programming since it started the implementation of known methods popular during those days and until today. It also paved the way for similar studies using other methods such as those by Lemke, Beale and earlier pioneers of mathematical programming and optimization.